Efficiency Enhancement of Secure Multi-Party Protocols for Privacy and Privilege Protection
プライバシーと権限を守るための安全な多者間プロトコルの効率化に関する研究

by

Jun Furukawa

Submitted to the Department of Information and Communication Engineering
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

THE UNIVERSITY OF TOKYO

September 2006

Certified by Kanta Matsuura
Associate Professor, Institute of Industrial Science
Thesis Supervisor

Efficiency Enhancement of Secure Multi-Party Protocols for Privacy and Privilege Protection
プライバシーと権限を守るための安全な多者間プロトコルの効率化に関する研究

by

Jun Furukawa

Submitted to the Department of Information and Communication Engineering
on June 16, 2006, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

Secure multi-party protocols are cryptographic protocols which multiple players engage in. Many application specific multi-party protocols have been proposed, which include electronic voting, electronic cash, electronic auction, broadcast encryption, traitor tracing, anonymous authentication, group signature, secret sharing, conference (group) key distribution, group key generation, etc. These multi-party protocols are designed mainly so as to carry out social activities in the network. The technique of general multi-party computation that enables multi-party to securely compute any efficient function is known. However, its results are mostly far from practical efficiency in the sense of computational, communication, and round complexity. This is the major reason why a large number of specific constructions of multi-party protocols has been proposed. Since a multiple number of players engage in multi-party protocols, they often have complex security requirements. Such circumstance, besides the fact that the number of players itself is large, makes it hard to construct efficient multi-party protocols. Thus, efficiency enhancement is the major and crucial interest in these designing of multi-party protocols. Among these examples of multi-party protocols presented above, electronic voting, broadcast encryption, and group signature can be listed as most useful multi-party protocols. This dissertation presents several efficiency enhanced variants of these protocols. They are an efficient publicly verifiable shuffle scheme, an efficient publicly verifiable shuffle and decryption scheme, an efficient publicly verifiable hybrid mix-net scheme, an aggregate shuffle argument scheme, an efficient compiler from Σ-protocol to deniable zero-knowledge argument, a black-box traitor revocable broadcast encryption scheme, a group signature scheme for separate and distributed authorities, and an efficient group signature based on bilinear mappings. These protocols are efficient enough for practical purposes.

Thesis Supervisor: Kanta Matsuura
Title: Associate Professor, Institute of Industrial Science

Contents

Acknowledgments

I had been a member of Imai Laboratory in the first two and half year during the doctorial course in the University of Tokyo. I want to express my gratitude to Professor Hideki Imai for encouraging me to join his laboratory and giving me many practical advises during my stay in his laboratory. After professor Hideki Imai left the university, I have joined Matsuura Laboratory to complete my Ph.D., where I could get the advice necessary to finish this dissertation from Professor Kanta Matsuura. I express my gratitude to him. Study in Imai Laboratory was very exciting. I could enjoy stimulating discussions with Goichiro Hanaoka, Rui Zhang, Nuttapong Attrapadung, and other members. I have been candidate for the Ph.D. and worked for NEC Corporation simultaneously for three years. I am grateful for Kazue Sako, giving me nice themes to study in NEC Corporation. I am also grateful for Hiroshi Miyauchi, my former supervisor in NEC Corporation. Without his support and arrangement, I could not go to study in Imai Laboratory.

Chapter 1

Preliminaries

Cryptography had originally arisen as measures of private communication between two parties through wire-tapped channel. It has been then developed to include measures to guarantee authenticity. In cryptography, these are two party protocols now known as public key encryption, symmetric key encryption, signature, message authentication, etc.

While these primitives for two parties remain to be central interest of cryptography, modern cryptography has expanded to encompass protocols for multiple of parties. Such a multi-party protocols include general multi-party computation, electronic voting, electronic cash, electronic auction, broadcast encryption, traitor tracing, anonymous authentication, group signature, secret sharing, conference (group) key distribution, group key generation, etc. These multi-party protocols are designed mainly so as to carry out social activities within the network.

Some of multi-party protocols can be constructed by combining simple two-party protocols. Such a simple primitive based construction, however, often ends up with inefficiency, especially when the number of participants are huge. The technique of general multi-party computation enables multi-party to securely compute any efficient function but its results are often far from practical efficiency in the sense of computational, communication, and round complexity. This is the major reason that a large number of specific constructions of multi-party protocols has been proposed. Thus, efficiency enhancement is the major and crucial interest in these designing of multi-party protocols.

Among these examples of multi-party protocols presented above, electronic voting, broadcast encryption, and group signature can be listed as most useful multi-party protocols. This dissertation presents several efficiency enhanced variants of these protocols. These protocols are efficient enough for practical purposes.

1.1 Mix-net

Electronic voting system has a potential to enable us to engage in a voting from our homes, from office, during a travel, or during a hospital stay. Although the notion of electronic voting is promising, it has severe and conflicting requirements which tends to make the system terribly inefficient. Since the number of voters who engage a voting can be huge, only deliberately constructed scheme can serve for practical purpose. The most promising cryptographic protocols that can realize electronic voting systems are publicly verifiable mix-nets. Hence, proposing efficient electronic voting system can be achieved by proposing efficient publicly verifiable mix-nets.

Mix-Nets are composed of a sequence of shuffles and decryptions by multiple and independent mixers. Shuffles are core parts of mix-nets. First they are given multiple of ciphertexts as an input. Next they permute them and then reencrypt the results of the permutation. Finally, they outputs the resulting ciphertexts. Because of reencryption and permutation, input ciphertexts and output ciphertexts of a shuffle are unlinkable to every one except the mixer itself. Hence, the relation between inputs and outputs of a mix-net is disclosed to no one unless all the mixer collude. If shuffles and decryptions in a mix-net is publicly verifiable, that is, the correctness of permutation and reencryption and that of decryption can be verified by any player, the mix-net will be also publicly verifiable.

When such a publicly verifiable mix-net is applied to electronic voting, the anonymity of voters is guaranteed by unlinkability of the input and the output ciphertexts of the mix-net and the correctness of tallying is guaranteed by its public verifiability. Hence the publicly verifiable mix-nets are protocols that can securely realize electronic voting system.

The correctness of shuffles need to be proved without sacrificing their unlinkability, which is often done by a zero-knowledge argument system. It is true that any language in NP can be proven in zero-knowledge, however, the result of such a general approach does not have enough efficient for practical purposes. Indeed, it is crucial to make the shuffle argument part efficient to make the total mix-net efficient. In Chapter 2, we propose an efficient protocol for proving the correctness of shuffling and an efficient protocol for simultaneously proving the correctness of both shuffling and decryption. The former protocol is, as of our proposal, the most efficient in computational and communication complexity among 3-move honest verifier perfect zero-knowledge protocols for proving a shuffling of ElGamal cipher-texts. The protocol still has very close efficiency to the most efficient one. The latter protocol is also the most efficient in computational, as of our proposal, in communication, and round complexity, as a whole, in proving the correctness of both shuffling and decryption of ElGamal cipher-texts.

In Chapter 3, we propose an efficient publicly verifiable hybrid mix-net. The publicly verifiable mix-net we consider in Chapter 2 was only efficient for short ciphertexts and was not suitable for mixing long messages. Hybrid-mix nets are mix-nets that are suitable for handling long ciphertexts of hybrid type. Previous hybrid mix-net can mix long messages but did not have public verifiability. The proposed scheme is efficient enough to treat large scale electronic questionnaires of long messages as well as voting with write-ins, and offers public verifiability of the correctness of the tally. A cost we pay for public verifiability is that the length of the input ciphertext grows in linear in the number of mixers. The scheme is provably secure if we assume random oracles, semantic security of a one-time symmetric-key cryptosystem, and intractability of decision Diffie-Hellman problem.

Verifiers in publicly verifiable mix-net in Chapter 2 are required to verify the correctness of each mixer. Hence, if the number of mixers is large, the cost for verifying mix-nets will be large. Since the anonymity of mix-nets are guaranteed only when at least one mixer is honest, the number of mixers need to be large to achieve high anonymity. But increasing the number of mixers increase the cost for verification.

In Chapter 4, we propose a novel scheme to prove the correctness of mix-net that is composed of multiple shufflings, in such a way that the computational complexity of its verifier does not depend on the number of its composite shufflings. We call this scheme an *aggregate shuffle argument scheme*. Although a similar scheme proposed by Abe in Eurocrypt '98 exists, our scheme is much more efficient. In fact, the computational cost required for the verifier in our scheme is less than 1/60 of that in Abe's scheme. We achieved such an efficiency since we constructed our protocol based on our efficient shuffle arguments in Chapter 2, while Abe's scheme exploits the shuffle proof proposed in Sako et al. in Eurocrypt '95. We also proposed a formal model and security requirements of aggregate shuffle argument schemes.

Mix-nets are good candidate cryptographic protocols for electronic voting. However, it is not a perfect substitution for non electronic voting system. One major problem is receipt freeness of voting systems. That is, each voter should not be able to prove whom he have voted to third parties. Without this such a property, vote-buying will be easy. One solution is to make the receipts of voting center to be non transferable to third parties. We also give an efficient and general technique to make non transferable proof system. That is a technique to make deniable zero-knowledge argument.

Pass showed a 2-move deniable zero-knowledge argument scheme for any $\mathcal{NP}$ language in the random oracle model at CRYPTO 2003. However, this scheme is very inefficient because it relies on the cut and choose paradigm (via straight-line witness extractable technique). In Chapter 5, we propose a very efficient compiler that transforms any Σ-protocol to a 2-move deniable zero-knowledge argument scheme in the random oracle model, which is also a resettable zero-knowledge and resettably-sound argument of knowledge. Since there is no essential loss of efficiency in our transform, we can obtain a very efficient undeniable signature scheme and a very efficient deniable authentication scheme. We achieved to balance all properties such as efficiency, deniability, resettable zero-knowledge, and resettably-sound argument of knowledge in the random oracle model, which is impossible by black-box reduction in the standard model.

1.2 Black-Box Traitor Revocable Broadcast Encryption

Broadcast encryption schemes enable senders to distribute ciphertexts of some contents to registered receivers. The schemes have applications in pay-TV, DVD system etc. Its primary purpose can be achieve by simple encryption scheme if one distributes the secret key of the encryption scheme to all the registered users. However, modern broadcast encryption schemes require more functionalities. The schemes require their ciphertexts to be decryptable only by non-revoked users and require a manager that can trace the receiver, who has leaked its key from a pirate decoder, to be provided. These requirements can be satisfied if the broadcasted ciphertexts are simple combinations of all the ciphertexts, each encrypted for each receiver. But this is unrealistically inefficient.

Black-box traitor revocable broadcast encryption schemes are encryption schemes that enable senders to efficiently broadcast ciphertexts to a large set of receivers in a way that only non-revoked receivers can decrypt them and that, given a pirate decoder, enable a tracer to identify traitors by black-box accessing the pirated decoder and to revoke traitors so identified. Since the number of receivers in broadcast encryption schemes is potentially large, pirate decoders are always a serious threat, and that is why traitor revocable schemes are useful for them. Such schemes are useful for distributing contents with copyright since one can effectively disable pirate decoders.

Broadcast encryption schemes enable senders to efficiently broadcast ciphertexts to a large set of receivers in a way that only non-revoked receivers can decrypt them.

In Chapter 6 we propose a black-box traitor revocable broadcast encryption scheme in which the size of each of its private keys is constant, the size of the public key is proportional to the number of receivers, and the sizes of its ciphertexts are sub-linear with respect to the number of receivers. The encryption procedure in our scheme requires only a public key. The tracing procedure in it requires a secret key and black-box access to a resettable pirate decoder. The security of our scheme is proved in the generic bilinear group model.

1.3 Group Signature

Group signatures schemes enable a member of a group to sign on behalf of its group without revealing his own identity. The scheme also realizes a special authority that can identify actual signers in case of dispute. Group signatures have many applications in which user anonymity is required such as in anonymous credential systems, identity escrow, voting and bidding, and electronic cash systems.

In group signature schemes, their authorities have rather strong privilege. They can add new member and trace members at any time. Such a strong privilege is often hard to manage in practical operation. In Chapter 7, we propose a new group signature scheme that simultaneously provides the following two properties : (1) the membership authority is able to add a user but not to identify an actual signer, while the tracing authority is able to identify the actual signer but not to add a user, (2) for further decentralization, these two authorities are respectively distributed among multiple entities in a manner efficient enough for practical applications. Previous group signature schemes, as of our proposal, have only offered one or the other of these two properties. Further, we formalize the security properties in a group signature scheme that has the former property and in which either of the two authorities is potentially dishonest.

Although earlier group signature schemes required large computational cost and long signatures, many of recently proposed schemes are very efficient. These successes of efficient constructions open the way for using them in practical applications. In Chapter 8, we propose a new group signature scheme which is secure if we assume the Decision Diffie-Hellman assumption, the q-Strong Diffie-Hellman assumption, and the existence of random oracles. Within these assumption, the proposed scheme is the most efficient among the all previous group signature schemes in signature length and in computational complexity.

Chapter 2

Publicly Verifiable Shuffle, Shuffle and Decryption

2.1 Introduction

A mix-net [42] protocol is useful in applications which require anonymity, such as voting. Crucial to a mix-net protocol is the execution of multiple rounds of shuffling and decryption by multiple, independent mixers, so that none of the output decryptions can be linked to any of the input encryptions.

To ensure the correctness of output, it is desirable to achieve the property of public verifiability. Early studies, such as those by Sako and Kilian [137] and Abe [2], required vast amounts of computation to prove and verify the correctness of a shuffling in a mix-net without sacrificing unlinkability. However, recently proposed protocols [69, 70, 85, 120] were sufficiently efficient and practical. The protocols of [69, 70] use the property of permutation matrixes, and the protocols of [85, 120] use the fact that polynomials remain invariant under the permutations of their roots. The protocols of [70], [85] , and [120] require the respective computation of $18k$, $12k$, and $42k$ modular exponentiations to prove and verify the correctness of a shuffling of k data. The protocol of [69] requires $19k$ modular exponentiations to prove and verify both shuffling and decryption. Groth's protocol [85] is the most efficient.

A result of these recent works is that proving the correctness of decryption now costs as much as proving the correctness of shuffling. Hence, decreasing the cost of proving decryption has also become important for mix-net. It is now also important in these protocols to decrease round complexity. The round complexity of the interactive protocols is one of their most important complexity measures. Indeed, substantial research efforts have been invested into reducing round complexity in many cryptographic protocols. In particular, no efficient 3-round zero-knowledge argument for shuffling is known. Indeed, the 3-round protocol proposed in [70] is not zero-knowledge.

In this chapter, we present two protocols. The first protocol is a 3-round special honest-verifier perfect zero-knowledge argument for shuffling, which is the most efficient among less-than-7-rounds zero-knowledge arguments for shuffling. The second protocol is a 5-round protocol that proves the correctness of both shuffling and decryption, which is the most efficient in computational, communication, and round complexity, as a whole, in proving the correctness of both shuffling and decryption of ElGamal ciphertexts.

The first protocol is a refinement of the non zero-knowledge protocol in [70] which is not zero-knowledge. We introduced more random elements into the original protocol to make it turn into a zero-knowledge protocol. We also reduced its computational and communication complexity. The second protocol is based on the first protocol. Here, we reduce the cost for decryption by choosing a strategy of simultaneously proving the correctness of shuffling and decryption. The protocol of shuffle-decryption was first presented in [128]. The technique we used here is the same as that introduced in [69]. However, as is mentioned in [69], the protocol of [69] is not zero-knowledge, and this simultaneously proving technique never yields a zero-knowledge protocol.

Although the second protocol is not zero-knowledge, we propose a formal definition for the core requirement of unlinkability in arguments for ElGamal shuffle-decryption and proved that the proposed argument is secure in the sense of this definition. We call this property "complete permutation hiding (CPH)." If an argument for shuffle-decryption is CPH, polynomially-bounded honest verifiers, who may have some partial information about its permutation, will learn nothing new about its permutation from an interaction with a prover in an overwhelming number of cases of input even if the arguments with the same private key are repeatedly executed, whereas if the argument is zero-knowledge, verifiers will learn nothing new in every case of input.

The definition of security for argument for shuffle-decryption stated in [69] is "No polynomially bounded adversary can compute any partial information of the permutation from the protocol". Unlike our new definition, this definition does not mention the case where the verifier has already obtained partial information before the protocol begins and where the shuffle-decryptions with the same private key are repeatedly executed. These cases seem to occur quite often.

A simple combination of two zero-knowledge protocols of an argument for shuffling and of an argument for decryption also does not yield a zero-knowledge protocol since the intermediate state cannot be simulated. And our second protocol is not weaker than this simply combined protocol.

We also analyzed efficiency of proposed protocols. Our first protocol requires roughly $15k$ exponentiations, $1344k$ bits of communication, and three rounds to prove in perfect zero-knowledge the correctness of a shuffling of k-data. The most efficient previously known 3-round perfect zero-knowledge proof or argument for shuffling is the protocol of [137], which requires vast amounts of computation, i.e., $640k$ modular exponentiations. Our second protocol requires roughly $14k$ exponentiations, $1344k$ bits of communication, and five rounds to prove and verify the correctness of both shuffling and decryption of k-data. To prove and verify the correctness of both shuffling and decryption of k-data with Groth's protocol [85] by using the standard technique[1] for proving the correctness of decryption, we require $15k$ exponentiations, $2528k$ bits of communication, and seven rounds.

This chapter is organized as follows. Section 2.2 introduces the properties of permutation matrixes. Section 2.3 proposes a 3-round honest verifier perfect zero-knowledge argument for shuffling. Section 2.4 proposes a protocol by which we are able to simultaneously prove the correctness of a shuffle-decryption in an efficient way. Section 2.5 proposes a definition for the requirement of unlinkability in arguments for shuffle-decryption. Section 2.6 compares the efficiency of our protocols to prior work.

2.2 Permutation Matrix

We define a permutation matrix $(A_{ij})_{i,j=1,\cdots,k}$ as follows:

Definition 1 *Let q be a prime. A matrix $(A_{ij})_{i,j=1,\cdots,k}$ is a* permutation matrix over $\mathbb{Z}/q\mathbb{Z}$ *if it satisfies*

$$A_{ij} = \begin{cases} 1 \bmod q & \textit{if } \phi(i) = j \\ 0 \bmod q & \textit{otherwise} \end{cases}$$

for a permutation function $\phi : \{1, \ldots, k\} \to \{1, \ldots, k\}$.

The following theorems are the key observations used to construct the proposed protocols.

Theorem 1 ([70] Theorem 1) *Let q be a prime. A matrix $(A_{ij})_{i,j=1,\ldots,k}$ is a permutation matrix over $\mathbb{Z}/q\mathbb{Z} \Leftrightarrow$*

$$\sum_{h=1}^{k} A_{hi}A_{hj}A_{hg} = \delta_{ijg} \quad (2.1)$$

$$\sum_{h=1}^{k} A_{hi}A_{hj} = \delta_{ij} \quad (2.2)$$

for all $i, j,$ and g.

[1]This technique is explained in Section 2.6.

Here

$$\begin{aligned} \delta_{ij} &\triangleq \begin{cases} 1 \pmod{q}, & \textit{if} \quad i = j \\ 0 \pmod{q}, & \textit{if} \quad i \neq j \end{cases} \\ \delta_{ijg} &\triangleq \delta_{ij}\delta_{jg} \end{aligned}$$

Proof. ($\Rightarrow$) is trivial. ($\Leftarrow$); Let us first show that there is exactly one non-zero element in each row vector of (A_{ij}) and then show the same for each column vector. Let C_i be an i-th column vector of matrix $(A_{ij})_{i,j=1,\ldots,k}$. From Eq. (2.2), $\sum_{h=1}^{k}(A_{hi}A_{hi})(A_{hg}) = \delta_{ig}$. Thus, $rank(A_{ij}) = k$, i.e., there is at least one non-zero element in each row and each column. Next, let us consider a vector $C_i \odot C_j (i \neq j)$ for which the operator $\odot$ is defined as $(a_1 \ldots a_k) \odot (b_1 \ldots b_k) = (a_1 b_1 \ldots a_k b_k)$. Define a vector $\hat{C} = \sum_{l=0}^{k} \kappa_l C_l$ for an arbitrary κ_l. From the facts that $(\hat{C}, C_i \odot C_j) = \sum_{l=1} \kappa_l \delta_{lij} = 0$ and that linear combinations of $\{C_l\}$ generate the space $(\mathbb{Z}/q\mathbb{Z})^k$, we obtain $C_i \odot C_j = \vec{0}$. This means that for any h, i and j such that $i \neq j$, either $A_{hi} = 0$ or $A_{hj} = 0$. Therefore, the number of non-zero elements in each row vector of (A_{ij}) is at most 1, and thus exactly 1.

From the above observations, matrix (A_{ij}) contains exactly k non-zero elements. Since $C_i \neq \vec{0}$ for all i, the number of non-zero elements in each column vector is also 1. Thus, there is exactly one non-zero element in each row vector and each column vector of matrix (A_{ij}) if Eqs. (2.2) and (2.1) hold.

From Eq. (2.2), the unique non-zero element e_i in the $i-th$ row must satisfy $e_i^2 = 1 \bmod q$, and from Eq. (2.1), it must satisfy $e_i^3 = 1 \bmod q$. This leads to $e_i = 1$ and to the fact that matrix $(A_{ij})_{i,j=1,\ldots,k}$ is a permutation matrix over $\mathbb{Z}/q\mathbb{Z}$. □

Theorem 2 *For* $q \bmod 3 = 2$*, a matrix* $(A_{ij})_{i,j=1,\ldots,n}$ *is a permutation matrix over* $\mathbb{Z}/q\mathbb{Z} \Leftrightarrow$ *Eq. (2.1) holds.*

Proof. ($\Rightarrow$) is trivial. ($\Leftarrow$); From the proof of Theorem 1 1, if Eq.(2.1) holds, then there is only one non-zero element e_i in each i-th row and it must satisfy $e_i^3 = 1 \bmod q$. Because $q \bmod 3 = 2$ implies that 1 is the only cubic root of 1 in $\mathbb{Z}/q\mathbb{Z}$, e_i must be 1. Therefore, matrix $(A_{ij})_{i,j=1,\ldots,n}$ is a permutation matrix over $\mathbb{Z}/q\mathbb{Z}$.

□

The soundness of our first protocol depends directly on Theorem 1. The soundness of our second protocol depends directly on Theorem 2.

2.3 Perfect Zero-knowledge Argument for Shuffling

In this section, we propose a 3-round honest verifier perfect zero-knowledge argument for shuffling which is the most efficient among all previous less-than-7-round zero-knowledge arguments for shuffling.

2.3.1 Notation

Let p, q be two primes s.t. $q|p-1$, $\mathbb{G}_q$ be an order q subgroup of $(\mathbb{Z}/p\mathbb{Z})^*$, $g_0(\neq 1)$ be an element of $\mathbb{G}_q$, $x \in \mathbb{Z}/q\mathbb{Z}$ be a private key, $m_0 = {g_0}^x \bmod p$ be a public key used for re-encryption in shuffling, and k be the number of ElGamal cipher-texts to be shuffled. Let $\{(g_i, m_i) \in \mathbb{G}_q^2\}_{i=1,\dots,k}$ be ElGamal cipher-texts to be shuffled by the public-key m_0. The resulting cipher-texts is denoted as $\{(g'_i, m'_i) \in \mathbb{G}_q^2\}_{i=1,\dots,k}$. Treating the public key g_0, m_0 as if it were an element in a cipher-text vector may be awkward, but it gives a more compact and unified representation to variables. Let P be a prover who shuffles and proves the correctness of shuffling to a verifier V.

Let $F_k \triangleq \{f_\nu \in_R \mathbb{G}_q\}_{\nu=-4,\dots,k}$ be $k+5$ elements of $\mathbb{G}_q$ that are uniformly and randomly generated so that neither P nor V can generate non-trivial integers $a, \{a_\nu\}_{\nu=-4,\dots,k}$ satisfying $g_0^a \prod_{\nu=-4}^{k} {f_\nu}^{a_\nu} \equiv 1 \pmod p$ with non-negligible probability. The public key is a set, $\{p, q, g_0, m_0, F_k\}$.

2.3.2 ElGamal Shuffling

ElGamal shuffling is a procedure that, given g_0, m_0 and k ElGamal cipher-texts $\{(g_i, m_i)\}_{i=1,\dots,k}$, outputs ElGamal ciphertexts

$$(g'_i, m'_i) = ({g_0}^{s_i} g_{\phi^{-1}(i)}, {m_0}^{s_i} m_{\phi^{-1}(i)}) \bmod p \tag{2.3}$$

for $i = 1, \dots, k$ where $\{s_i \in_R \mathbb{Z}/q\mathbb{Z}\}_{i=1,\cdots,k}$ and a permutation of indices $\phi : \{1, \dots, k\} \to \{i = 1, \dots, k\}$ are chosen uniformly and randomly.

Shuffling of ElGamal cipher-texts results in the following two important properties:

1. There exists a permutation ϕ such that equations $D_x((g'_i, m'_i)) = D_x((g_{\phi^{-1}(i)}, m_{\phi^{-1}(i)}))$ hold for all i. Here, $D_x(\cdot)$ is a decryption algorithm that uses the private key x that corresponds to m_0.

2. As long as the decision Diffie-Hellman problem is difficult to solve, no polynomially bounded algorithm, given only $p, q, g_0, m_0, \{(g_i, m_i)\}_{i=1,\cdots,k}$, $\{(g'_i, m'_i)\}_{i=1,\cdots,k}$, has an advantage over the random-guessing algorithm in guessing any part of permutation ϕ for uniformly and randomly chosen g_0, m_0, s_i, ϕ.

Using a permutation matrix (A_{ji}), which corresponds to a permutation ϕ, we find that Eq. (2.3) can be expressed as

$$(g'_i, m'_i) = ({g_0}^{s_i} \prod_{j=1}^{k} g_j^{A_{ji}}, {m_0}^{s_i} \prod_{j=1}^{k} m_j^{A_{ji}}) \bmod p. \tag{2.4}$$

Therefore, proving the correctness of the shuffle is equivalent to proving the existence of an $s_i \in \mathbb{Z}/q\mathbb{Z}$ for $i = 1, \ldots, k$ and a permutation matrix $(A_{ji})_{i,j=1,\ldots,k}$ which satisfies Eq. (2.4).

2.3.3 Protocol Structure

The zero-knowledge argument for shuffling we will propose in this section is almost the same as the protocol proposed in [70]. The proposed protocol and the protocol of [70] are roughly composed of three components. Those in [70] are, (i) generation of $\{f'_i\}_{i=1,\cdots,k}$ and an argument of knowledge of s_i and a general matrix (A_{ji}) that satisfy

$$f'_i = f_0^{s_i} \prod_{j=1}^{k} f_j^{A_{ji}} \bmod p \quad i = 1, \cdots, k, \tag{2.5}$$

for uniformly and randomly chosen $\{f_\mu \in_R \mathbb{G}_q\}_{\mu=0,\ldots,k}$, (ii) proof that (A_{ji}) whose knowledge proved in (i) is a permutation matrix using Theorem 1, and (iii) proof that s_i and (A_{ji}) whose knowledge is proved in (i) also satisfies Eq. (2.4). In Proof (ii), there are commitment, challenge, and response phase.

The main difference between our protocol and the protocol of [70] is that we have introduced the values $f_{-4}, f_{-3}, f_{-2}, f_{-1}$ in the proposed protocol. Because of these values f'_is in the commitment are modified from $f'_i = f_0^{A_{0i}} f_{\phi^{-1}(i)}$ to $f'_i = f_{-4}^{A_{-4i}} f_{-3}^{A_{-3i}} f_{-2}^{A_{-2i}} f_{-1}^{A_{-1i}} f_0^{A_{0i}} f_{\phi^{-1}(i)}$. As a result, we have more redundancy $(A_{-4i}, A_{-3i}, A_{-2i}, A_{-1i})$ to generate the value $\{f'_i\}_{i=1,\ldots,k}$. Then we adjusted $A_{-4i}, A_{-3i}, A_{-2i}$ so that some values in the commitment become zero, which decreased the number of terms in checking equations in the response phase[2]. And because of the redundancy gained by A_{-1i}, the proposed protocol enjoys zero-knowledgeness unlike the protocol in [70].

The other difference between them is with respect to the verification of Eq. (2.9). A verification that Eq. (2.9) holds is equivalent to verifying that equations

$$\begin{aligned} \prod_{\nu=-4}^{k} {f_\nu}^{r_\nu} &= f'_0 \prod_{i=1}^{k} {f'_i}^{c_i} \pmod p \\ \prod_{\nu=-4}^{k} {f_\nu}^{r'_\nu} &= \tilde{f}'_0 \prod_{i=1}^{k} {f'_i}^{{c_i}^2} \pmod p \end{aligned}$$

[2]The equations in [70] that correspond to Equations (2.12) and (2.13) are Equations (15) and (16) in [70]. We can see that terms quadratic and linear to elements of the challenge disappear in the proposed protocol.

hold. The verifier is able to additionally check that the second equation holds with negligible overhead. Since, r'_{-2} in this second equation plays the role of λ' in [70], we can omit Equation (14) in [70].

2.3.4 Proposed Zero-Knowledge Argument

Let us next introduce our argument for shuffling.

ElGamal Shuffling

P uniformly and randomly chooses $\{A_{0i} \in_R \mathbb{Z}/q\mathbb{Z}\}_{i=1,\cdots,k}$ and a permutation matrix $(A_{ji})_{i,j=1,\cdots,k}$. P shuffles k ElGamal ciphertexts $\{(g_i, m_i)\}_{i=1,\cdots,k}$ to $\{(g'_i, m'_i)\}_{i=1,\cdots,k}$ as

$$\begin{aligned} (g'_i, m'_i) &= (\prod_{\nu=0}^{k} {g_\nu}^{A_{\nu i}}, \prod_{\nu=0}^{k} {m_\nu}^{A_{\nu i}}) \bmod p \qquad (2.6) \\ &= ({g_0}^{A_{0i}} g_{\phi^{-1}(i)}, {m_0}^{A_{0i}} m_{\phi^{-1}(i)}) \bmod p. \end{aligned}$$

Here, the matrix $(A_{\nu i})_{\nu=0,\ldots,k;i=1,\ldots,k}$ in Eq. (2.6) is a $(k+1)\times k$ matrix that contains a permutation matrix whose size is $k \times k$ as a sub-matrix. It should be remembered here that, in the rest of the chapter, just as the size of matrices may often change, so will the range over which the indices of the matrix A run.

Proving a Shuffle

Commitment: P uniformly and randomly chooses $\{A_{\nu 0}, A'_\nu\}_{\nu=-4,\ldots,k}$ and $\{A_{-1i}\}_{i=1,\cdots,k}$ from $\mathbb{Z}/q\mathbb{Z}$, and then computes:

$$\begin{aligned} A_{-2i} &= \sum_{j=1}^{k} 3{A_{j0}}^2 A_{ji} \bmod q \quad i = 1, \cdots, k \\ A_{-3i} &= \sum_{j=1}^{k} 3A_{j0} A_{ji} \bmod q \quad i = 1, \cdots, k \\ A_{-4i} &= \sum_{j=1}^{k} 2A_{j0} A_{ji} \bmod q \quad i = 1, \cdots, k \\ f'_\mu &= \prod_{\nu=-4}^{k} {f_\nu}^{A_{\nu\mu}} \bmod p \quad \mu = 0, \cdots, k \qquad (2.7) \\ \tilde{f}'_0 &= \prod_{\nu=-4}^{k} f_\nu^{A'_\nu} \bmod p \qquad (2.8) \\ g'_0 &= \prod_{\nu=0}^{k} {g_\nu}^{A_{\nu 0}} \bmod p \end{aligned}$$

$$
\begin{aligned}
m'_0 &= \prod_{\nu=0}^{k} {m_\nu}^{A_{\nu 0}} \bmod p \\
w &= \sum_{j=1}^{k} {A_{j0}}^3 - A_{-20} - A'_{-3} \bmod q \\
\dot{w} &= \sum_{j=1}^{k} {A_{j0}}^2 - A_{-40} \bmod q.
\end{aligned}
$$

Then, P sends $g'_0, m'_0, \tilde{f}'_0, \{f'_\mu\}_{\mu=0,\cdots,k}, w, \dot{w}$ to V as a commitment.

Challenge: V uniformly and randomly chooses $\{c_i \in \mathbb{Z}/q\mathbb{Z}\}_{i=1,\cdots,k}$ and sends it to P.

Response: P sends V the following response:

$$
\begin{aligned}
r_\nu &= \sum_{\mu=0}^{k} A_{\nu\mu} c_\mu \bmod q \quad \nu = -4, \cdots, k \\
r'_\nu &= \sum_{i=1}^{k} A_{\nu i} {c_i}^2 + A'_\nu \quad \nu = -4, \cdots, k
\end{aligned}
$$

where $c_0 = 1 \bmod q$.

Verification:

V accepts the shuffle if the following equations hold for a uniformly and randomly chosen $\alpha \in_R \mathbb{Z}/q\mathbb{Z}$:

$$\prod_{\nu=-4}^{k} {f_\nu}^{r_\nu + \alpha r'_\nu} \equiv f'_0 \tilde{f}'^{\alpha}_0 \prod_{i=1}^{k} f'^{c_i + \alpha {c_i}^2}_i \pmod p \tag{2.9}$$

$$\prod_{\nu=0}^{k} {g_\nu}^{r_\nu} \equiv \prod_{\mu=0}^{k} {g'_\mu}^{c_\mu} \pmod p \tag{2.10}$$

$$\prod_{\nu=0}^{k} {m_\nu}^{r_\nu} \equiv \prod_{\mu=0}^{k} {m'_\mu}^{c_\mu} \pmod p \tag{2.11}$$

$$\sum_{j=1}^{k} (r_j^3 - c_j^3) \equiv r_{-2} + r'_{-3} + w \pmod q \tag{2.12}$$

$$\sum_{j=1}^{k} (r_j^2 - c_j^2) \equiv r_{-4} + \dot{w} \pmod q. \tag{2.13}$$

View: The view of this protocol is

$$p, q, g_0, m_0, \{(g_i, m_i)\}_{i=1,\ldots,k}, \{(g'_i, m'_i)\}_{i=1,\ldots,k},$$

$$\{f_\nu\}_{\nu=-4,\dots,k}, \{f'_\mu\}_{\mu=0,\dots,k}, \tilde{f}'_0, g'_0, m'_0,$$
$$w, \dot{w}, \{c_i\}_{i=1,\dots,k}, \{r_\nu\}_{\nu=-4,\dots,k}, \{r'_\nu\}_{\nu=-4,\dots,k}$$

2.3.5 Properties of the Proposed Argument for Shuffling

Theorem 3 *The protocol is complete.*

Proof. It is clear. □

Theorem 4 *The protocol is sound as long as the discrete logarithm problem is difficult to solve.*

Proof.

Lemma 1 *If V accepts the protocol with a probability p, which implies, Eq. (2.9) holds, then*

$$\prod_{\nu=-4}^{k} f_\nu{}^{r_\nu} = \prod_{\mu=0}^{k} f'_\mu{}^{c_\mu} \pmod p \tag{2.14}$$
$$\prod_{\nu=-4}^{k} f_\nu{}^{r'_\nu} = \tilde{f}'_0 \prod_{i=1}^{k} f'^{c_i{}^2}_i \pmod p \tag{2.15}$$

will hold with the probability at least $p - 1/2^{|q|}$.

Proof. From

$$\prod_{\nu=-4}^{k} f_\nu{}^{r_\nu+\alpha r'_\nu} = f'_0 \tilde{f}'_0{}^\alpha \prod_{\mu=-1}^{k} f'^{c_i+\alpha c_i{}^2}_i \pmod p$$
$$\Leftrightarrow \frac{\prod_{\nu=-4}^{k} f_\nu{}^{r_\nu}}{\prod_{\mu=0}^{k} f'_\mu{}^{c_\mu}} = \left(\frac{\tilde{f}'_0 \prod_{i=1}^{k} f'^{c_i{}^2}_i}{\prod_{\nu=-4}^{k} f_\nu{}^{r'_\nu}} \right)^\alpha \pmod p,$$

it is clear that the probability that V will choose α such that Eq. (2.9) holds is negligible if Eqs. (2.14) and (2.15) do not hold. □

Lemma 2 *If V accepts the protocol with a probability p, then there is a polynomially bounded knowledge extractor K that can, within an expected number of steps bounded by $\mathcal{O}(n/p)$, extract from P an $\{A_{\nu\mu}\}_{\nu=-4,\dots,k;\mu=0,\dots,k}$ which satisfies Eq. (2.7) and an $\{A'_\nu\}_{\nu=-4,\dots,k}$ which satisfies Eq. (2.8).*

Proof. Let us call every challenge $\{(c_\mu)_{\mu=0,\cdots,k}\}$ to which P can compute responses $\{(r_\nu)_{\nu=-4,\dots,k}\}$ such that Eq. (2.9) holds for overwhelming probability as *acceptable challenge*. From Lemma 1, Eq. (2.14) also holds for these challenges. Suppose that K has chosen n linearly independent acceptable challenges.

Then, K extracts, from P, n responses that corresponds to these challenges, from which K is able to extract an $\{A_{\nu\mu}\}_{\nu=-4,\dots,k;\mu=0,\dots,k}$ that satisfies the relation:

$$r_\nu \equiv \sum_{\mu=0}^{k} A_{\nu\mu} c_\mu \pmod q \quad \nu = -3, \cdots, k$$

for every $n+1$ challenge and response. Such an $\{A_{\nu\mu}\}_{\nu=-4,\dots,k;\mu=0,\dots,k}$ satisfies Eq. (2.7).

Now, we estimate the expected number of steps K is required to obtain these acceptable challenges. Suppose that K has obtained ℓ linearly independent acceptable challenges. If K tries to find other accepting challenges by randomly choosing challenges, the expected number of times K chooses challenges until it succeeds in finding an acceptable challenge is $1/p$. The probability that the obtained acceptable challenge is linearly independent to already obtained ℓ acceptable challenges is at least $1 - (|q|^\ell/p|q|^n)$, which is overwhelming. Therefore, with the expected number of steps K is required to obtains n linearly independent acceptable challenges is $\mathcal{O}(n/p)$.

With respect to $\{A'_\nu\}_{\nu=-4,\dots,k}$, the proof is similar. □

Lemma 3 *Assume that there is a knowledge extractor K that can extract an $\{A_{\nu\mu}\}_{\nu=-4,\dots,k;\mu=0,\dots,k}$ and an $\{A'_\nu\}_{\nu=-4,\dots,k}$ from P that satisfies Eq. (2.7) and (2.8). If there is also a knowledge extractor K' that can extract an $\{r_\nu, r'_\nu\}_{\nu=-4,\dots,k}$ from P that satisfies Eq. (2.14) and (2.15) such that either*

$$\begin{aligned} r_\nu &\not\equiv \sum_{\mu=0}^{k} A_{\nu\mu} c_\mu \pmod q \\ \textit{or} \qquad r'_\nu &\not\equiv \sum_{i=1}^{k} A_{\nu i} {c_i}^2 + A'_\nu \pmod q \end{aligned}$$

for some $\nu = -4, \dots, k$, then K' can generate non-trivial integers $\{a_\nu\}_{\nu=-4,\dots,k}$ satisfying $\prod_{\nu=-4}^{k} {f_\nu}^{a_\nu} \equiv 1 \pmod p$ with overwhelming probability.

Proof. The following $\{f_\nu\}$-based expression will be a non-trivial representation of 1:

$$\prod_{\nu=-4}^{k} {f_\nu}^{\sum_{\mu=0}^{k} A_{\nu\mu} c_\mu - r_\nu} \equiv 1 \pmod p$$

$$\prod_{\nu=-4}^{k} {f_\nu}^{\sum_{i=1}^{k} A_{\nu i} {c_i}^2 + A'_\nu - r_\nu} \equiv 1 \pmod p$$

□

Lemma 4 *Assume that $\{A_{\nu\mu}\}_{\nu=-4,\dots,k;\mu=0,\dots,k}$, $\{A_\nu\}_{\nu=-4,\dots,k}$ and w are givens. Also assume that*

$\{r_i\}_{i=1,\ldots,k}, r_{-2}$ and r'_{-3} will be generated, for a given $\{c_i\}_{i=1,\cdots,k}$, as

$$\begin{aligned} r_\nu &= \sum_{\mu=0}^{k} A_{\nu\mu} c_\mu \bmod q \quad \nu = -3, 1, 2, \ldots, k \\ r'_{-3} &= \sum_{i=1}^{k} A_{-3i} {c_i}^2 + A'_{-3} \bmod q \end{aligned}$$

If the given $\{A_{ij}\}_{i=1,\ldots,k;j=1,\ldots,k}$ does not satisfy Eq. (2.1), then Eq. (2.12) will not hold with overwhelming probability for a uniformly and randomly chosen $\{c_i\}_{i=1,\cdots,k}$.

Proof. Eq. (2.12) is equivalent to

$$\begin{aligned} &\sum_{i=1}^{k}(\sum_{h=1}^{k} A_{hi}A_{hi}A_{hi} - \delta_{iii}){c_i}^3 \\ &+3\sum_{i\neq j}(\sum_{h=1}^{k} A_{hi}A_{hi}A_{hj} - \delta_{iij}){c_i}^2 c_j \\ &+6\sum_{i>j>k}(\sum_{h=1}^{k} A_{hi}A_{hj}A_{hk} - \delta_{ijk})c_i c_j c_k \\ &\equiv \sum_{i=1}^{k}\sum_{j=1}^{k}\left\{(\sum_{h=1}^{k} 3A_{h0}A_{hi}A_{hj} - A'_{-3i}\delta_{ij})\right\} c_i c_j \\ &+\sum_{i=1}^{k}\left\{(\sum_{h=1}^{k} 3A_{h0}^2 A_{hi} - A_{-2i})\right\} c_i \\ &+\left\{(\sum_{h=1}^{k} A_{h0}^3 - A_{-20} - A'_{-30}) - w\right\} \pmod q. \end{aligned}$$

The first three terms of the expression above and the fact that $2, 3 \not| q$ indicates that if Eq. (2.1) does not hold for some set $\{i, j, k\}$, then the probability will be negligible that Eq. (2.12) will hold for a uniformly and randomly chosen $\{c_i\}_{i=1,\cdots,k}$. □

Lemma 5 *Assume that $\{A_{\nu\mu}\}_{\nu=-4,\ldots,k;\mu=0,\ldots,k}$ and $\dot{w}$ are givens. Also assume that $\{r_i\}_{i=1,\ldots,k}$ and r_{-4} will be generated for a given $\{c_i\}_{i=1,\cdots,k}$ as*

$$r_\nu = \sum_{\mu=0}^{k} A_{\nu\mu} c_\mu \bmod q \quad \nu = -4, \ldots, k.$$

If the given $\{A_{\nu\mu}\}_{i=1,\ldots,k;j=1,\ldots,k}$ does not satisfy Eq. (2.2), then Eq. (2.13) will not hold with overwhelming probability for a uniformly and randomly chosen $\{c_i\}_{i=1,\cdots,k}$.

Proof. Because Eq. (2.13) is equivalent to

$$\begin{aligned}
&\sum_{i=1}^{k}(\sum_{h=1}^{k} A_{hi}A_{hi} - \delta_{ii}){c_i}^2 \\
&2\sum_{i>j}(\sum_{h=1}^{k} A_{hi}A_{hj} - \delta_{ij})c_ic_j \\
&+\sum_{i=1}^{k}(\sum_{h=1}^{k} 2A_{h0}A_{hi} - A_{-4i})c_i \\
&+\sum_{h=1}^{k} A_{h0}^2 - A_{-30} - \dot{w} \equiv 0 \pmod{q}.
\end{aligned}$$

□

Lemma 6 *Assume that* $\{g_\nu, m_\nu\}_{(\nu=0,\ldots,k)}$ *and* $\{A_{\nu\mu}\}_{(\nu=0,\ldots,k;\mu=0,\ldots,k)}$ *are givens. Also assume that* $\{r_\nu\}_{\nu=0,\ldots,k}$ *will be generated for a given* $\{c_i\}_{i=1,\cdots,k}$ *as*

$$r_\nu = \sum_{\mu=0}^{k} A_{\nu\mu}c_\mu \bmod q \quad \nu = 0,\ldots,k.$$

If the equations

$$g'_\mu \equiv \prod_{\nu=0}^{k} {g_\nu}^{A_{\nu\mu}} \pmod{p} \quad \mu = 0,\cdots,k \tag{2.16}$$

$$m'_\mu \equiv \prod_{\nu=0}^{k} {m_\nu}^{A_{\nu\mu}} \pmod{p} \quad \mu = 0,\cdots,k \tag{2.17}$$

do not hold, then Eqs. (2.10) and (2.11) will not hold with overwhelming probability for a uniformly and randomly chosen $\{c_i\}_{i=1,\cdots,k}$.

Proof. Eq. (2.10) is equivalent to

$$\prod_{\mu=0}^{k}\left(\frac{\prod_{\nu=0}^{k} {g_\nu}^{A_{\nu\mu}}}{g'_\mu}\right)^{c_\mu} \equiv 1 \pmod{p}.$$

If Eq. (2.16) does not hold, then Eq. (2.10) will not hold with overwhelming probability for a uniformly and randomly chosen $\{c_i\}_{i=1,\cdots,k}$.

By the same logic, if Eq. (2.17) does not hold, then (2.11) will not hold with overwhelming probability for a uniformly and randomly chosen $\{c_i\}_{i=1,\cdots,k}$. □

From Lemmata 1, 2, and 3, there is a knowledge extractor K that can extract an $\{A_{\nu\mu}\}_{\nu=-4,\ldots,k;\mu=0,\ldots,k}$

and a $\{A'_\nu\}_{\nu=-4,\ldots,k}$ from P that respectively satisfy Eqs. (2.7) and (2.8) within an expected number of steps bounded by $\mathcal{O}(n/p)$. Further, K can either, from values extracted from P, (1) only generate a response that satisfies equations

$$\begin{aligned} r_\nu &\equiv \sum_{\mu=0}^{k} A_{\nu\mu} c_\mu \pmod q \qquad \nu = -4, \ldots, k \\ r'_\nu &\equiv \sum_{i=1}^{k} A_{\nu i} c_i + A'_\nu \pmod q \qquad \nu = -4, \ldots, k \end{aligned}$$

for a given challenge, or (2) generate non-trivial integers $\{a_\nu\}_{\nu=-4,\ldots,k}$ satisfying $\prod_{\nu=-4}^{k} f_\nu{}^{a_\nu} \equiv 1 \pmod p$ with overwhelming probability, with overwhelming probability. Note that (2) will happen with negligible probability as long as solving the discrete logarithm problem is difficult.

From Lemmata 4 and 5, the $\{A_{\nu\mu}\}_{(\nu=1,\ldots,k;\mu=1,\ldots,k)}$ that can be extracted by K from P satisfies Eqs. (2.1) and (2.2). From Theorem 1, sub-matrix $(A_{ij})_{i,j=1,\cdots,k}$ is a permutation matrix. From Lemma 6, the equations

$$\begin{aligned} g'_\mu &\equiv \prod_{\nu=0}^{k} g_\nu{}^{A_{\nu\mu}} \pmod p \quad \mu = 0, \cdots, k \\ m'_\mu &\equiv \prod_{\nu=0}^{k} m_\nu{}^{A_{\nu\mu}} \pmod p \quad \mu = 0, \cdots, k \end{aligned}$$

hold for the $\{A_{\nu\mu}\}_{(\nu=0,\ldots,k;\mu=0,\ldots,k)}$ above.

Therefore, as long as solving the discrete logarithm problem is difficult, there is a knowledge extractor K that can extract an $\{A_{\nu i}\}_{(\nu=0,\ldots,k;i=1,\ldots,k)}$ from P satisfying Eq. (2.6) within an expected number of steps bounded by $\mathcal{O}(n/p)$, and its sub-matrix $(A_{ij})_{i,j=1,\cdots,k}$ will be a permutation matrix if V accepts the protocol. □

Theorem 5 *The protocol is honest-verifier perfect zero-knowledge.*

Proof. The following simulator proves the theorem. Given p, q, g_0, m_0, $\{g_i, m_i\}_{i=1,\cdots,k}$, $\{g'_i, m'_i\}_{i=1,\cdots,k}$, $\{f_\nu\}_{(\nu=-4,\ldots,k)}$, the simulator uniformly and randomly chooses r_ν, r'_ν, c_i, f'_i for $\mu = 0, \ldots, k; \nu = -4, \ldots, k$ and $i = 1, \ldots, k$. It then generates $\tilde{f}'_0, f'_0, g'_0, m'_0, w$ and $\dot{w}$ so as to satisfy Eqs. (2.14), (2.15), (2.10), (2.11), (2.12), and (2.13). □

2.4 An Efficient Argument for Shuffle-Decryption

In this section, we propose an argument for shuffle-decryption that is the most efficient of all previous protocols, as a whole, to prove the correctness of both shuffling and decryption of ElGamal cipher-texts.

The protocol requires five rounds. Although the protocol is not zero-knowledge, we can prove that the verifier learn nothing new about its permutation.

2.4.1 Notation

Let p, q be two primes such that $q|p-1$ and $q \bmod 3 = 2$. Note that we have imposed an additional condition to q. Let $\mathbb{G}_q$ be an order q subgroup of $(\mathbb{Z}/p\mathbb{Z})^*$, $g_0(\neq 1)$ be an element of $\mathbb{G}_q$, k be the number of ElGamal cipher-texts to be shuffled, and ℓ be the number of shuffler-decrypters. Let $x'^{(\lambda)} \in_R \mathbb{Z}/q\mathbb{Z}$ be a private key of the λ-th shuffle-decrypter which is used for the partial decryption and $y^{(\lambda)} = {g_0}^{x'^{(\lambda)}} \bmod p$ be the corresponding public key. Let ${m_0}^{(\lambda)} = \prod_{\kappa=1}^{\lambda} y^{(\kappa)} \bmod p$ be a public key used for the shuffling of the λ-th shuffle-decrypter.

Let $\{(g_i^{(\ell)}, m_i^{(\ell)})\} = \{({g_0}^{\bar{r}_i}, (m_0^{(\ell)})^{\bar{r}_i} M_i)\}_{i=1,\ldots,k} \bmod p$ be ElGamal cipher-texts to be input to the ℓ-th shuffle-decrypter where $\{M_i \in \mathbb{G}_q\}_{i=1,\ldots,k}$ is a set of encrypted plain texts, and $\{\bar{r}_i \in_R \mathbb{Z}/q\mathbb{Z}\}_{i=1,\ldots,k}$ be elements of $\mathbb{Z}/q\mathbb{Z}$, which are uniformly and randomly chosen. Let $\{(g_i^{(\lambda)}, m_i^{(\lambda)})\}_{i=1,\ldots,k}$ be cipher-texts to be input to the λ-th shuffle-decrypter, who shuffles them with the public key $(g_0, m_0^{(\lambda)})$ and then partially decrypts them with the private key $x'^{(\lambda)}$. The resulting cipher-texts are $\{(g_i'^{(\lambda)}, m_i'^{(\lambda)})\}_{i=1,\ldots,k} = \{(g_i^{(\lambda-1)}, m_i^{(\lambda-1)})\}_{i=1,\ldots,k}$ which are passed to the $(\lambda-1)$-th shuffle-decrypter. In the rest of the chapter, we only consider the λ-th shuffle-decrypter and omit the index (λ).

We assume another public key $F_k \triangleq \{f_\nu \in_R \mathbb{G}_q\}_{\nu=-2,\ldots,k}$ which are $k+3$ elements of $\mathbb{G}_q$ that are uniformly and randomly generated so that neither P nor V can generate non-trivial integers a, $\{a_\nu\}_{\nu=-2,\ldots,k}$ satisfying $g_0^a \prod_{\nu=-2}^{k} {f_\nu}^{a_\nu} \equiv 1 \pmod{p}$ with non-negligible probability. The public key is a set $\{p, q, g_0, m_0, y, F_k\}$.

2.4.2 ElGamal Shuffle-Decryption

ElGamal shuffle-decryption is a combination procedure of ElGamal shuffling and partial decryption that, given g_0, m_0, y and k ElGamal cipher-texts $\{(g_i, m_i)\}_{i=1,\cdots,k}$, outputs ElGamal cipher-texts

$$(g_i', m_i') \quad = \quad ({g_0}^{s_i} g_{\phi^{-1}(i)}, {g_i'}^{-x'} {m_0}^{s_i} m_{\phi^{-1}(i)}) \bmod p$$

for $i = 1, \ldots, k$ where $\{s_i \in_R \mathbb{Z}/q\mathbb{Z}\}_{i=1,\cdots,k}$ and ϕ are chosen uniformly and randomly. Here, the multiplication by ${g_i'}^{-x'}$ in the second term has the effect of partial decryption[3].

[3]Note that this multiplier is not ${g_i}^{-x'}$ which is part of input cipher-texts. The first element of the left side of the equation is used in the second element of the right side of the equation to define the second element of the left side of the equation.

2.4.3 Protocol Structure and Tricks for Efficiency

In [69], a technique to simultaneously prove the correctness of shuffling and decryption is proposed. The verifiable shuffle-decryption protocol proposed in [69] is the result of application of this technique to the protocol proposed in [70]. The argument for shuffle-decryption we will propose in this section is the result of application of this simultaneously proving technique to the zero-knowledge argument for shuffling proposed in the previous section with little modification. We briefly sketch this technique.

We first consider the following 4-round zero-knowledge proof for decryptions of cipher-texts. The prover wants to prove that $\{(g'_i, m'_i)\}_{i=1,\ldots,k}$ is the valid partial decryptions of $\{(g'_i, \bar{m}_i)\}_{i=1,\ldots,k}$ with a private key x' such that $y = {g_0}^{x'} \bmod p$, that is, the relation $m'_i/\bar{m}_i = {g'_i}^{x} \bmod p$ holds for all $i = 1, \ldots, k$.

1. The verifier sends to the prover uniformly and randomly chosen $\{c_j \in_R \mathbb{Z}/q\mathbb{Z}\}_{j=1,\ldots,k}$.

2. The prover chooses $z' \in_R \mathbb{Z}/q\mathbb{Z}$ uniformly and randomly and then computes

$$\begin{aligned} \zeta &= \prod_{j=1}^{k} {g'_j}^{c_j} \bmod p \\ y' &= g^{z'} \ , \ \ \eta' = \zeta^{z'} \bmod p \end{aligned}$$

   and sends y', η' to the verifier.

3. The verifier sends to the prover uniformly and randomly chosen $c' \in_R \mathbb{Z}/q\mathbb{Z}$.

4. The prover sends to the verifier

$$r' = c'x' + z' \bmod q$$

5. The verifier computes

$$\zeta = \prod_{j=1}^{k} {g'_j}^{c_j} \ , \ \ \eta = \prod_{j=1}^{k} (m'_j/\bar{m}_j)^{c_j} \bmod p$$

   and accepts if

$$g^{r'} = y^{c'} y' \quad , \quad \zeta^{r'} = \eta^{c'} \eta' \bmod p$$

   hold.

We have noticed that some of the computations done in the proposed argument for shuffling in the previous section and in this proof for decryption are similar. For example, $\zeta = \prod_{j=1}^{k} {g'_j}^{c_j}$ mod

p with respect to the verifier's challenge $\{c_j\}$, is also computed in the argument for shuffling, $\eta = \prod_{j=1}^{n}(m'_j/\bar{m}_j)^{c_j} = \prod_{j=1}^{k} {m'_j}^{c_j} / \prod_{j=1}^{k} {\bar{m}_j}^{c_j} \bmod p$ in the proof for decryption and $\prod_{j=1}^{k} {\bar{m}_j}^{c_j} \bmod p$ in the argument for shuffling have the same factor.

We merged the two protocols in the following way: Two proofs share the same challenge $\{c_i\}_{i=1,\ldots,k}$ so that both prover and verifier need to compute the above mentioned values only once. Since η, which is a combination of $\{g'_j\}_{i=1,\ldots,k}$ and the challenge $\{c_i\}_{i-1,\ldots,k}$, does not appear in the protocol proposed in [85], we cannot apply this technique directly to that protocol.

Although, as is mentioned in [69], the protocol resulting from this simultaneously proving technique cannot be zero-knowledge, we can prove its security. Moreover, since we do not require the protocol to be zero-knowledge any more, we can modify the protocol, so long as the protocol remains secure at the same security level, so as to be more efficient. In particular, the protocol requires less random numbers to remain at the same security level, thus we reduce the set of extra public parameters $\{f_{-4}, f_{-3}, f_{-2}, f_{-1}\}$ to $\{f_{-3}, f_{-2}, f_{-1}\}$.

We have made one more modification to the original protocol. Since the proposed protocol adopts the prime q such that $q \bmod 3 = 2$, verifiers do not need to confirm that Equation (2.2) holds [4] any more. Hence we do not need r_{-4} and reduce the set of extra public parameters $\{f_{-3}, f_{-2}, f_{-1}\}$ to $\{f_{-2}, f_{-1}\}$.

2.4.4 Proposed Argument

We now describe our proposed argument for ElGamal shuffle-decryption.

ElGamal Shuffle-Decryption

P uniformly and randomly chooses $\{A_{0i} \in_R \mathbb{Z}/q\mathbb{Z}\}_{i=1,\cdots,k}$ and a permutation matrix $(A_{ji})_{i,j=1,\cdots,k}$ and then shuffles and decrypts k ElGamal cipher-texts $\{(g_i, m_i)\}_{i=1,\cdots,k}$ to $\{(g'_i, m'_i)\}_{i=1,\cdots,k}$ as

$$\begin{aligned}(g'_i, m'_i) &= ({g_0}^{A_{0i}} g_{\phi^{-1}(i)}, {g'_i}^{-x'} {m_0}^{A_{0i}} m_{\phi^{-1}(i)}) \bmod p \\ &= (\prod_{\nu=0}^{k} {g_\nu}^{A_{\nu i}}, {g'_i}^{-x'} \prod_{\nu=0}^{k} {m_\nu}^{A_{\nu i}}) \bmod p. \end{aligned} \tag{2.18}$$

In this protocol, the witness W_n is a set $\{x', (A_{ji})_{i,j=1,\ldots,k}, \{A_{0i}\}_{i=1,\ldots,k}\}$, and the common input X_n is a set $\{p, q, y, g_0, m_0, F_k, (g_i, m_i)_{i=1,\ldots,k}, (g'_i, m'_i)_{i=1,\ldots,k}\}$. P is given X_n and W_n, and V is given X_n.

[4]We can omit a verification of the equation that corresponds to Equation (2.13).

Proving a Shuffle-Decryption

Commitment-1: P uniformly and randomly chooses $\{A_{\nu 0}, A'_\nu \in_R \mathbb{Z}/q\mathbb{Z}\}_{\nu=-2,\dots,k}$ and then computes:

$$\begin{aligned}
A_{-1i} &= \sum_{j=1}^{k} 3A_{j0}A_{ji} \bmod q \quad i = 1, \cdots, k \\
A_{-2i} &= \sum_{j=1}^{k} 3{A_{j0}}^2 A_{ji} \bmod q \quad i = 1, \cdots, k \\
f'_\mu &= \prod_{\nu=-2}^{k} {f_\nu}^{A_{\nu\mu}} \bmod p \quad \mu = 0, \cdots, k \\
\tilde{f}'_0 &= \prod_{\nu=-2}^{k} {f_\nu}^{A'_\nu} \bmod p \\
g'_0 &= \prod_{\nu=0}^{k} {g_\nu}^{A_{\nu 0}} \bmod p \\
m'_0 &= \prod_{\nu=0}^{k} {m_\nu}^{A_{\nu 0}} \bmod p \\
w &= \sum_{j=1}^{k} {A_{j0}}^3 - A_{-20} - A'_{-1} \bmod q
\end{aligned}$$

Then, P sends $g'_0, m'_0, w, \tilde{f}'_0, \{f'_\mu\}_{\mu=0,\cdots,k}$to V as a commitment.

Challenge-1: V uniformly and randomly chooses $\{c_i\}_{i=1,\cdots,k}$ from $\mathbb{Z}/q\mathbb{Z}$ and sends it to P.

Response-1: P sends V the following response:

$$\begin{aligned}
r_\nu &= \sum_{\mu=0}^{k} A_{\nu\mu}c_\mu \bmod q \quad \nu = -2, \cdots, k \\
r'_\nu &= \sum_{i=1}^{k} A_{\nu i}{c_i}^2 + A'_\nu \bmod q \quad \nu = -2, \cdots, k
\end{aligned}$$

where $c_0 = 1 \bmod p$.

Commitment-2: P then computes

$$\zeta = \prod_{i=1}^{k} g_i'^{c_i} \bmod p.$$

P uniformly and randomly chooses $\beta \in_R \mathbb{Z}/q\mathbb{Z}$, computes the following commitment, and sends it to V:

$$\eta = \zeta^{x'} \bmod p \ , \quad \eta' = \zeta^\beta \bmod p$$

$$y' = {g_0}^{\beta} \bmod p. \tag{2.19}$$

Challenge-2: V uniformly and randomly chooses c' from $\mathbb{Z}/q\mathbb{Z}$ and sends it to P.

Response-2: P sends V the following response: $r' = c'x' + \beta \bmod q$

Verification: V computes

$$\zeta = \prod_{i=1}^{k} g_i'^{c_i} \bmod p.$$

V accepts the shuffle-decryption if the following equations hold for a uniformly and randomly generated $\alpha \in_R \mathbb{Z}/q\mathbb{Z}$:

$$\prod_{\nu=-2}^{k} {f_\nu}^{r_\nu + \alpha r'_\nu} \equiv f'_0 \tilde{f}_0'^{\alpha} \prod_{i=1}^{k} f_i'^{c_i + \alpha {c_i}^2} \pmod p \tag{2.20}$$

$$\prod_{\nu=0}^{k} {g_\nu}^{r_\nu} \equiv \zeta g'_0 \pmod p \tag{2.21}$$

$$\prod_{\nu=0}^{k} {m_\nu}^{r_\nu} \equiv \eta \prod_{\mu=0}^{k} {m'_\mu}^{c_\mu} \pmod p \tag{2.22}$$

$$\sum_{j=1}^{k} (r_j^3 - c_j^3) \equiv r_{-2} + r'_{-1} + w \pmod q \tag{2.23}$$

$${g_0}^{r'} \equiv y^{c'} y' \pmod p \tag{2.24}$$

$$\zeta^{r'} \equiv \eta^{c'} \eta' \pmod p \tag{2.25}$$

The view $View_V^P(X_n, W_n)$ of this protocol is

$$\begin{aligned}
&p, q, y, g_0, m_0, \{(g_i, m_i)\}_{i=1,\dots,k}, \{(g'_i, m'_i)\}_{i=1,\dots,k},\\
&\{f_\nu\}_{\nu=-2,\dots,k}, f'_0, \{f'_i\}_{i=1,\dots,k},\\
&\tilde{f}'_0, g'_0, m'_0, w, \{c_i\}_{i=1,\dots,k},\\
&\{r_\nu\}_{\nu=-2,\dots,k}, \{r'_\nu\}_{\nu=-2,\dots,k}, \eta, \eta', y', c', r'.
\end{aligned}$$

2.4.5 Properties of the Proposed Argument for Shuffle-Decryption

Theorem 6 *The protocol is complete.*

Proof. It is clear. □

Theorem 7 *The protocol is sound as long as the discrete logarithm problem is difficult to solve.*

Proof.

Lemma 7 *If V accepts the protocol with a probability p, i.e., Eq. (2.20) holds, then*

$$\prod_{\nu=-2}^{k} f_\nu{}^{r_\nu} = \prod_{\mu=0}^{k} f'_\mu{}^{c_\mu} \pmod p \tag{2.26}$$

$$\prod_{\nu=-2}^{k} f_\nu{}^{r'_\nu} = \tilde{f}'_0 \prod_{i=1}^{k} f'_i{}^{c_i{}^2} \pmod p \tag{2.27}$$

will hold with the probability p.

Proof. From

$$\prod_{\nu=-2}^{k} f_\nu{}^{r_\nu+\alpha r'_\nu} = f'_0 \tilde{f}'_0{}^{\alpha} \prod_{\mu=-1}^{k} f'_i{}^{c_i+\alpha c_i{}^2} \pmod p$$
$$\Leftrightarrow \frac{\prod_{\nu=-2}^{k} f_\nu{}^{r_\nu}}{\prod_{\mu=0}^{k} f'_\mu{}^{c_\mu}} = \left(\frac{\tilde{f}'_0 \prod_{i=1}^{k} f'_i{}^{c_i{}^2}}{\prod_{\nu=-2}^{k} f_\nu{}^{r'_\nu}}\right)^{\alpha} \pmod p,$$

it is clear that the probability that V will choose α such that Eq. (2.20) holds is negligible if Eqs. (2.26) and (2.27) do not hold. □

Lemma 8 *If V accepts the protocol with a probability p, then there is a knowledge extractor K that can, within an expected number of steps bounded by $\mathcal{O}(n/p)$, extract an $\{A_{\nu\mu}\}_{\nu=-2,\dots,k;\mu=0,\dots,k}$ and $\{A'_\nu\}_{\nu=-2,\dots,k}$ from P satisfying*

$$f'_\mu = \prod_{\nu=-2}^{k} f_\nu{}^{A_{\nu\mu}} \pmod p \quad \mu = 0, \cdots, k \tag{2.28}$$

$$\tilde{f}'_0 = \prod_{\nu=-2}^{k} f_\nu{}^{A'_\nu} \pmod p \tag{2.29}$$

Proof. Eqs. (2.26) and (2.27) hold with a probability p. Hence, the lemma can be proved in the same way as Lemma 2. □

Lemma 9 *Assume there is a knowledge extractor k that can extract $\{A_{\nu\mu}\}_{\nu=-2,\dots,k;\mu=0,\dots,k}$ and $\{A'_\nu\}_{\nu=-2,\dots,k}$ from P satisfying Eqs. (2.28) and (2.29). If there is also a knowledge extractor K' that can extract an $\{r_\nu, r'_\nu\}_{\nu=-2,\cdots,k}$ that satisfies Eqs. (2.26) and (2.27) such that either*

$$r_\nu \not\equiv \sum_{\mu=0}^{k} A_{\nu\mu} c_\mu \pmod q$$

$$\textit{or} \quad r'_\nu \not\equiv \sum_{i=1}^{k} A_{\nu i} {c_i}^2 + A'_\nu \pmod q$$

for some $\nu = -2, \ldots, k$*, then* K' *can generate non-trivial integers* $\{a_\nu\}_{\nu=-2,\cdots,k}$ *satisfying* $\prod_{\nu=-2}^{k} {f_\nu}^{a_\nu} = 1 \pmod p$ *with overwhelming probability.*

Proof. The following $\{f_\nu\}$-based expression will be a non-trivial representation of 1:

$$\begin{aligned} \prod_{\nu=-2}^{k} {f_\nu}^{\sum_{\mu=0}^{k} A_{\nu\mu} c_\mu - r_\nu} &= 1 \pmod p \\ \prod_{\nu=-2}^{k} {f_\nu}^{\sum_{i=1}^{k} A_{\nu i} {c_i}^2 + A'_\nu - r'_\nu} &= 1 \pmod p \end{aligned}$$

□

Lemma 10 *Assume* $\{A_{\nu\mu}\}_{\nu=-2,\ldots,k;\mu=0,\ldots,k}$*,* $\{A'_\nu\}_{\nu=-2,\cdots,k}$ *and* w *are givens. Also assume* $\{r_\nu, r'_\nu\}_{\nu=-2,\cdots,k}$ *will be generated for a given* $\{c_i\}_{i=1,\cdots,k}$ *as*

$$\begin{aligned} r_\nu &= \sum_{\mu=0}^{k} A_{\nu\mu} c_\mu \bmod q \quad \nu = -2, \cdots, k \\ r'_\nu &= \sum_{i=1}^{k} A_{\nu i} {c_i}^2 + A'_\nu \bmod q \quad \nu = -2, \cdots, k. \end{aligned}$$

If the given $\{A_{\nu\mu}\}_{\nu=-3,\ldots,k;\mu=0,\cdots,k}$ *does not satisfy Eq. (2.1), then Eq. (2.23) will not hold with overwhelming probability for a uniformly and randomly chosen* $\{c_i\}_{i=1,\cdots,k}$*.*

Proof. It can be shown in a way similar to that used for Lemma 4. □

Lemma 11 *Assume* $\{g_\nu\}_{\nu=0,\ldots,k}$ *and* $\{A_{\nu\mu}\}_{\nu=0,\ldots,k;\mu=0,\ldots,k}$ *are givens. Also assume* $\{r_\nu\}_{\nu=0,\ldots,k}$ *will be generated for a given* $\{c_i\}_{i=1,\cdots,k}$ *as*

$$r_\nu = \sum_{\mu=0}^{k} A_{\nu\mu} c_\mu \bmod q \quad \nu = 0, \ldots, k$$

If equation

$$g'_\mu = \prod_{\nu=0}^{k} {g_\nu}^{A_{\nu\mu}} \pmod p \quad \mu = 0, \cdots, k \tag{2.30}$$

does not hold, then Eq. (2.21) will not hold with overwhelming probability for a uniformly and randomly chosen $\{c_i\}_{i=1,\cdots,k}$*.*

Proof. Eq. (2.21) is equivalent to

$$\prod_{\mu=0}^{k}\left(\frac{\prod_{\nu=0}^{k} {g_\nu}^{A_{\nu\mu}}}{g'_\mu}\right)^{c_\mu} = 1 \pmod p.$$

If Eq. (2.30) does not hold, then Eq. (2.21) will not hold with overwhelming probability for a uniformly and randomly chosen $\{c_i\}_{i=1,\cdots,k}$. □

Lemma 12 *The following three items hold:*

1. *If Eq. (2.24) holds with non-negligible probability, then there is a knowledge extractor K that can extract x' and β satisfying equation*

$$y = g_0^{x'} \bmod p, \tag{2.31}$$

and Eq. (2.19).

2. *Assume that β and x' satisfy Eqs. (2.31) and (2.19). If Eq. (2.24) holds, then*

$$r' = c'x' + \beta \bmod q \tag{2.32}$$

will hold.

3. *Assume β, x' and ζ are givens. Also assume r' will be generated for a given c' as*

$$r' = c'x' + \beta \bmod q$$

If equations

$$\begin{aligned} \eta &= \zeta^{x'} \quad \left(= (\prod_{i=1}^{k} {g'_i}^{c_i})^{x'}\right) \pmod p \\ \eta' &= \zeta^{\beta} \pmod p \end{aligned}$$

do not hold, then Eq. (2.25) will not hold with overwhelming probability for a uniformly and randomly chosen c'.

Proof. The proof is straight-forward and, hence, has been omitted here. □

Lemma 13 *Assume $\{g_\nu, m_\nu\}_{\nu=0,\ldots,k}$, $(A_{\nu\mu})_{\nu=0,\ldots,k;\mu=0,\ldots,k}$, $\{g'_j\}_{j=1,\cdots,k}$ and x' are givens. Also assume* *that $\{r_\nu\}_{\nu=0,\dots,k}$ and η will be generated, for a given $\{c_i\}_{i=1,\cdots,k}$, as*

$$\begin{aligned} r_\nu &= \sum_{\mu=0}^{k} A_{\nu\mu} c_\mu \bmod q \quad \nu = 0,\dots,k \\ \eta &= (\prod_{j=1}^{k} {g'_j}^{c_j})^{x'} \bmod p. \end{aligned} \tag{2.33}$$

If equation

$$m'_i = {g'_i}^{-x'} \prod_{\nu=0}^{k} {m_\nu}^{A_{\nu i}} \pmod p \quad \mu = 0,\cdots,k \tag{2.34}$$

does not hold, then Eq. (2.22) will not hold with overwhelming probability for a uniformly and randomly chosen $\{c_i\}_{i=1,\cdots,k}$.

Proof. After substituting Eq. (2.33), Eq. (2.22) will be equivalent to

$$\prod_{\nu=0}^{k} {m_\nu}^{r_\nu} = (\prod_{i=1}^{k} {g'_i}^{c_i})^{x'} \prod_{\mu=0}^{k} {m'_\mu}^{c_\mu} \bmod p$$
$$\prod_{\nu=0}^{k} {m_\nu}^{r_\nu} = m'_0 \prod_{i=1}^{k} ({g'_i}^{x'} m'_i)^{c_i} \bmod p.$$

Thus, by the same logic as Lemma 11, if equation

$${g'_i}^{x'} m'_i = \prod_{\nu=0}^{k} {m_\nu}^{A_{\nu i}} \bmod p \ ,$$

which is equivalent to Eq. (2.34), does not hold, then Eq. (2.22) will not hold with overwhelming probability for a uniformly and randomly chosen $\{c_i\}_{i=1,\cdots,k}$. □

From Lemmata 7, 8, and 9, there is a knowledge extractor K that can extract an $\{A_{\nu\mu}\}_{\nu=-2,\dots,k;\mu=0,\dots,k}$ and an $\{A'_\nu\}_{\nu=-2,\dots,k}$ from P that will respectively satisfy Eqs. (2.26) and (2.27) within an expected number of steps bounded by $\mathcal{O}(n/p)$. Further, K can either, from values extracted from P, (1) generate only a response which satisfies equations

$$r_\nu = \sum_{\mu=0}^{k} A_{\nu\mu} c_\mu \bmod q \qquad \nu = -2,\dots,k$$

$$r'_\nu = \sum_{i=1}^{k} A_{\nu i} {c_i}^2 + A'_\nu \bmod q \qquad \nu = -2, \ldots, k$$

for a given challenge or (2) generate non-trivial integers $\{a_\nu\}_{\nu=-2,\ldots,k}$ satisfying $\prod_{\nu=-2}^{k} {f_\nu}^{a_\nu} = 1 \pmod p$ with overwhelming probability. Note that (2) will happen with negligible probability as long as solving the discrete logarithm problem is difficult.

From Lemma 10, the sub-matrix $(A_{ij})_{i,j=1,\cdots,k}$ that can be extracted by K from P satisfies Eq. (2.1). From the corollary of Theorem 1 and the fact $q \bmod 3 = 2$, sub-matrix $(A_{ij})_{i,j=1,\cdots,k}$ is a permutation matrix. From Lemma 11, the equation

$$g'_i = \prod_{\nu=0}^{k} {g_\nu}^{A_{\nu i}} \bmod p \quad i = 1, \cdots, k$$

holds for the above $\{A_{\nu i}\}_{\nu=0,\ldots,k;i=1,\ldots,k}$.

From Lemmas 12 and 13, the equation

$$m'_i = {g'_i}^{-x} \prod_{\nu=0}^{k} {m_\nu}^{A_{\nu i}} \bmod p \quad i = 1, \cdots, k$$

also holds for the above $\{A_{\nu i}\}_{\nu=0,\ldots,k;i=1,\ldots,k}$.

Therefore, as long as solving the discrete logarithm problem is difficult, if an honest verifier V accepts the protocol with non-negligible probability, there exists a polynomially bounded knowledge extractor K that can extract an $\{A_{\nu i}\}_{\nu=0,\ldots,k;i=1,\ldots,k}$ and x' from P that satisfies Eqs. (2.18) and $y = {g_0}^{x'} \pmod p$ within an expected number of steps bounded by $\mathcal{O}(n/p)$, and its submatrix $(A_{ij})_{i,j=1,\cdots,k}$ will be a permutation matrix.

Therefore, the protocol is sound as long as solving the discrete logarithm problem is difficult. $\square$

2.5 Complete Permutation Hiding

The proposed argeument for shuffle-decryption has a property that, if the decision Diffie-Hellman problem is difficult to solve, honest verifiers learn nothing new about its permutation from an interaction with a prover in an overwhelming number of cases of input, even if the protocols with the same private key are repeatedly executed. We formalize such a security notion and prove that the protocol satisfies this notion in this section.

2.5.1 Intuitive Security

The fact that the proposed argument is not less secure than the simple combination of shuffling and decryption is intuitively understood as in the following. Consider the proof system in which a common input to a prover is $p, q, y, g_0, m_0, f_0, \{(g_i, m_i, f_i)\}_{i=1,\ldots,k}, \{(g'_i, \bar{m}_i, \bar{f}'_i)\}_{i=1,\ldots,k}$ and $\{m'_i\}_{i=1,\ldots,k}$. Where

$$\begin{aligned}(g'_i, \bar{m}_i, \bar{f}'_i) &= ({g_0}^{r_i} g_{\pi(i)}, {m_0}^{r_i} m_{\pi(i)}, {f_0}^{r_i} f_{\pi(i)}) \bmod p \\ m'_i &= {g'_i}^{-x} \bar{m}_i \bmod p.\end{aligned}$$

Here, the first equation can be taken as a shuffle of generalized ElGamal cipher-texts composed of three elements instead of two elements. This shuffling as well as the ordinary shuffling, leaks no knowledge about its permutation as long as the decision Diffie-Hellman problem is difficult to solve. The second equation is a partial decryption of shuffled input cipher-texts. The only obstacle that prevents simulating the whole proposed argument for shuffle-decryption is the intermediate state $\{(\bar{m}_i, \bar{f}'_i)\}_{i=1,\ldots,k}$[5]. This is unsimulatable without the knowledge of $\{r_i\}$ and ϕ. Since the simple combined protocol also needs to reveal such an unsimulatable intermediate state, we can intuitively conclude that the proposed protocol is no less secure than the simple combination of two perfect zero-knowledge protocol. Since the above shuffled-cipher-texts-like intermediate states are part of the argument, we are required to consider the security of the argument in the same manner in which we consider the security of plain shuffle-decryption.

2.5.2 Adaptive Chosen Cipher-Text Attack

As discussed in the previous subsection, the proposed argument for ElGamal shuffle-decryption is as secure as the simple combination of shuffling and decryption. However, the security of ElGamal shuffle-decryption itself is not so trivial. This is because, in the mix-net protocol composed of a sequence of shuffle-decryptions, the last shuffle-decrypter outputs a permuted list of "plain-texts". This means mix-net behaves as a kind of decryption oracle. Thus, adaptive chosen cipher-text attack is effective for adversaries.

An example of such an attack can be seen as in the following. Consider the case where a user U_1 sends a plain ElGamal cipher-text to a mix-net composed of ElGamal shuffle-decryptions, and suppose that the mix-net accepts this cipher-text. Then, an adversary may obtain this cipher-text, re-encrypt it, and send the resulting cipher-text to the mix-net. At the end, mix-net outputs a list of plain-texts. If the adversary finds two same plain-texts, then the possibility that this cipher-text is the one that U_1 generated is meaningfully high.

[5]Since there is a subtle difference between $\bar{f}'_i$ and f'_i, rigorous logic is more complicated.

The above observation implies that the security of arguments for shuffle-decryption depends on the security of input cipher-texts. Hence, we assume that cipher-texts input to the mix-net are ElGamal cipher-texts that are secure under adaptive chosen cipher-text attack (IND-CCA2 secure) instead of plain ElGamal cipher-texts. And validity of cipher-text needs to be publicly checkable. An example of such a cryptosystem can be found in the literature [142]. This cryptosystem can be proved to be IND-CCA2 secure assuming the random oracle model. One can, in this cryptosystem, extract the random number used for generating a ElGamal cipher-text by rewinding the user at least with the number of the step quadratic to that of real protocol. We assume input to the mix-net is cipher-texts of such a cryptosystem.

2.5.3 Complete Permutation Hiding

We propose here the notion of *complete permutation hiding* (CPH) as a core requirement of unlinkability in arguments for ElGamal shuffle-decryption. If an argument for shuffle-decryption is CPH, polynomially-bounded honest verifiers, who may have some partial information about its permutation, will learn nothing new about its permutation from an interaction with a prover in an overwhelming number of cases of input even if the protocols with the same private key are repeatedly executed, whereas if the protocol is zero-knowledge, verifiers will learn nothing new in every case of input.

A verifier in CPH may obtain knowledge about permutation as long as it happens only in a negligible number of cases of common input X_n and witness W_n that the generator G_R (defined below) outputs. This is because we take the success probability of adversaries over the distribution of outputs of G_R.

Let I_n be a *set* of domain parameters $1^n, p, q$, where p and q are primes and are the lengths of the polynomial of n, private key $\bar{x}$, plain texts $\{M_i \in \mathbb{G}_q\}_{i=1,\ldots,k}$, and random tapes of U and G_R. Let *enc*(U) be an *encoding of a probabilistic polynomial time (PPT) Turing machine* U which generates cipher-texts $\{(g_i, m_i)\}_{i=1,\ldots,k}$ input to the shuffle-decryption procedure. We assume the existence of a knowledge extractor that can extract $\{\bar{r}_i\}_{i=1,\ldots,k}$ such that ${g_0}^{\bar{r}_i} = g_i$ from U.

Existence of such a knowledge extractor can be justified as in the following. Suppose that input cipher-texts $\{(g_i, m_i)\}_{i=1,\ldots,k}$ to shuffle-decrypter are $\{(g_i^{(\lambda)}, m_i^{(\lambda)})\}_{i=1,\ldots,k}$, and suppose that the input cipher-texts to the mix-net are $\{(g_i^{(\ell)}, m_i^{(\ell)})\}_{i=1,\ldots,k}$. Then $\{(g_i^{(\lambda)}, m_i^{(\lambda)})\}_{i=1,\ldots,k}$ are already shuffled and decrypted by $\ell - \lambda + 1$ shuffle-decrypters. If proofs for these shuffles and decryptions are accepted, shuffle-decrypters are able to collaborately generate $\{\bar{s}_i\}_{i=1,\ldots,k}$ and ϕ such that $g_i^{(\lambda)} = g_{\phi^{-1}(i)}^{(\ell)} {g_0}^{\bar{s}_i}$. And one can extract $\tilde{s}_i$ such that $g_i^{(\ell)} = {g_0}^{\tilde{s}_i}$ by rewinding the user who generated g_i. Hence, there exists an extractor that can extract $\bar{r}_i$ such that $g_i^{(\lambda)} = {g_0}^{\bar{r}_i}$.

Definition 2 *Given* $I_n(= \{1^n, p, q, \bar{x} \in \mathbb{Z}/q\mathbb{Z}, \{M_i \in \mathbb{G}_q\}_{i=1,\cdots,k}, Z_n\})$ *and enc*(U), instance generator G_R *chooses* $g_0 \in_R \mathbb{G}_q, x' \in_R \mathbb{Z}/q\mathbb{Z}$, $\{s_i \in_R \mathbb{Z}/q\mathbb{Z}\}_{i=1,\ldots,k}$, *and a permutation* ϕ *uniformly and randomly* *and computes:*

$$\begin{aligned} m_0 &= {g_0}^{x'+\bar{x}}, y = {g_0}^{x'} \bmod p \\ (g_i, m_i) &= U(I_n, g_0, y) \in \mathbb{G}_q \times \mathbb{G}_q \\ (g'_i, m'_i) &= ({g_0}^{s_i} g_{\phi^{-1}(i)}, {g_i}^{-x'} {m_0}^{s_i} m_{\phi^{-1}(i)}) \bmod p. \end{aligned}$$

G_R then outputs common input X_n and witness W_n:

$$\begin{aligned} X_n &= \{p, q, y, \bar{x}, g_0, m_0, \\ &\quad \{(g_i, m_i)\}_{i=1,\cdots,k}, \{(g'_i, m'_i)\}_{i=1,\cdots,k}\}, \\ W_n &= \{\phi, \{s_i\}_{i=1,\cdots,k}, x'\}. \end{aligned}$$

Where, U is a PPT Turing machine such that there exists a knowledge extractor that can extract $\bar{r}_i$ that satisfies $g_i = {g_0}^{\bar{r}_i}$ by rewinding U and choosing the random oracle. And $\bar{x}$ is the sum of other prover's private keys in the mix-net.

In the above definition, U is a PPT Turing machine that plays the role of (malicious and colluding) players who generate cipher-texts $\{(g_i, m_i)\}$. Although U is determined before the public parameter is generated, it does not lose generality because it has this public parameter as an input. In a case where U plays the role of honest players, it outputs

$$(g_i, m_i) = ({g_0}^{\bar{r}_i}, M_i {m_0}^{\bar{r}_i}) \bmod p \quad i = 1, \cdots, k$$

using random numbers $\{\bar{r}_i\}_{i=1,\ldots,k}$.

We say X_n and W_n *satisfy relation R* if the following equations are satisfied:

$$\begin{aligned} m_0 &\equiv {g_0}^{x'+\bar{x}}, y \equiv {g_0}^{x'} \pmod{p} \\ (g'_i, m'_i) &\equiv ({g_0}^{s_i} g_{\phi^{-1}(i)}, {g_i}^{-x'} {m_0}^{s_i} m_{\phi^{-1}(i)}) \pmod{p}. \end{aligned}$$

We denote this fact as $(X_n, W_n) \in R$. If there exists a witness W_n for a common input X_n that satisfies $(X_n, W_n) \in R$, common input X_n is a *correct shuffle-decryption.* Generator G_R outputs such an X_n.

Definition 3 *Let $View_V^P(X_n, W_n)$ be V's* view of an interaction *with P, which is composed of the common input X_n, messages V receives from P, random tape input to V, and messages V sends to P during joint computation employing X_n, where P has auxiliary input W_n such that $(X_n, W_n) \in R$. $View_V^P$ is an abbreviation of $View_V^P(X_n, W_n)$.*

We consider the case when a semi-honest verifier may collude with malicious players who encrypt the cipher-texts and other provers who shuffle and decrypt in the same mix-net. Such a verifier and players may obtain partial information regarding the plain texts $\{M_i\}$, private key $\bar{x}$, random tapes of players, and even a part of the permutation ϕ in addition to $View_V^P$. Moreover, they may obtain the results of other shuffle-decryptions executed by the same prover.

Then it is reasonable to describe this extra information that the verifier may have as $H(I_n, enc(U), X_n, \phi)$ and input cipher-texts generated by the malicious player as $U(I_n, g_0, y)$ using PPT Turing machines $H(\cdot)$ and $U(\cdot)$. Note that $\{s_i\}_{i=1,\cdots,k}$ are not included in the arguments of H, because we consider only the case where the prover never reveals these values to anyone and never uses the same $\{s_i\}_{i=1,\cdots,k}$ for other shuffle-decryptions.

Even though the verifier and the players may obtain the results of other shuffle-decryptions executed by the same prover who uses x', we do not include x' but include y into the input of U and H. Moreover, we assume that there exists a PPT Turing machine K such that the distribution of $View_V^P$ for such H and U and that of $K(I_n, g_0, y, enc(U), \phi)$ are the same. We denote this as $View_V^P \approx K(I_n, g_0, y, enc(U), \phi)$. The exclusion of x' is crucial because it enables us to consider the security of arguments for shuffle-decryption over the distribution of X_n, which includes that of x'. This is exactly the same way that we consider the security of plain shuffle-decryption. We describe information about the permutation ϕ that verifiers try to learn as $f(\phi)$ using PPT Turing machine f. This description can be justified because the expression $f(\phi)$ is sufficient to express any bit of ϕ and any kind of check sum for ϕ.

Now we can say that an argument for shuffle-decryption hides its permutation completely with respect to G_R - i.e., CPH occurs - if there exists a probabilistic polynomial time algorithm E'^E (which has black box access to E) with inputs X_n and $H(I_n, enc(U), X_n, \phi)$ that suffers no disadvantage with respect to learning anything about the permutations compared to any probabilistic polynomial time verifier E having input $View_V^P$ and $H(I_n, enc(U), X_n, \phi)$ even if the protocols with the same private key are repeatedly executed. This leads to:

Definition 4 (complete permutation hiding) *Let* $enc(U)$ *be an* encoding of a probabilistic polynomial time (PPT) Turing machine U *which generates cipher-texts* $\{(g_i, m_i)\}_{i=1,\ldots,k}$ *input to the shuffle-decryption procedure. Suppose that there exists a knowledge extractor that can extract* $\{\bar{r}_i\}_{i=1,\ldots,k}$ *such that* ${g_0}^{\bar{r}_i} = g_i$ *from* U.

Then, an argument for shuffle-decryption (P, V, G_R) *achieves* complete permutation hiding *if*

$$\begin{aligned}&{}^{\exists}E'^E\ {}^{\forall}E\ {}^{\forall}H\ {}^{\forall}f\ {}^{\forall}U\ {}^{\forall}c > 0\ {}^{\exists}N\ {}^{\forall}n > N\ {}^{\forall}I_n \\ &\quad \Pr[E(View_V^P, H(I_n, enc(U), X_n, \phi)) = f(\phi)] \\ &\quad < \Pr[E'^E(X_n, H(I_n, enc(U), X_n, \phi)) = f(\phi)]\end{aligned}$$

$$+\frac{1}{n^c}, \tag{2.35}$$

and

$$\exists_K \quad View_V^P \approx K(I_n, g_0, y, enc(U), \phi)$$

where E', E, H, f, U, K are PPT Turing machines. The left probability in Eq.(2.35) is taken over the distribution of the random tapes input to G_R, [6] *$P, V, H,$ and E. The right probability in Eq.(2.35) is taken over the distribution of the random tapes input to G_R, H, E', and E. E' may use E as a black box.*

Here, the definition includes the case where $\{(g'_i, m'_i)\}_{i=1,\ldots,k}$ is totally decrypted cipher-texts, that is, the case where $\{m'_i\}_{i=1,\ldots,k}$ are plain-texts.

If an argument for shuffle-decryption is CPH, we can say that for an input cipher-texts set $\{(g_i, m_i)\}$ and its corresponding output cipher-texts set $\{(g'_i, m'_i)\}$, whatever an honest verifier who has partial information $(H(I_n, enc(U), X_n, \phi))$ about the common input (X_n) and previously executed protocols (K) can learn $(f(\phi))$ about the permutation (ϕ) after interacting with a prover, can also - in an overwhelming number of cases of common input (X_n)- be efficiently computed from that common input (X_n) and that partial information $(H(I_n, enc(U), X_n, \phi))$ alone using a PPT Turing machine E' without interaction with the prover as long as the prover has chosen the private key x', permutation ϕ, and random numbers $\{s_i\}_{i=1,\cdots,k}$ uniformly and randomly.

Note that we are considering the case even where malicious and colluding players, who have the results of other shuffle-decryptions with the same x', are engaged in generating $\{(g_i, m_i)\}$ of common input. Hence, CPH guarantees security when shuffle-decryptions with the same private key are repeatedly executed.

Theorem 8 *If the decision Diffie-Hellman problem is difficult to solve, the proposed argument for shuffle-decryption (P, V, G_R) is complete-permutation-hiding.*

Proof. Let us define an extended version of the decision Diffie-Hellman problem, DDH_k^m. For primes p, q such that $q|p-1$, let $\Theta_{mk}^{(pq)}$ be a set consisting of $(k+1) \times m$ elements of $\mathbb{G}_q$ as

$$\Theta_{mk}^{(pq)} = (\Theta_{i\nu})_{i=1,\ldots,m;\nu=0,\ldots,k} \quad ,$$

and let R_k^m be a set consisting of all different $\Theta_{mk}^{(pq)}$, where the number of whose elements is $q^{m(k+1)}$.

Let D_k^m be the subset of R_k^m consisting of all the elements $\Theta_{mk}^{(pq)}$ of R_k^m that satisfy

$$\log_{\Theta_{10}} \Theta_{j0} = \log_{\Theta_{1\nu}} \Theta_{j\nu} \pmod q$$

[6] *Since the probability is taken over a distribution containing x', we have excluded any adversary who knows x'.*

for $\nu = 1, \ldots, k; j = 2, \ldots, m$, and the number of whose elements is $q^{(k+m)}$.

Definition 5 *Given a $\Theta_{mk}^{(pq)}$ chosen uniformly and randomly from either a set R_k^m or a set D_k^m, the DDH_k^m problem is that of deciding whether the given $\Theta_{mk}^{(pq)}$ has been chosen from D_k^m or R_k^m.*

The DDH_2^2 problem is a decision Diffie-Hellman problem.

Lemma 14 *Solving the DDH_k^m problem is as difficult as solving a decision Diffie-Hellman problem. That is,*

$$\begin{aligned}
&{}^{\forall}D\ {}^{\forall}c > 0\ {}^{\exists}N\ {}^{\forall}n > N \\
&\quad |\Pr(D[\Theta_{mk}^{(pq)} \in_R D_k^m] = 1) \\
&\quad - \Pr(D[\Theta_{mk}^{(pq)} \in_R R_k^m] = 1)| \\
&\quad \leq \frac{1}{n^c}
\end{aligned}$$

holds if and only if the decision Diffie-Hellman problem is difficult to solve, where D is a polynomially bounded algorithm.

Proof. The lemma can be proved by a hybrid argument. (ref. Lemmas 1 and 2 in [70].) □

Let us next define a simulator S of the proposed protocol (P, V) [7]. We will show that if the protocol is not CPH, there will be a distinguisher that can distinguish the distribution of any simulated view from that of a real protocol view, where probability is taken over "the random tape of G_R" as well as other random tapes. And we then show that distinguishing them is equivalent to solving the decision Diffie-Hellman problem.

Definition 6 *Given X_n, simulator S uniformly and randomly generates $\{r_\nu, r'_\nu\}_{\nu=-2,\cdots,k}$, $\{c_i\}_{i=1,\cdots,k}$, $r', c' \in_R \mathbb{Z}/q\mathbb{Z}$, $\{f'_i \in_R \mathbb{G}_q\}_{i=1,\cdots,k}$, and $\eta \in_R \mathbb{G}_q$. It then generates:*

$$\begin{aligned}
f'_0 &= \prod_{\nu=-2}^{k} {f_\nu}^{r_\nu} \prod_{i=1}^{k} {f'_i}^{-c_i} \bmod p \\
\tilde{f}'_0 &= \prod_{\nu=-2}^{k} {f_\nu}^{r'_\nu} \prod_{i=1}^{k} {f'_i}^{-{c_i}^2} \bmod p \\
g'_0 &= \prod_{\nu=0}^{k} {g_\nu}^{r_\nu} \prod_{i=1}^{k} {g'_i}^{-c_i} \bmod p
\end{aligned}$$

[7]If we want to prove that the protocol is computational zero-knowledge, we must show the existence of a simulator whose output is, even to a distinguisher who knows x', indistinguishable from the actual interaction. This is impossible here; i.e., the above protocol is not zero-knowledge. Though a distinguisher who knows x' is able to distinguish a real protocol-transcript from the output of this simulator (defined below), we are able to prove, by using this simulator, that the protocol hides permutations completely.

$$
\begin{aligned}
m'_0 &= \eta^{-1} \prod_{\nu=0}^{k} {m_\nu}^{r_\nu} \prod_{i=1}^{k} {m'_i}^{-c_i} \bmod p \\
w &= \sum_{j=1}^{k} ({r_j}^3 - {c_j}^3) - r_{-2} - r'_{-1} \bmod q \\
y' &= {g_0}^{r'} y^{-c'} \bmod p \\
\eta' &= \zeta^{r'} \eta^{-c'} \bmod p.
\end{aligned}
$$

S outputs:

$$
\begin{aligned}
&p, q, y, g_0, m_0, \{(g_i, m_i)\}_{i=1,\ldots,k}, \{(g'_i, m'_i)\}_{i=1,\ldots,k}, \\
&\{f_\nu\}_{\nu=-2,\ldots,k}, f'_0, \{f'_i\}_{i=1,\ldots,k}, \tilde{f}'_0, \\
&g'_0, m'_0, w, \{c_i\}_{i=1,\ldots,k}, \\
&\{r_\nu\}_{\nu=-2,\ldots,k}, \{r'_\nu\}_{\nu=-2,\ldots,k}, \eta, \eta', y', c', r'.
\end{aligned}
$$

Let us next define an algorithm M which generates distribution of view from $\Theta^{(pq)}_{2k}$.

Definition 7 *(**Algorithm** M.) Given $I_n = \{1^n, p, q, \bar{x}, \{M_i\}_{i=1,\cdots,k}, Z_n\}$, enc$(U)$, and $\Theta^{(pq)}_{3k}$, M uniformly and randomly chooses $f_{-2}, f_{-1}, \{f_\nu\}_{i=1,\cdots,k}$ and generates*

$$
\begin{aligned}
f_0 &= \Theta_{30} \\
(g_i, m_i) &= U(I_n, g_0, y, F_k) \in (\mathbb{G}_q \times \mathbb{G}_q)
\end{aligned}
$$

where $F_k = \{f_\nu\}_{\nu=-2,\ldots,k}$. M then extracts $\{\bar{r}_i\}_{i=1,\cdots,k}$ such that $g_i = {g_0}^{\bar{r}_i} \bmod p$ using U as a black box. M uniformly and randomly chooses $\{c_i\}_{i=1,\cdots,k}, \{r_\nu, r'_\nu\}_{\nu=-3,\cdots,k}, r', c' \in_R \mathbb{Z}/q\mathbb{Z}$, and a $k \times k$ permutation matrix $(A_{ji})_{i,j=1,\cdots,k}$. Then, for $i = 1, \ldots, k$, it generates:

$$
\begin{aligned}
g_0 &= \Theta_{10} \ , \quad y = \Theta_{20} \ , \quad m_0 = y {g_0}^{\bar{x}} \bmod p \ , \\
g'_i &= \Theta_{1i} \prod_{j=1}^{k} {g_j}^{A_{ji}} \bmod p \\
m'_i &= (y^{-\sum_{j=1}^{k} \bar{r}_j A_{ji}}) {\Theta_{1i}}^{\bar{x}} \prod_{j=1}^{k} {m_j}^{A_{ji}} \bmod p \ , \\
A_{i0} &= r_i - \sum_{j=1}^{k} A_{ij} c_j \bmod q \\
A_{-1i} &= \sum_{j=1}^{k} 3 A_{j0} A_{ji} \bmod q
\end{aligned}
$$

$$
\begin{aligned}
A_{-2i} &= \sum_{j=1}^{k} 3{A_{j0}}^2 A_{ij} \bmod q \ , \\
\zeta &= \prod_{i=1}^{k} g_i'^{c_i} \bmod p \ , \\
\eta &= y^{\sum_{i,j=1}^{k} \bar{r}_j A_{ji} c_i} \prod_{i=1}^{k} {\Theta_{2i}}^{c_i} \bmod p \\
\eta' &= \zeta^{r'} \eta^{-c'} \bmod p \\
g_0' &= \zeta^{-1} \prod_{\nu=0}^{k} {g_\nu}^{r_\nu} \bmod p \\
m_0' &= \eta^{-1} \prod_{\nu=0}^{k} {m_\nu}^{r_\nu} \prod_{j=1}^{k} m_j'^{-c_j} \bmod p \\
y' &= {g_0}^{r'} y^{-c'} \bmod p \ , \\
f_i' &= f_{-2}^{A_{-2i}} f_{-1}^{A_{-1i}} \Theta_{3i} \prod_{j=1}^{k} {f_j}^{A_{ji}} \bmod p \\
f_0' &= \prod_{\nu=-2}^{k} {f_\nu}^{r_\nu} \prod_{j=1}^{k} f_j'^{-c_j} \bmod p \\
\tilde{f}_0' &= \prod_{\nu=-2}^{k} {f_\nu}^{r_\nu'} \prod_{j=1}^{k} f_j'^{-{c_j}^2} \bmod p \\
w &= \sum_{j=1}^{k} (r_j^3 - c_j^3) - r_{-2} - r_{-1}' \pmod q
\end{aligned}
$$

M outputs:

$$
\begin{aligned}
&p, q, y, g_0, m_0, \{(g_i, m_i)\}_{i=1,\dots,k}, \{(g_i', m_i')\}_{i=1,\dots,k}, \\
&\{f_\nu\}_{\nu=-2,\dots,k}, f_0', \{f_i'\}_{i=1,\dots,k}, \\
&\tilde{f}_0', g_0', m_0', w, \{c_i\}_{i=1,\dots,k}, \\
&\{r_\nu\}_{\nu=-2,\dots,k}, \{r_\nu'\}_{\nu=-2,\dots,k}, \eta, \eta', y', c', r'.
\end{aligned}
$$

as $View^\dagger(I_n, U, \Theta_{3k}^{(pq)})$. $View^\dagger$ *is an abbreviation of* $View^\dagger(I_n, U, \Theta_{3k}^{(pq)})$. *M also outputs* (A_{ji}) *as* $(A_{ji})^\dagger_{i,j=1,\cdots,k}$.

Lemma 15 *Assume* X_n *is the output of* G_R*, whose input is a set* $I_n = \{1^n, p, q, \bar{x}, \{M_i\}_{i=1,\cdots,k}, Z_n\}$ *and an enc*(U) *Also assume* $View_V^P$ *is a view of* (P, V) *with respect to* F_k*, the above enc*(U)*,* I_n *and uniformly and randomly chosen* g_0, y*. If* $\Theta_{3k}^{(pq)} \in_R D_k^3$*, then the distribution of* $View^\dagger$ *for the above* I_n *and enc*(U) *is equivalent to that of the above* $View_V^P$ *for the above* I_n *and enc*(U)*, where the distribution* *of $View^{\dagger}$ is considered with respect to $f_{-2}, f_{-1}, \{f_i, c_i\}_{i=1,\cdots,k}, \{r_\nu, r'_\nu\}_{\nu=-2,\ldots,k}, r', c'$, $(A_{ji})_{i,j=1,\cdots,k}$ and $\Theta_{3k}^{(pq)}$ and the distribution of $View_V^P$ is considered with respect to the random tapes input to G_R, P, and V and with respect to F_k.*

Proof. The proof is clear from the following correspondences (for $i = 1, \ldots, k$):

$$\begin{aligned}
&\Theta_{10} \Leftrightarrow g_0, \Theta_{1i} \Leftrightarrow g_0^{A_{0i}},\\
&\Theta_{20} \Leftrightarrow y, \Theta_{2i} \Leftrightarrow y^{A_{0i}},\\
&\Theta_{30} \Leftrightarrow f_0, \Theta_{3i} \Leftrightarrow f_0^{A_{0i}}
\end{aligned}$$

□

Lemma 16 *Assume X_n is the output of G_R, whose input is a set $I_n = \{1^n, p, q, \bar{x}, \{M_i\}_{i=1,\cdots,k}\}$ and $enc(U)$. Also assume $View^*$ is an output of Simulator S whose input is X_n. If $\Theta_{3k}^{(pq)} \in_R R_k^3$, then the distribution of $View^{\dagger}$ for the above I_n and $enc(U)$ is equivalent to that of the above $View^*$ for the above I_n, where the distribution of $View^{\dagger}$ is considered with respect to $f_{-2}, f_{-1}, \{f_i, c_i\}_{i=1,\cdots,k}, \{r_\nu, r'_\nu\}_{\nu=-2,\ldots,k}$, $r', c', (A_{ji})_{i,j=1,\cdots,k}$ and $\Theta_{3k}^{(pq)}$ and the distribution of $View^*$ is considered with respect to the random tapes input to G_R, S, and M and with respect to F_k.*

Proof. In $View^{\dagger}$, there is sufficient randomness, originating from the elements in $\Theta_{3k}^{(pq)}$, to match the randomness of a shuffle and that of $f'_i, \eta \in_R \mathbb{G}_q$ uniformly and randomly chosen by the simulator. □

Lemma 17 *There exists a PPT Turing machine K such that its output distribution on input of I_n, g_0, y, $enc(U), F_k, \phi$ i.e.,$K(I_n, g_0, y, enc(U), \phi)$ is the same as that of $View_V^P$.*

Proof. Since $\bar{r}_i$ such that $g_i = {g_0}^{\bar{r}_i} \bmod p$ is extractable by the PPT Turing machine, K is able to generate m'_i and η as

$$\begin{aligned}
m'_i &= g_i'^{-x'} \prod_{\nu=0}^{k} {m_\nu}^{A_{\nu i}}\\
&= y^{A_{0i}} y^{\sum_{j=1}^{k} \bar{r}_j A_{ji}} \prod_{\nu=0}^{k} {m_\nu}^{A_{\nu i}} \bmod p\\
\eta &= \prod_{i=1}^{k} \left(y^{-A_{0i}} y^{-\sum_{j=1}^{k} \bar{r}_j A_{ji}} \right)^{c_i} \bmod p
\end{aligned}$$

without the knowledge of x'. Other variables in $View_V^P$ are perfectly simulatable from the above values. □

We are able to prove that the contraposition of the first statement of the theorem holds. That is, we are able to prove that, if the negation of Formula (2.35) holds, then there exists a polynomially bounded algorithm that can solve the DDH_k^3 problem.

Assume that the negation of Formula (2.35), that is,

$$\begin{aligned}
&{}^{\forall}E'^{E}\ {}^{\exists}E\ {}^{\exists}H\ {}^{\exists}f\ {}^{\exists}U\ {}^{\exists}c>0\ {}^{\forall}N\ {}^{\exists}n>N\ {}^{\exists}I_n\\
&\quad \Pr[E(View_V^P, H(I_n, enc(U), X_n, (A_{ji}))) = f((A_{ji}))]\\
&\quad \geq \Pr[E'^{E}(X_n, H(I_n, enc(U), X_n, (A_{ji}))) = f((A_{ji}))]\\
&\quad +\frac{1}{n^c},
\end{aligned}$$

holds. If we let E' be an algorithm that first generates a simulated view $View'^P_V$ from X_n using the simulator defined in Definition 6, and then guess the permutation using E as a black box, then the formula

$$\begin{aligned}
&{}^{\exists}E\ {}^{\exists}U\ {}^{\exists}c>0\ {}^{\forall}N\ {}^{\exists}n>N\ {}^{\exists}I_n\\
&\quad \big|\Pr[E(View_V^P, H(I_n, enc(U), X_n, (A_{ji}))) = f((A_{ji}))]\\
&\quad - \Pr[E(View'^P_V, H(I_n, enc(U), X_n, (A_{ji}))) = f((A_{ji}))]\big|\\
&\qquad \geq \frac{1}{n^c},
\end{aligned}$$

holds. If we include the definitions of H, f and E into E', then the formula

$$\begin{aligned}
&{}^{\exists}E'\ {}^{\exists}U\ {}^{\exists}c>0\ {}^{\forall}N\ {}^{\exists}n>N\ {}^{\exists}I_n\\
&\quad \big|\Pr[E'(View_V^P, I_n, enc(U), (A_{ji})) = 1]\\
&\quad -\Pr[E'(View'^P_V, I_n, enc(U), (A_{ji})) = 1]\big| \geq \frac{1}{n^c}
\end{aligned}$$

holds. From Lemma 15 the distribution of $\{View_V^P, I_n, enc(U), (A_{ji})\}$ is equivalent to that of $\{View^\dagger, I_n, enc(U), (A_{ji})^\dagger\}$ when $\Theta_{3k}^{(pq)} \in_R D_k^3$. From Lemma 16 the distribution of $\{View'^P_V, I_n, enc(U), (A_{ji})\}$ is equivalent to that of $\{View^\dagger, I_n, enc(U), (A_{ji})^\dagger\}$ when $\Theta_{3k}^{(pq)} \in_R R_k^3$. Therefore, we can construct a distinguisher D by using algorithms M and E as black boxes such that the formula

$$\begin{aligned}
&{}^{\exists}D\ {}^{\exists}c>0\ {}^{\forall}N\ {}^{\exists}n>N\\
&\quad |\Pr(D[\Theta_{3k}^{(pq)} \in_R D_k^3] = 1)
\end{aligned}$$

$$- \Pr(D[\Theta_{3k}^{(pq)} \in_R R_k^3] = 1)| \geq \frac{1}{n^c}$$

holds. This means that the DDH_k^3 problem can be solved by using E. Therefore, the contraposition of the lemma has been proved.

The second statement is satisfied from Lemma 17 □

A complete description of distinguisher D is as follows:

Definition 8 *Given $\Theta_{3k}^{(pq)}$ of size n, D first chooses F_k uniformly and randomly, and chooses arbitrary $\{M_i\}_{i=1,\cdots,k}$, and $\mathbb{Z}/q\mathbb{Z}$ elements $\bar{x}$, Z, and $enc(U)$. It gives these values $(I_n, enc(U))$ and $\Theta_{3k}^{(pq)}$ to M, and then obtains $View^\dagger$, and permutation matrix $(A_{ji})^\dagger$ from M.*

Next, D gives $View^\dagger, enc(U), I_n$, and $(A_{ji})^\dagger$ to the adversary E' and obtains response 1 or 0. Finally, D outputs the response of E'.

□

2.6 Efficiency

In this section, we compare the efficiency of the proposed protocol described in Section 2.3 (Pro-I) and the proposed protocol described in Section 2.4 (Pro-II) to (FS), the protocol proposed in [70], (Fmmos), the protocol proposed in [69], and (Groth), the protocol proposed in [85]. We have assumed the lengths of p and q to be 1024 and 160.

	FS	Fmmos	Groth	Pro-I	Pro-II
S-P	$8k$		$6k$	9k	
S-V	$10k$		$6k$	6k	
SD-P	$(9k)$	$9k$	$(7k)$	(10k)	$8k$
SD-V	$(12k)$	$10k$	$(8k)$	(8k))	$6k$

Table 2.1: Numbers of exponentiations required in each protocol

Let us first compare them, in Table 2.1[8], by the number of exponentiations used in each protocol when the number of cipher-texts is k. "S-P" and "S-V" denote the number of exponentiations required for P and V to prove and verify a shuffle. 'SD-P" and "SD-V" denote the number of exponentiations required for P and V to prove and verify a shuffle-decryption. The numbers for (FS), (Fmmos), (Groth), and (Pro-I) are the sum of those required to prove a shuffle and those required to prove a decryption. The 3-round protocol for proving decryption is as follows.

[8]The computational cost required for provers of protocol "Pro-I" can be reduced by $1k$ if $q \bmod 3 = 2$

1. Given shuffled cipher-texts $\{(g'_i, \bar{m}_i)\}_{i,\ldots,k}$ and shuffle-decrypted cipher-texts $\{(g'_i, m'_i)\}_{i,\ldots,k}$, a prover first chooses $\{\beta_i \in_R \mathbb{Z}/q\mathbb{Z}\}_{i=1,\ldots,k}$ uniformly and randomly and then generates and sends to a verifier:

$$\bar{g}_i = {g'_i}^{\beta_i} \bmod p.$$

2. Next, the verifier chooses $\{c_i \in_R \mathbb{Z}/q\mathbb{Z}\}_{i\ldots,k}$ and sends it back to the prover.

3. Next, the prover returns

$$r'_i = x'c_i + \beta_i \bmod q.$$

4. Finally, the verifier checks that

$$ {g'_i}^{r'_i} = (\bar{m}_i/m'_i)^{c_i}\bar{g}_i \bmod p$$

holds.

As is mentioned in Subsection 2.4.3, the technique of proving the correctness of both shuffling and decryption simultaneously is not directly available to the protocol proposed in [85] since η, which is a combination of $\{g'_j\}_{i=1,\cdots,k}$ and the challenge $\{c_i\}_{i=1,\cdots,k}$, does not appear in that protocol. And since a more efficient proof for decryption is not known, one of the most efficient ways for proving the correctness of both shuffling and decryption using [85] is to use the above presented proof for decryption.

If we adopt the computation tools described in [107], such as the simultaneous multiple exponentiation algorithm and the fixed-base comb method, the cost for computing exponentiations can be heuristically reduced. We estimated that multiple exponentiations cost a 1/3 and exponentiations by the fixed-base comb method cost a 1/12 (when the number of cipher-texts is large) of that of single exponentiation. Estimations done in this way are given in Table 2.2. Even if two protocols require the same number of exponentiations, one that can benefit from fixed-base comb method more than the other is able to reduce its computational cost more than the other.

Table 2.3 lists the number of communication bits and number of rounds required for protocols. "S" denotes the number of communication bits used when proving a shuffle, "SD" denotes the number of communication bits used when proving a shuffle-decryption, and "rounds" denotes the number of rounds required for protocols. The numbers for (FS), (Fmmos), (Groth), and (Pro-I) include intermediate state bits, i.e., those of shuffled cipher-texts.

Table 2.4 lists the level of security achieved by protocols. Here "DL" denotes that the corresponding

	FS	Fmmos	Groth	Pro-I	Pro-II
S-P	$1.4k$		$1.75k$	1.75k	
S-V	$3.3k$		$1.75k$	2k	
SD-P	$(2.4k)$	$1.75k$	$(2.75k)$	(2.75k)	$1.9k$
SD-V	$(4.5k)$	$3.3k$	$(3k)$	(3.2k))	$2k$

Table 2.2: Cost of computation required in each protocol

	FS	Fmmos	Groth	Pro-I	Pro-II
S	$5044k$		$1184k$	1344k	
SD	$(6388k)$	$5044k$	$(2528k)$	(2688k)	$1344k$
rounds	3	5	7	3	5

Table 2.3: Communication bits required in each protocols

protocol is sound as long as the discrete logarithm problem is difficult to solve. For Groth's protocol, we consider the case when it shuffles ElGamal cipher-texts. "CPH" denotes complete permutation hiding. "PZK" denotes perfect zero-knowledge. The protocol proposed in [69] can also be proved to be CPH. The protocol proposed in [70] can also be proved to be CPH assuming $y = 1$. The security of this protocol is proved also in [122]

Our protocols and the protocols of [69, 70] require a rather long public parameter F_k. Although the protocol of [85] also requires such a parameter, it can be reduced greatly at the cost of increasing the amount of both computation and communication.

From Tables 2.2, 2.3, and 2.4, we can conclude that our zero-knowledge argument for shuffling is the most efficient among the less-than-7-round perfect zero-knowledge arguments for shuffling. The only comparable protocol is that of [137], which requires $640k$ exponentiations.

From Tables 2.2 and 2.3, we can conclude that computational complexity with our argument for shuffle-decryption protocol represents a 32% improvement in efficiency over that of (Groth)[85], while communication complexity improves by 47%. Our protocol requires two rounds less than that of Groth's [85].

	FS	Fmmos	Groth	Pro-I	Pro-II
soundness	DL	DL	DL	DL	DL
anonymity	CPH	CPH	PZK	PZK	CPH

Table 2.4: Security levels of each protocol

2.7 Robustness

For the proposed argument for shuffle-decryption to be robust, we may consider that it is desirable to provide a method to make threshold shuffle-decryption. Since a shuffle and decryption are executed in succession by a single prover, it is not easy to make threshold shuffle-decryption. However, if each prover distributes his secret key to other provers in a desired threshold, robustness can be provided easily. In case any prover quits shuffle-decryption after some provers had shuffle-decrypted the cipher-texts, honest provers are able to collaborately recover the secret key of the malicious prover and make explicit decryption with that secret key. With this strategy, unless the number of honest provers is less than an appropriately configured threshold, honest provers are able to shuffle-decrypt all the cipher-texts. Since the cost for distributing secret keys depends on the number of provers but not on the number of cipher-texts, its cost is negligible when the number of cipher-texts is huge.

2.8 Conclusion

In this chapter, we have proposed an efficient argument for shuffling and an efficient argument for shuffle-decryption. The former is most efficient in computational and in communication complexity among 3-move honest verifier perfect zero-knowledge arguments for shuffling of ElGamal cipher-texts. The latter protocol is the most efficient in computational, communication, and round complexity, as a whole, in proving the correctness of both shuffling and decryption of ElGamal cipher-texts. We also proposed a formal definition for the core requirement of unlinkability in arguments for shuffle-decryption, and then proved the latter protocol enjoys this property.

Chapter 3

Hybrid mix-net

3.1 Introduction

Mix-net[42] schemes are useful for applications which require anonymity, such as voting and anonymous questionnaires. To ensure the correctness of output, it is desirable to have the property of public verifiability. A typical realization of a publicly verifiable mix-net scheme is, as we have considered in Chapter 2, that based on a zero-knowledge proof system for shuffling of ElGamal ciphertexts[2, 65, 66, 69, 70, 85, 120, 137].

However, these schemes only achieve their best efficiency when sets of ciphertexts are of a short length, one fixed to the length determined by the employed encryption algorithm. A typical length is, say 160 bits long. In order to verifiably shuffle texts of a fixed but a longer length, a straightforward approach, for example, is to divide each text into blocks of the predetermined short bits, re-encrypt each block, and concatenate them to form a string. Then, this block-wisely re-encrypted string is permuted. This approach requires public key operations and shuffle-proving operations for each block, thus the computational cost is linear in the length of the input. Another kind of mix-net scheme, referred to as "hybrid mixing,"[83, 91, 92, 126], is able to shuffle long ciphertexts efficiently, but the correctness of its shuffling is not publicly verifiable. It is only mutually verifiable among its component mixers.

Neither of these approaches would be applicable, when long messages, such as those which might form the replies to questionnaires or write-in votes, are to be tallied, and when the correctness of the tallying needs to be publicly verifiable. A scheme that is publicly verifiable and is capable of shuffling ciphertexts of long but common length efficiently is yet to be proposed.

Our Contributions

We propose here the first efficient hybrid-mixing scheme that provides public verifiability. In our scheme, the number of zero-knowledge arguments for proving a shuffle does not depend on the length of the input ciphertext. Although the resulting mix-net does not provide full public verifiability of the hybrid decryption in the case when a user and a mixer collude, the best adversary can do is to switch the input between a valid and an invalid one. Moreover, all the users whose input failed decrypted correctly are traced. We prove the security properties of the proposed scheme assuming the random oracle model, semantically secure one-time symmetric-key cryptosystem, and intractability of decision Diffie-Hellman problem.

In the course of constructing the verifiable hybrid-mix, we have developed (1) a new IND-ME-CCA secure [152] encryption scheme that multiply uses IND-CCA2 secure hybrid encryption and (2) a 3-move efficient perfect zero-knowledge argument for shuffle-and-decryption of ElGamal ciphertexts.

Construction of Our Mix-Net Scheme

A commonly adopted construction of publicly verifiable mix-net is a combination of IND-CCA2 secure encryption scheme that is suitable for shuffling, a secure group decryption scheme to be performed by mixers, and zero-knowledge arguments for shuffle and decryption. For example, a typical construction [2, 69] is a combination of IND-CCA2 secure ElGamal type cryptosystem, a threshold decryption scheme of ElGamal ciphertexts, and zero-knowledge arguments for shuffling and decrypting ElGamal ciphertexts. Such construction provides secure verifiable mix-net but suffers from the restriction that the input message should be within the domain of ElGamal encryption scheme.

Our construction follows the above approach, but we use a hybrid encryption instead of a plain ElGamal encryption so that we can handle long messages. In order to achieve the group decryption property, we designed a new multiple encryption scheme where hybrid encryptions are repeatedly applied each public key of the mixers. In order to achieve the secure threshold decryption property and IND-CCA2 secure property we devised our multiple encryption scheme to achieve the IND-ME-CCA secure [152] property with repetitive IND-CCA2 secure hybrid encryptions. For public verifiability, we add encryption of a hash of the plaintext in the ciphertext in each repetition to achieve the publicly verifiable hybrid decryption.

We also provide a zero-knowledge argument for shuffle of the proposed multiple encryption scheme. For this purpose, we use a perfect zero-knowledge argument for shuffle-and-decryption of ElGamal ciphertexts. We can also use the scheme in [86]. We note that a cost we pay for public verifiability is that the length of the input ciphertext grows in linear in the number of mixers unlike the scheme in [126].

The rest of the chapter is organized as follows. Section 3.2 proposes IND-ME-CCA secure multiple encryption scheme using IND-CCA2 secure hybrid encryption scheme with efficient verifiable decryption property. Section 3.5 proposes an efficient perfect zero-knowledge argument for shuffle-and-decryption of ElGamal ciphertexts. Section 3.3 illustrates our publicly verifiable mix-net scheme, Section 3.4 discusses with analysis on the efficiency of our mix-net scheme.

3.2 IND-ME-CCA Secure Multiple Encryption in Hybrid Setting

We present here our multiple encryption scheme that is suitable to be used in a verifiable mix net. We adopted the model from [152] with modifications to add auxiliary output of *Dec* algorithm to suit for use in mix-net. A multiple encryption scheme *ME* is a public key encryption scheme, which consists of a set of a key generation algorithm (*MKey-Gen*), an encryption algorithm (*MEnc*), and a decryption algorithm (*MDec*). Each of these algorithms invokes respective algorithms of a public key encryption scheme (*Key-Gen*, *Enc*, *Dec*) for multiple times. We also adopt the security notion of multiple encryption called IND-ME-CCA which is introduced in [152]. This notion differs from IND-CCA2 security in the sense that the adversaries are allowed to access key exposure oracle. We note that our multiple encryption scheme additionally offers public verifiability of decryption, which is a property not considered in [152].

We provide a formal model and the definitions in the following:

Definition 9 (Multiple Encryption) *Multiple encryption scheme ME is a set of following algorithms (Key-Gen, Enc, Dec, MKey-Gen, MEnc, MDec):*

Key-Gen: *A probabilistic key generation algorithm that, given a security parameter* 1^n*, outputs a pair of a public key and a secret key* (pk, sk)*.*

Enc: *A probabilistic encryption algorithm that, given the public key pk and a message* M*, outputs a ciphertext* C*.*

Dec: *An probabilistic algorithm that, given a ciphertext* C *and the secret key sk, outputs a message* M*(or* $\perp$*) and auxiliary data.*

MKey-Gen: *An algorithm that, given a security parameter* 1^n*, invokes Key-Gen* m *times to output a public key* $\{pk^{(j)}\}_{j=1,\ldots,m}$ *and a secret key* $\{sk^{(j)}\}_{j=1,\ldots,m}$*. We assume that the message space is* $\mathcal{M}$ *and the ciphertext space is* $\mathcal{C}$*.*

MEnc: *An algorithm that, given* $M \in \mathcal{M}$ *and a public key* $\{pk^{(j)}\}_{j=1,\ldots,m}$*, repeatedly invokes Enc* m *times, where in the first invocation Enc takes* $pk^{(m)}$ *and* M *as input, in the second invocation*

Enc takes $pk^{(m-1)}$ and the output of the previous invocation as input, and so on. The output is a ciphertext $C \in \mathcal{C}$, which is thus multiply encrypted M.

MDec**:** *A probabilistic algorithm that, given $C \in \mathcal{C}$ and a secret key $\{sk^{(j)}\}_{j=1,\ldots,m}$, repeatedly invokes Dec m times to output $M \in \mathcal{M}$ and auxiliary data generated by Dec's. In j-th invocation, MDec gives $sk^{(j)}$ to Dec.*

Definition 10 (IND-ME-CCA [152]) *Assume any polynomially bounded adversary $\mathcal{A}$ plays the following game with ME. First, key generation algorithm MKey-Gen is run. The public key $PK = \{pk^{(j)}\}_{j=1,\ldots,m}$ is given to $\mathcal{A}$ and an challenging oracle $\mathcal{CO}$. The secret key $SK = \{sk^{(j)}\}_{j=1,\ldots,m}$ is given to a decryption oracle $\mathcal{DO}$ and a key exposure oracle $\mathcal{KE}$. If two messages $M_0, M_1 \in \mathcal{M}$ and $b \in \{0,1\}$ are given to $\mathcal{CO}$, it outputs $C_* = MEnc(M_b) \in \mathcal{C}$. If index $j \in \{1,\ldots,m\}$ is given to $\mathcal{KE}$, it returns $sk^{(j)}$. $\mathcal{A}$ is allowed to invoke $\mathcal{KE}$ only $(m-1)$ times. When $C \in \mathcal{C}$ such that $C \neq C_*$ is given, $\mathcal{DO}$ executes MDec to obtain decryption of C and auxiliary data. $\epsilon_{neg}(n)$ is a negligible function of n and state is an intermediate state of $\mathcal{A}$. We call this ME IND-ME-CCA secure. if*

$$\Pr\left[b = b' \;\middle|\; \begin{array}{l} (PK, SK) \leftarrow \mathit{MKey\text{-}Gen}(1^n), \\ (M_0, M_1, \mathit{state}) \leftarrow \mathcal{A}(PK)^{\mathcal{DO},\mathcal{KE}} \\ b \leftarrow_R \{0,1\}, \\ C_* \leftarrow \mathcal{CO}(M_0, M_1, b), \\ b' \leftarrow \mathcal{A}(PK, \mathit{state})^{\mathcal{DO},\mathcal{KE}} \end{array} \right] < \frac{1}{2} + \epsilon_{neg}(n).$$

3.2.1 The Idea of the Proposed Multiple Encryption Scheme

In the proposed encryption scheme, the following three operations are repeated m times using m independent symmetric keys $\{K^{(j)}\}_{j=1,\ldots,m}$ and m public keys $\{X^{(j)}\}_{j=1,\ldots,m}$. (1) Symmetric-key encryption of the input ciphertext $\mu^{(j)}$ using a symmetric key $K^{(j)}$, and ElGamal encryption of this symmetric key $\{K^{(j)}\}$. (2) Hashing the input ciphertext $\mu^{(j)}$, and ElGamal encryption of this hashed value, and (3) generating a proof that the user himself was engaged in this repetition.

The first procedure is the main function of hybrid encryption scheme. The second procedure generates hashed value that will be used to verify decryptions. If we compare this value to the hash of a decrypted text using symmetric-key cryptosystem, we can efficiently verify that that the symmetric-key decryption has been operated correctly. The case of colluding user and one of the decrypter is discussed in Remark 1.

The third procedure comprises the core technique to make the total encryption scheme IND-ME-

CCA secure. The main purpose of the procedure is to make sure that the user himself performed the encryption, and not an adversary who has eavesdropped some intermediate data. A simplified description for the third procedure is as follows: Let G be a generator of an elliptic curve E [1] and $[r]G$ denote scalar multiplication of G by $r \in \mathbb{Z}/q\mathbb{Z}$. In $(m-j+1)$-th repetition of the third procedure, user generates $(E^{(j)}, J^{(j)}) = ([r^{(j)}]X^{(j)}, [r^{(j)}]G + J^{(j+1)})$, which is an encryption of $J^{(j+1)} = [r^{(m)} + \cdots + r^{(j+1)}]G$ and provides the knowledge of the random number $r^{(j)}$. After m-th repetition, the output is $([r^{(1)}]X^{(1)}, [r^{(m)} + \cdots + r^{(1)}]G)$. The user also provides the proof on the knowledge of $r^{(m)} + \cdots + r^{(1)}$.

The scheme is so designed to achieve non-malleability, that is, that even if a malicious adversary copied a j-th partially decrypted ciphertext and submitted a modified version of this copy to the mix-net, he will be detected. This is because the adversary will fail to prove the knowledge of at least one random number used for generating ciphertexts unless he can solve the discrete logarithm problem. Therefore, by adding these operations, we can restrict the accepted ciphertexts to be those which the submitter himself engaged in very repetition himself.

3.2.2 Proposed Multiple Encryption with Verifiable Decryption

We define $n, n', q, E, G, \mathcal{O}, \mathcal{H}_E, \mathcal{H}_q$, and $\mathcal{H}_\sigma$ as follows: n is a security parameter, q is a prime such that $|q| = n$ and $q \bmod 3 = 2$, E is a set of points on an elliptic curve of an order q, G is a generator of E, $\mathcal{O}$ is the zero of E, $\mathcal{H}_E, \mathcal{H}_q$, and $\mathcal{H}_\sigma$ are cryptographic hash functions that map, respectively, arbitrary strings to points in E, arbitrary strings to elements of $\mathbb{Z}/q\mathbb{Z}$, and arbitrary points in E to n' bit strings. Let $(\mathrm{enc}_\kappa, \mathrm{dec}_\kappa)$ be an encryption algorithm and a decryption algorithm of a symmetric-key cryptosystem which is semantically secure under chosen plaintext attack, where κ is a symmetric-key of length n'. The message space $\mathcal{M}$ is $\{0,1\}^\ell$ and the corresponding ciphertext space of *MEnc* is denoted by $\mathcal{C}$.

We first describe the key generation phase.

MKey-Gen: Given a security parameter 1^n, first chooses an arbitrary set of parameters $(q, E, G, \ell, \mathcal{H}_E, \mathcal{H}_q, \mathcal{H}_\sigma, (\mathrm{enc}_\kappa, \mathrm{dec}_\kappa), \mathcal{M}, \mathcal{C})$, which we call $\mathcal{D}$, then *MKey-Gen* invokes *Key-Gen* m times with $\mathcal{D}$ to obtain public keys $\{X^{(j)}\}_{j=1,\ldots,m}$ and secret keys $\{x^{(j)}\}_{j=1,\ldots,m}$. Finally, *MKey-Gen* outputs $\mathcal{D}, \{X^{(j)}\}_{j=1,\ldots,m}$, and $\{x^{(j)}\}_{j=1,\ldots,m}$.

Key-Gen: Given $\mathcal{D}$ in j-th invocation, outputs a randomly chosen secret key $x^{(j)} \in_R \mathbb{Z}/q\mathbb{Z}$ and an ElGamal public key

$$X^{(j)} := [x^{(j)}]G. \tag{3.1}$$

[1]Any DDH hard cyclic group suits for our purpose. But, we use the notation of scalar multiplication in elliptic curves since we need many superscripts and subscripts

We will now show how to multiply encrypt a message $M \in \mathcal{M}$. The first invocation of *Enc* is performed on the message M using a public key $X^{(m)}$. Then, its output, $\mu^{(m-1)}$, will be the input to the next invocation of *Enc*. The output $\mu^{(0)}$ of the last invocation together with the additional proof of knowledge will be the final output of our multiple encryption algorithm *Enc*. Let a message space and a ciphertext space of $(m-j+1)$-th invocation of *Enc* be denoted as $\mathcal{M}^{(j)}$ and $\mathcal{C}^{(j)}$, where $\mathcal{C}^{(j)} = \mathcal{M}^{(j-1)}$.

As we will see below in Eq.(3.2), each element in $\mu^{(j-1)} \in \mathcal{C}^{(j)}$ can be represented as a structure of multiple data $\mu^{(m)} = (M, \mathcal{O})$.

*Enc***:** The ElGamal public key $X^{(j)}$ and a message $\mu^{(j)} \in \mathcal{M}^{(j)}$ are given. *Enc* randomly chooses $K^{(j)} \in_R E$, a tuple $(r_K^{(j)}, r_H^{(j)}, r_J^{(j)}) \in_R (\mathbb{Z}/q\mathbb{Z})^3$, and a tuple $(s_K^{(j)}, s_H^{(j)}, s_J^{(j)}) \in_R (\mathbb{Z}/q\mathbb{Z})^3$. The message $\mu^{(j)}$ is encrypted in hybrid manner, that is, data $K^{(j)}$ is encrypted with ElGamal public key $X^{(j)}$, while $K^{(j)}$ transformed to a symmetric key $\kappa^{(j)}$ and used to encrypt $\mu^{(j)}$ with symmetric encryption $enc_{\kappa^{(j)}}$

$$\begin{aligned} (E_K^{(j)}, D_K^{(j)}) \quad &:= \quad \left([r_K^{(j)}]X^{(j)}, [r_K^{(j)}]G + K^{(j)}\right), \\ \kappa^{(j)} := \mathcal{H}_\sigma(K^{(j)}) \quad &, \quad \chi^{(j)} := \mathrm{enc}_{\kappa^{(j)}}(\mu^{(j)}). \end{aligned}$$

The input message $\mu^{(j)}$ is hashed and then encrypted with ElGamal public key $X^{(j)}$ to be used in verification of the decryption.

$$\begin{aligned} H^{(j)} \quad &:= \quad \mathcal{H}_E(\mu^{(j)}) \\ (E_H^{(j)}, D_H^{(j)}) \quad &:= \quad \left([r_H^{(j)}]X^{(j)}, [r_H^{(j)}]G + H^{(j)}\right). \end{aligned}$$

The element $J^{(j+1)}$ in $\mu^{(j)}$ is encrypted to be used to prove that the user himself was engaged in the computation, which operation makes total multiple encryption IND-ME-CCA secure.

$$(E_J^{(j)}, J^{(j)}) \quad := \quad \left([r_J^{(j)}]X^{(j)}, [r_J^{(j)}]G + J^{(j+1)}\right)$$

The rest of the data are introduced to make the procedure *Enc* IND-CCA2 secure [119] by using the technique presented in [142]. The data $P_K^{(j)}, P_H^{(j)}$, and $P_J^{(j)}$ are twin encryption parts of the above public-key encryptions and the data $(c^{(j)}, t_K^{(j)}, t_H^{(j)}, t_J^{(j)})$ is a non-interactive proof of these twin encryptions.

$$\begin{aligned} &(F_K^{(j)}, F_H^{(j)}, F_J^{(j)}) \\ &\quad := \quad ([s_K^{(j)}]X^{(j)}, [s_H^{(j)}]X^{(j)}, [s_J^{(j)}]X^{(j)}) \end{aligned}$$

$$
\begin{aligned}
Y_K^{(j)} &:= \mathcal{H}_E(\mathcal{D}, \chi^{(j)}, X^{(j)}, E_K^{(j)}, D_K^{(j)}, E_H^{(j)}, \\
&\quad D_H^{(j)}, E_J^{(j)}, J^{(j)}, F_K^{(j)}, F_H^{(j)}, F_J^{(j)}, \texttt{key}) \\
Y_H^{(j)} &:= \mathcal{H}_E(\mathcal{D}, \chi^{(j)}, X^{(j)}, E_K^{(j)}, D_K^{(j)}, E_H^{(j)}, \\
&\quad D_H^{(j)}, E_J^{(j)}, J^{(j)}, F_K^{(j)}, F_H^{(j)}, F_J^{(j)}, \texttt{hsh}) \\
Y_J^{(j)} &:= \mathcal{H}_E(\mathcal{D}, \chi^{(j)}, X^{(j)}, E_K^{(j)}, D_K^{(j)}, E_H^{(j)}, \\
&\quad D_H^{(j)}, E_J^{(j)}, J^{(j)}, F_K^{(j)}, F_H^{(j)}, F_J^{(j)}, \texttt{jnt}) \\
&(P_K^{(j)}, P_H^{(j)}, P_J^{(j)}) \\
&:= ([r_K^{(j)}]Y_K^{(j)}, [r_H^{(j)}]Y_H^{(j)}, [r_J^{(j)}]Y_J^{(j)}) \\
&(Q_K^{(j)}, Q_H^{(j)}, Q_J^{(j)}) \\
&:= ([s_K^{(j)}]Y_K^{(j)}, [s_H^{(j)}]Y_H^{(j)}, [s_J^{(j)}]Y_J^{(j)}) \\
c^{(j)} &:= \mathcal{H}_q(\mathcal{D}, \chi^{(j)}, X^{(j)}, E_K^{(j)}, D_K^{(j)}, \\
&\quad E_H^{(j)}, D_H^{(j)}, E_J^{(j)}, J^{(j)}, Y_K^{(j)}, \\
&\quad Y_H^{(j)}, Y_J^{(j)}, P_K^{(j)}, P_H^{(j)}, P_J^{(j)}, \\
&\quad F_K^{(j)}, Q_K^{(j)}, F_H^{(j)}, Q_H^{(j)}, F_J^{(j)}, Q_J^{(j)}) \\
&(t_K^{(j)}, t_H^{(j)}, t_J^{(j)}) \\
&:= (s_K^{(j)} - c^{(j)} r_K^{(j)}, s_H^{(j)} - c^{(j)} r_H^{(j)}, \\
&\qquad s_J^{(j)} - c^{(j)} r_J^{(j)}) \bmod q
\end{aligned}
$$

Here, $\texttt{key}, \texttt{hsh}$, and $\texttt{jnt}$ are strings. Then *Enc* outputs a ciphertext

$$
\begin{aligned}
\mu^{(j-1)} &:= \{\chi^{(j)}, c^{(j)}, E_K^{(j)}, D_K^{(j)}, E_H^{(j)}, \\
&\quad D_H^{(j)}, E_J^{(j)}, J^{(j)}, P_K^{(j)}, P_H^{(j)}, \\
&\quad P_J^{(j)}, t_K^{(j)}, t_H^{(j)}, t_J^{(j)}) \qquad \in \mathcal{C}^{(j)}.
\end{aligned} \tag{3.2}
$$

*MEnc***:** A message $M \in \mathcal{M}$ and a public key $\{X^{(j)}\}_{j=1,\ldots,m}$ are given. *MEnc* first sets $J^{(m+1)} = \mathcal{O}$ and $\mu^{(m)} = (M, J^{(m+1)})$. Next, *MEnc* repeatedly invokes *Enc* m times to generate

$$
\begin{aligned}
\mu^{(0)} &:= \mathit{Enc}(X^{(1)}, \mathit{Enc}(\cdots \\
&\qquad \mathit{Enc}(X^{(m-1)}, \mathit{Enc}(X^{(m)}, \mu^{(m)}))\cdots))
\end{aligned}
$$

Next, *MEnc* proves the knowledge of sum of the randomness $\sum_{j=1}^{m} r_J^{(j)}$ used in the all invocations.

For that purpose, it randomly chooses $s_J^{(0)} \in_R \mathbb{Z}/q\mathbb{Z}$ and generates

$$\begin{aligned}
R_J^{(0)} := [s_J^{(0)}]G \quad &, \quad G_J := \mathcal{H}_E(\mathcal{D}, \mu^{(0)}, R_J^{(0)}), \\
R_J := [s_J^{(0)}]G_J \quad &, \quad J := [\sum_{j=1}^{m} r_J^{(j)}]G_J, \\
c^{(0)} := &\quad \mathcal{H}_q(\mathcal{D}, \mu^{(0)}, G_J, J, R_J^{(0)}, R_J, \mathcal{U}), \\
t_J^{(0)} := &\quad s_J^{(0)} - c^{(0)} \sum_{j=1}^{m} r_J^{(j)} \bmod q
\end{aligned}$$

where $\mathcal{U}$ is an identity of user. Finally, *MEnc* outputs a ciphertext

$$C \;:=\; (\mu^{(0)}, J, c^{(0)}, t_J^{(0)}) \in \mathcal{C} \tag{3.3}$$

*Dec***:** The secret key $x^{(j)}$ and ciphertext $\mu^{(j-1)} \in \mathcal{C}^{(j)}$ are given. We assume here $\mu^{(j-1)}$ are parsed as Eq.(3.2). *Dec* first computes

$$\begin{aligned}
&(F_K^{(j)}, F_H^{(j)}, F_J^{(j)}) \\
&\quad = \quad ([t_K^{(j)}]X^{(j)} + [c^{(j)}]E_K^{(j)}, \\
&\qquad\qquad [t_H^{(j)}]X^{(j)} + [c^{(j)}]E_H^{(j)}, \\
&\qquad\qquad [t_J^{(j)}]X^{(j)} + [c^{(j)}]E_J^{(j)}) \\
Y_K^{(j)} &= \quad \mathcal{H}_E(\mathcal{D}, \chi^{(j)}, X^{(j)}, E_K^{(j)}, D_K^{(j)}, E_H^{(j)}, \\
&\qquad D_H^{(j)}, E_J^{(j)}, J^{(j)}, F_K^{(j)}, F_H^{(j)}, F_J^{(j)}, \texttt{key}) \\
Y_H^{(j)} &= \quad \mathcal{H}_E(\mathcal{D}, \chi^{(j)}, X^{(j)}, E_K^{(j)}, D_K^{(j)}, E_H^{(j)}, \\
&\qquad D_H^{(j)}, E_J^{(j)}, J^{(j)}, F_K^{(j)}, F_H^{(j)}, F_J^{(j)}, \texttt{hsh}) \\
Y_J^{(j)} &:= \quad \mathcal{H}_E(\mathcal{D}, \chi^{(j)}, X^{(j)}, E_K^{(j)}, D_K^{(j)}, E_H^{(j)}, \\
&\qquad D_H^{(j)}, E_J^{(j)}, J^{(j)}, F_K^{(j)}, F_H^{(j)}, F_J^{(j)}, \texttt{jnt})
\end{aligned}$$

Next, *Dec* verifies if

$$\begin{aligned}
c^{(j)} = \quad & \mathcal{H}_q(\mathcal{D}, \chi^{(j)}, X^{(j)}, E_K^{(j)}, D_K^{(j)}, \\
& E_H^{(j)}, D_H^{(j)}, E_J^{(j)}, J^{(j)}, Y_K^{(j)}, \\
& Y_H^{(j)}, Y_J^{(j)}, P_K^{(j)}, P_H^{(j)}, P_J^{(j)}, \\
& F_K^{(j)}, [t_K^{(j)}]Y_K^{(j)} + [c^{(j)}]P_K^{(j)},
\end{aligned}$$

$$\begin{aligned}&F_H^{(j)}, [t_H^{(j)}]Y_H^{(j)} + [c^{(j)}]P_H^{(j)},\\ &F_J^{(j)}, [t_J^{(j)}]Y_J^{(j)} + [c^{(j)}]P_J^{(j)})\end{aligned} \tag{3.4}$$

hold. If not, *Dec* outputs $\perp_1^{(j)}$ and stops. Next, *Dec* computes

$$K^{(j)\dagger} := D_K^{(j)} - [1/x^{(j)}]E_K^{(j)} \tag{3.5}$$

$$H^{(j)\dagger} := D_H^{(j)} - [1/x^{(j)}]E_H^{(j)} \tag{3.6}$$

$$J^{(j+1)\dagger} := J^{(j)} - [1/x^{(j)}]E_J^{(j)} \tag{3.7}$$

$$\kappa^{(j)} := \mathcal{H}_\sigma(K^{(j)\dagger}) \tag{3.8}$$

$$\mu^{(j)} := \mathrm{dec}_{\kappa^{(j)}}(\chi^{(j)}) \tag{3.9}$$

and outputs $H^{(j)\dagger}$ and $J^{(j+1)\dagger}$. Next, if either of the following equations does not hold,

$$H^{(j)\dagger} = \mathcal{H}_E(\mu^{(j)}) \ , \ \ J^{(j+1)\dagger} = J^{(j+1)} \in \mu^{(j)} \tag{3.10}$$

Dec outputs $K^{(j)\dagger}$ and $\perp_2^{(j)}$ and stops. Finally, *Dec* outputs $\mu^{(j)} \in \mathcal{M}^{(j)}$.

MDec: A ciphertext $C \in \mathcal{C}$ and the secret key $\{x^{(j)}\}_{j=1,\dots,m}$ are given. *MDec* first computes

$$\begin{aligned} R_J^{(0)} &= [t_J^{(0)}]G + [c^{(0)}]J^{(1)},\\ G_J &= \mathcal{H}_E(\mathcal{D}, \mu^{(0)}, R_J^{(0)}) \end{aligned}$$

and verifies if

$$\begin{aligned} c^{(0)} = \quad & \mathcal{H}_q(\mathcal{D}, \mu^{(0)}, G_J, J,\\ & R_J^{(0)}, [t_J^{(0)}]G_J + [c^{(0)}]J, \mathcal{U}) \end{aligned} \tag{3.11}$$

holds. If not *MDec* outputs $\perp$ and stops. Next, *MDec* recursively invokes *Dec* m-times to generate

$$\begin{aligned} \mu^{(m)} &= (M, \mathcal{O})\\ &= \mathit{Dec}(\mathit{Dec}(\cdots \mathit{Dec}(\mathit{Dec}(\mu^{(0)}))\cdots)). \end{aligned}$$

If any of invocations of *Dec* stops, *MDec* also stops. *MDec* outputs the output of each invoked *Dec*. Finally, *MDec* outputs first elements M of $\mu^{(m)}$ and stops.

Theorem 9 *The proposed multiple encryption scheme is IND-ME-CCA secure and Encryption scheme*

(Key-Gen, Enc, Dec) is IND-CCA2 secure assuming random oracles, intractability of decision Diffie-Hellman problem, and semantic security of the one-time symmetric-key cryptosystem.

Proof.

The following Lemma 18 is repeatedly used.

Lemma 18 *[48] Let S_1, S_2 and F be events defined on some probability space. Suppose that the event $S_1 \vee \bar{F}$ occurs if and only if $S_2 \vee \bar{F}$ occurs. Then*

$$|\Pr[S_1] - \Pr[S_2]| \leq \Pr[F]$$

We define Game 0−6 in the following. Each Game is an interactive computation between an adversary $\mathcal{A}$ and a simulator $\mathcal{S}$.

Definition 11 (Game 0) *Game 0 is simply the usual game used to define IND-ME-CCA security.*

Initially, $\mathcal{S}$ runs MKey-Gen, obtaining a domain parameter $E, G, \ell, \mathcal{H}_q, \mathcal{H}_E, \mathcal{H}_\sigma, (\mathrm{enc}_\kappa, \mathrm{dec}_\kappa)$, a secret key $\{x^{(k)}\}_{j=1,\ldots,m}$, and a public key $\{X^{(k)}\}_{j=1,\ldots,m}$. The simulator gives the domain parameter and the public key to $\mathcal{A}$.

During the execution of the game, $\mathcal{A}$ makes a number of decryption queries. For each query, $\mathcal{S}$ runs MDec to execute decryption. $\mathcal{A}$ may also makes at most $m-1$ key exposure queries. For each query in which $\mathcal{A}$ sends index j, $\mathcal{S}$ sends $x^{(j)}$ to $\mathcal{A}$. $\mathcal{A}$ may also send a pair of plaintext $M_0, M_1 \in \{0,1\}^\ell$ to the challenging oracle once. In such a case, $\mathcal{S}$ randomly chooses $b \in_R \{0,1\}$ and generates target ciphertext C_* by running MEnc. $\mathcal{S}$ gives C_* to $\mathcal{A}$. $\mathcal{A}$ may not make a decryption query with respect to this target ciphertext. $\mathcal{A}$ may also send a query to random oracle $\mathcal{H}_E, \mathcal{H}_q$, or $\mathcal{H}_\sigma$. If the query was not the same to any of previous queries, $\mathcal{S}$ randomly chooses an element respectively from $E, \mathbb{Z}/q\mathbb{Z}$, or $\{0,1\}^{n'}$ and sends it to $\mathcal{A}$. At the end of the game, $\mathcal{A}$ outputs $\hat{b} \in \{0,1\}$. Let X_0 be the event that $\hat{b} = b$.

We also consider Game 1, Game 2, etc. These games will be quite similar to Game 0 in their overall structure, and will only differ from Game 0 in terms of how the simulator works. However, in each game, there will be well defined bits $\hat{b}$ and b, so that in Game i, we always define X_i to the event that $\hat{b} = b$ in that game. The all elements in C_* and its partial decryptions are indexed by $*$. Other elements in C_i and its partial decryptions are indexed by i. Our goal is to prove that $|\Pr[X_0] - 1/2|$ is negligible.

Definition 12 (Game 1) *Game 1 is the same as Game 0, except that $\mathcal{S}$ randomly chooses $j^* \in \{1, \ldots, m\}$ at the beginning of the game and rejects key exposure queries with respect j^*-th key.*

Then, we have

Lemma 19

$$|\Pr[X_0] - 1/2| = m|\Pr[X_1] - 1/2|$$

□

Definition 13 (Game 2) *Game 2 is the same as Game 1 except the followings:*

1. *At the beginning of the game, $\mathcal{S}$ randomly chooses $G_{J*} \in_R E$. $\mathcal{S}$ uses this value in C_*.*

2. *At the beginning of the game, $\mathcal{S}$ randomly chooses $y_{J*}^{(j^*)} \in_R \mathbb{Z}/q\mathbb{Z}$ and computes*

$$Y_{J*}^{(j^*)} = [y_{J*}^{(j^*)}]G.$$

$\mathcal{S}$ uses this value in C_*.

3. *If $\mathcal{A}$ sends*

$$(\mathcal{D}, \chi_i^{(j)}, E_{Ki}^{(j)}, D_{Ki}^{(j)}, E_{Hi}^{(j)}, D_{Hi}^{(j)},$$
$$E_{Ji}^{(j)}, J_i^{(j)}, F_{Ki}^{(j)}, F_{Hi}^{(j)}, F_{Ji}^{(j)}, \texttt{jnt})$$

to $\mathcal{H}_E$ for $j > j^*$, $\mathcal{S}$ randomly chooses $y_{Ji}^{(j)} \in_R \mathbb{Z}/q\mathbb{Z}$ and generates

$$Y_{Ji}^{(j)} = [y_{Ji}^{(j)}]G_{J*}.$$

4. *If $\mathcal{A}$ sends*

$$(\mathcal{D}, \mu_i^{(0)}, R_{Ji}^{(0)})$$

to $\mathcal{H}_E$, $\mathcal{S}$ randomly chooses $y_{Ji}^{(0)} \in_R \mathbb{Z}/q\mathbb{Z}$ and generates

$$G_{Ji} = [y_{Ji}^{(0)}]G_{J*}. \tag{3.12}$$

These are only conceptual modification. Hence

Lemma 20

$$\Pr[X_1] = \Pr[X_2].$$

□

Definition 14 (Game 3) *Game 3 is the same as Game 2, except that $\mathcal{S}$ randomly chooses $G_{J*}^{(j)} \in_R E$ and generates*

$$\begin{aligned}&(E_{J*}^{(j^*)}, P_{J*}^{(j^*)})\\ &:= ([x^{(j^*)}](G_{J*}^{(j^*)} - G_{J*}^{(j^*+1)}), [y_{J*}^{(j^*)}](G_{J*}^{(j^*)} - G_{J*}^{(j^*+1)})).\end{aligned}$$

Then $\mathcal{S}$ generates the rest of $\mu_*^{(j^*-1)}$ so as to be valid by appropriately choosing random oracles.

Since, the distribution of C_* is not modified, we have

Lemma 21

$$\Pr[X_2] = \Pr[X_3].$$

□

Definition 15 (Game 4) *Game 4 is the same as Game 3, except that if $\mathcal{A}$ ever submits decryption query with respect to C_i such that*

$$\mu_i^{(j-1)} = \mu_*^{(j-1)}, \tag{3.13}$$

$\mathcal{S}$ rejects the given ciphertext.

Let F_4 be the event that $\mathcal{S}$ applies this rejection rule to C_i.

Then, we have

Lemma 22

$$|\Pr[X_3] - \Pr[X_4]| \leq \Pr[F_4].$$

Moreover

$$\Pr[F_4] \leq \epsilon_{cdh}(n) + \epsilon_{neg}(n)$$

where $\epsilon_{cdh}(n)$ is the success probability that one can solve Diffie-Hellman problem using resources similar to those of $\mathcal{A}$ and $\epsilon_{neg}(n)$ is a negligible function of n.

Proof. We can easily build a game that takes $(G, G_1, G_2) \in E^3$ as an input and solve Diffie-Hellman problem using $\mathcal{A}$ if $\mathcal{S}$ assigns

$$(G, G_{J*}, J_*^{(j^*)}) \quad = \quad (G, G_1, G_2)$$

The fact that a tuple (A, B, C, D) is a Diffie-Hellman instance is denoted as $(A, B, C, D) \in \mathcal{DH}$. If C_i will not be rejected by from the first to $j^* - 1$-th invocations of *Dec*, then the following tuples found in C_i satisfy

$$(G, G_{Ji}, J_i^{(1)}, J_i) \quad \in \quad \mathcal{DH} \tag{3.14}$$

$$(G, Y_{Ji}^{(j)}, J_i^{(j)} - J_i^{(j+1)}, P_{Ji}^{(j)}) \quad \in \quad \mathcal{DH} \quad j = 1, \ldots, j^* - 1 \tag{3.15}$$

where $J_i^{(j^*)} = J_*^{(j^*)}$. It is clear that if Eq. (3.14) does not hold, the probability that $c^{(0)}$ be the value such that there exists $t_J^{(0)}$ that satisfies Eq. (3.11) is negligible. It is also clear that if

$$(X^{(j)}, Y_{Ji}^{(j)}, E^{(j)}, P_{Ji}^{(j)}) \in \mathcal{DH}$$

does not hold for any $j = 1, \ldots, j^* - 1$, the probability that $c^{(j)}$ be the value such that there exists $t_J^{(j)}$ that satisfies Eq. (3.4) is negligible. From Eqs. (3.1), (3.7), and (3.10),

$$(G, Y_{Ji}^{(j)}, J_i^{(j)} - J_i^{(j+1)}, P_{Ji}^{(j)}) \in \mathcal{DH}.$$

From Eq.(3.12),

$$(G, G_{J*}, J_i^{(j)} - J_i^{(j+1)}, [1/y_{Ji}^{(j)}]P_{Ji}^{(j)}) \in \mathcal{DH}.$$

Taking the sum of these tuples,

$$(G, G_{J*}, J_i^{(1)} - J_*^{(j^*)}, \sum_{j=1}^{j^*-1} [1/y_{Ji}^{(j)}]P_{Ji}^{(j)}) \in \mathcal{DH}. \tag{3.16}$$

From Eqs. (3.14) and (3.12),

$$(G, G_{J*}, J_i^{(1)}, [1/y_{Ji}^{(0)}]J_i) \quad \in \quad \mathcal{DH} \tag{3.17}$$

From Eqs. (3.16) and (3.17),

$$(G, G_{J*}, J_*^{(j^*)}, [1/y_{Ji}^{(0)}]J_i - \sum_{j=1}^{j^*-1}[1/y_{Ji}^{(j)}]P_{Ji}^{(j)}) \in \mathcal{DH}.$$

Therefore, we have solved Diffie-Hellman problem. □

Since $\mathcal{A}$ is not allowed to sends to the decryption oracle C_i such that Eq. (3.13) holds, IND-CCA2 security of encryption scheme of (*Key-Gen*, *Enc*, *Dec*) implies IND-ME-CCA security of the proposed scheme. The following lemmas 23-25 proves IND-CCA2 security of encryption scheme (*Key-Gen*, *Enc*, *Dec*).

Definition 16 (Game 5) *Let $\mathcal{DH}_3$ be all set of*

$$\begin{pmatrix} A & B & C \\ D & E & F \end{pmatrix} \in E^6$$

such that

$$\log_A D = \log_B E = \log_C F.$$

Game 5 is the same as Game 4 except the followings:

1. *$\mathcal{S}$ randomly chooses*

$$\begin{aligned} &G, X^{(j^*)}, \\ &Y_{K*}^{(j^*)}, D_{K*}^{(j^*)}, E_{K*}^{(j^*)}, P_{K*}^{(j^*)} \\ &Y_{H*}^{(j^*)}, D_{H*}^{(j^*)}, E_{H*}^{(j^*)}, P_{H*}^{(j^*)} \\ &Y_{J*}^{(j^*)}, J_*^{(j^*)}, E_{J*}^{(j^*)}, P_{J*}^{(j^*)} \end{aligned}$$

under the condition that

$$\begin{pmatrix} G & X^{(j^*)} & Y_{K*}^{(j^*)} \\ D_{K*}^{(j^*)} - K_*^{(j^*)} & E_{K*}^{(j^*)} & P_{K*}^{(j^*)} \end{pmatrix} \in \mathcal{DH}_3$$
$$\begin{pmatrix} G & X^{(j^*)} & Y_{H*}^{(j^*)} \\ D_{H*}^{(j^*)} - H_*^{(j^*)} & E_{H*}^{(j^*)} & P_{H*}^{(j^*)} \end{pmatrix} \in \mathcal{DH}_3$$
$$\begin{pmatrix} G & X^{(j^*)} & Y_{J*}^{(j^*)} \\ J_*^{(j^*)} - J_*^{(j^*+1)} & E_{J*}^{(j^*)} & P_{J*}^{(j^*)} \end{pmatrix} \in \mathcal{DH}_3$$

2. *If $\mathcal{A}$ sends*

$$(\mathcal{D}, \chi_i^{(j^*)}, E_{Ki}^{(j^*)}, D_{Ki}^{(j^*)}, E_{Hi}^{(j^*)}, D_{Hi}^{(j^*)}, E_{Ji}^{(j^*)}, J_i^{(j^*)}, \texttt{key})$$

to $\mathcal{H}_E$, $\mathcal{S}$ randomly chooses $y_{Ki}^{(j^*)} \in_R \mathbb{Z}/q\mathbb{Z}$ and generates

$$Y_{Ki}^{(j^*)} = [y_{Ki}^{(j^*)}]G.$$

Let us define

$$y_{Hi}^{(j^*)}, y_{Ji}^{(j^*)}, Y_{Hi}^{(j^*)}, Y_{Ji}^{(j^*)}$$

in the same way.

3. *$\mathcal{S}$ randomly chooses*

$$\begin{aligned}&(E_{K*}^{(j^*)}, E_{H*}^{(j^*)}, E_{J*}^{(j^*)}, P_{K*}^{(j^*)}, P_{H*}^{(j^*)}, P_{J*}^{(j^*)}) \in_R E^6,\\ &(c_*^{(j^*)}, t_{K*}^{(j^*)}, t_{H*}^{(j^*)}, t_{J*}^{(j^*)}) \in_R (\mathbb{Z}/q\mathbb{Z})^4\end{aligned}$$

and generates

$$(F_{K*}^{(j)}, Q_{K*}^{(j)}, F_{H*}^{(j)}, Q_{H*}^{(j)}, F_{J*}^{(j)}, Q_{J*}^{(j)})$$

so that Eq. (3.4) is satisfied by choosing random oracles.

4. *$\mathcal{S}$ rejects decryption of C_i if any of the following equations holds:*

$$Y_{Ki}^{(j^*)} = Y_{K*}^{(j^*)}, y_{Hi}^{(j^*)} = Y_{H*}^{(j^*)}, Y_{Ji}^{(j^*)} = Y_{J*}^{(j^*)} \tag{3.18}$$

Let F_5 be the event that $\mathcal{S}$ applies this rejection rule to C_i.

5. *For computing $K_i^{(j)\dagger}, H_i^{(j)\dagger}$, and $J_i^{(j+1)\dagger}$, $\mathcal{S}$ computes as*

$$\begin{aligned}K_i^{(j^*)\dagger} &= D_K^{(j^*)} - [1/y_{Ji}^{(j^*)}]P_K^{(j^*)}\\ H_i^{(j^*)\dagger} &= D_H^{(j^*)} - [1/y_{Ji}^{(j^*)}]P_H^{(j^*)}\\ J_i^{(j^*+1)\dagger} &= J_i^{(j^*)} - [1/y_{Ji}^{(j^*)}]P_{Ji}^{(j^*)}\end{aligned}$$

The distribution of elements in C_* in this game are exactly the same to those in Game 4. It should be noted that the above construction is always possible for any given $\mu^{(j^*)}$, which allows adversary of IND-CCA2 game to choose any target ciphertext. Hence, we have

Lemma 23

$$|\Pr[X_4] - \Pr[X_5]| \le \Pr[F_5] \le \epsilon_{neg}(n).$$

Proof. If equation

$$\begin{aligned}
&(\chi_i^{(j^*)}, E_{Ki}^{(j^*)}, D_{Ki}^{(j^*)}, E_{Hi}^{(j^*)}, D_{Hi}^{(j^*)},\\
&E_{Ji}^{(j^*)}, J_i^{(j^*)}, F_{Ki}^{(j^*)}, F_{Hi}^{(j^*)}, F_{Ji}^{(j^*)})\\
&= (\chi_*^{(j^*)}, E_{K*}^{(j^*)}, D_{K*}^{(j^*)}, E_{H*}^{(j^*)}, D_{H*}^{(j^*)},\\
&E_{J*}^{(j^*)}, J_*^{(j^*)}, F_{K*}^{(j^*)}, F_{H*}^{(j^*)}, F_{J*}^{(j^*)}),
\end{aligned} \tag{3.19}$$

does not hold, the probability that any of Eqs.(3.18) holds is negligible.

Suppose that Eq. (3.19) holds, then, if

$$\begin{aligned}
&(P_{Ki}^{(j^*)}, P_{Hi}^{(j^*)}, P_{Ji}^{(j^*)}, Q_{Ki}^{(j^*)}, Q_{Hi}^{(j^*)}, Q_{Ji}^{(j^*)})\\
&= (P_{K*}^{(j^*)}, P_{H*}^{(j^*)}, P_{J*}^{(j^*)}, Q_{K*}^{(j^*)}, Q_{H*}^{(j^*)}, Q_{J*}^{(j^*)})
\end{aligned}$$

does not hold, i.e., $C_i \neq C_*$, the probability that randomly chosen $c_i^{(j^*)}$ be such that there exists $(t_{Ki}^{(j^*)}, t_{Hi}^{(j^*)}, t_{Ji}^{(j^*)})$ satisfying Eq. (3.4) is negligible. Therefore

$$\Pr[F_5] \le \epsilon_{neg}(n).$$

□

Definition 17 (Game 6) *Game 6 is the same as Game 5 except $\mathcal{S}$ randomly chooses*

$$\begin{aligned}
&G, X^{(j^*)},\\
&Y_{K*}^{(j^*)}, D_{K*}^{(j^*)}, E_{K*}^{(j^*)}), P_{K*}^{(j^*)}\\
&Y_{H*}^{(j^*)}, D_{H*}^{(j^*)}, E_{H*}^{(j^*)}), P_{H*}^{(j^*)}\\
&Y_{J*}^{(j^*)}, J_*^{(j^*)}, E_{J*}^{(j^*)}), P_{J*}^{(j^*)}
\end{aligned}$$

without any condition.

Then, we have

Lemma 24

$$|\Pr[X_5] - \Pr[X_6]| < \epsilon_{ddh}(n)$$

where $\epsilon_{ddh}(n)$ is the success probability that one can solve decision Diffie-Hellman problem using resources similar to those of $\mathcal{A}$.

Proof. We can easily build a game that takes $(G_1, G_2, G_3, G_4) \in E^4$ as an input and decide whether input is in $\mathcal{DH}$ or not using $\mathcal{A}$, which is solving decision Diffie-Hellman problem. Let us randomly choose $u_{11}, u_{12}, u_{21}, u_{22}, u_{23}, u_{24}, u_{31}, u_{32}, u_{33}, u_{34}$ from $\mathbb{Z}/q\mathbb{Z}$ and assign

$$\begin{pmatrix} G & X^{(j^*)} & Y^{(j^*)}_{K*} \\ D^{(j^*)}_{K*} - K^{(j^*)}_{*} & E^{(j^*)}_{K*}) & P^{(j^*)}_{K*} \end{pmatrix}$$
$$= \begin{pmatrix} G_1 & G_2 & [u_{11}]G_1 + [u_{12}]G_2 \\ G_3 & G_4 & [u_{11}]G_3 + [u_{12}]G_4 \end{pmatrix}$$
$$\begin{pmatrix} G & X^{(j^*)} & Y^{(j^*)}_{H*} \\ D^{(j^*)}_{H*} - H^{(j^*)}_{*} & E^{(j^*)}_{H*}) & P^{(j^*)}_{H*} \end{pmatrix}$$
$$= \begin{pmatrix} G_1 & G_2 & [u_{21}]G_1 + [u_{22}]G_2 \\ [u_{23}]G_1 + [u_{24}]G_3 & [u_{23}]G_2 + [u_{24}]G_4 & [u_{21}u_{23}]G_1 + [u_{21}u_{24}]G_3 + [u_{22}u_{23}]G_2 + [u_{22}u_{24}]G_4 \end{pmatrix}$$
$$\begin{pmatrix} G & X^{(j^*)} & Y^{(j^*)}_{J*} \\ J^{(j^*)}_{*} - J^{(j^*+1)}_{*} & E^{(j^*)}_{J*}) & P^{(j^*)}_{J*} \end{pmatrix}$$
$$= \begin{pmatrix} G_1 & G_2 & [u_{31}]G_1 + [u_{32}]G_2 \\ [u_{33}]G_1 + [u_{34}]G_3 & [u_{33}]G_2 + [u_{34}]G_4 & [u_{31}u_{33}]G_1 + [u_{31}u_{34}]G_3 + [u_{32}u_{33}]G_2 + [u_{32}u_{34}]G_4 \end{pmatrix}$$

in Game 6. Then, distribution of output of this game is exactly the same to that of Game 5 if $(G_1, G_2, G_3, G_4) \in \mathcal{DH}$ and is exactly the same to that of Game 6 if (G_1, G_2, G_3, G_4) is randomly chosen from E^4.

$\square$

Now, we have

Lemma 25

$$|\Pr[X_6] - 1/2| < \epsilon_{enc}(n')$$

where $\epsilon_{enc}(n')$ *is the success probability that one can break semantic security of one-time encryption scheme* $(\mathrm{enc}_\kappa, \mathrm{dec}_\kappa)$ *using resources similar to those of* $\mathcal{A}$. □

From Lemmas 23-25,

$$|\Pr[X_4] - 1/2| \leq \epsilon_{neg}(n) + \epsilon_{ddh}(n) + \epsilon_{enc}(n')$$

which implies IND-CCA2 security of hybrid encryption scheme used in the proposed multiple encryption scheme.

From Lemmas 19-25,

$$|\Pr[X_0] - 1/2| \leq m(\epsilon_{cdh}(n) + 2\epsilon_{neg}(n) + \epsilon_{ddh}(n) + \epsilon_{enc}(n'))$$

which implies IND-ME-CCA security of the proposed multiple encryption scheme. □

The possible output of *Dec* and *MDec* are either results of decryption or one of the set of symbols $\{\perp, \perp_1^{(j)}, \perp_2^{(j)}\}$. Each symbol represents that the input ciphertext was improperly generated. In order to prove the validity of the decryption result or these symbols, we introduce interactive protocols *Ver* and *MVer*. *Ver* is performed between a prover $\mathcal{P}$ and a verifier $\mathcal{V}$ to prove the validity of the output of *Dec*, and *MVer* is performed between a prover $\mathcal{MP}$ and a verifier $\mathcal{MV}$ to prove the validity of the output of *MDec*.

Ver: $\mathcal{P}$ and $\mathcal{V}$ are given $\mu^{(j-1)} \in \mathcal{C}^{(j)}$ and the output *Out* of *Dec*. $\mathcal{P}$ is also given $x^{(j)}$. Suppose $Out = \perp_1$. Then, $\mathcal{V}$ evaluates Eq.(3.4). If this holds, $\mathcal{V}$ rejects $\mathcal{P}$ and stops. Otherwise $\mathcal{V}$ accepts $\mathcal{P}$ and stops.

Suppose that $Out \neq \perp_1$. $\mathcal{V}$ first checks if Eq.(3.4) holds or not. If it does not hold, $\mathcal{V}$ rejects $\mathcal{P}$ and stops. Next, $\mathcal{P}$ proves to $\mathcal{V}$ in zero-knowledge the knowledge of $x^{(j)}$ satisfying Eqs, (3.1), (3.6), and (3.7). $\mathcal{V}$ rejects $\mathcal{P}$ and stops if this proof in unacceptable. Next, if the output of *Dec* was $\perp_2$, $\mathcal{P}$ additionally proves to $\mathcal{V}$ in zero-knowledge the knowledge of $x^{(j)}$ satisfying Eqs, (3.1) and (3.5). $\mathcal{V}$ rejects $\mathcal{P}$ and stops if this proof in unacceptable. Finally, $\mathcal{V}$ accepts $\mathcal{P}$ and stops if neither of the above proofs are unacceptable.

MVer $\mathcal{MP}$ and $\mathcal{MV}$ are given a ciphertext $C \in \mathcal{C}$ and the output *Out* of *MDec*. $\mathcal{MP}$ is given secret keys $\{x^{(j)}\}_{j=1,\ldots,m}$. Suppose $Out = \perp$. Then, $\mathcal{MV}$ evaluates Eq. (3.11). If this holds, $\mathcal{MV}$ rejects $\mathcal{MP}$ and stops. Otherwise $\mathcal{MV}$ accepts $\mathcal{MP}$ and stops.

Suppose *Out* $\neq \perp$. $\mathcal{MV}$ first checks if Eq. (3.11) holds or not. If it does not, $\mathcal{MV}$ rejects $\mathcal{MP}$ and stops. Next, $\mathcal{MP}$ plays the role of $\mathcal{P}$ in all *Ver* with respect to all *Dec* invoked by *MDec*. If any of $\mathcal{V}$ rejects $\mathcal{P}$ in *Ver*, $\mathcal{MV}$ rejects $\mathcal{MP}$. Otherwise, $\mathcal{MV}$ accepts $\mathcal{MP}$.

From the construction, one can easily observe that the following theorems hold.

Theorem 10 *Suppose that $C \in \mathcal{C}$ is an encryption of $M \in \mathcal{M}$ generated by honestly following Algorithm MEnc. $\mathcal{MP}$ will be accepted by $\mathcal{MV}$ in MVer when M is input to $\mathcal{MP}$ and $\mathcal{MV}$ as the output of MDec. Here, we assume random oracles.* □

Theorem 11 *No polynomial time adversary is able to output $(C, M', M) \in \mathcal{C} \times (\mathcal{M})^2$ such that $M' \neq M$ and the adversary will be accepted by $\mathcal{MV}$ as $\mathcal{MP}$ in MVer with non negligible probability both when they are given M and when they are given M'. Here, we assume random oracles.* □

Remark 1 *As Theorem 3 assures, no ciphertext can be decrypted in two valid messages, no matter how the ciphertext is generated. But our scheme can not prevent an adversary to generate a ciphertext that can be treated in two ways by a malicious decryptor: either it is decrypted to a 'correct' message, or claimed invalid(i.e. malicious decryptor outputs $\perp_2$)*

The strategy for the adversary and the corrupted j-th decrypter to generate such ciphertext is as follows. The adversary executes multiple encryption up to $m-j+1$-th iteration and obtain $\chi^{(j)} \in \mu^{(j-1)}$. But then he starts the next iteration starting with a different data, say $\chi^{(j)*}$, and process correctly then. Thus generated ciphertext will be processed without any problem until the j-th iteration of decryption. If j-th decryptor is honest, he will output $\perp_2$ and $K^{(j)}$ and show that the decryption of $\mathcal{H}_E(\mathrm{dec}_{\mathcal{H}_\sigma(K^{(j)})}(\chi^{(j*)}))$ does not coincide with $H^{(j)\dagger} = \mathcal{H}_E(\mathrm{dec}_{\mathcal{H}_\sigma(K^{(j)})}(\chi^{(j)}))$. A malicious j-th decryptor has a choice, whether to honestly claim the ciphertext is malicious, or to secretly replace $\chi^{(j)*}$ with $\chi^{(j)}$ obtained from the adversary and continue the procedure assuming nothing has happened. The semantic advantage of this attack is the adversary can decide his input to be either valid message or invalid, not at the submission of the encrypted input but in the midst of decryption, when the colluding decryptor proceeds.

3.3 Publicly Verifiable Mix-net under Hybrid Construction

Now we will present our publicly verifiable mix-net that can mix long messages. The idea of the construction is as follows: The input to the mix-net will be a list of ciphertext, where each ciphertext is generated by a user following the multiple encryption scheme proposed in Section 3.2. Within a mix-net, each mixer performs *Dec* to each ciphertext in the input list. He then permute the list of the decrypted data which will be the input list to the next mixer. Thus the mixers comprise *MDec* of the proposed multiple encryption scheme but the correspondence between the input ciphertext and the output message is hidden due to the permutation.

In Section 3.2, we have shown that the decryption can be done verifiably. In this section, we present how to do so without revealing the permutation. The way we achieved the verifiability of decryption is by adding an encryption of the hash value of resulting decryption in the input ciphertext. If we compare the set of encryptions of the hashed values in the input list and the set of values where hash function is applied to the decrypted messages, the latter should be a set of decryptions of the former set. Therefore, by performing the zero-knowledge argument of shuffle-and-decrypt on these sets, we can prove that the output list is a correct decryption of the input list.

We note that [86] provides 7-move zero-knowledge argument of shuffle-and-decrypt. We provide in Section 3.5 an alternative scheme which provide 3-move zero-knowledge argument but the detail is omitted for space limitations.

Our mix-net satisfies anonymity property defined in [3], whose definition is presented in the following:

Definition 18 (Anonymity)[3]. *Let $\mathcal{A}$ be an adversary that plays the following game. At any moment of the game, $\mathcal{A}$ is allowed to corrupt up to t_u users and t_m servers. Once corrupted, the user or the server is thoroughly controlled by $\mathcal{A}$.*

1. *$(y, x) \leftarrow G_{mix}(1^n)$. Public-key y and each shared decryption key x_i of x is given to j-th mixer $\mathcal{S}^{(j)}$.*

2. *$\mathcal{A}$ is given y and allowed to invoke any mixer $\mathcal{S}^{(j)}$ an arbitrary number of times for arbitrary chosen input ciphertexts (i.e., $\mathcal{A}$ can use any of $\{\mathcal{S}^{(j)}\}_j$ as a decryption oracle).*

3. *$\mathcal{A}$ outputs $L_C = (\mu_1, \ldots, \mu_k)$ that is a list of messages chosen from message space $\mathcal{M}$ (depends on y).*

4. *Choose a permutation π of $\{1, \ldots, k\}$. Each U_i is given $\mu_{\pi(i)}$ privately and outputs ciphertext C_i. If U_i is corrupted and outputs nothing, let C_i be an empty string. Let $\mathbf{C} = \{C_1, \ldots, C_k\}$.*

5. *The set of $\{\mathcal{S}^{(j)}\}_j$ performs mix processing on $\mathbf{C}$.*

6. *$\mathcal{A}$ is again allowed to invoke any of $\{\mathcal{S}^{(j)}\}_j$ an arbitrary number of times for arbitrarily chosen input ciphertexts except for the ones included in $\mathbf{C}$.*

7. *$\mathcal{A}$ outputs $(i^*, j^*) \in \{1, \ldots, k\}$. The restriction is that $U_{i^*} \notin U_A$ (i.e., U_{i^*} has never been corrupted).*

$\mathcal{A}$ wins the game if $\pi(i^*) = j^*$. Mix-net is anonymous against (t_u, t_m)-limited adaptive and active adversary $\mathcal{A}$ if the probability that any polynomial-time (t_u, t_m)-limited adaptive and active adversary $\mathcal{A}$

wins the above game is at most $\frac{1}{k-t_u} + \epsilon(n)$ *where* $\epsilon(n)$ *is negligible in* n. *Probability is taken over the coin flips of* $G_{mix}, \{U_i\}_{i=1,\ldots,k}, \{\mathcal{S}^{(j)}\}_{j=1,\ldots,m}, \mathcal{A}$ *and the choice of* π.

3.3.1 Proposed Mix-net

Players of our mix-net are m mixers $\{\mathcal{S}^{(j)}\}_{j=1,\ldots,m}$, n users $\{\mathcal{U}_i\}_{i=1,\ldots,k}$, and a verifier $\mathcal{V}$. The scheme is composed of the following steps: (1) Setup, (2) Public-Key Generation, (3) Message Encryption, and (4) Shuffle and Prove. We assume that the input to a mixer $\mathcal{S}^{(j)}$, which is an output of the previous mixer, has been publicly verified of its correctness. If this assumption is not appropriate, $\mathcal{S}^{(j)}$ needs to verify previous Mixers $\{\mathcal{S}^{(h)}\}_{h<j}$.

Setup: In Setup, domain parameters of the scheme are determined. They are a security parameter n, an elliptic curve E of prime order q, a randomly chosen generator G of the curve E, the length ℓ of the messages, a semantically secure one-time symmetric-key cryptosystem $(\mathrm{enc}_\kappa, \mathrm{dec}_\kappa)$, and cryptographic hash functions $\mathcal{H}_E, \mathcal{H}_q$, and $\mathcal{H}_\sigma$.

Public-key Generation: Given the domain parameters, each server $\mathcal{S}^{(j)}$ generates its own secret key $x^{(j)}$ and the corresponding public-key $X^{(j)}$ as described in Section 3.2.

Message Encryption: Each user $\mathcal{U}_i$ encrypts the message M_i of length ℓ following the encryption scheme proposed in Section 3.2, and sends the ciphertext C_i to $\mathcal{S}^{(1)}$ with a signature of $\mathcal{U}_i$.

Shuffle and Prove: Suppose $\mathcal{S}^{(1)}$ is given $\{C_i\}_{i=1,\ldots,k}$. Mixers $\{\mathcal{S}^{(j)}\}$ collaboratively decrypts all multiply encrypted ciphertexts $\{C_i\}_{i=1,\ldots,k}$ as in the following. We assume every ciphertext in $\{C_i\}$ and $\{\mu_i^{(j)}\}$ are parsed as Eqs. (3.3) and (3.2).

1. $\mathcal{S}^{(1)}$ verifies the validity of each signature attached to $\{C_i\}_{i=1,\ldots,k}$. $\mathcal{S}^{(1)}$ also verifies that each $c_i^{(0)} \in C_i$ include correct $\mathcal{U}_i$. From the all elements $\mu_i^{(0)}$ in C_i that are accepted in the above verification, $\mathcal{S}^{(1)}$ generates the the set $\{\mu_i^{(0)}\}_{i=1,\ldots,k^{(1)}}$. Here and in the following $k^{(j)}$ is the number of ciphertexts given to $\mathcal{S}^{(j)}$.

2. From $j = 1$ to $j = m$ do the following:

   (i) Each $\mathcal{S}^{(j)}$ receives a set $\{\mu_i^{(j)}\}_{i=1,\ldots,k^{(j)}}$.

   (ii) $\mathcal{S}^{(j)}$ deletes μ_i which does not satisfy Eq.(3.4) from the set.

   (iii) $\mathcal{S}^{(j)}$ continue decryption of each ciphertext in the set as Eqs. (3.5), (3.6), (3.7), (3.8), and (3.9).

(iv) For each ciphertext μ_i that does not satisfy Eqs. (3.10), $\mathcal{S}^{(j)}$ reveals $H^{(j)\dagger}$, $J^{(j+1)\dagger}$, and $K^{(j)\dagger}$ as *Dec* does. Then $\mathcal{S}^{(j)}$ proves to $\mathcal{V}$ in zero-knowledge the knowledge of $x^{(j)}$ satisfying (3.1), (3.5), (3.6), and (3.7) as in *Ver* does.

(v) From ciphertexts μ_i's that satisfy Eqs. (3.10), $\mathcal{S}^{(j)}$ generates a set $\{\mu_i\}_{i=1,\dots,k^{(j+1)}}$. Here indices are reallocated. $\mathcal{S}^{(j)}$ randomly permutes the order of the corresponding $\{\mu_i^{(j+1)}, H_i^{(j)\dagger}, J_i^{(j+1)\dagger}\}_{i=1,\dots,k^{(j+1)}}$ and reallocates their indices. $\mathcal{S}^{(j)}$ proves to $\mathcal{V}$ in zero-knowledge, using the argument proposed in Section 3.5 or in [86], the knowledge of $x^{(j)}$ and a permutation of $\{1,\dots,k^{(j+1)}\}$ that prove the validity of generating a set $\{H_i^{(j)\dagger}, J_i^{(j+1)\dagger}\}_{i=1,\dots,k^{(j+1)}}$ from $\{\mu_i\}_{i=1,\dots,k^{(j+1)}}$.

3. $\{M_i\}_{i=1,\dots,k^{(m)}}$ output by $\mathcal{S}^{(m)}$ is the output of the mix-net.

Theorem 12 *The proposed mix-net is anonymous assuming random oracles, intractability of decision Diffie-Hellman problem, and semantic security of the one-time symmetric-key cryptosystem.*

Proof. First of all, $\mathcal{A}$ is unable to obtain any knowledge from interaction in which $S^{(j)}$ proves the correctness of decryption of public-key encryption parts. This is because all of these interactions are zero-knowledge. Note that in the proposed mix-net, we only consider the case when the public key M_0 used for shuffle and the public key Y corresponding to the secret key used for decryption are the same. In such a case, we assume that $\{G'_i\}_{i=1,\dots,k}$ is no longer included in the result of shuffle-and-decryption. However, we include them in the first message of our argument. This inclusion does not prevent the argument from being zero-knowledge. This is because, since $\{G'_i \in E\}_{i=1,\dots,k}$ is randomly distributed, the simulator of the argument is able to perfectly simulate it by randomly choosing it from E^k.

The following correspondences between abilities of an adversary $\mathcal{A}$ against anonymity of mix-net and those of an adversary $\mathcal{A}'$ against IND-ME-CCA security exists.

1. Since each ciphertext C_i includes identities of user who have generated it, $\mathcal{A}$ is unable to submit a copy of C_i with the signature of other users. This restriction corresponds to the restriction that $\mathcal{A}'$ is not allowed to send target ciphertext to decryption oracle.

2. An ability of $\mathcal{A}$ to corrupt up to $m-1$ mixers corresponds to the ability of $\mathcal{A}'$ to invoking the key exposure oracle for $m-1$ times.

3. An ability of $\mathcal{A}$ to choose L_C corresponds to the ability of $\mathcal{A}'$ to choose target ciphertexts.

4. An ability of $\mathcal{A}$ to invoke mix-net for the arbitrary number of times corresponds to the ability of $\mathcal{A}'$ to invoke decryption oracle for an arbitrary number of times.

Hence, $\mathcal{A}$ is unable to send a copy of $\mu_i^{(j^*+1)}$, which is a partial decryption of $C_i \in \mathbf{C}$. Therefore, IND-CCA2 security of (*Key-Gen*, *Enc*, *Dec*) implies anonymity of the proposed mix-net.

□

From the property of our hybrid encryption scheme described in Remark 1, our mix-net does not provide full public verifiability unlike the one proposed in [137, 2, 70, 120, 85, 66]. However, it satisfies a stronger notion of robustness than that proposed in [92] as follows: (1) Our scheme allows any verifier to verify the validity of the output to a certain extent whereas that in [92] only allows mixers to verify it. (2) The scheme in [92] allows colluding mixer and user to find two translations (renderings) for a ciphertext. Thus, such colluders are able to arbitrary switch messages during their mixing procedure without being detected. However, our scheme only allows them to switch messages between a valid message and an invalid message as described in Remark 1.

Definition 19 *We say a ciphertext $C \in \mathcal{C}$ is* correct *if MDec, given C, outputs a message in $m \in \mathcal{M}$. We call m as the* correct decryption *of C.*

Definition 20 *Let MDec^f be the same as MDec except that it uses the pair (f, f^{-1}) of a permutation f of strings and its inverse instead of the pair $(\mathrm{enc}_\kappa, \mathrm{dec}_\kappa)$. We say a message $m \in \mathcal{M}$ is the* correct translation *of a ciphertext $C \in \mathcal{C}$ if there exists some f such that MDec^f, given C, outputs m.*

Definition 21 *(Correctness) Let $\mathcal{I}$ be a set of all ciphertexts input to the mix-net and $\mathcal{O}$ be the set of all the output messages of mix-met. We say a $\mathcal{O}$ is* correct *with respect to $\mathcal{I}$ if the following hold:*

1. *Let $\mathcal{I}' \subset \mathcal{I}$ be the full set of correct ciphertexts. Then, there exists $\mathcal{O}' \subset \mathcal{O}$ that contains a unique correct decryption of every element in $\mathcal{I}'$.*

2. *There exists $\hat{\mathcal{I}} \subset \mathcal{I} \setminus \mathcal{I}'$ such that $\hat{\mathcal{O}} := \mathcal{O} \setminus \mathcal{O}'$ contains the full set of the correct translation of $\hat{\mathcal{I}}$.*

Definition 22 *(Robustness) We say a mix-net is* robust *if the following hold:*

1. *$\mathcal{V}$ accepts the protocol if all the mixers are honest.*

2. *The output $\mathcal{O}$ of the mix-net is correct with respect to its input $\mathcal{I}$ with overwhelming probability if $\mathcal{V}$ accepts the protocol.*

3. *It is computationally difficult to find two correct translations for any ciphertext $C \in \mathcal{C}$.*

4. *For ciphertexts that are not decrypted into a message in $\mathcal{M}$ but into $\perp, \perp_1$, or $\perp_2$, the mixers are able to find their corresponding users in publicly verifiable manner.*

	# of SM	Com. bits
user	$15m + 3$	$2080m + \ell + 480$
j-th mixer	$32n$	$(2080j + \ell + 960)k$
verifying j-th mixer	$32n$	$(2080j + \ell + 960)k$

Table 3.1: Complexity

In our mix-net, as noted in Remark 1, a malicious user colluding with a mixer can generate a ciphertext C which the mixer can decide to provide the unique correct translation of C or it claims it invalid. Any ciphertexts that is claimed invalid will be traced back in a publicly verifiable manner to the original user who submitted the ciphertext. It is indeed a weakness in our mix-net, that a collusion has advantage of delaying the decision of the input ciphertext, although the choice is between a valid message and an invalid message. It may be a problem if it is the last mixer who enjoys this advantage since he can decide his choice after seeing the other inputs. To make this advantage insignificant, one can make the output of the mix-net to be a ciphertext that additionally need to be decrypted in a threshold manner. The same strategy is also used in the hybrid mix-net of [92].

Theorem 13 *The protocol is robust assuming random oracles.*

Proof. The theorem follows from Theorems 10 and 11 and the construction of the scheme. Here, random oracles are used only within the IND-ME-CCA encryption scheme. □

3.4 Efficiency of the Proposed Verifiable Mix-net

Most costly computation in our scheme is scalar multiplications over the elliptic curve. But the amount of these computations does not depend on the length of input ciphertext ℓ. The only computations that depends on the length ℓ are hash function and symmetric-key operations, which is negligibly small compared to scalar multiplications. Therefore, we estimate the computational cost by the number of scalar multiplications required in our scheme. We also estimate the amount of data that each player need to communicate. These results of estimation are given in Table 3.1, where "# of SM" means the number of scalar multiplications, "Com. bits" means the length of communication, $|q|$ is 160, m is the number of mixers, and k is the number of users. Here, the cost for mixers does not include the cost for verifying other mixers.

We now give estimates in how fast our scheme can be implemented. Suppose that $k = 100,000$, $m = 3$, and that available computers are Pentium III (700MHz) machines. Suppose that it takes 100μ sec to compute a single scalar multiplication in E by one computer. Such an implementation and experiment are reported in [100]. Suppose that each mixer has two machines, one machine for verifications of the previous mixers, and the other for shuffling and proving the correctness of its shuffle. We assume mixers generate Fiat-Shamir transformations of proofs so that any mixer is able to verify the proofs independently and in parallel without assuming trusted verifier. Such operations make the total time required for the proposed verifiable mix-net linear in the number of the mixers. Using variants of fixed-based comb method and simultaneous modular exponentiations in [107], we can estimate the total time required for mix-net, which tallies $100,000$ long ciphertexts and proves and verifies its correctness by 3 mixers, to be less than 10 minutes.

3.5 3-move Perfect Zero-Knowledge Argument for Shuffle-and-decrypt

In this section, we discuss here an alternative way to efficiently prove shuffle-and-decryption in zero-knowledge. Since using the 7-move scheme presented in [86] suffices to build our publicly verifiable hybrid mix-net, a novel property of our scheme is only that it is 3-move.

Our 3-move scheme is based on the perfect zero knowledge argument for shuffle proposed in [66] combined with a decryption proof. [66] is perfect zero-knowledge version of [70]. As discussed in [69], a sequential composition of zero-knowledge arguments on shuffle and that on decryption does not provide zero-knowledge argument on shuffle-and-decrypt since the intermediate state is not simulatable. So the combination is not straight forward. Using the compilers proposed in [110], we can easily derive black-box or concurrent zero-knowledge variants of the proposed arguments.

3.5.1 Notation

Let $n, q, E, \mathcal{O}$ be as defined in Section 3.2. Let G_0 be a generator of E, $(G_i, M_i) \in (E \otimes E)_{i=1,\ldots,k}$ be ElGamal ciphertexts to be shuffled and decrypted, where n is the number of ciphertexts. The resulting output ciphertexts are denoted as $\{(G'_i, M'_i) \in E^2\}_{i=1,\ldots,k}$. Let $M_0 = [x]G_0$ be the public key used for the shuffle, where $x \in \mathbb{Z}/q\mathbb{Z}$ is the corresponding private key. Let $x' \in_R \mathbb{Z}/q\mathbb{Z}$ be the private key used for partial decryption, and $Y = [x']G_0$ be the corresponding public key. Here, the public key is the set, $\{q, E, G_0, M_0, (Y)\}$. $\mathcal{P}$ is a prover who shuffles-and-decrypts and proves the validity of shuffle-and-decryption to a verifier $\mathcal{V}$. Let π be a permutation $\pi : \{1, \ldots, k\} \to \{1, \ldots, k\}$, matrix $(\pi_{ij})_{i,j=1,\cdots,k}$ be a permutation matrix over $\mathbb{Z}/q\mathbb{Z}$. We say $(\pi_{ij})_{i,j=1,\cdots,k}$ is a permutation matrix if there exists a permutation π that satisfies

$$\pi_{ij} = \begin{cases} 1 \bmod q & \text{if } \pi(i) = j \\ 0 \bmod q & \text{otherwise} \end{cases} .$$

In the proposed mix-net, we only consider the case $M_0 = Y$ and $x = x'$. Although the decryptions in Eqs. (3.5), (3.6), and (3.7) are not decryption of conventional ElGamal ciphertexts, the proposed argument can be applied.

3.5.2 ElGamal Shuffle-and-Decryption

ElGamal shuffling is a procedure that, given G_0, M_0 and n ElGamal ciphertexts $\{(G_i, M_i)\}_{i=1,\ldots,k}$, outputs ElGamal ciphertexts

$$(G'_i, M'_i) = ([s_i]G_0 + G_{\pi^{-1}(i)}, [s_i]M_0 + M_{\pi^{-1}(i)})$$

for $i = 1, \ldots, k$ where $\{s_i \in_R \mathbb{Z}/q\mathbb{Z}\}_{i=1,\ldots,k}$ and a permutation of indices $\pi : \{1, \ldots, k\} \to \{i = 1, \ldots, k\}$ are randomly chosen.

Input ciphertexts and resulting ciphertexts of ElGamal shuffling are the same as sets. And at the same time, no polynomially bounded machine, given an input and its result of shuffling, has an advantage over the random-guessing algorithm in guessing any part of permutation π for randomly chosen G_0, M_0, s_i, π.

ElGamal shuffle-and-decryption is a combination procedure of ElGamal shuffling and decryption that, given n ElGamal ciphertexts $\{(G_i, M_i)\}_{i=1\ldots,k}$, outputs ElGamal ciphertexts

$$(G'_i, M'_i) = \tag{3.20}$$

$$([s_i]G_0 + G_{\pi^{-1}(i)}, [-x']G'_i + [s_i]M_0 + M_{\pi^{-1}(i)}) \tag{3.21}$$

for $i = 1, \ldots, k$ where $\{s_i \in_R \mathbb{Z}/q\mathbb{Z}\}_{i=1,\ldots,k}$ and π are chosen uniformly and randomly. Here, the addition of $[-x']G'_i$ in the second term has the effect of partial decryption.

Using a permutation matrix (π_{ji}) which corresponds to a permutation π and replacing $\{s_i\}_{i=1,\ldots,k}$ with $\{\pi_{0i} \in \mathbb{Z}/q\mathbb{Z}\}_{i=1,\ldots,k}$, we find that Eq. (3.21) can be expressed as

$$(G'_i, M'_i) \quad = \quad (\sum_{\nu=0}^{k}[\pi_{\nu i}]G_\nu, [-x']G'_i + \sum_{\nu=0}^{k}[\pi_{\nu i}]M_\nu) \tag{3.22}$$

for $i = 1, \ldots, k$. Therefore, proving the correctness of the shuffle is equivalent to proving the existence of an $\pi_{0i} \in \mathbb{Z}/q\mathbb{Z}$ for $i = 1, \ldots, k$, x', and a permutation matrix $(\pi_{ji})_{i,j=1,\ldots,k}$ which satisfies Eqs. (3.22) and $Y = [x']G_0$.

Here, the matrix $(\pi_{\nu i})_{\nu=0,\ldots,k;i=1,\ldots,k}$ in Eq. (3.22) is a $(k+1)\times k$ matrix that contains a permutation matrix whose size is $k \times k$ as a sub-matrix. It should be remembered here that, in the rest of the chapter, just as the size of matrices may often change, so will the range over which the indices of the matrix π run.

3.5.3 Idea of the Proposed Zero-Knowledge Argument

In [66], a prover is given an input ciphertexts set $\{(G_i, M_i)\}_{i=1,\ldots,k}$ and its shuffled ciphertexts set $\{(G'_i, M'_i)\}_{i=1,\ldots,k}$ such that

$$(G'_i, M'_i) = ([s_i]G_0 + G_{\pi^{-1}(i)}, [s_i]M_0 + M_{\pi^{-1}(i)})$$

for $i = 1,\ldots,k$ holds for witness set $\{s_i \in \mathbb{Z}/q\mathbb{Z}\}_{i=1,\ldots,k}$ and a permutation π. Then the prover proves to the verifier the knowledge of the witness by exploiting the property of permutation matrices.

Now we consider constructing a zero-knowledge argument for shuffle-and-decryption by composing the above zero-knowledge argument for shuffle and a zero-knowledge proof for decryption. As previously stated, the intermediate state, that is, $(M'_i)_{i=1,\ldots,k}$, is unsimulatable. The rest of the message that prover generates is simulatable.

We avoid this problem by using

$$\bar{M}_i = [s''_i]M_{-1} + [s_i]M_0 + M_{\pi^{-1}(i)} \quad i = 1,\ldots,k$$

instead of the above $(M'_i)_{i=1,\ldots,k}$, where $\{s''_i \in_R \mathbb{Z}/q\mathbb{Z}\}_{i=1,\ldots,k}$ is randomly chosen elements in $(\mathbb{Z}/p\mathbb{Z})^*$ and M_{-1} is a public parameter. Since $\bar{M}_i$ can be considered as a perfect hiding commitment of M'_i, a simulation of $\bar{M}_i$ will be possible.

We need some new methods to prove the knowledge of committed $\{s_i\}_{i=1,\ldots,k}$ instead of knowledge of simple exponents, which can be obtained using similar tricks that appears in [66]. Applying these methods we are able to construct a perfect zero-knowledge argument for shuffle-and-decryption of Elements ciphertexts.

3.5.4 Proposed Zero-Knowledge Argument for Shuffle-and-Decryption

Let us next introduce a 3-round honest-verifier perfect zero-knowledge argument for shuffle-and-decryption. We assume M_{-1} and $\{F_\nu \in_R E\}_{\nu=-4,\ldots,k}$ are $n+6$ points in E that are randomly generated so that neither $\mathcal{P}$ nor $\mathcal{V}$ can generate non-trivial elements $c, b, \{a_\nu\}_{\nu=-4,\ldots,k}$ in $\mathbb{Z}/q\mathbb{Z}$ satisfying $[c]G_0 + [b]M_{-1} +$

$\sum_{\nu=-4}^{k}[a_\nu]F_\nu = \mathcal{O}$ with non-negligible probability. We assume $c_0 = 1 \bmod q$.

Proving a Shuffle Decryption

Commitment: $\mathcal{P}$ uniformly and randomly chooses $\{\pi_{\nu 0}, \pi'_\nu\}_{\nu=-4,\ldots,k}$, $\{\pi_{-1i}\}_{i=1,\ldots,k}$, $\{\pi_{-2i}\}_{i=1,\ldots,k}$, and $\{t_i\}_{i=1,\ldots,k}$ from $\mathbb{Z}/q\mathbb{Z}$, and then computes:

$$\begin{aligned}
\pi_{-3i} &= \sum_{j=1}^{k} 3{\pi_{j0}}^2\pi_{ji} \bmod q \quad i=1,\cdots,k \\
\pi_{-4i} &= \sum_{j=1}^{k} 3\pi_{j0}\pi_{ji} \bmod q \quad i=1,\cdots,k \\
Y'_i &= [t_i]G_0 \quad i=1,\cdots,k \\
F'_\mu &= \sum_{\nu=-4}^{k}[\pi_{\nu\mu}]F_\nu \quad \mu=0,\cdots,k && (3.23) \\
\bar{M}_\mu &= \sum_{\nu=-1}^{k}[\pi_{\nu i}]M_\nu \quad \mu=0,\cdots,k && (3.24) \\
\widetilde{F}'_0 &= \sum_{\nu=-4}^{k}[\pi'_\nu]F_\nu && (3.25) \\
G'_0 &= \sum_{\nu=0}^{k}[\pi_{\nu 0}]G_\nu \\
w &= \sum_{j=1}^{k}{\pi_{j0}}^3 - \pi_{-30} - \pi'_{-4} \bmod q \\
\bar{M}' &= [\pi_{-10}]M_{-1} + \sum_{i=1}^{k}[t_i]G'_i
\end{aligned}$$

Then, $\mathcal{P}$ sends $\{F'_\mu\}_{\mu=0,\ldots,k}, \{\bar{M}_\mu\}_{\mu=0,\ldots,k}, \widetilde{F}'_0, G'_0, w, \bar{M}', \{Y'_i\}_{i=1,\ldots,k}$ to $\mathcal{V}$ as a commitment.

Challenge: $\mathcal{V}$ uniformly and randomly chooses $\{c_i \in \mathbb{Z}/q\mathbb{Z}\}_{i=1,\ldots,k}$ and sends it to $\mathcal{P}$.

Response: $\mathcal{P}$ sends $\mathcal{V}$ the following response:

$$\begin{aligned}
r_\nu &= \sum_{\mu=0}^{k}\pi_{\nu\mu}c_\mu \bmod q \quad \nu=-4,\cdots,k \\
r'_\nu &= \sum_{i=1}^{k}\pi_{\nu i}{c_i}^2 + \pi'_\nu \quad \nu=-4,\cdots,k \\
\bar{r}_i &= x'c_i + t_i \bmod q \quad i=1,\cdots,k
\end{aligned}$$

Verification:

$\mathcal{V}$ accepts the shuffle if the following equations hold for a randomly chosen $d \in_R \mathbb{Z}/q\mathbb{Z}$:

$$\sum_{\nu=-4}^{k} [r_\nu + dr'_\nu]F_\nu = F'_0 + [d]\widetilde{F}'_0 + \sum_{i=1}^{k}[c_i + dc_i{}^2]F'_i \tag{3.26}$$

$$\sum_{\nu=0}^{k}[r_\nu]G_\nu = \sum_{\mu=0}^{k}[c_\mu]G'_\mu \tag{3.27}$$

$$\sum_{\nu=-1}^{k} [r_\nu]M_\nu = \sum_{\mu=0}^{k}[c_\mu]\bar{M}_\mu \tag{3.28}$$

$$\sum_{j=1}^{k}(r_j^3 - c_j^3) = r_{-3} + r'_{-4} + w \pmod q \tag{3.29}$$

$$\bar{M}' + \sum_{i=1}^{k}[c_i](\bar{M}_i - M'_i) = [r_{-1}]M_{-1} + \sum_{i=1}^{k}[\bar{r}_i]G'_i \tag{3.30}$$

$$[\bar{r}_i]G_0 = [c_i]Y + Y'_i \quad i = 1, \cdots, k \tag{3.31}$$

Theorem 14 *The proposed protocol is argument for shuffle-and-decryption.*

Proof. The theorem can be proved in the same way that Theorem 4 is proved. Here, proofs for lemmas are omitted.

Lemma 26 *If $\mathcal{V}$ accepts the protocol with a probability p, which implies, Eq. (3.26) holds, then*

$$\sum_{\nu=-4}^{k} [r_\nu]F_\nu = \sum_{\mu=0}^{k}[c_\mu]F'_\mu \tag{3.32}$$

$$\sum_{\nu=-4}^{k} [r'_\nu]F_\nu = \tilde{F}'_0 + \sum_{i=1}^{k}[c_i{}^2]F'_i \tag{3.33}$$

will hold with the probability at least $p - 1/2^n$.

Lemma 27 *If $\mathcal{V}$ accepts the protocol with a probability p, then there is a polynomially bounded knowledge extractor K that can, within an expected number of steps bounded by $\mathcal{O}(k/p)$, extract from $\mathcal{P}$ an $\{\pi_{\nu\mu}\}_{\nu=-4,\ldots,k;\mu=0,\ldots,k}$ which satisfies Eq. (3.23) and an $\{\pi'_\nu\}_{\nu=-4,\ldots,k}$ which satisfies Eq. (3.25).*

Lemma 28 *Assume that there is a knowledge extractor K that can extract an $\{\pi_{\nu\mu}\}_{\nu=-4,\ldots,k;\mu=0,\ldots,k}$ and an $\{\pi'_\nu\}_{\nu=-4,\ldots,k}$ from $\mathcal{P}$ that satisfies Eq. (3.23) and (3.25). If there is also a knowledge extractor K' that can extract an $\{r_\nu, r'_\nu\}_{\nu=-4,\ldots,k}$ from $\mathcal{P}$ that satisfies Eqs. (3.32) and (3.33) such that either*

$$r_\nu = \sum_{\mu=0}^{k} \pi_{\nu\mu}c_\mu \bmod q$$

or

$$r'_\nu = \sum_{i=1}^{k} \pi_{\nu i} {c_i}^2 + \pi'_\nu \bmod q$$

for some $\nu = -4, \ldots, k$*, then* K' *can generate non-trivial integers* $\{a_\nu\}_{\nu=-4,\ldots,k}$ *satisfying* $\sum_{\nu=-4}^{k} [a_\nu] F_\nu = 1 \bmod p$ *with overwhelming probability.*

Lemma 29 *Assume that* $\{\pi_{\nu\mu}\}_{\nu=-4,\ldots,k;\mu=0,\ldots,k}$*,* $\{\pi_\nu\}_{\nu=-4,\ldots,k}$ *and* w *are givens. Also assume that* $\{r_i\}_{i=1,\ldots,k}, r_{-3}$ *and* r'_{-4} *will be generated, for a given* $\{c_i\}_{i=1,\ldots,k}$*, as*

$$\begin{aligned} r_\nu &= \sum_{\mu=0}^{k} \pi_{\nu\mu} c_\mu \bmod q \quad \nu = -3, 1, 2, \ldots, k \\ r'_{-4} &= \sum_{i=1}^{k} \pi_{-4i} {c_i}^2 + \pi'_{-4} \bmod q \end{aligned}$$

If the given $\{\pi_{ij}\}_{i=1,\ldots,k;j=1,\ldots,k}$ *is not a permutation matrix, then Eq. (3.29) will not hold with overwhelming probability for a uniformly and randomly chosen* $\{c_i\}_{i=1,\ldots,k}$ *[70].*

Lemma 30 *Assume that* $\{G_\nu, M_\nu\}_{\nu=0,\ldots,k}$ *and* $\{\pi_{\nu\mu}\}_{\nu=0,\ldots,k;\mu=0,\ldots,k}$ *are givens. Also assume that* $\{r_\nu\}_{\nu=0,\ldots,k}$ *will be generated for a given* $\{c_i\}_{i=1,\ldots,k}$ *as*

$$r_\nu = \sum_{\mu=0}^{k} \pi_{\nu\mu} c_\mu \bmod q \quad \nu = 0, \ldots, k.$$

If the equations

$$\begin{aligned} G'_\mu &= \sum_{\nu=0}^{k} [\pi_{\nu\mu}] G_\nu \quad \mu = 0, \cdots, k, \\ \bar{M}_\mu &= \sum_{\nu=-1}^{k} [\pi_{\nu\mu}] M_\nu \quad \mu = 0, \cdots, k \end{aligned}$$

do not hold, then Eqs. (3.27) and (3.28) will not hold with overwhelming probability for a uniformly and randomly chosen $\{c_i\}_{i=1,\ldots,k}$*.*

Lemma 31 *If* $\mathcal{V}$ *accepts the protocol with a probability p, then there is a polynomially bounded knowledge extractor K that can, within an expected number of steps bounded by* $\mathcal{O}(1/p)$*, extract from* $\mathcal{P}$ *an* $\{t_i\}_{i=1,\ldots,k}$ *and* x' *which satisfies*

$$Y'_i = [t_i] G_0 \quad i = 1, \cdots, k$$

$$Y \quad = \quad [x']G_0.$$

Moreover

$$\bar{r}_i \quad = \quad x'c_i + t_i \bmod q$$

Lemma 32 *Assume that $\{G_\nu, M_\nu\}_{\nu=0,\ldots,k}$ and $\{\pi_{\nu\mu}\}_{\nu=0,\ldots,k;\mu=0,\ldots,k}$, $\{t_i\}_{i=1,\ldots,k}$, and x' are givens. Also assume that $\{r_\nu\}_{\nu=0,\ldots,k}$ and $\{\bar{r}_i\}_{i=1,\ldots,k}$ will be generated for a given $\{c_i\}_{i=1,\ldots,k}$ as*

$$\begin{aligned} r_{-1} &= \sum_{\mu=0}^{k} \pi_{-1\mu} c_\mu \bmod q \\ \bar{r}_i &= x'c_i + t_i \bmod q \quad i = 1, \ldots, k. \end{aligned}$$

If the equations

$$\begin{aligned} \left(\bar{M}_i - M'_i\right) &= [x']G'_i + [\pi_{-1i}]M_{-1} \quad i = 1, \cdots, k \\ Y &= [x']G \end{aligned}$$

do not hold, then Eqs. (3.30) and (3.30) will not hold with overwhelming probability for a uniformly and randomly chosen $\{c_i\}_{i=1,\ldots,k}$.

From Lemmas 26, 27, and 28, there is a knowledge extractor K that can extract an $\{\pi_{\nu\mu}\}_{\nu=-4,\ldots,k;\mu=0,\ldots,k}$ and a $\{\pi'_\nu\}_{\nu=-4,\ldots,k}$ from $\mathcal{P}$ that respectively satisfy Eqs. (3.23) and (3.25) within an expected number of steps bounded by $\mathcal{O}(n/p)$. Further, K can either, from values extracted from $\mathcal{P}$, (1) only generate a response that satisfies equations

$$\begin{aligned} r_\nu &= \sum_{\mu=0}^{k} \pi_{\nu\mu} c_\mu \bmod q \qquad \nu = -4, \ldots, k \\ r'_\nu &= \sum_{i=1}^{k} \pi_{\nu i} c_i + \pi'_\nu \bmod q \qquad \nu = -4, \ldots, k \end{aligned}$$

for a given challenge, or (2) generate non-trivial integers $\{a_\nu\}_{\nu=-4,\ldots,k}$ satisfying $\sum_{\nu=-4}^{k}[a_\nu]F_\nu = 1$ with overwhelming probability. Note that (2) will happen with negligible probability as long as solving the discrete logarithm problem is difficult.

From Lemma 29, the $\{\pi_{\nu\mu}\}_{\nu=1,\ldots,k;\mu=1,\ldots,k}$ that can be extracted by K from $\mathcal{P}$ is a permutation matrix. From Lemma 30, the equations

$$\begin{aligned} G'_\mu &= \sum_{\nu=0}^{k}[\pi_{\nu\mu}]G_\nu = [\pi_{0i}]G_0 + G_{\pi^{-1}(i)} \quad \mu = 0,\cdots,k \\ \bar{M}_\mu &= \sum_{\nu=-1}^{k}[\pi_{\nu\mu}]M_\nu \quad \mu = 0,\cdots,k \end{aligned}$$

hold for the $\{\pi_{\nu\mu}\}_{\nu=0,\ldots,k;\mu=0,\ldots,k}$ above.

From Lemmas 31,28, and 32,

$$M'_i = \bar{M}_i - [x']M'_i + [\pi_{-1i}]M_{-1} \quad i = 1,\cdots,k$$

and from Lemma 30,

$$\begin{aligned} M'_i &= [-x']G'_i + \sum_{\nu=0}^{k}[\pi_{\nu i}]M_\nu \\ &= [-x']G'_i + [A_{0i}]M_0 + M_{\pi^{-1}(i)} \quad \mu = 0,\cdots,k \end{aligned}$$

Therefore, as long as solving the discrete logarithm problem is difficult, there is a knowledge extractor K that can extract an $\{\pi_{\nu i}\}_{\nu=0,\ldots,k;i=1,\ldots,k}$ and x' from $\mathcal{P}$ satisfying Eq. (3.22) within an expected number of steps bounded by $\mathcal{O}(n/p)$, and its sub-matrix $(\pi_{ij})_{i=1,\ldots,k}j$ will be a permutation matrix if $\mathcal{V}$ accepts the protocol. □

Theorem 15 *The proposed protocol is perfect zero-knowledge.*

Proof.
View: The view of this protocol is

$$\begin{aligned} & q, E, G_0, M_0, Y, \\ & \{(G_i, M_i)\}_{i=1,\ldots,k}, \{(G'_i, M'_i)\}_{i=1,\ldots,k}, M_{-1}, \\ & \{F_\nu\}_{\nu=-4,\ldots,k}, \{F'_\mu\}_{\mu=0,\ldots,k}, \{\bar{M}_\mu\}_{\mu=0,\ldots,k}, \\ & \widetilde{F}'_0, G'_0, w, \bar{M}', \{Y'_i\}_{i=1,\ldots,k}, \{c_i\}_{i=1,\ldots,k}, \\ & \{r_\nu\}_{\nu=-4,\ldots,k}, \{r'_\nu\}_{\nu=-4,\ldots,k}, \{\bar{r}_i\}_{i=1,\ldots,k} \end{aligned}$$

The following simulator proves the theorem:

Given

$$\begin{aligned}&q, E, G_0, M_0, Y, \{G_i, M_i\}_{i=1,\dots,k}, \{G'_i, M'_i\}_{i=1,\dots,k},\\ &M_{-1}, \{F_\nu\}_{(\nu=-4,\dots,k)},\end{aligned}$$

a simulator uniformly and randomly chooses

$$\begin{aligned}&\{r_\nu\}_{\nu=-4,\dots,k}, \{r'_\nu\}_{\nu=-4,\dots,k}, \{\bar{r}_i\}_{i=1,\dots,k},\\ &\{c_i\}_{i=1,\dots,k}, \{F'_i\}_{i=1,\dots,k}, \{\bar{M}_i\}_{i=1,\dots,k}.\end{aligned}$$

It then generates $\{Y'_i\}_{i=1,\dots,k}, \bar{M}', w, \bar{M}_0, \widetilde{F}'_0, F'_0, G'_0$, so as to satisfy Eqs.(3.27), (3.28), (3.29), (3.30), (3.31), and

$$\begin{aligned}\sum_{\nu=-4}^{k}[r_\nu]F_\nu &= \sum_{\mu=0}^{k}[c_\mu]F'_\mu,\\ \sum_{\nu=-4}^{k}[r'_\nu]F_\nu &= \widetilde{F}'_0 + \sum_{i=1}^{k}[{c_i}^2]F'_i.\end{aligned}$$

The correspondences between the randomness of the argument view and that of the simulated view are as in the following.

$$\begin{aligned}&\{r_\nu\}_{\nu=-4,\dots,k} \Leftrightarrow \{\pi_{\nu 0}\}_{\nu=-4,\dots,k},\\ &\{r'_\nu\}_{\nu=-4,\dots,k} \Leftrightarrow \{\pi'_\nu\}_{\nu=-4,\dots,k}\\ &\{\bar{r}_i\}_{i=1,\dots,k} \Leftrightarrow \{t_i\}_{i=1,\dots,k},\\ &\{c_i\}_{i=1,\dots,k} \Leftrightarrow \{c_i\}_{i=1,\dots,k}\\ &\{F'_i\}_{i=1,\dots,k} \Leftrightarrow \{\pi_{-2i}\}_{i=1,\dots,k},\\ &\{\bar{M}_i\}_{i=1,\dots,k} \Leftrightarrow \{\pi_{-1i}\}_{i=1,\dots,k}\end{aligned}$$

□

Conclusion

We proposed here the first efficient publicly verifiable hybrid mix-net. The scheme is composed of newly developed IND-ME-CCA secure multiple encryption scheme whose component encryption is IND-CCA secure hybrid encryption scheme. The decryption in this multiple encryption scheme is publicly and efficiently verifiable. The proposed scheme is efficient enough to treat large scale electronic questionnaires of long messages as well as voting with write-ins, and offers public verifiability of the correctness of the tally. Indeed, the heaviest operation for proving or verifying the correctness of each mixer is computation of scalar multiplication/modular exponentiation for 32 times the number of input ciphertexts. The scheme is provably secure if we assume random oracles, semantic security of a one-time symmetric-key cryptosystem, and intractability of decision Diffie-Hellman problem.

Chapter 4

Aggregate Shuffle Argument Scheme

4.1 Introduction

The notion of mix-net, whose core ingredient is the execution of multiple rounds of shuffling and decryption by multiple and independent mixers so that none of the output decryptions can be linked to any of the input encryptions, is useful in applications which require anonymity, such as voting. Although it is desirable to achieve the property of public verifiability to ensure the correctness of output, it requirs a rather large amount of computation. Reducing this computational cost has been one of the major problems in the study of mix-net.

There are roughly two types of previous works that reduce the amount of computation for verification of mix-net. The straight forward way is to increase the efficiency of each component. In [65, 66, 69, 70, 85, 120, 86, 150], efficient and practical schemes for proving the correctness of each component shuffling or a pair of shuffling and decryption are proposed. On the other hand, in [1], Abe proposed a scheme to prove the correctness of total mix-net shufflings in such a way that the amount of computation of its verifier does not depend on the number of shufflings executed in the mix-net. We call such a scheme an aggregate shuffle argument scheme. The essence of the latter scheme is to consider the whole mix-net as a single shuffle and let all mixers collaborately play the role of its single prover. Then, the verification cost for the mix-net will be reduced to that for a single shuffle. Here, the proof of shuffle collaborately generated is the one proposed by Sako et al. in [137].

Since the computational cost for the verifier in the Sako's scheme in [137] is huge, so is that for the verifier in Abe's scheme. Hence, Abe's scheme is advantageous over the former type of schemes only when a large number of shufflings is executed in a single mix-net. However, in such a situation, neither the former nor the latter types of schemes are efficient enough to be applied to very large-scale voting.

In this chapter, we propose a novel aggregate shuffle argument scheme that is much more efficient than the one proposed in [1] by exploiting schemes proposed in [66, 70] instead of the scheme proposed in [137]. We also propose a model and security requirements of aggregate shuffle argument schemes.

The basic idea of our scheme is to consider the whole mix-net as a single shuffle and let all mixers collaborately play the role of a prover of the zero-knowledge argument for shuffling proposed in [66]. This is certainly possible by the technique of general multi-party computation. However, such a general solution requires of each mixer an unrealistic amount of computation, an amount on the order of the square of the number of mixers. Hence, we have developed a specifically designed new zero-knowledge argument for shuffling in which multiple mixers can efficiently collaborate to play the role of its single prover. As a result, our scheme requires of each mixer an amount of computation that is linear to the number of all the mixers.

Suppose that the number of ciphertexts to be mixed is k and that of mixers is λ. Then, the verifier in our scheme is required to compute $10k$ exponentiations to verify the correctness of total mixing, and each mixer in our scheme is required to compute $28k$ and $13k(\lambda - 1)$ exponentiations, respectively, to prove the correctness of its shuffling and to verify the correctness of shuffling of all other mixers. On the other hand, each mixer and verifier in the scheme proposed in [1] are required to compute, respectively, $640k$ and $320k + 640k\lambda$ exponentiations. If we use the scheme proposed in [85] for multiple times, each mixer and verifier are required to compute, respectively, $6k\lambda$ and $6k + 6k\lambda$ exponentiations.

It should be mentioned here that, although it is theoretically possible for the verifiers to collaborately and interactively play the role of the single verifier in our mix-net, we do not consider it as a realistic solution for very large-scale voting such as one for a national election. Rather, as is the case considered in [69], we consider the case that all mixers collaborately generate a Fiat-Shamir transformation of their proof instead of interactively proving their correctness. Then, each of the verifiers is able to independently verify their correctness.

Note that, although Abe's scheme can be fixed trivially, it does not satisfy our security requirements. Each mixer needs to additionally verify the correctness of the output of the previous mixer to be a secure aggregate shuffle argument scheme in our sense.

An Example Application: We show an example case in which our proposed scheme is effective. Suppose that a mix-net is applied to very large-scale voting such as a national election in which voter anonymity is strongly required. In such a case, since the coalition of all the mixers is able to link input encryptions to output decryptions, an increase of the number of mixers in the mix-net is strongly required.

However, in such a large-scale voting, even if one of the previous efficient schemes [65, 66, 69, 70, 85, 120, 86, 150] is used for proving each shuffling, the amount of computation for provers and verifiers will unavoidably be large. Therefore, increasing the number of mixers, i.e., achieving high level of anonymity, puts a heavy computation load on mixers and verifiers. Such a situation is especially undesirable for verifiers, since some of them, such as voters in general, may have only small computational resources.

Since the amount of computation for verifiers in our scheme does not depend on the number of mixers, it is easy to increase the number of mixers in our scheme, i.e., it is easy to achieve a high level of anonymity. Moreover, since our scheme exploits a variant of the very efficient arguments for shuffling proposed in [70, 66], it can handle large-scale voting unlike the scheme proposed in [1].

Organization: This chapter is organized as follows. Section 4.2 describes the model and security requirements for aggregate shuffle argument schemes. Section 4.3 describes the basic idea of our proposed scheme by introducing an example aggregate permutation argument scheme. Section 4.4 proposes an aggregate shuffle argument scheme which is secure only when some of the players are assumed to be honest and discusses its security requirements. Section 4.5 presents a method to transform the scheme proposed in Section 4.4 into a full scheme which is secure even if all players are malicious. Section 4.6 estimates the efficiency of our scheme and compares it with prior works.

4.2 Aggregate Shuffle Argument Schemes

4.2.1 ElGamal Shuffling

We introduce here shufflings of ElGamal encryptions, on which our aggregate shuffle argument scheme is based. Let E be an additive/multiplicative cyclic group of prime order q in which the discrete logarithm problem is difficult to solve, G be a generator of E, and $\{(G_i^{(1)}, M_i^{(1)}) \in E^2\}_{i=1,\dots,k}$ be k ElGamal ciphertexts to be shuffled by using the public-key $M \in E$. Let $\mathcal{O}$ denote the zero of E. Let $[x]G$ denote multiplication/modular-exponentiation of $G \in E$ by $x \in \mathbb{Z}/q\mathbb{Z}$.

ElGamal shuffling is a procedure that, given G, M and k ElGamal ciphertexts $\{(G_i, M_i) \in E^2\}_{i=1,\dots,k}$, uniformly and randomly choose permutation of indices $\phi : \{1, \dots, k\} \to \{i = 1, \dots, k\}$, randomly choose random numbers $\{\varphi_i \in_R \mathbb{Z}/q\mathbb{Z}\}_{i=1,\cdots,k}$ for reencryptions, and then outputs ElGamal ciphertexts

$$\{(G'_i, M'_i)\}_{i=1,\dots,k} = \{([\varphi_i]G + G_{\phi(i)}, [\varphi_i]M + M_{\phi(i)})\}_{i=1,\dots,k}.$$

Shuffling of ElGamal ciphertexts results in the following two important properties (as a shuffling):

1. There exists a permutation ϕ such that equation $D_x((G'_i, M'_i)) = D_x((G_{\phi(i)}, M_{\phi(i)}))$ holds for all i. Here, $D_x(\cdot)$ is a decryption algorithm that uses the private key x that corresponds to M.

2. As long as the decision Diffie-Hellman problem is difficult to solve, no polynomially bounded algorithm, given only $q, G, M, \{(G_i, M_i)\}_{i=1,\cdots,k}$, $\{(G'_i, M'_i)\}_{i=1,\cdots,k}$, has an advantage over the random-guessing algorithm in guessing any part of permutation ϕ for uniformly and randomly chosen G, M, φ_i, ϕ.

4.2.2 Model of Aggregate Shuffle Argument Schemes

Let $y \leftarrow \mathsf{A}(x)$ denote an algorithm A that outputs y when x is input. Let $\langle y_1, y_2\rangle \leftarrow \mathsf{IP}_{\mathcal{A}_1,\mathcal{A}_2}\langle x_1, x_2\rangle$ denote an interactive protocol IP between $\mathcal{A}_1$ and $\mathcal{A}_2$. Here, inputs to $\mathcal{A}_1$ and $\mathcal{A}_2$ are, respectively, x_1 and x_2, and outputs of $\mathcal{A}_1$ and $\mathcal{A}_2$ at the end of the protocol are, respectively, y_1 and y_2.

Let k be the number of ciphertexts input to the mix-net, and λ be the number of mixers. Players in aggregate shuffle argument scheme are a set of λ mixers ($\{\mathcal{M}^{(\gamma)}\}_{\gamma=1,\ldots,\lambda}$), an aggregator $\mathcal{A}$, and a verifier $\mathcal{V}$. Let *pkey* denote a public key for shufflings, $\mathit{wit}^{(\gamma)}$ denote the witness of $\mathcal{M}^{(\gamma)}$ for its shuffling, $\mathit{icset}^{(\gamma)}$ denote input ciphertexts of $\mathcal{M}^{(\gamma)}$, and $\mathit{ocset}^{(\gamma)}$ denote output ciphertexts of $\mathcal{M}^{(\gamma)}$. Each $\mathit{wit}^{(\gamma)}$ consists of a permutation of k elements and k random number to be used for reencryptions. We assume that output ciphertexts $\mathit{ocset}^{(\gamma)}$ of $\mathcal{M}^{(\gamma)}$ are input ciphertexts $\mathit{icset}^{(\gamma+1)}$ of $\mathcal{M}^{(\gamma+1)}$ for $\gamma = 1, \ldots, \lambda-1$ and that $\mathit{icset}^{(1)}$ are ciphertexts input to the mix-net and $\mathit{ocset}^{(\lambda)}$ are ciphertexts output by the mix-net. We also let $\mathit{icset}^{(\lambda+1)}$ denote $\mathit{ocset}^{(\lambda)}$.

Definition 23 *(Aggregate Shuffle Argument Scheme) An aggregate shuffle argument scheme consists of two algorithms* (Setup, Shuff) *and two interactive protocols* (Ind-Arg, Agg-Arg).

Setup: *A probabilistic setup algorithm that, given a security parameter k, outputs pkey of size k.*

$$\mathit{pkey} \leftarrow \mathsf{Setup}(1^k)$$

Shuff: *A probabilistic shuffling algorithm for each $\mathcal{M}^{(\gamma)}$ that, given pkey, $\mathit{wit}^{(\gamma)}$ and $\mathit{icset}^{(\gamma)}$, outputs $\mathit{ocset}^{(\gamma)}$.*

$$\mathit{ocset}^{(\gamma)} \leftarrow \mathsf{Shuff}(\mathit{pkey}, \mathit{wit}^{(\gamma)}, \mathit{icset}^{(\gamma)})$$

Ind-Arg: *An interactive protocol between each $\mathcal{M}^{(\gamma)}$ and $\mathcal{A}$ in which $\mathcal{M}^{(\gamma)}$ proves to $\mathcal{A}$ the validity of its shuffle. During this protocol, $\mathcal{A}$ is allowed to interact with $\{\mathcal{M}^{(\iota)}\}_{\iota=1,\ldots,\gamma-1,\gamma+1,\ldots,\lambda}$ through other instances of* Ind-Arg *protocols and with $\mathcal{V}$ through* Agg-Arg. *$\mathcal{M}^{(\gamma)}$ is given pkey, $\mathit{wit}^{(\gamma)}$, $\mathit{icset}^{(\gamma)}$, and $\mathit{ocset}^{(\gamma)}$. $\mathcal{A}$ is given pkey, and $\{\mathit{icset}^{(\zeta)}\}_{\zeta=1,\ldots,\lambda+1}$. At the end of the protocol, $\mathcal{A}$ outputs acc* *or rej.*

$$\langle \emptyset, acc/rej \rangle \leftarrow$$
$$\mathsf{Ind\text{-}Arg}_{\mathcal{M}^{(\gamma)},\mathcal{A}}\langle (pkey, wit^{(\gamma)}, icset^{(\gamma)}, ocset^{(\gamma)}), (pkey, \{icset^{(\zeta)}\}_{\zeta=1,\ldots,\lambda+1}) \rangle$$

Agg-Arg**:** *An interactive aggregation protocol between $\mathcal{A}$ and $\mathcal{V}$ in which $\mathcal{A}$ proves to $\mathcal{V}$ the validity of the output of a mix-net. During this protocol, $\mathcal{A}$ is allowed to interact with $\{\mathcal{M}^{(\gamma)}\}_{\gamma=1,\ldots,\lambda}$ through* Ind-Arg *protocols. $\mathcal{A}$ is given pkey and $\{icset^{(\zeta)}\}_{\zeta=1,\ldots,\lambda+1}$. $\mathcal{V}$ is given pkey, $icset^{(1)}$, and $ocset^{(\lambda)}$. At the end of the protocol $\mathcal{V}$ outputs acc or rej.*

$$\langle \emptyset, acc/rej \rangle \leftarrow \mathsf{Agg\text{-}Arg}_{\mathcal{A},\mathcal{V}}\langle (pkey, \{icset^{(\zeta)}\}_{\zeta=1,\ldots,\lambda+1}), (pkey, icset^{(1)}, ocset^{(\lambda)}) \rangle$$

4.2.3 Security Requirements of Aggregate Shuffle Argument Scheme

We say an aggregate shuffle argument scheme is secure and efficient if it has the following completeness, soundness, zero-knowledge, and aggregation properties. Our goal is to propose a secure and efficient aggregate shuffle argument scheme.

Completeness is defined in the ordinary way as:

Definition 24 *(completeness) If all players are honest, $ocset^{(\gamma)}$ is the shuffling of $icset^{(\gamma)}$, $\mathcal{A}$ outputs acc after the end of each* Ind-Arg*, and $\mathcal{V}$ outputs acc after the end of each* Agg-Arg.

The soundness is defined as:

Definition 25 *(Soundness) The scheme is* sound *if the following three statements hold.*

1. *Interactive protocol $\mathsf{Agg\text{-}Arg}_{\mathcal{A},\mathcal{V}}$ is an argument of shuffling from $icset^{(1)}$ to $ocset^{(\lambda)}$.*
2. *Each interactive protocol $\mathsf{Ind\text{-}Arg}_{\mathcal{M}^{(\gamma)},\mathcal{A}}$ is an argument of shuffling from $icset^{(\gamma)}$ to $ocset^{(\gamma)}$.*
3. *If $\mathcal{A}$ accepts all $\{\mathcal{M}^{(\gamma)}\}_{\gamma=1,\ldots,\lambda}$ and $\mathcal{A}$ is honest, $\mathcal{V}$ accepts $\mathcal{A}$.*

Although, what the verifier wants to verify is only the correctness of total mix-met, it is often convenient if the aggregator is able to detect which mixer has cheated. Hence, we included the second and the third properties in the soundness as well as the first one.

A permutation that each mixer has chosen must not be leaked through $\mathsf{Ind\text{-}Arg}_{\mathcal{M}^{(\gamma)},\mathcal{A}}$.

Definition 26 *(Zero-knowledge) The scheme is* zero-knowledge *if each interactive protocol $\mathsf{Ind\text{-}Arg}_{\mathcal{M}^{(\gamma)},\mathcal{A}}$ is zero-knowledge.*

All of the above properties can be achieved by simply using any zero-knowledge argument for shuffling for multiple times. Hence, efficiency is the one of the most important factors for defining aggregate shuffle argument schemes.

Definition 27 *(Aggregation) We say an aggregate shuffle argument schemes for a mix-net has* aggregation *property if the computational and communication cost for its verifier does not depend on the number of mixers that compose the mix-net.*

4.3 Basic Idea

Let q, E, and $\mathcal{O}$ be as defined in the previous section. In this section, we consider an example of an aggregate argument scheme for simple permutation of $\{G_j \in E\}_{j=1,\ldots,k}$. Note that unlike shufflings, this protocol does not hide its permutation. Hence, the example scheme is not secure. Aggregate permutation argument schemes are an analogy of aggregate shuffle argument schemes in which the shuffle protocol Shuff is replaced by the simple permutation protocol Perm. This example scheme is simple but is rich enough to illustrates the essence of our aggregate shuffle argument scheme.

In the next section, we propose a basic aggregate shuffle argument scheme which is secure if players are forced to be honest in a particular circumstance. The difference between simple permutation and ElGamal shuffling is existence of masks in the latter scheme. These masks are transformed as shufflings are applied to ciphertexts. This basic scheme is constructed so that provers can collaborately cancel these transformation of additional masks. With this cancellation, the provers are able to exploit the simple property that the aggregate permutation scheme has. The data for cancellation required for γ-th mixer, are collaborately generated by all other mixers and the aggregator. This basic scheme guarantees zero-knowledge property of $\mathsf{Ind\text{-}Arg}_{\mathcal{M}^{(\gamma)},\mathcal{A}}$ only when these collaboration is done honestly.

Finally, the restriction in the basic scheme can be eliminated by requiring each mixer to verify other mixers, which results in the full scheme.

4.3.1 Permutation Matrix

Let $\delta_{ij} := 1 \bmod q$ if $i = j$ and $\delta_{ij} = 0$ otherwise. Let $\delta_{ijg} := \delta_{ij}\delta_{jg}$. We define permutation matrices over $\mathbb{Z}/q\mathbb{Z}$ as follows:

Definition 28 *Let q be a prime and $\phi(\cdot)$ be any permutation function $\phi : \{1, \ldots, k\} \to \{1, \ldots, k\}$.*

A matrix $(\phi_{ij})_{i,j=1,\cdots,k}$ is a permutation matrix over $\mathbb{Z}/q\mathbb{Z}$ *if and only if it satisfies*

$$\phi_{ij} = \begin{cases} 1 \bmod q & \textit{if } \phi(i) = j \\ 0 \bmod q & \textit{otherwise} \end{cases} .$$

The following Theorem 1 and Theorem 2 hold with respect to permutation matrices. Proofs for them are found in [70] and [65].

Theorem 16 (Theorem 1 in [70]) *Let q be a prime. A matrix $(\phi_{ij})_{i,j=1,\dots,k}$ is a permutation matrix over $\mathbb{Z}/q\mathbb{Z}$ if and only if both the equations $\sum_{h=1}^{k} \phi_{hi}\phi_{hj}\phi_{hg} = \delta_{ijg} \bmod q$ and $\sum_{h=1}^{k} \phi_{hi}\phi_{hj} = \delta_{ij} \bmod q$ hold for all i, j, and g.*

Theorem 17 (Theorem 2 in [65]) *For prime $q \bmod 3 = 2$, a matrix $(\phi_{ij})_{i,j=1,\dots,n}$ is a permutation matrix over $\mathbb{Z}/q\mathbb{Z}$ if and only if the equation $\sum_{h=1}^{k} \phi_{hi}\phi_{hj}\phi_{hg} = \delta_{ijg} \bmod q$ holds for all i, j, and g.*

4.3.2 Aggregate Permutation Argument Scheme

The following example scheme is an aggregate permutation argument scheme.

Setup: *pkey* is G and q such that $q \bmod 3 = 2$.

Perm: *pkey* and $icset^{(\gamma)} = \{G_i^{(\gamma)} \in E\}_{i=1,\dots,k}$, $wit^{(\gamma)} = \{(\phi_{ij}^{(\gamma)})_{i,j=1,\dots,k}, \emptyset\}$ is given to $\mathcal{M}^{(\gamma)}$. We assume that each element of $icset^{(\gamma)}$ is randomly generated so that it is difficult for $\mathcal{M}^{(\gamma)}$ to compute $\{a_i \in \mathbb{Z}/q\mathbb{Z}\}_{i=1,\dots,k}$ such that $\sum_{i=1}^{k}[a_i]G_i^{(\gamma)} = 0$. Then, $\mathcal{M}^{(\gamma)}$ outputs

$$ocset^{(\gamma)} = \{G_i^{(\gamma+1)} = \sum_{j=1}^{k}[\phi_{ij}^{(\gamma)}]G_j^{(\gamma)}\}_{i=1,\dots,k}.$$

Note that $ocset^{(\gamma)}$ is a simple permutation of $icset^{(\gamma)}$.

Ind-Arg: $\mathcal{A}$ sends $\{c_i^{(\gamma+1)} \in \mathbb{Z}/q\mathbb{Z}\}_{i=1,\dots,k}$ to $\mathcal{M}^{(\gamma)}$ and then $\mathcal{M}^{(\gamma)}$ returns

$$\{c_j^{(\gamma)}\}_{j=1,\dots,k} = \{\sum_{i=1}^{k} c_i^{(\gamma+1)}\phi_{ij}^{(\gamma)}\}_{j=1,\dots,k}.$$

Here, $\mathcal{A}$ chooses $\{c_j^{(\lambda)} \in_R \mathbb{Z}/q\mathbb{Z}\}_{j=1,\dots,k}$ as the one given by $\mathcal{V}$ through Agg-Arg. $\mathcal{A}$ accepts $\mathcal{M}^{(\gamma)}$ if the following equations hold:

$$\sum_{j=1}^{k}[c_j^{(\gamma)}]G_j^{(\gamma)} = \sum_{i=1}^{k}[c_i^{(\gamma+1)}]G_i^{(\gamma+1)} \quad , \quad \sum_{j=1}^{k}(c_j^{(\gamma)})^3 = \sum_{i=1}^{k}(c_i^{(\gamma+1)})^3$$

Agg-Arg: $\mathcal{V}$ sends randomly chosen $\{c_j^{(\lambda+1)} \in_R \mathbb{Z}/q\mathbb{Z}\}_{j=1,\dots,k}$ to $\mathcal{A}$. Then $\mathcal{A}$ sends $\{c_j^{(1)}\}_{j=1,\dots,k}$ to $\mathcal{V}$. $\mathcal{V}$ accepts if the following equations hold:

$$\sum_{j=1}^{k}[c_j^{(1)}]G_j^{(1)} = \sum_{i=1}^{k}[c_i^{(\lambda+1)}]G_i^{(\lambda+1)} \quad , \quad \sum_{j=1}^{k}(c_j^{(1)})^3 = \sum_{i=1}^{k}(c_i^{(\lambda+1)})^3$$

As is in the case of the example scheme that appears in the very beginning of Section 4.1 in [70], Ind-Arg is also an argument of permutation: $\{G_j^{(\gamma)}\}_{j=1,\ldots,k} \mapsto \{G_i^{(\gamma+1)}\}_{i=1,\ldots,k}$.

From the fact that equations

$$\sum_{j=1}^{k}[c_j^{(1)}]G_j^{(1)} = \sum_{i=1}^{k}[c_i^{(2)}]G_i^{(2)} = \cdots = \sum_{i=1}^{k}[c_i^{(\lambda+1)}]G_i^{(\lambda+1)} \tag{4.1}$$

$$\sum_{j=1}^{k}(c_j^{(1)})^3 = \sum_{i=1}^{k}(c_i^{(2)})^3 = \cdots = \sum_{i=1}^{k}(c_i^{(\lambda+1)})^3 \tag{4.2}$$

hold, Agg-Arg is also an argument of permutation. Computational cost for $\mathcal{V}$ is equal to that for verification of a single permutation.

Note that even from $\{c_j^{(\gamma+1)} \in_R \mathbb{Z}/q\mathbb{Z}\}_{j=1,\ldots,k}$ and $\{c_j^{(\gamma)} \in_R \mathbb{Z}/q\mathbb{Z}\}_{j=1,\ldots,k}$ which is an output of Ind-Arg protocol, permutation $\phi^{(\gamma)}$ is extractable.

4.4 Basic Aggregate Shuffle Argument Scheme

In this section, we propose a basic aggregate shuffle argument scheme that partially satisfies the proposed security requirements. That is, Ind-Arg protocol between $\mathcal{M}^{(\gamma)}$ and $\mathcal{A}$ is a zero-knowledge argument only if $\{\mathcal{M}^{(\tau)}\}_{\tau=1,\ldots,\gamma-1,\gamma+1,\ldots,\lambda}$ and $\mathcal{A}$ are honest.

4.4.1 Overview of the Strategy for the basic scheme

There are mainly three steps in our strategy for converting the example scheme to the basic aggregate shuffle argument scheme. They are (1) replacing each of its arguments of permutation with one of shuffling, (2) modifying each of its arguments to a zero-knowledge protocol, and (3) modifying each of its arguments so that its input for shufflings no longer needs to be randomly chosen.

The schemes in [70, 66] are constructed by adopting a similar strategy and we follow this strategy for construction of the basic scheme. Once we replace permutation by shuffling, we can have no simple relations between values $\{c_i^{(\gamma)}\}_{i=1,\ldots,k}$, $\{c_i^{(\gamma+1)}\}_{i=1,\ldots,k}$, $\{G_i^{(\gamma)}\}_{i=1,\ldots,k}$, and $\{G_i^{(\gamma+1)}\}_{i=1,\ldots,k}$ unlike those in the example scheme. This is because of the masking values in shuffling are transformed by the shuffling and hide the simple relations.

Therefore, the main problem that we must solve in constructing an aggregate shuffle argument scheme from the schemes in [70, 66] is how to compensate for the difference between the values in the example scheme and those in the full scheme so that we can still exploit the simple relations, i.e., Equations (4.1) and (4.2). This compensation is done by the value H that appears in our proposed scheme. Although such a compensation is provided for only single shuffling in [66, 70], in our scheme, the compensation must be provided for all of the shufflings in a mix-net. Our scheme offers such a compensation by passing the compensatory values from mixer to mixer and modifying them.

Finally, we eliminate the condition that the inputs are required to be chosen randomly. This can be done by preparing randomly chosen elements independent to input ciphertext and simultaneously executing the same shuffling to both of them.

The resulting zero-knowledge argument in which all the collaborated mixers play the role of single prover is not exactly the same as the scheme proposed in [66] but its slightly modified version. This argument is presented in Section 4.7. Hence, the security of our scheme can be reduced to that of this protocol. It is easy to see, from the theorems in [66], that this protocol is a zero-knowledge argument, as is that proposed in [66].

In our scheme, mixers and an aggregator exchange ElGamal ciphertexts many times (From Step 2 to Step 8), which makes it appear to be a very complicated scheme. However, this is only a multi-party computation of an encryption of the value H, which is an element of the commitment in a zero-knowledge argument for shuffling. Except for this multi-party computation, it is easy to see that each Ind-Arg and Agg-Arg are essentially equivalent to the simple zero-knowledge argument for shuffling given in Section 4.7.

A shuffling can be represented by a combination of multiplication by a matrix and addition of a vector to input ciphertexts. Hence, for compensation of masks transfered by these operation, we are required to transform these masks in opposite way. Hence, the most of complicated computations are composed of multiplication by a matrices and addition of vectors.

The full scheme in which mixers and aggregator may be malicious will be proposed in the next section.

4.4.2 Notation

Let $q, E, \mathcal{O}, G, M, \lambda, \{\mathcal{M}^{(\gamma)}\}_{\gamma=1,\dots,\lambda}$ be as defined before. Suppose that λ mixers $\{\mathcal{M}^{(\gamma)}\}_{\gamma=1,\dots,\lambda}$ collaboratedly mix k ElGamal ciphertexts $\mathit{icset}^{(1)} = \{(G_i^{(1)}, M_i^{(1)}) \in E^2\}_{i=1,\dots,k}$ by the public-key M. Here, each γ-th mixer $\mathcal{M}^{(\gamma)}$ shuffles ElGamal ciphertexts $\{(G_i^{(\gamma)}, M_i^{(\gamma)}) \in E^2\}_{i=1,\dots,k}$ to $\{(G_i^{(\gamma+1)}, M_i^{(\gamma+1)}) \in E^2\}_{i=1,\dots,k}$ using the public-key M. The resulting ciphertexts are $\{(G_i^{(\lambda+1)}, M_i^{(\lambda+1)}) \in E^2\}_{i=1,\dots,k}$. The public key is a set, $\{q, E, G, M\}$.

We generalize matrix operations as in the following:

1. Let $M_{k,m}(X)$ denote a set of all k times m matrices whose components are in X, where we consider the case $X = \mathbb{Z}/q\mathbb{Z}, X = E$, and X is ciphertext space of ElGamal ciphertexts EC.

2. Suppose that $A \in M_{k,m}(X)$, then we let A_{ij} denote (i,j) component of A.

3. Suppose that $A \in M_{k,m}(X)$ and $B \in M_{m,n}(Y)$, then we let AB denote a matrix in $M_{k,n}(Z)$ such that its (i,j) component is $\sum_{g=1}^{m} A_{ig} \cdot B_{gj}$. If $X = Y = Z = \mathbb{Z}/q\mathbb{Z}$, "$\cdot$" is multiplication in $\mathbb{Z}/q\mathbb{Z}$. If $X = \mathbb{Z}/q\mathbb{Z}, Y = Z = E$, "$\cdot$" is multiplication/modular-exponentiation, and is denoted by $[A_{ig}]B_{gj}$ instead. If $Y = \mathbb{Z}/q\mathbb{Z}, X = Z = E$, "$\cdot$" is multiplication/modular-exponentiation, and is denoted by $B_{gj}[A_{ig}]$ instead.

4. Suppose $C \in M_{k,m}(EC)$ and Y is the public key of an ElGamal encryption scheme, then we let $Y * C \in M_{k,m}(EC)$ be such that $(Y * C)_{ij}$ is reencryption of C_{ij} by Y.

   Suppose that $\varphi \in M_{k,m}(\mathbb{Z}/q\mathbb{Z})$, then $Y * \varphi \in M_{k,m}(EC)$ such that $(Y * \varphi)_{ij}$ is encryption of $[\varphi_{ij}]G$ by Y.

5. Suppose that $\{\phi^{(\gamma)}\}_{\gamma=1,\dots,\lambda}$ is a set of matrices. Then a product $\prod_{\gamma=1}^{\lambda} \phi^{(\gamma)}$ denotes $\phi^{(\lambda)}\phi^{(\lambda-1)}\cdots\phi^{(1)}$. Note the order of matrices in the product.

6. Suppose that $A \in M_{k,m}(X)$ and $B \in M_{k,m}(Y)$, then $A \circ B \in M_{k,m}(Z)$ is such that $(A \circ B)_{ij} = A_{ij} \cdot B_{ij}$ where $\cdot$ is a product of elements between X and Y which results in Z.

7. $[0]$ and $[1]$ are the element of $M_{1,k}(\mathbb{Z}/q\mathbb{Z})$ whose components are all, respectively, $0 \in \mathbb{Z}/q\mathbb{Z}$ and $1 \in \mathbb{Z}/q\mathbb{Z}$.

4.4.3 The Proposed Scheme with Honest Players

Setup:

Given a security parameter, Setup outputs $pkey = q, E, G, M$ where $q \bmod 3 = 2$ and the size of q is the given security parameter.

Shuff:

Let $icset^{(\gamma)} = \{(G_i^{(\gamma)}, M_i^{(\gamma)}) \in E^2 = EC\}_{i=1,\dots,k}$ be such that $(G^{(\gamma)}, M^{(\gamma)}) \in M_{k,1}(E)^2$.

1. $\mathcal{M}^{(\gamma)}$ randomly chooses a permutation matrix $\phi^{(\gamma)} \in M_{k,k}(\mathbb{Z}/q\mathbb{Z})$ and $\varphi^{(\gamma)} \in k, 1(\mathbb{Z}/q\mathbb{Z})$.

2. $\mathcal{M}^{(\gamma)}$ generates

$$(G^{(\gamma+1)}, M^{(\gamma+1)}) = ([\phi^{(\gamma)}]G^{(\gamma)} + [\varphi^{(\gamma)}]G, [\phi^{(\gamma)}]M^{(\gamma)} + [\varphi^{(\gamma)}]M).$$

3. $ocset^{(\gamma)}$ is $\{(G_i^{(\gamma+1)}, M_i^{(\gamma+1)})\}_{i=1,\dots,k}$.

Ind-Arg and Agg-Arg:

Since Ind-Arg and Agg-Arg are executed in an interleaving way, we will describe both of them together. In the following procedures, interactions between $\mathcal{M}^{(\gamma)}$ and $\mathcal{A}$ belong to Ind-Arg and interactions between $\mathcal{A}$ and $\mathcal{V}$ belong to Agg-Arg.

Suppose that $F^{(1)} \in M_{k,1}(E)$ and $F \in E$ are generated from a common reference string or an output of a random oracle so that no one is able to generate non trivial $a \in M_{1,k}(\mathbb{Z}/q\mathbb{Z})$ and $b, c \in \mathbb{Z}/q\mathbb{Z}$ such that $[a]F^{(1)} + [b]F + [c]G = \mathcal{O}$. We assume $\bar{F}^{(1)} = \bar{G}^{(1)} = \bar{M}^{(1)} = \mathcal{O}$.

1. From $\gamma = 1$ to $\gamma = \lambda$, the following steps are executed:

   (i) $\mathcal{M}^{(\gamma)}$ is given $F^{(\gamma)}, \bar{F}^{(\gamma)}, \bar{G}^{(\gamma)}$, and $\bar{M}^{(\gamma)}$.

   (ii) $\mathcal{M}^{(\gamma)}$ randomly chooses $\psi^{(\gamma)} \in M_{1,k}(\mathbb{Z}/q\mathbb{Z})$ and $\rho^{(\gamma)} \in_R \mathbb{Z}/q\mathbb{Z}$.

   (iii) $\mathcal{M}^{(\gamma)}$ generates

$$\begin{aligned}
F^{(\gamma+1)} &= [\phi^{(\gamma)}]F^{(\gamma)} + [\varphi^{(\gamma)}]F \\
\bar{F}^{(\gamma+1)} &= [\psi^{(\gamma)}]F^{(\gamma)} + [\rho^{(\gamma)}]F + \bar{F}^{(\gamma)} \\
\bar{G}^{(\gamma+1)} &= [\psi^{(\gamma)}]G^{(\gamma)} + [\rho^{(\gamma)}]G + \bar{G}^{(\gamma)} \\
\bar{M}^{(\gamma+1)} &= [\psi^{(\gamma)}]M^{(\gamma)} + [\rho^{(\gamma)}]M + \bar{M}^{(\gamma)}
\end{aligned}$$

   (iv) $\mathcal{M}^{(\gamma)}$ sends $F^{(\gamma+1)}, \bar{F}^{(\gamma+1)}, \bar{G}^{(\gamma+1)}$, and $\bar{M}^{(\gamma+1)}$ to $\mathcal{A}$, which forwards them to $\mathcal{M}^{(\gamma+1)}$. In the case $\gamma = \lambda$, $\mathcal{A}$ forwards them to $\mathcal{V}$.

2. Each of $\{\mathcal{M}^{(\gamma)}\}_{\gamma=1,\dots,\lambda}$ generates a secret key/public key pair

$$(y^{(\gamma)} \in \mathbb{Z}/q\mathbb{Z}, Y^{(\gamma)} = [y^{(\gamma)}]G)$$

   of ElGamal encryption and prove the knowledge of $y^{(\gamma)}$ to $\mathcal{A}$. Let Y be $\sum_{\gamma=1}^{\lambda} Y^{(\gamma)}$. All $Y^{(\gamma)}$ are sent to $\mathcal{A}$.

3. Let $C_1^{(1)}$ be $Y * [0]$.

   From $\gamma = 1$ to $\gamma = \lambda$, the following steps are executed:

   (i) $\mathcal{M}^{(\gamma)}$ is given $C_1^{(\gamma)}$.

   (ii) $\mathcal{M}^{(\gamma)}$ generates

$$C_1^{(\gamma+1)} = (\psi^{(\gamma)}\phi^{(\gamma)*}) * Y + (C_1^{(\gamma)}[\phi^{(\gamma)*}]) * Y$$

   where $\phi^{(\gamma)*}$ is the inverse matrix of $\phi^{(\gamma)}$.

(iii) $\mathcal{M}^{(\gamma)}$ sends $C_1^{(\gamma+1)}$ to $\mathcal{A}$, which forwards it to $\mathcal{M}^{(\gamma+1)}$. In the case $\gamma = \lambda$, $\mathcal{A}$ forwards it to $\mathcal{V}$

4. Let $C_2^{(\lambda+1)} = C_1^{(\lambda+1)}$ From $\gamma = \lambda$ to $\gamma = 1$, the following steps are executed:

   (i) $\mathcal{M}^{(\gamma)}$ is given $C_2^{(\gamma+1)}$.

   (ii) $\mathcal{M}^{(\gamma)}$ generates

$$C_2^{(\gamma)} = (C_2^{(\gamma+1)}[\phi^{(\gamma)}]) * Y$$

   (iii) $\mathcal{M}^{(\gamma)}$ sends $C_2^{(\gamma)}$ to $\mathcal{A}$, which forwards it to $\mathcal{M}^{(\gamma-1)}$. In the case $\gamma = 1$, $\mathcal{A}$ forwards it to no one.

   Each $\mathcal{M}^{(\gamma)}$ obtains

$$C_2^{(\gamma)} = \left(\left(\sum_{\gamma'=\lambda}^{1} \psi^{(\gamma')} \prod_{\zeta'=\lambda}^{\gamma'} \phi^{(\zeta')*} \right) \prod_{\zeta=\gamma}^{\lambda} \phi^{(\zeta)} \right) * Y$$

5. Let $C_3^{(1)} = C_2^{(1)}$ From $\gamma = 1$ to $\gamma = \lambda$, the following steps are executed:

   (i) $\mathcal{M}^{(\gamma)}$ is given $C_3^{(\gamma)}$.

   (ii) $\mathcal{M}^{(\gamma)}$ generates

$$C_3^{(\gamma+1)} = (C_2^{(\gamma)} \circ [\psi^{(\gamma)}]) * Y + (C_3^{(\gamma)}[\phi^{(\gamma)}]) * Y.$$

   (iii) $\mathcal{M}^{(\gamma)}$ sends $C_3^{(\gamma+1)}$ to $\mathcal{A}$, which forwards it to $\mathcal{M}^{(\gamma+1)}$. In the case $\gamma = \lambda$, $\mathcal{A}$ forwards it to $\mathcal{V}$

6. Let $C_4^{(\lambda+1)} = C_3^{(\lambda+1)}$ From $\gamma = \lambda$ to $\gamma = 1$, the following steps are executed:

   (i) $\mathcal{M}^{(\gamma)}$ is given $C_4^{(\gamma+1)}$.

   (ii) $\mathcal{M}^{(\gamma)}$ generates

$$C_4^{(\gamma)} = (C_4^{(\gamma+1)}[\phi^{(\gamma)}]) * Y$$

   (iii) $\mathcal{M}^{(\gamma)}$ sends $C_4^{(\gamma)}$ to $\mathcal{A}$, which forwards it to $\mathcal{M}^{(\gamma-1)}$. In the case $\gamma = 1$, $\mathcal{A}$ forwards it to no one.

   Each $\mathcal{M}^{(\gamma)}$ obtains

$$C_4^{(\gamma)} = \left\{ \left((\sum_{\gamma=\lambda}^{1} \psi^{(\gamma)} \prod_{\zeta=\lambda}^{\gamma} \phi^{(\zeta)*}) \circ (\sum_{\gamma'=\lambda}^{1} \psi^{(\gamma')} \prod_{\zeta'=\lambda}^{\gamma'} \phi^{(\zeta')*}) \right) \prod_{\zeta=\gamma}^{\lambda} \phi^{(\zeta)} \right\} * Y$$

7. Let $C_5^{(1)} = C_4^{(1)}$ From $\gamma = 1$ to $\gamma = \lambda$, the following steps are executed:

   (i) $\mathcal{M}^{(\gamma)}$ is given $C_5^{(\gamma)}$.

   (ii) $\mathcal{M}^{(\gamma)}$ generates

$$C_5^{(\gamma+1)} \quad = \quad (C_4^{(\gamma)} \circ [\psi^{(\gamma)}]) * Y + (C_5^{(\gamma)}[\phi^{(\gamma)}]) * Y.$$

   (iii) $\mathcal{M}^{(\gamma)}$ sends $C_5^{(\gamma+1)}$ to $\mathcal{A}$, which forwards it to $\mathcal{M}^{(\gamma+1)}$. In the case $\gamma = \lambda$, $\mathcal{A}$ forwards it to $\mathcal{V}$.

8. Each of $\{\mathcal{M}^{(\gamma)}\}_{i=1,\dots,\lambda}$ sends to $\mathcal{A}$ the following ElGamal encryptions:

$$\begin{aligned} C_7^{(\gamma)} &= ([[1](\psi^{(\gamma)} \circ \psi^{(\gamma)} \circ \psi^{(\gamma)})]G + [\theta^{(\gamma)}]Y, [\theta^{(\gamma)}]G) \\ C_8^{(\gamma)} &= ([3(\psi^{(\gamma)} \circ \psi^{(\gamma)})\phi^{(\gamma)}]G + [t^{(\gamma)}]Y, [t^{(\gamma)}]G) \\ C_9^{(\gamma)} &= ([3\psi^{(\gamma)}\phi^{(\gamma)}]G + [s^{(\gamma)}]Y, [s^{(\gamma)}]G) \end{aligned}$$

9. $\mathcal{V}$ randomly chooses $c^{(\lambda+1)} \in_R M_{1,k}(\mathbb{Z}/q\mathbb{Z})$ and sends it to $\mathcal{A}$, which forwards it to $\mathcal{M}^{(\lambda)}$ (Agg-Arg).

10. Let $r^{(\lambda+1)} = c^{(\lambda+1)}$ and $\bar{r}^{(\lambda+1)} = 0 \in \mathbb{Z}/q\mathbb{Z}$.

    From $\gamma = \lambda$ to $\gamma = 1$, the following steps are executed:

    (i) $\mathcal{M}^{(\gamma)}$ is given $r^{(\gamma+1)}$ and $\bar{r}^{(\gamma+1)}$.

    (ii) $\mathcal{M}^{(\gamma)}$ generates

$$\begin{aligned} r^{(\gamma)} &= r^{(\gamma+1)}\phi^{(\gamma)} + \psi^{(\gamma)} \\ \bar{r}^{(\gamma)} &= r^{(\gamma+1)}\varphi^{(\gamma)} + \rho^{(\gamma)} + \bar{r}^{(\gamma+1)} \end{aligned}$$

    (iii) $\mathcal{M}^{(\gamma)}$ sends $r^{(\gamma)}$ and $\bar{r}^{(\gamma)}$ to $\mathcal{A}$, which forwards it to $\mathcal{M}^{(\gamma-1)}$. In the case $\gamma = 1$, $\mathcal{A}$ forwards it to $\mathcal{V}$.

11. $\mathcal{A}$ computes

$$C_6 \quad = \quad [c^{(\lambda+1)}]C_1^{(\lambda+1)} + [c^{(\lambda+1)} \circ c^{(\lambda+1)}]C_3^{(\lambda+1)} + [1]C_5^{(\lambda+1)}$$

    and sends it to all $\{\mathcal{M}^{(\gamma)}\}_{\gamma=1,\dots,\lambda}$.

12. $\mathcal{A}$ and each $\mathcal{M}^{(\gamma)}$ compute a ciphertext:

$$C_{10}^{(\gamma)} \quad = \quad [c^{(\gamma+1)}]C_7^{(\gamma)} + [c^{(\gamma+1)} \circ c^{(\gamma+1)}]C_8^{(\gamma)} + [1]C_9^{(\gamma)}.$$

Then each $\mathcal{M}^{(\gamma)}$ sends $\mathcal{A}$ the decryption $H^{(\gamma)}$ of $C_{10}^{(\gamma)}$ and proves the validity of its decryption. $\mathcal{A}$ accepts $\mathcal{M}^{(\gamma)}$ if the following equations hold:

$$\begin{aligned}
[\bar{r}^{(\gamma)} - \bar{r}^{(\gamma+1)}]F + [r^{(\gamma)}]F^{(\gamma)} &= \bar{F}^{(\gamma+1)} - \bar{F}^{(\gamma)} + [r^{(\gamma+1)}]F^{(\gamma+1)} && (4.3)\\
[\bar{r}^{(\gamma)} - \bar{r}^{(\gamma+1)}]G + [r^{(\gamma)}]G^{(\gamma)} &= \bar{G}^{(\gamma+1)} - \bar{G}^{(\gamma)} + [r^{(\gamma+1)}]G^{(\gamma+1)} && (4.4)\\
[\bar{r}^{(\gamma)} - \bar{r}^{(\gamma+1)}]M + [r^{(\gamma)}]M^{(\gamma)} &= \bar{M}^{(\gamma+1)} - \bar{M}^{(\gamma)} + [r^{(\gamma+1)}]M^{(\gamma+1)} && (4.5)\\
[\sum_{j=1}^{k}((r_j^{(\gamma)})^3 - (r_j^{(\gamma+1)})^3)]G &= H^{(\gamma)}. && (4.6)
\end{aligned}$$

13. Each $\{\mathcal{M}^{(\gamma)}\}_{\gamma=1,\ldots,\lambda}$ sends $\mathcal{A}$ a partial decryption of C_6 with respect to their $\{y^{(\gamma)}\}$. Then, $\mathcal{A}$ generates H which is the decryption of C_6 and sends it to $\mathcal{V}$.

14. $\{\mathcal{M}^{(\gamma)}\}_{\gamma=1,\ldots,\lambda}$ jointly prove the validity of C_6 to $\mathcal{V}$ through $\mathcal{A}$. The detail is omitted but is basic.

15. $\mathcal{V}$ accepts the shuffle if the following equations hold:

$$\begin{aligned}
[\bar{r}^{(1)}]F + [r^{(1)}]F^{(1)} &= \bar{F}^{(\lambda+1)} + [c^{(\lambda+1)}]F^{(\lambda+1)}\\
[\bar{r}^{(1)}]G + [r^{(1)}]G^{(1)} &= \bar{G}^{(\lambda+1)} + [c^{(\lambda+1)}]G^{(\lambda+1)}\\
[\bar{r}^{(1)}]M + [r^{(1)}]M^{(1)} &= \bar{M}^{(\lambda+1)} + [c^{(\lambda+1)}]M^{(\lambda+1)}\\
[\sum_{j=1}^{k}((r_j^{(1)})^3 - (c_j^{(\lambda+1)})^3)]G &= H.
\end{aligned}$$

4.4.4 Security of the Proposed Scheme

Lemma 33 *The proposed scheme is complete.*

Proof. The completeness follows from trivial but patient computations. □

Lemma 34 *Each* Ind-Arg *protocol between $\mathcal{M}^{(\gamma)}$ and $\mathcal{A}$ is a computational zero-knowledge argument for shuffling if $\{\mathcal{M}^{(\tau)}\}_{\tau=1,\ldots,\gamma-1,\gamma+1,\ldots,\lambda}$ and $\mathcal{A}$ are honest.*

Proof. Suppose that $F^{(\gamma)}, \bar{F}^{(\gamma)}, r^{(\gamma+1)}$, and $\bar{r}^{(\gamma+1)}$ are randomly generated so that no one is able to generate $a \in M_{1,k}(\mathbb{Z}/q\mathbb{Z}), b, c \in \mathbb{Z}/q\mathbb{Z}$, such that $[a]F^{(\gamma)} + [b]F + [c]G = \mathcal{O}$. But these are cases when $\{\mathcal{M}^{(\tau)}\}_{\tau=1,\ldots,\gamma-1,\gamma+1,\ldots,\lambda}$ and $\mathcal{A}$ are honest. Then, it is easy to see that each protocol is essentially the same as the zero-knowledge argument presented in Section 4.7.2. □

Lemma 35 Agg-Arg *protocol between $\mathcal{A}$ and $\mathcal{V}$ is a computational zero-knowledge argument for shuffling if $\mathcal{V}$ is honest.*

	$\mathcal{V}$	$\mathcal{M}^{(\gamma)}$	$\mathcal{A}$
# of multiplications/exponentiations in E	$10k$	$28k$	$13k\lambda$
communication bits	$1,500k$	$5,500k$	$5,500k\lambda$

Table 4.1: Complexity of our scheme

Proof. It is easy to see that the protocol is essentially the same as the zero-knowledge argument presented in Section 4.7.2. □

4.5 Full Scheme

In the scheme proposed in the previous section, each Ind-Arg between $\mathcal{M}^{(\gamma)}$ and $\mathcal{A}$ is a zero-knowledge argument only if $\{\mathcal{M}^{(\tau)}\}_{\tau=1,\ldots,\gamma-1,\gamma+1,\ldots,\lambda}$ and $\mathcal{A}$ are honest. This honesty can be guaranteed if the scheme additionally includes the following procedures:

1. Each of $\mathcal{M}^{(\gamma)}$ proves to $\{\mathcal{M}^{(\zeta)}\}_{\zeta=1,\ldots,\gamma-1,\gamma+1,\ldots,\lambda}$ in zero-knowledge that it correctly generated $F^{(\gamma+1)}$ from $F^{(\gamma)}$. This can be done by using the scheme proposed in [66], [70], or the scheme proposed in Section 4.7.2.

2. Each of $\mathcal{M}^{(\gamma)}$ verifies that Equations (4.3)-(4.6) hold for all $\gamma = 1, \ldots, \lambda$.

This completes the description of the full scheme. Note that the above additional procedures are executed by only $\{\mathcal{M}^{(\gamma)}\}_{\gamma=1,\ldots,\lambda}$ and $\mathcal{A}$. Hence, the cost of $\mathcal{V}$ does not increase.

In the former procedure, each $\mathcal{M}^{(\gamma)}$ is able to verify that $F^{(\gamma)}$ and $\bar{F}^{(\gamma)}$ are generated correctly by $\{\mathcal{M}^{(\zeta)}\}_{\zeta=1,\ldots,\gamma-1}$. Each $\mathcal{M}^{(\gamma)}$ is able to verify that $\{F^{(\zeta)}\}_{\zeta=\gamma+1,\ldots,\lambda}$ $\{\bar{F}^{(\zeta)}\}_{\zeta=\gamma+1,\ldots,\lambda}$ is generated correctly by $\{\mathcal{M}^{(\zeta)}\}_{\zeta=1,\ldots,\lambda}$ from the former procedure. Hence, if $\mathcal{M}^{(\zeta)}$ verifies that if Equations (4.3)-(4.6) hold for all $\gamma = \zeta+1, \ldots, \lambda$ in the latter procedure, $\mathcal{M}^{(\zeta)}$ is able to confirm that $c^{(\zeta+1)}$ and $\bar{c}^{(\zeta+1)}$ are generated correctly by $\{\mathcal{M}^{(\gamma)}\}_{\gamma=\zeta+1,\ldots,\lambda}$. This can be easily understood from Lemma 3 in [66]. Therefore, the condition for Lemmas 34 and 35 are guaranteed to be satisfied in the case when all the players cannot be trusted in the full scheme.

Moreover, since all the $\{\mathcal{M}^{(\gamma)}\}_{\gamma=1,\ldots,\lambda}$ are required to be honest for $\mathcal{A}$ to accept all Ind-Arg, the following Lemma clearly holds.

Lemma 36 *If $\mathcal{A}$ accepts all* Ind-Arg*, $\mathcal{V}$ accepts* Agg-Arg*.* □

Therefore, the following theorem holds.

Theorem 18 *The full scheme satisfies completeness, soundness, and zero-knowledge properties.* □

	$\mathcal{V}$	$\mathcal{M}^{(\gamma)}$	$\mathcal{A}$
# of multiplications/exponentiations in E	$640k$	$320k$	$640k\lambda$
communication bits	$750,000k$	$1,500,000k$	$1,500,000k\lambda$

Table 4.2: Complexity of the scheme in [1]

	$\mathcal{V}$	$\mathcal{M}^{(\gamma)}$	$\mathcal{A}$
# of multiplications/exponentiations in E	$10k\lambda$	$6k$	$10k\lambda$
communication bits	$1,500k\lambda$	$1,500k$	$1,500k\lambda$

Table 4.3: Complexity of a scheme that uses [66]

4.6 Efficiency

In this section, we estimate the computational cost for each player in our proposed scheme and show that the scheme has an aggregation property. Then we compare the efficiency of our scheme with that of previous schemes.

The most costly computation in our scheme is multiplications/modular-exponentiations in E. In comparison, the cost of other computations is negligibly small. Therefore, we estimate the computational cost according to the number of multiplications/modular-exponentiations required in our scheme. We also estimate the amount of data that each player needs to communicate.

We first consider the case when $\mathcal{A}$ can be trusted but $\{\mathcal{M}^{(\gamma)}\}_{\gamma=1,\ldots,\lambda}$ may be malicious. In this case, $\{\mathcal{M}^{(\gamma)}\}_{\gamma=1,\ldots,\lambda}$ do not verify each other, but $\mathcal{A}$ verifies them. The results of estimation are given in Table 4.1, where we assumed that $|q|$ is 160 and elements of E can be represented by a 161-bit string. For a comparison, an estimated computational and communication complexity of the aggregate shuffle argument scheme of Abe [1] are given in Table 4.2, and those of a scheme in which the ordinary shuffle argument proposed in [66] is repeatedly used multiple times are given in Table 4.3 (This scheme allows the case when $q \bmod 3 = 1$).

In the general case when $\mathcal{A}$ cannot be trusted, every $\mathcal{M}^{(\gamma)}$ is required to prove the correctness of $F^{(\gamma)}$ to all $\{\mathcal{M}^{(\zeta)}\}_{\zeta=1,\ldots,\gamma-1,\gamma+1,\ldots,\lambda}$. Here, to avoid the execution of these protocols independently to every other mixer, we assume that all $\{\mathcal{M}^{(\zeta)}\}_{\zeta=1,\ldots,\gamma-1,\gamma+1,\ldots,\lambda}$ collaborately play the role of a single verifier to $\mathcal{M}^{(\gamma)}$. This is possible by using a multi-party coin-tossing protocol. Therefore, in the general case when $\mathcal{A}$ cannot be trusted, the computational cost that each $\mathcal{M}^{(\gamma)}$ additionally requires is that of $\mathcal{A}$. The operation of this scheme would be more practical if each mixer published Fiat-Shamir transformation of their proofs instead of interactively proving their correctness to the single verifier that is formed by the collaboration of all other mixers. This would enable all mixers to independently verify the correctness of the prover. Now we have proved the last theorem:

Theorem 19 *The proposed scheme has an aggregation property.* □

4.7 A New Computational Zero-knowledge Argument for Shuffling

We introduce a computational zero-knowledge argument for shuffling, the security of which the security of our scheme is reduced to.

4.7.1 Notation

Let q, E, G, M, k be as defined before. Let $\{(G_i^{(2)}, M_i^{(2)})\}_{i=1,\dots,k}$ be a shuffled result of $\{(G_i^{(1)}, M_i^{(1)})\}_{i=1,\dots,k}$ by using the public-key M, such that $(G^{(1)}, M^{(1)}) \in M_{k,1}(E)^2$ and $(G^{(2)}, M^{(2)}) \in M_{k,1}(E)^2$. The public key is a set, $\{q, E, G, M\}$. A permutation matrix $\phi^{(1)} \in M_{k,k}(\mathbb{Z}/q\mathbb{Z})$ and a column vector $\varphi^{(1)} \in M_{k,1}(\mathbb{Z}/q\mathbb{Z})$ is the witness of the above shuffle, i.e.,

$$(G^{(2)}, M^{(2)}) = ([\varphi^{(1)}]G + [\phi^{(1)}]G^{(1)}, [\varphi^{(1)}]M + [\phi^{(1)}]M^{(1)}) \tag{4.7}$$

Therefore, Proving the correctness of the shuffle is equivalent to proving the existence of an $\varphi^{(1)} \in M_{k,1}(\mathbb{Z}/q\mathbb{Z})$ and a permutation matrix $\phi^{(1)} \in M_{k,k}(\mathbb{Z}/q\mathbb{Z})$ which satisfies Eq. (4.7). $\mathcal{P}$ is a prover and $\mathcal{V}$ is a verifier.

4.7.2 Zero-Knowledge Argument for Shuffling

Let us next introduce an argument for shuffling.

1. Both $\mathcal{P}$ and $\mathcal{V}$ are given q, E, G, M, $(G^{(1)}, M^{(1)})$ and $(G^{(2)}, M^{(2)})$. $\mathcal{P}$ is additionally given $\varphi^{(1)}$ and $\phi^{(1)}$.

2. $\mathcal{P}$ and $\mathcal{V}$ engage in a coin-tossing protocol to generate a random $F \in_R E, F^{(1)} \in M_{k,1}(E)$.

3. $\mathcal{P}$ uniformly and randomly chooses $\psi \in_R \mathbb{Z}/q\mathbb{Z}, \psi^{(1)} \in_R M_{1,k}(\mathbb{Z}/q\mathbb{Z})$, $t \in_R \mathbb{Z}/q\mathbb{Z}, t^{(1)} \in_R M_{1,k}(\mathbb{Z}/q\mathbb{Z})$, $s^{(1)} \in_R M_{1,k}(\mathbb{Z}/q\mathbb{Z})$, and $y \in_R \mathbb{Z}/q\mathbb{Z}$, and then computes:

$$\begin{aligned} Y &= [y]G \\ F^{(2)} &= [\varphi^{(1)}]F + [\phi^{(1)}]F^{(1)} \\ F' &= [\psi]F + [\psi^{(1)}]F^{(1)} \end{aligned} \tag{4.8}$$

$$\begin{aligned}
G' &= [\psi]G + [\psi^{(1)}]G^{(1)} \\
M' &= [\psi]M + [\psi^{(1)}]M^{(1)} \\
(C, D) &= ([[1](\psi^{(1)} \circ \psi^{(1)} \circ \psi^{(1)})]G + [t]Y, [t]G) \\
(C', D') &= ([3(\psi^{(1)} \circ \psi^{(1)})\phi^{(1)}]G + [t^{(1)}]Y, [t^{(1)}]G) \\
(C'', D'') &= ([3\psi^{(1)}\phi^{(1)}]G + [s^{(1)}]Y, [s^{(1)}]G)
\end{aligned}$$

Then, $\mathcal{P}$ sends $Y, F^{(2)}, F', G', M', (C, D), (C', D'), (C'', D'')$ to $\mathcal{V}$ as a commitment.

4. V randomly chooses $c^{(1)} \in_R M_{1,k}(\mathbb{Z}/q\mathbb{Z})$ and sends it to P.

5. P generates $y' \in \mathbb{Z}/q\mathbb{Z}$ and generates

$$\begin{aligned}
r^{(1)} &= \psi^{(1)} + c^{(1)}\phi^{(1)} \\
r &= \psi + c^{(1)}\varphi^{(1)} \\
H &= (C + [c^{(1)} \circ c^{(1)}]C' + [c^{(1)}]C'') - [y](D + [c^{(1)} \circ c^{(1)}]D' + [c^{(1)}]D'') \\
H' &= [y'](D + [c^{(1)} \circ c^{(1)}]D' + [c^{(1)}]D'').
\end{aligned}$$

Then $\mathcal{P}$ sends $r, r^{(1)}$, H, and H' to $\mathcal{V}$.

6. $\mathcal{V}$ randomly chooses $c' \in \mathbb{Z}/q\mathbb{Z}$ and sends it to $\mathcal{P}$.

7. $\mathcal{P}$ generates

$$r' = c'y + y'$$

and sends it to $\mathcal{V}$.

8. $\mathcal{V}$ accepts the shuffle if the following equations hold:

$$\begin{aligned}
[r'](D + [c^{(1)} \circ c^{(1)}]D' + [c^{(1)}]D'') &= [c'](C + [c^{(1)} \circ c^{(1)}]C' + \sum_{i=1}^{k}[c^{(1)}]C' - H) + H' \\
[r]F + [r^{(1)}]F^{(1)} &= F' + [c^{(1)}]F^{(2)} && (4.9) \\
[r]G + [r^{(1)}]G^{(1)} &= G' + [c^{(1)}]G^{(2)} && (4.10) \\
[r]M + [r^{(1)}]M^{(1)} &= M' + [c^{(1)}]M^{(2)} && (4.11) \\
[\sum_{j=1}^{k}(r_j^{(1)3} - c_j^{(1)3})]G &= H && (4.12)
\end{aligned}$$

Otherwise, V rejects the shuffle.

4.7.3 Properties of the Proposed Argument for Shuffling

Theorem 20 *The protocol is complete, sound as long as the discrete logarithm problem is difficult to solve, and honest-verifier computational zero-knowledge.*

The theorem can be proven in the same way that Theorem 3, 4, and 5 in [66] are proven.

4.8 Completeness

The following relations aid in understanding the completeness of the protocol.

1. Since $ocset^{(\lambda)}$ is

$$\begin{aligned}
&(G^{(\lambda+1)}, M^{(\lambda+1)}) \\
=\ &([\prod_{\gamma=1}^{\lambda}\phi^{(\gamma)}]G^{(1)} + [\sum_{\gamma=1}^{\lambda}(\prod_{\theta=\gamma+1}^{\lambda}\phi^{(\theta)})\varphi^{(\gamma)}]G,\ [\prod_{\gamma=1}^{\lambda}\phi^{(\gamma)}]M^{(1)} + [\sum_{\gamma=1}^{\lambda}(\prod_{\theta=\gamma+1}^{\lambda}\phi^{(\theta)})\varphi^{(\gamma)}]M),
\end{aligned}$$

it is clearly a shuffle of $icset^{(1)}$.

2. The following relations hold with respect $F^{(\lambda+1)}, \bar{F}^{(\lambda+1)}, \bar{G}^{(\lambda+1)}, \bar{M}^{(\lambda+1)}$:

$$\begin{aligned}
F^{(\lambda+1)} &= [\prod_{\gamma=1}^{\lambda}\phi^{(\gamma)}]F^{(1)} + [\sum_{\gamma=1}^{\lambda}(\prod_{\theta=\gamma+1}^{\lambda}\phi^{(\theta)})\varphi^{(\gamma)}]F \\
\bar{F}^{(\lambda+1)} &= \sum_{\gamma=1}^{\lambda}[\psi^{(\gamma)}]F^{(\gamma)} + \sum_{\gamma=1}^{\lambda}[\rho^{(\gamma)}]F \\
&= \sum_{\gamma=1}^{\lambda}[\psi^{(\gamma)}\prod_{\eta=1}^{\gamma-1}\phi^{(\eta)}]F^{(1)} + \left[\sum_{\gamma=1}^{\lambda}\psi^{(\gamma)}\sum_{\eta=1}^{\gamma-1}(\prod_{\theta=\eta+1}^{\gamma-1}\phi^{(\theta)})\varphi^{(\eta)} + \sum_{\gamma=1}^{\lambda}\rho^{(\gamma)}\right]F \\
&= \sum_{\gamma=1}^{\lambda}[\psi^{(\gamma)}\prod_{\eta=1}^{\gamma-1}\phi^{(\eta)}]F^{(1)} \\
&\qquad + \left[\sum_{\{\gamma,\eta|\gamma>\eta,(\gamma,\eta)\in(1,\ldots,k)^2\}}\psi^{(\gamma)}(\prod_{\theta=\eta+1}^{\gamma-1}\phi^{(\theta)})\varphi^{(\eta)} + \sum_{\gamma=1}^{\lambda}\rho^{(\gamma)}\right]F \\
\bar{G}^{(\lambda+1)} &= \sum_{\gamma=1}^{\lambda}[\psi^{(\gamma)}\prod_{\eta=1}^{\gamma-1}\phi^{(\eta)}]G^{(1)} \\
&\qquad + \left[\sum_{\{\gamma,\eta|\gamma>\eta,(\gamma,\eta)\in(1,\ldots,k)^2\}}\psi^{(\gamma)}(\prod_{\theta=\eta+1}^{\gamma-1}\phi^{(\theta)})\varphi^{(\eta)} + \sum_{\gamma=1}^{\lambda}\rho^{(\gamma)}\right]G \\
\bar{M}^{(\lambda+1)} &= \sum_{\gamma=1}^{\lambda}[\psi^{(\gamma)}\prod_{\eta=1}^{\gamma-1}\phi^{(\eta)}]M^{(1)}
\end{aligned}$$

$$+\left[\sum_{\{\gamma,\eta|\gamma>\eta,(\gamma,\eta)\in(1,\ldots,k)^2\}}\psi^{(\gamma)}\,(\prod_{\theta=\eta+1}^{\gamma-1}\phi^{(\theta)})\varphi^{(\eta)}+\sum_{\gamma=1}^{\lambda}\rho^{(\gamma)}\right]M.$$

3. The following relation holds with respect to $C_1^{(\lambda+1)}$:

$$C_1^{(\lambda+1)} \;=\; \left(\sum_{\gamma=\lambda}^{1}\psi^{(\gamma)}\prod_{\zeta=\lambda}^{\gamma}\phi^{(\zeta)*}\right)*Y$$

4. The following relation holds with respect to $C_3^{(\lambda+1)}$:

$$C_3^{(\lambda+1)} \;=\; \left(\left(\sum_{\gamma=\lambda}^{1}\psi^{(\gamma)}\prod_{\zeta=\lambda}^{\gamma}\phi^{(\zeta)*}\right)\circ\left(\sum_{\gamma'=\lambda}^{1}\psi^{(\gamma')}\prod_{\zeta'=\lambda}^{\gamma'}\phi^{(\zeta')*}\right)\right)*Y.$$

5. The following relation holds with respect to $C_5^{(\lambda+1)}$:

$$C_5^{(\lambda+1)}$$
$$=\;\left\{\left(\sum_{\gamma=\lambda}^{1}\psi^{(\gamma)}\prod_{\zeta=\lambda}^{\gamma}\phi^{(\zeta)*}\right)\circ\left(\sum_{\gamma'=\lambda}^{1}\psi^{(\gamma')}\prod_{\zeta'=\lambda}^{\gamma'}\phi^{(\zeta')*}\right)\circ\left(\sum_{\gamma'=\lambda}^{1}\psi^{(\gamma')}\prod_{\zeta'=\lambda}^{\gamma'}\phi^{(\zeta')*}\right)\right\}*Y.$$

6. The following relation holds with respect to $r^{(1)}, \bar{r}^{(1)}$:

$$\begin{aligned}
r^{(1)} &= c^{(\lambda+1)}\left(\prod_{\gamma=1}^{\lambda}\phi^{(\gamma)}]\right)+\sum_{\gamma=1}^{\lambda}\psi^{(\gamma)}\prod_{\eta=1}^{\gamma-1}\phi^{(\eta)}\\
\bar{r}^{(1)} &= \sum_{\gamma=1}^{\lambda}(r^{(\gamma+1)}\varphi^{(\gamma)}+\rho^{(\gamma)})\\
&= \sum_{\gamma=1}^{\lambda}\left(\left(c^{(\lambda+1)}\left(\prod_{\tau=\gamma+1}^{\lambda}\phi^{(\tau)}]\right)+\sum_{\tau=\gamma+1}^{\lambda}\psi^{(\tau)}\prod_{\eta=\gamma+1}^{\tau-1}\phi^{(\eta)}\right)\varphi^{(\gamma)}+\rho^{(\gamma)}\right)\\
&= c^{(\lambda+1)}\sum_{\gamma=1}^{\lambda}\left(\prod_{\tau=\gamma+1}^{\lambda}\phi^{(\tau)}]\right)\varphi^{(\gamma)}\\
&\qquad+\sum_{\{\tau,\gamma|\tau>\gamma,(\tau,\gamma)\in(1,\ldots,k)^2\}}\psi^{(\tau)}(\prod_{\eta=\gamma+1}^{\tau-1}\phi^{(\eta)})\varphi^{(\gamma)}+\sum_{\gamma=1}^{\lambda}\rho^{(\gamma)}.
\end{aligned}$$

4.9 Conclusion

We proposed a new aggregate shuffle argument scheme which is much more efficient that that proposed by Abe in [1]. The computational cost required for the verifier in our scheme is less than 1/60 of that in Abe's scheme. This is mainly because our scheme is based on the scheme in [66] while Abe's scheme is based on the scheme in [137]. Our scheme is particularly effective for large-scale voting such as in a national election.

Chapter 5

Compiler from Σ-Protocol to Deniable Zero-Knowledge

5.1 Introduction

Zero-knowledge interactive proof systems, first proposed by Goldwasser, Micali and Rackoff [82], have the significant property that they leak no knowledge other than the validity of the proven assertion. It has been shown in [79] that every NP-statement can be proved in zero-knowledge if one-way functions exist. Because of these properties, these proof systems have been found to be very important tools in many cryptographic applications.

The original definition of zero-knowledge considered the setting in which a single prover and a verifier execute only one instance of a protocol. However, in more realistic settings, where many computers are connected through the Internet and protocols may be concurrently executed, many verifiers may interact with the same (or many) prover(s) simultaneously. Proof systems that are zero-knowledge even in such a setting are called concurrent zero-knowledge (cZK).

The term "concurrent zero-knowledge" was coined by Dwork, Naor, and Sahai in [54], and they observed that the zero-knowledge property does not necessarily carry over to the concurrent setting. Indeed, Goldreich and Krawczyk showed in [78] the existence of protocols that are ordinary zero-knowledge and yet fail dramatically to be zero-knowledge in the concurrent scenario. Moreover, Kilian, Petrank, and Rackoff showed a negative result in [97] such that any language that has a 4-move black-box cZK proof argument is in BPP. Canetti, Kilian, Petrank, and Rosen proved in [39] that black-box cZK proof systems for any non-trivial language require a non-constant number ($\tilde{\Omega}(\log k)$) of rounds.

Despite these negative results, many protocols that achieve round efficiency and adequate security in concurrent settings have been presented under some additional assumptions. These include cZK under the timing assumption [54], resettable zero-knowledges (rZK) in the public-key model [37] and the weak public-key model [108], cZK in the auxiliary string model [50], universally composable zero-knowledge in the common reference string (CRS) model [36], etc.

Now even though the notions of cZK in the CRS model and the auxiliary string model achieve a kind of zero-knowledgeness, they lose some of the spirit of the original definition. In particular, as is mentioned in [54], these models are not sufficient to yield the property of deniability. An interactive protocol is called deniable zero-knowledge if the transcript of its interaction does not leak any evidence of interaction. For example the simulators in the CRS model and the auxiliary string model are powerful enough to control their strings, while the verifiers in these models are never able to control them. As a result, a verifier interacting with a prover in these models are able to output a transcript that cannot be generated by the verifier alone. Hence, the verifier's possession of such a transcript is an evidence of its interaction with the prover.

The question of whether or not there exists a constant-round deniable cZK argument under additional assumptions was studied by Pass [129]. He showed that no black-box constant-round deniable cZK argument for non-trivial language exists in the CRS model. It is also shown there that a 2-move constant-round straight-line witness extractable deniable cZK argument exists for any $\mathcal{NP}$-language in the random oracle (RO) model. However, this argument system is inefficient since it relies on the cut and choose technique. Fischlin also proposed in [62] a communication efficient straight-line witness extractable zero-knowledge proof that can be applied to deniable cZK argument. However, this argument still requires rather large computational complexity.

Besides proving the existence of a certain kind of zero-knowledge protocol for every language in $\mathcal{NP}$, it is also important for practical applications to construct a compiler which transforms Σ-protocols to certain kind of zero-knowledge proof systems or arguments. Σ-protocols are 3-move special honest verifier zero-knowledge protocols with special soundness property. We call such a compiler, a *Σ-compiler.* Σ-compilers are useful in practical point of view since many efficient Σ-protocols for many relations are proposed until now. Σ-compilers for cZK argument and that for rZK and concurrently sound protocol with small overhead are proposed by [50] and [153], respectively. The results of Pass [129] and Fischlin [62] mentioned above are indeed proposal of Σ-compiler for 2-move straight-line witness extractable deniable zero-knowledge argument in the random oracle model. However, their Σ-compiler have large overheads.

Our Contribution:

In this chapter, we propose a more efficient Σ-compiler for deniable zero-knowledge argument *with very small overhead* in the random oracle model. We also proved that our Σ-compiler simultaneously provides the following properties:

- The resulting protocol is **deniable** resettable zero-knowledge (rZK).
- The resulting protocol is 2-move (constant-round).
- The computational and communication overhead of compilation is very small.
- The resulting protocol is resettably-sound argument of knowledge (RSAK).

We note that the result of Pass also provides resettable soundness and can, with slight modification, provide rZK property. (These facts have not been shown before.) In this sense, the essential improvement in our scheme is with respect to efficiency.

The overhead of our compiler is computation, verification, and transmission of Fiat-Shamir transformations of any Σ-protocol, i.e., NIZK-argument. This Σ-protocol is chosen independently to the proven statement and we choose the most efficient one within an allowed assumptions. On the other hand, the compiler in [129] requires the verifier to generate the corresponding NIZK argument by Cut & Choose method instead of Fiat-Shamir transformation. Thus, its overhead will be larger than that of ours in the proportion of the security parameter to one. Fischlin [62] improved efficiency of communication complexity but its computational overhead is till large.

Although our protocol itself is efficient, it does not provide a straight-line simulator. It only provides a rather complicated but still polynomially bounded simulator. This can be compared to cZK protocols under the timing assumption [54] and rZK protocols in the public-key models [37, 108]. All types of protocols achieve deniability with efficient non straight-line simulator in different models. Our protocol is the first that provides efficient non straight-line simulator in the random oracle model.

Here, rZK and RSAK [11] are, respectively, stronger notions of cZK and argument of knowledge. The requirement for rZK is more restricting than that of cZK in the sense that proof systems or arguments must be cZK even if verifiers in these protocols are able to reset provers. Meanwhile, protocols which are still argument of knowledge against provers who can reset verifiers are called RSAK.

The notion of RSAK was proposed by Barak et al. in [11]. As is pointed out in [37], rZK arguments of knowledge are impossible to achieve for non-trivial languages as long as the ability of knowledge extractor is limited to black-box oracle access to the prover. By exchanging the roles of provers and verifiers, and those of simulators and knowledge extractors, this impossibility holds for zero-knowledge RSAK. However, the negative result are only with respect to the standard definition and may not hold in the random oracle model where simulators and knowledge extractors are more powerful than, respectively, the provers and verifiers in the sense that they are able to control random oracles.

Applications:

The security of Chaum's undeniable signature scheme was recently formally proved in the random oracle model [124], where the confirmation protocol and the disavowal protocol are both 4-move. Now by applying our Σ-compiler, we can obtain a very efficient 2-move undeniable signature scheme.

Further, our protocols are concurrent zero-knowledge and resettably-sound argument of knowledge while Chaum's protocols are not. Hence, our protocols remain secure in a setting where parties in protocols are implemented by devices, which cannot reliably keep state (e.g., smart card), being maliciously reset to prior state. And, the resulting protocols are available in the setting when it is impossible or too costly to generate fresh randomness on the fly.

Another application is deniable identification. In Schnorr's identification scheme, a cheating verifier (Bob) will compute his challenge as a hashed value of the first message of the prover (Alice). Then the transcript of the protocol is an evidence of the fact that Alice executed the protocol with Bob. So the privacy of Alice is not protected. In this sense, Schnorr's identification scheme is not deniable. Now by applying our Σ-compiler, we can obtain a very efficient 2-move deniable identification scheme. These applications are shown in Section 5.5.

Organization:

This chapter is organized as follows. Section 5.2 describes the basic concepts involved in constructing the proposed compiler in the random oracle model. Section 5.3 proposes our Σ-compiler for 2-move deniable rZK and RSAK with no essential loss of efficiency in the random oracle model. Section 5.4 discusses the efficiency of our compiler. Section 5.5 demonstrates the example of compilation and discusses its advantage over the previous scheme.

5.2 Basic Concepts and Approach

5.2.1 Deniable ZK in the Random Oracle Model

We consider the random oracle model, where a prover P and a (malicious) verifier V^* have access to a random oracle O. In the definition of zero-knowledge, however, a distinguisher D does not have access to O. Hence S has only to generate a view of V^* by providing V^* with a fake random oracle O' which S can manipulate arbitrarily.

Therefore, V^* cannot necessarily generate his view by himself in the *real* world, where D has access to O. This means that V^* can use the view as an evidence of the fact that P executed the protocol, and P cannot deny it. Indeed, V^* can show the view as an evidence to the third party who has access to O.

On the other hand, in the definition of *deniable* zero-knowledge, D has access to the random oracle O. So S must be able to generate a view of V^* which cannot be distinguished from the real one by D

who has access to O. Therefore, in a *deniable* zero-knowledge protocol, there is no evidence of the fact that P executed the protocol because V^* can generate his view. Hence P can deny that fact.

5.2.2 Our Approach

In the model of *deniable* zero-knowledge protocols, a prover P, a (malicious) verifier V^*, and a distinguisher D have access to the same random oracle O as shown below.

$$(O \leftrightarrow P) \leftrightarrow (V^* \leftrightarrow O) \text{ and } D \leftrightarrow O. \tag{5.1}$$

In this model, P cannot see the queries of V^* to O nor control the answers of O.

In the simulation of S, however, the simulator S can provide V^* with a fake random oracle O' as follows.

$$O \leftrightarrow S \leftrightarrow (V^* \leftrightarrow O') \text{ and } D \leftrightarrow O. \tag{5.2}$$

In the simulated world, S can see the queries of V^* to O' and control the answers of O'.

In the methods of Pass and Fischlin, S sees the queries of V^* to O', but does not control the answers of O'. On the other hand, we construct S which both sees the queries of V^* and controls the answers of O'. This is a critical part of our approach. We use a similar technique for knowledge extractor as well.

5.2.3 Notation and Security Definitions

Definition 29 *A function $f(n)$ is* negligible *if* $^\forall c > 0\ ^\exists N\ ^\forall n > N, \quad f(n) < \frac{1}{n^c}$

Definition 30 *Let $R \subset \{0,1\}^* \times \{0,1\}^*$ be a relation. We say that (x, w)* satisfies *R if $(x, w) \in R$, where x is called an* common input *and w is called a* witness*. Define $L_R = \{x|\ ^\exists w$ s.t., $(x, w) \in R\}$. Also let $R_n = R \cap (\{0,1\}^n \times \{0,1\}^n)$.*

Definition 31 *A deterministic Turing machine G_R is called a* generator for relation R *if there exists a polynomial $Q(\cdot)$ such that on input of a random tape $r_G \in \{0,1\}^{Q(n)}$, it produces $(x, w) \in R_n$.*

We say a generator G_R is an invulnerable generator for relation R *if for any polynomial time nonuniform algorithm A,* $\Pr[(x, A(x)) \in R]$ *is negligible in n. Here, $(x, w) \leftarrow G_R$ and the probability is taken over the distribution of random tapes of G_R and C.*

The following generator G_R is an example of invulnerable generator when the discrete logarithm problem is difficult to solve: G_R takes a random tape in $\{0,1\}^n$ as an input, chooses a prime p of size n and $g \in (\mathbb{Z}/p\mathbb{Z})^*$ such that the order q of $\langle g \rangle$ is a prime and close to n, randomly choose $\alpha \in \mathbb{Z}/q\mathbb{Z}$, and outputs $(x, w) = ((g, y = g^\alpha), \alpha)$.

Notation 1 *Let RO denote a random oracle and $RO(t)$ denote the output of the random oracle RO when it is given t.*

Canetti et al. defined the notion of rZK in [37]. Here, we introduce its deniable variant in the random oracle model.

Convention 1 *Without loss of generality, we assume that each message of the verifier contains the entire communication history in the session that the message relates to. Furthermore, we assume that the prover is memoryless: it responds to each message based solely on the received message and on its input and random input.*

Definition 32 *An interactive protocol (P, V) for a relation R is said to be* (black-box) deniable rZK in the random oracle model *if, there exists a probabilistic polynomial time simulator S such that, for every probabilistic polynomial time adversary V^*, the following two distribution ensembles are computational indistinguishable by every probabilistic polynomial time distinguisher who can access a random oracle RO: Let each distribution be indexed by a sequence of common inputs $\bar{x} = (x_i)_{i=1,\ldots,poly(n)}$ and the corresponding sequence of prover's auxiliary-inputs $\bar{w} = (w_i)_{i=1,\ldots,poly(n)}$ such that $(x_i, w_i) \in R_n$ for all i.*

Distribution 1: *This is defined by the following random process which depends on P and V^*.*

1. *Randomly select and fix RO and $t = poly(n)$ random-tapes, $\{r_i\}_{i=1,\ldots,t}$, for P, resulting in deterministic strategies $P^{(i,j)} = P_{x_i,w_i,r_j}$ defined by $P_{x_i,w_i,r_j}(\alpha) = P(x_i, w_i, r_j, \alpha)$, for $i, j \in \{1, \ldots, t\}$. Each $P^{(i,j)}$ is called an incarnation of P. P is allowed to access the random oracle RO.*
2. *Machine V^* is allowed to arbitrarily interact with all the incarnations of P (i.e., V^* sends arbitrary messages to each of the $P^{(i,j)}$ and obtains the responses of $P^{(i,j)}$ to such messages) and the random oracle RO. Once V^* decides it is done interacting with the $P^{(i,j)}$'s, it (i.e., V^*) produces an output based on its view of these interactions.*

Distribution 2: *The output of $S(\bar{x})$. RO is randomly selected and fixed at first. S has black-box access to the random oracle RO and V^{**} and is able to control a random oracle RO^*. V^{**} is the same as V^* except that the random oracle that V^{**} accesses is RO^* rather than RO.*

It is important to notice that the simulator is able to control the random oracle, i.e., choose the outputs of the random oracle, that the verifier accesses but is unable to control the one that the distinguisher accesses. The latter property is the key feature of deniability. The former property comes from the fact that the simulator can black-box access the verifier and is not essential for deniability. Our simulator leverages this property for simulation while the simulator of Pass only uses the property that it can catch random oracle queries of the verifier but does not fully leverage the former property. To leverage this property, our simulator rewinds the verifier for polynomial times.

Barak et al. defined the notion of RSAK in [11]. Here , we introduce its random oracle variant in the following.

Definition 33 *A* resetting attack *of a cheating prover P^* on a* resettable verifier V *is defined by the following two-step random process, indexed by a security parameter n.*

1. *Uniformly select and fix $t = poly(n)$ random tapes, denoted $\{r_i\}_{i=1,\ldots,t}$, for V, resulting in deterministic strategies $V^{(j)}(x) = V_{x,r_j}$ defined by $V_{x,r_j}(\alpha) = V(x, r_j, \alpha)$, where $x \in \{0,1\}^n$ and $j \in \{1, \ldots, t\}$. Each $V^{(j)}(x)$ is called an* incarnation *of V.*

2. *On input 1^n, machine P^* is allowed to initiate $poly(n)$-many interactions with the $V^{(j)}(x)$'s. The activity of P^* proceeds in rounds. In each round P^* chooses $x \in \{0,1\}^n$ and $j \in \{1, \ldots, t\}$, thus defining $V^{(j)}(x)$, and conducts a complex session with it.*

Definition 34 *We say that a protocol (P, V) is a* resettably-sound *argument for a relation R if the following two conditions hold:*

Resettable-Completeness: *Consider an arbitrary polynomial-size resetting attack, and suppose that in some session, after selecting an incarnation $V^{(j)}(x)$, the attacker follows the strategy P. Then, if $x \in L_R$ then $V^{(j)}(x)$ rejects with negligible probability.*

Resettable-soundness: *For every polynomial-size resetting attack, the probability that in some session the corresponding $V^{(j)}(x)$ has accepted and $x \notin L_R$ is negligible.*

Definition 35 *We then say that a protocol (P, V) is* RSAK for R in the random oracle model *if,*

1. *(P, V) is a resettably-sound argument for L_R, and*

2. *There exists a probabilistic expected polynomial oracle machine E such that for every polynomial-time resetting attack P^*, the probability that $E^{P^*}(1^n)$ outputs a witness for the input selected in the last session is at most negligibly smaller than the probability that P^* convinces V in the last session.*

   Here, E has black-box access to the random oracle RO and P^{**} and is able to control a random oracle RO^*. P^{**} is the same as P^* except that the random oracle that P^{**} accesses is RO^* rather than RO.

The focus on the last session of the resetting attack is done for simplicity and is valid without loss of generality.

5.2.4 Σ-Protocols and Σ-Compilers

Σ-protocols are first introduced by Cramer, Damgård and Schoenmakers in [46]. Informally, a Σ-protocol is itself a 3-round public-coin special honest verifier zero-knowledge protocol with special soundness in the knowledge-extraction sense. Since its introduction, Σ-protocols have been proved a very powerful cryptographic tool and are widely used in numerous important cryptographic applications including digital signatures by using the famous Fiat-Shamir methodology [60].

Definition 36 *3-round public-coin protocol for a pair of prover and verifier (P, V) denoted by $(A_R, C_R, Z_R, V_R, S_R)$ is said to be a* Σ-protocol for relation R *if the following hold:*

Let a, e, z denote for the first, the second and the third message respectively of the protocol (P, V). Let (x, w) be such that $(x, w) \in R_n$ and a random tape $r_P \in \{0,1\}^n$. A_R, for P, is a function that generates the first message a from $(x, w) \in R_n$ and $r_P \in \{0,1\}^n$. C_R denotes the space from which e is randomly chosen by V. C_R is implicitly indexed by n. Z_R, for P, is a function that generates z from x, w, e and r_P. V_R, for V, is a function such that $qV_R(a, e, z) = 1/0$ denote the acceptance/rejection of V. S_R, for the simulator, is a function that generates a simulated view (x, a, e, z) from x, e, and the random tape $r_S \in \{0,1\}^n$.

- *Completeness. If P and V follow the protocol, V always accepts.*

- *Special soundness. From any common input x and any pair of accepting conversations on input x, (a, e, z) and (a, e', z') where $e \neq e'$, one can efficiently compute w such that $(x, w) \in R$.*

- *Special honest verifier zero-knowledge (SHVZK). There exists a polynomial-time simulator S_R, which on input x and a random challenge string e, outputs an accepting conversation of the form (a, e, z), with the same probability distribution as conversations between the honest P, V on input x.*

Σ-compilers are compilers from Σ-protocols for a relation R to a certain kind of zero-knowledge proof system or argument for the relation R.

Definition 37 *A compiler is said to be Σ-compiler for zero-knowledge protocol with a property C if the compiler, given a description of Σ-protocol for relation R, gives a description of a zero-knowledge proof/argument of knowledge of witness of relation R with the property C.*

5.3 Proposed Σ-Compiler in the Random Oracle Model

5.3.1 Σ-Compiler

Now we present our Σ-compilers which output 2-move deniable rZK and RSAK protocols in the random oracle model. respectively. Suppose that there exists a Σ-protocol for a relation R. Let P and V be a prover and a verifier, respectively, and let x be a common input and w be the private input to P, where $(x, w) \in R$. Then our 2-move protocol proceeds as follows.

1. V chooses a random $(\bar{x}, \bar{w}) \in R$ and sends $\bar{x}$ to P. V then proves that he knows $\bar{w}$ non-interactively.

2. P proves that she knows w or $\bar{w}$ non-interactively.

We use Fiat-Shamir transformation [60] to construct non-interactive arguments, and use the technique of [46] to construct the P's message (OR protocol).

Although our protocol is very simple, it has not been known that it is deniable zero-knowledge. This is probably because many people believed that the straight-line extractability of the witness of the verifier is necessary. Our main contribution is then to prove that the above simple construction is indeed deniable zero-knowledge.

We first present Construction 1 which is deniable zero-knowledge only, but not rZK. We next show Construction 3 which is rZK, where Construction 3 is obtained by applying a known technique to Construction 1.

Let n be a security parameter, and suppose that there exists a Σ-protocol denoted by (A_R, C_R, Z_R, S_R) for a relation R_n. Let x be a common input and w be the private input to P, where $(x, w) \in R_n$. Also suppose that there exists a Σ-protocol denoted by $(A_{\bar{R}}, C_{\bar{R}}, Z_{\bar{R}}, S_{\bar{R}})$ for a relation $\bar{R}_n$ which has an invulnerable generator $G_{\bar{R}}$, where $\bar{R}_n$ can be R_n. We assume that $C_R = C_{\bar{R}}$ and $\frac{1}{|C_R|} = \frac{1}{|C_{\bar{R}}|}$ is negligible in n.

P has random tapes $r_E, r_S, r_P \in \{0,1\}^n$, and V has tapes $r_V, r_G \in \{0,1\}^n$. They are allowed to have access to a random oracle RO whose output is uniformly distributed over $C_R = C_{\bar{R}}$.

Construction 1 *1. V generates $(\bar{x}, \bar{a}, \bar{z})$ as follows, and sends them to P.*

(i) V generates $(\bar{x}, \bar{w}) \in \bar{R}_n$ by giving r_G to $G_{\bar{R}}$.

(ii) V generates $\bar{a}$ by giving $(\bar{x}, \bar{w}, r_V)$ to $A_{\bar{R}}$ and sends $(\bar{x}, \bar{a})$ to RO.

(iii) V receives the output of the random oracle as $\bar{e}$.

(iv) V generates $\bar{z}$ by giving $(\bar{x}, \bar{w}, \bar{e}, r_V)$ to $Z_{\bar{R}}$.

2. P generates $((a, e, z), (\bar{a}', \bar{e}', \bar{z}'))$ as follows, and sends them to V.

(i) *P computes $\bar{e} = RO(\bar{x}, \bar{a})$, and verifies that $V_R(\bar{x}, \bar{a}, \bar{e}, \bar{z}) = 1$.*

(ii) *P generates $\bar{e}' \in C_{\bar{R}}$ by using the random tape r_E, and P generates a simulated view $(\bar{x}, \bar{a}', \bar{e}', \bar{z}')$ for the relation $\bar{R}$ by giving $(\bar{x}, \bar{e}', r_S)$ to $S_{\bar{R}}$.*

(iii) *P generates a by giving (x, w, r_P) to A_R, and P sends $(x, a, \bar{x}, \bar{a}')$ to RO.*

(iv) *Let d be the output of the random oracle. Then P let $e = d \oplus \bar{e}'$*

(v) *P generates z by giving (x, w, e, r_P) to Z_R.*

3. *V accepts iff $RO(x, a, \bar{x}, \bar{a}') = e \oplus \bar{e}'$, $V_{\bar{R}}(\bar{x}, \bar{a}', \bar{e}', \bar{z}') = 1$, and $V_R(x, a, e, z) = 1$*

An example that enjoys this construction is shown in Section 5.5.2.

5.3.2 RSAK of the Compiled Protocol

Theorem 21 *The above protocol is RSAK for relation R in the random oracle model.*

We first give the idea of the proof.

The message sent from the verifier to the prover is a Fiat-Shamir transformation of a Σ-protocol for the relation $\bar{R}$. Hence, even resetting provers, who are not allowed to choose random oracles, are unable to extract the witness for the relation $\bar{R}$.

The message sent from the prover to the verifier is a Fiat-Shamir transformation of a proof of knowledge of either the witness for the relation R or the witness for the relation $\bar{R}$. Hence, if the knowledge extractor runs the prover for multiple of times by choosing different random oracles for each time, the knowledge extractor is able to extract either of witnesses. However, the prover is unable to extract the witness for the relation $\bar{R}$ as we know, the witness that the knowledge extractor is able to extract is the one for the relation R. After all, the above strategy is a standard style for proof of argument of knowledge.

The heart of the proof is to show that the expected running time of the knowledge extractor is bounded by a polynomial of n.

We are now going to present the knowledge extractor E for the compiled protocol, which proves the theorem

Proof. Let $\mathfrak{R}_D$ be the set of all random oracles that maps $\{0,1\}^*$ to a domain D. Let $\bar{V}$ be a set of honest verifiers $\{V^{(j)}\}_{j=1,\ldots,t=poly(n)}$ and and $(\bar{r}_V, \bar{r}_G)$ be a set $\{(r_V^{(j)}, r_G^{(j)}) \in (\{0,1\}^n)^2\}_{j=1,\ldots,t}$. We assume each $(r_V^{(j)}, r_G^{(j)})$ is given to $V^{(j)}$.

Suppose that $q(n)$-time P^* with a random tape $r_{P^*} \in \{0,1\}^{poly(n)}$, after accessing random oracles RO and $\bar{V}$ with a random tape $(\bar{r}_V, \bar{r}_G)$, successes or fails to convinces $\bar{V}$ with respect to the common input selected in the last session.

Let $\mathfrak{S}$ be the set of all $(r_{P^*}, \bar{r}_V, \bar{r}_G, RO) \in \{0,1\}^{poly(n)} \times (\{0,1\}^n)^t \times (\{0,1\}^n)^t \times \mathfrak{R}_D$ and $\mathfrak{C}$ be the set of all $(r_{P^*}, \bar{r}_V, \bar{r}_G, RO)$ such that P^* convinces $\bar{V}$ in the last session. We say "a random oracle RO is given to E" if E is able to access RO.

Definition 38 *An knowledge extractor E is given black-box access to $q(n)$-time P^*. E is able to control the random oracle that P^* accesses. Suppose that $T := (r_{P^*}, \bar{r}_V, \bar{r}_G, RO) \in \mathfrak{S}$ is given to E. In a case $T \in \mathfrak{C}$, let $(x, (\bar{x}, \bar{a}, \bar{e}, \bar{z}), (a, e, z), (\bar{a}', \bar{e}', \bar{z}'))$ be the view of the last session, where $e + \bar{e}' = RO(x, a, \bar{x}, \bar{a}')$.*

E works as follows:

1. *If given T is in $\mathfrak{S} \setminus \mathfrak{C}$, E outputs $\perp$. Clearly, E is able to decide whether T is in $\mathfrak{C}$ or not.*

2. *If given T is in $\mathfrak{C}$, E repeatedly invokes the following extraction step until it obtains such an accepting view $(x_1, (\bar{x}_1, \bar{a}_1, \bar{e}_1, \bar{z}_1), (a_1, e_1, z_1), (\bar{a}'_1, \bar{e}'_1, \bar{z}'_1))$ that*

$$\bar{x}_1 = x \ , \ (\bar{x}_1, \bar{a}_1, \bar{e}_1, \bar{z}_1) = (\bar{x}, \bar{a}, \bar{e}, \bar{z}) \ , \ (a_1, \bar{a}'_1) = (a, \bar{a}') \tag{5.3}$$

hold. Note that the case when the relation $(e_1, z_1, \bar{e}'_1, \bar{z}'_1) = (e, z, \bar{e}', \bar{z}')$ also holds is included.

If $e_1 + \bar{e}'_1 = e + \bar{e}'$, E outputs $\perp$. Otherwise, E outputs either w such that $(x, w) \in R_n$ or $\bar{w}$ such that $(\bar{x}, \bar{w}) \in \bar{R}_n$, which is possible from property of Σ-protocols.

Extraction Step: *E interacts with P^* by obeying the strategy of the honest $\bar{V}$ with random tape $(\bar{r}_V, \bar{r}_G)$. Whenever P^* resets $V^{(j)}$, E uses the same $(r_V^{(j)}, r_G^{(j)})$ again to execute $V^{(j)}$. E simulates the random oracle that P^* accesses. Without loss of generality, we are able to consider only the case when P^* never sends same queries to the random oracle. The responses of the random oracle is chosen as in the following.*

(i) Until P sends the query $(x, a, \bar{x}, \bar{a}')$ to the random oracle, E interacts with P^* by using the same T. We call this random oracle query determining query.

(ii) Once P^* sends this determining query, E replaces the unused part of T with random elements. E interacts with P^* using this this modified random tapes and random oracle hereafter.

Lemma 37 *The probability that P^* fails to convince $\bar{V}$ and the probability that E outputs $\perp$ at Step 1 are the same.* □

Lemma 38 *The probability that E outputs $\perp$ in Step 2 is $\frac{1}{|C_R|}$.*

Proof. Since the value of $RO(x_1, a_1, \bar{x}_1, \bar{a}'_1)$ is chosen in Step 2 independently to that of $RO(x, a, \bar{x}, \bar{a}')$ for given T. The probability that both of them coincide each other is $\frac{1}{|C_R|}$. □

Lemma 39 *E runs in time expected polynomial of n.*

Proof. Let $\mathfrak{S}^T \subset \mathfrak{S}$ be the set of all possible elements in $\mathfrak{S}$ that can be chosen by E when it is given $T \in \mathfrak{C}$. Let $\mathfrak{C}^T \subset \mathfrak{S}^T$ be the set of all elements in $\mathfrak{S}^T$ that leads the pair of P^* and $\bar{V}$ to obtain an accepting view $(x, (\bar{x}, \bar{a}, \bar{e}, \bar{z}), (a, e, z), (\bar{a}', \bar{e}', \bar{z}'))$ that satisfies Equations (5.3). Let $\epsilon(T)$ denotes $\frac{|\mathfrak{C}^T|}{|\mathfrak{S}^T|}$.[1]

The expected running time T_E of E is

$$\begin{aligned} T_E &= \left(\frac{1}{|\mathfrak{S}|} \sum_{T \in \mathfrak{S} \setminus \mathfrak{C}} 1 + \frac{1}{|\mathfrak{S}|} \sum_{T \in \mathfrak{C}} \epsilon(T) \sum_{m=1}^{\infty} m(1-\epsilon(T))^{m-1} \right) q(n) \\ &= \left(\frac{|\mathfrak{S} \setminus \mathfrak{C}|}{|\mathfrak{S}|} + \frac{1}{|\mathfrak{S}|} \sum_{T \in \mathfrak{C}} \frac{1}{\epsilon(T)} \right) q(n) = \left(\frac{|\mathfrak{S} \setminus \mathfrak{C}|}{|\mathfrak{S}|} + \frac{1}{|\mathfrak{S}|} \sum_{T \in \mathfrak{C}} \frac{|\mathfrak{S}^T|}{|\mathfrak{C}^T|} \right) q(n). \end{aligned}$$

We note that E is also required to complete interaction with P^* in order to decides whether or not $T \in \mathfrak{C}$.

Let I be $\frac{1}{|\mathfrak{S}|} \sum_{T \in \mathfrak{C}} \frac{|\mathfrak{S}^T|}{|\mathfrak{C}^T|}$. Let $T \in \mathfrak{C}$ and $T' \in \mathfrak{C}$. We define equivalence relation $T \sim T'$ by $T' \in \mathfrak{C}^T$. Let $[T]$ be the equivalent class of T. Since $\mathfrak{S}^T = \mathfrak{S}^{T'}$ and $\mathfrak{C}^T = \mathfrak{C}^{T'}$ if $T \sim T'$,

$$I = \frac{1}{|\mathfrak{S}|} \sum_{T \in \mathfrak{C}/\sim} |\mathfrak{C}^T| \cdot \frac{|\mathfrak{S}^T|}{|\mathfrak{C}^T|} = \frac{1}{|\mathfrak{S}|} \sum_{T \in \mathfrak{C}/\sim} |\mathfrak{S}^T|.$$

Suppose that $T \in \mathfrak{S}$ is also in $\mathfrak{S}^{T'}$. Then, there exists a unique determining query sent by P^* in the interactions yield by T. Conversely, if we choose one random oracle query of the form $(x, a, \bar{x}, \bar{a}')$ sent by P^* in the interactions yield by T, there exists at most one equivalence class $[T']$ and corresponding $\mathfrak{S}^{T'}$ such that the corresponding determining query is this chosen random oracle query.

Since P^* can send random oracle queries at most $q(n)$ times, each $T \in \mathfrak{S}$ may be in $\mathfrak{C}^{T'}$ for at most $q(n)$ different equivalence class $[T']$. Hence,

$$I \le \frac{1}{|\mathfrak{S}|} \cdot q(n)|\mathfrak{S}| = q(n).$$

Therefore, the expected running time (step) T_E of E is

$$T_E \le \left(\frac{|\mathfrak{S}| - |\mathfrak{C}|}{|\mathfrak{S}|} + q(n) \right) q(n) \le (1 + q(n))q(n),$$

which is upper bounded by polynomial of n.

□

[1]This value can be considered as the probability that $\bar{V}$ accepts P^* when the determining query corresponding to T is sent.

Lemma 40 *The probability that E outputs $\bar{w}$ such that $(\bar{x}, \bar{w}) \in \bar{R}_n$ is negligible in n.*

Proof. Suppose that the lemma does not hold. Then we are able to show that there exists a polynomially bounded Turing machine C, which is given the output $\bar{x}$ of $G_{\bar{R}}$ with randomly chosen $r_G \in \{0,1\}^n$, is able to extracts $\bar{w}$ such that $(\bar{x}, \bar{w}) \in \bar{R}_n$ with non-negligible probability. This is against the assumption that $G_{\bar{R}}$ is invulnerable.

The extraction can be done by running E and P^*. However, in this execution, rather than using $\bar{x}$ generated by feeding a random tape $r_G \in \{0,1\}^n$ to $G_{\bar{R}}$ by itself, E uses a common input $\bar{x}$ generated by $G_{\bar{R}}$ which is given randomly chosen unknown r_G as a problem. Although E does not have the corresponding $\bar{w}$, E is able to simulate $\bar{a}, \bar{e}, \bar{z}$ by choosing an appropriate random oracle RO.

E fails to choose an appropriate RO oracle only with negligible probability in n, otherwise, the simulated view and the original view are indistinguishable to P^*. $\square$

The resettable completeness is clear. Hence, the theorem follows from Lemmas 37-40. $\square$

5.3.3 Deniable Zero-knowledge of the Compiled Protocol

Theorem 22 *The above protocol is deniable cZK in the random oracle model.*

We first present the idea of the simulation. We first consider an oversimplified case where V^* never aborts the protocol but always outputs $\bar{z}$ such that $V_{\bar{R}}(\bar{x}, \bar{a}, \bar{e}, \bar{z}) = 1$ whenever it sends $\bar{a}$ to RO as long as P does not abort. The idea of the simulation is based on the following two abilities of S:

1. S is able to perfectly simulate the view without choosing random oracles if it is able to extract the corresponding $\bar{w}$ when V^* outputs the first message $(\bar{x}, \bar{a})$ to P. In this case, the required number of steps of S is the same order to that of real P.

2. S is able to perfectly simulate the view if it is able to choose random oracles. In this case, the required number of steps of S is the same order to that of real P.

By the second ability, S is able to extracts the corresponding $\bar{w}$. By the first ability, S, by using this $\bar{w}$, is able to simulate the entire view that includes no chosen output values of random oracles.

Roughly, S works as in the following:

1. First, S begins simulation by interacting with V without the knowledge of any $\bar{w}$. During the this interaction V^* sends $(\bar{x}, \bar{a})$ to the random oracle such that S does not have the corresponding $\bar{w}$. Then, S completes to simulate the entire view twice by, each time, randomly choosing random oracles to obtain two $\bar{z}$. This is possible since V^* always output a corresponding $\bar{z}$ within polynomial time. Then, from these $\bar{z}$, S is able to generate the corresponding $\bar{w}$. The required number of

steps for these simulations is less than the twice the number of steps P is required to complete the protocol.

2. Next, S rewinds V^* to the phase where V^* sent $(\bar{x}, \bar{a})$ to the random oracle and continues simulation. If S is required to prove the knowledge of either w or $\bar{w}$, S does so using $\bar{w}$. This is possible for S without choosing random oracles. Then, continues simulating the view. If V^* again sends another $(\bar{x}, \bar{a})$ to random oracle such that S does not have the corresponding $\bar{w}$, S extracts the corresponding $\bar{w}$ by the same method described above. Repeating these procedures, S succeeds in simulating the view that includes no chosen output values of random oracles. Therefore, the protocol is deniable.

Since, V^* sends such a $(\bar{x}, \bar{a})$ less than the number of steps of real P, the total number of steps required for the simulation to extracts all $\bar{w}$ is less than two times the square of the number of the steps required for the real protocol.

In the general case where V^* may abort without outputting $\bar{z}$, the simulation gets to be much complicated. In this case, S repeats simulation with chosen output values of random oracles for many times to obtains two $\bar{z}$ for each $(\bar{x}, \bar{a})$ but gives up if its number of repetitions exceeds a certain number of times $(2n^2Q(n))$. Such a multiple trial is required since there may be (with some probability) a case when V^* aborts without outputting $\bar{z}$ when S is simulating with chosen output values of random oracles but outputs $\bar{z}$ when S is simulating with the output values of real random oracle. The number of times the S tries to obtain $\bar{z}$ is $2n^2Q(n)$, which is indexed by ctr-III in the description of the simulator in Definition 39. Here, $Q(n)$ is the running time of P^*, which is indexed by ctr-II in the description of the simulator.

It turns out that such a simulation can be successful with the probability larger than 1/2. Hence, repeating it for polynomial times enable the successful simulation with overwhelming probability. The number of time S repeats this simulation is n, which is indexed by ctr-I. in the description of the simulator. The total running time of the simulator is $n^3Q(n)^2$.

The above simulator is not black-box simulator since it needs to know the running time of verifier. However, it is easy to construct a black-box simulator from the proposed simulator. The black-box simulator executes the proposed simulator repeatedly until it complete simulation, by, in each execution, it increase the order of time that it assumes as the running time of verifier.

We now prove the theorem by presenting the simulator.

Proof. The following definition 39 describes the simulator S which simulate an interaction between P and $Q(n)$-time verifier V^* of Construction 1, where $Q(\cdot)$ is a polynomial. Although the following description is very complicated, its structure is very simple as described in the above idea of the simulation.

S works in two phases. One is simulation in phase 1 in which S uses the true random oracles and the other is simulation phase 2 in which S uses fake random oracles. S begins its simulation in the former phase and call the latter phase routine from it. These phases are indicated by an index phase.

In phase 1, S generates a view by accessing true random oracle and using the witnesses $\bar{w}$. However, S initially possess no $\bar{w}$. Hence, whenever S is sent the corresponding $\bar{x}$ for the first time, S enters (3(i)iii) to the phase 2 in which it try to extract $\bar{w}$. If S has already extracted the corresponding $\bar{w}$, then it uses $\bar{w}$ as a trapdoor for generating the view (3ii). Suppose S has all necessary witnesses $\bar{w}$, then S stops within $Q(n)$ steps (3(i)i and etc.). $\mathsf{ctr\text{-}II}$ counts the number of this repetitions.

However, S initially has no $\bar{w}$. Hence, the remaining problem is that how much step is required for S to extract each $\bar{w}$ in phase 2 and that whether S succeeds to extract all the necessary $\bar{w}$.

The strategy of S in phase 2 for extraction of each $\bar{w}$ is as follows: Complete interaction with V^* for $2n^2Q(n)$ times. $\mathsf{ctr\text{-}III}$ counts the number of this repetitions. This can be done in straight-line manner by using fake random oracles (the third "-" in 4(v)ii). In some of these executions, S may obtain the corresponding $(\bar{x}, \bar{a}, \bar{z})$. If S succeeded for more than twice, it succeeds to extract $\bar{w}$ (the second "-" in 4(v)ii) and everything is all right.

However, in some cases, S fails for extraction, that is, S receives the corresponding $\bar{z}$ at most once. In such a case, S assumes it will not receive that $\bar{z}$ in phase 1 also and thus the corresponding $\bar{w}$ is unnecessary. Then, S forgets that $\bar{w}$ and enters phase 1 again (4iii).

S gets into the problem if $\bar{w}$ is not extracted by S in phase 2 but is required in phase 1 (3(ii)iv). Although this happens with the probability smaller than 1/2, it happens with non-negligible probability. Hence, S repeats entire simulation for n times (2) for the total simulation to be successful with overwhelming probability. $\mathsf{ctr\text{-}I}$ counts the number of this repetitions.

Step 3i is only for managing the information for rewinding schedule. The list $\mathcal{L}_{ext}$ keeps data for on-going extraction in phase 2. The list $\mathcal{L}_{wit}$ keeps extracted witnesses. The list $\mathcal{L}_{RO}$ keeps input/output pairs of simulated random oracles.

Definition 39 (Simulator S) *S is given a common input x and V^*'s random tape. S uses V^*, as a black-box. S first invokes V^* by feeding it x and the random tape.*

1. *S sets $\mathsf{phase} = 1$, $\mathsf{ctr\text{-}I} = \mathsf{ctr\text{-}II} = \mathsf{ctr\text{-}III} = 0$, and $\mathcal{L}_{ext} = \mathcal{L}_{wit} = \mathcal{L}_{RO} = \emptyset$.*

2. *If $\mathsf{ctr\text{-}I} < n$, S increments $\mathsf{ctr\text{-}I}$ and continues. Otherwise; aborts simulation.*

3. *S works as in the followings ($\mathsf{phase} = 1$).*

   (i) In the case S caught random oracle query of the form $(\bar{x}, \bar{a})$ from V^*, S works as in the following.

i. *S sets* phase $= 2$. *If* ctr-II $\geq Q(n)$ *goto the step 2.*

ii. *S save every necessary information to rewind V^*, $\mathcal{L}_{RO}$, and* ctr-II *to this current states.*

iii. *If* phase $= 2$, *S sets* ctr-III $= 0$ *and goes to the step 4i in the* phase $= 2$.

iv. *S obtains $\bar{e} = RO(\bar{x}, \bar{a})$ from the random oracle, sends it to V^*, and adds an entry $((\bar{x}, \bar{a}), \bar{e})$ to list $\mathcal{L}_{RO}$ if there is no entry of the form $((\bar{x}, \bar{a}), \cdot)$ in $\mathcal{L}_{RO}$. Then, S increments* ctr-II *by* 1.

(ii) *In the case S received a message of the form $(\bar{x}, \bar{a}, \bar{e}, \bar{z})$ from V^*, S works as in the following:*

i. *S increments* ctr-II *by* 1.

ii. *S obtains $\bar{e} = RO(\bar{x}, \bar{a})$ from the random oracle, and S adds an entry $((\bar{x}, \bar{a}), \bar{e})$ to list $\mathcal{L}_{RO}$ if there is no entry of the form $((\bar{x}, \bar{a}), \cdot)$ in $\mathcal{L}_{RO}$.*

iii. *If $V(\bar{x}, \bar{a}, \bar{e}, \bar{z}) = 1$ does not hold, S goes to the step 3.*

iv. *S search the list $\mathcal{L}_{wit}$ and finds an entry $(\bar{x}, \bar{w})$. If S could not find it, S goes to the step 2.*

v. *S generates $((a, e, z), (\bar{a}', \bar{e}', \bar{z}'))$, which is a Fiat-Shamir transformation of the proof of knowledge of either $\bar{w}$ of w, as in the following:*

A. *S generates a simulated view (x, a, e, z) for the relation R_n^{ρ} by giving x, randomly chosen $e \in C_R$, and a fresh random tape.*

B. *S generates $\bar{a}'$ by giving $\bar{x}, \bar{w}$, and a fresh random tape to $A_{\bar{R}}$. Then obtains $d = RO(x, a, \bar{x}, \bar{a}')$ by giving $(x, a, \bar{x}, \bar{a}')$ to the true random oracle RO.*

C. *S adds an entry $((x, a, \bar{x}, \bar{a}'), d)$ to list $\mathcal{L}_{RO}$.*

D. *S let $\bar{e}' = d \oplus e$ and S generates $\bar{z}'$ by giving $\bar{x}, \bar{w}, \bar{e}'$, and the random tape used to generate $\bar{a}'$ to $Z_{\bar{R}}$.*

Then, S sends $((a, e, z), (\bar{a}', \bar{e}', \bar{z}'))$ to V^*.

(iii) *If V^* stops, S terminates its simulation.*

(iv) *Otherwise; S increments* ctr-II *by* 1 *and goes to the step 3.*

4. *S works as in the followings (*phase $= 2$*).*

(i) *S adds a entry $(\bar{x}, \bar{a})$ to list $\mathcal{L}_{ext}$.*

(ii) *S save every necessary information to rewind V^*, $\mathcal{L}_{RO}$, and* ctr-II *to this current states.*

(iii) *If* ctr-III $\geq 2n^2 Q(n)$, *S sets* phase $= 1$ *and rewinds to the step 3(i)ii in the* phase $= 1$.

(iv) *S randomly choose $\bar{e} \in C_{\bar{R}}$, sends it to V^* as the corresponding output of the random oracle, adds a entry $((\bar{x}, \bar{a}), \bar{e})$ to list $\mathcal{L}_{RO}$, and increments* ctr-II *by* 1.

(v) *Then, S works as in the following:*

i. *In the case S caught random oracle query of the form $(\bar{x}, \bar{a})$ from V^*, S increments* ctr-II *by 1 and works as in the following:*

   - *In the case when there is an entry $((\bar{x}, \bar{a}), \bar{e})$ in $\mathcal{L}_{RO}$, S sends $\bar{e}$ to V^*.*
   - *In the case when there is no entry $((\bar{x}, \bar{a}), \bar{e})$ in $\mathcal{L}_{RO}$, S chooses random $\bar{e} \in C_{\bar{R}}$, adds an entry $((\bar{x}, \bar{a}), \bar{e})$ in $\mathcal{L}_{RO}$, and sends $\bar{e}$ to V^*.*

ii. *In the case S received a message of the form $(\bar{x}, \bar{a}, \bar{e}, \bar{z})$ such that $V_{\bar{R}}(\bar{x}, \bar{a}, \bar{e}, \bar{z}) = 1$ and $((\bar{x}, \bar{a}), \bar{e})$ in $\mathcal{L}_{RO}$ from V^*, S increments* ctr-II *by 1 and works as in the following:*

   - *Suppose that there is an entry $(\bar{x}, \bar{a})$ in $\mathcal{L}_{ext}$. Then, S first modifies the entry to $(\bar{x}, \bar{a}, \bar{e}, \bar{z})$. Next, S rewinds the V^*, $\mathcal{L}_{RO}$, and* ctr-II *to Step 4ii.*
   - *Suppose that there is an entry $(\bar{x}, \bar{a}, \bar{e}', \bar{z}')$ in $\mathcal{L}_{ext}$ such that $\bar{e} \neq \bar{e}'$. Then, S first generates $\bar{w}$ such that $(\bar{x}, \bar{w}) \in \bar{R}_n$ from $(\bar{x}, \bar{a}, \bar{e}', \bar{z}')$ and $(\bar{x}, \bar{a}, \bar{e}, \bar{z})$. Next, S adds an entry $(\bar{x}, \bar{w})$ to $\mathcal{L}_{wit}$. Next, S deletes the entry from $\mathcal{L}_{ext}$. Finally, S sets* phase $= 1$ *and rewinds V^*, $\mathcal{L}_{RO}$, and* ctr-II *to Step 3(i)ii in the* phase $= 1$.
   - *Otherwise; S generates $((a, e, z), (\bar{a}', \bar{e}', \bar{z}'))$, which is a Fiat-Shamir transformation of the proof of knowledge of either $\bar{w}$ of w, as in the following:*

     A. *S generates a simulated view $(\bar{x}, \bar{a}', \bar{e}', \bar{z}')$ for the relation $\bar{R}_n$ by giving $\bar{x}$, randomly chosen $\bar{e}' \in C_{\bar{R}}$, and a fresh random tape to $S_{\bar{R}}$.*

     B. *S generates a simulated view (x, a, e, z) for the relation R_n by giving x, randomly chosen $e \in C_R$, and a fresh random tape to S_R.*

     C. *S writes an entry $((x, a, \bar{x}, \bar{a}'), d)$ in the list $\mathcal{L}_{RO}$, where $d = e \oplus \bar{e}'$. If an entry $((x, a, \bar{x}, \bar{a}'), d')$ such that $d \neq d'$ already exists, S fails and stops.*

     Then, S sends $((a, e, z), (\bar{a}', \bar{e}', \bar{z}'))$ to V^*.

iii. *In either case when V^* stopped or case when* ctr-II $\geq Q(n)$, *S sets* phase $= 1$, *increase* ctr-III *by 1, and rewinds V^*, $\mathcal{L}_{RO}$, and* ctr-II *to Step 3(i)ii in the* phase $= 1$.

Lemma 41 *Simulator S stops within $n^3 Q(n)^2$ steps. Hence, it is polynomial-time simulator.* □

Lemma 42 *Simulator S succeeds in simulating the view of protocols with overwhelming probability as long as $|\frac{1}{C_{\bar{R}}}|$ is negligible in n.*

Proof. Suppose that ϵ_n is the probability that V^* sends $(\bar{x}, \bar{a}, \bar{e}, \bar{z})$ such that $\bar{e} = RO(\bar{x}, \bar{a})$ and $V_{\bar{R}}(\bar{x}, \bar{a}, \bar{e}, \bar{z}) = 1$ when V^* sends $(\bar{x}, \bar{a})$ to RO (for a single value of ctr-III). Then, the probability η_n that at least one such $(\bar{x}, \bar{a}, \bar{e}, \bar{z})$ is sent by V^* in the first half of phase 2 (for ctr-III $= 0$ to ctr-III $= n^2 Q(n) - 1$

for a single value of $\mathsf{ctr\text{-}II}$) is

$$\eta_n = 1 - (1-\epsilon_n)^{n^2Q(n)}$$

The same for the latter half of phase 2 (for $\mathsf{ctr\text{-}III} = n^2Q(n)$ to $\mathsf{ctr\text{-}III} = 2n^2Q(n) - 1$ for a single value of $\mathsf{ctr\text{-}II}$).

From Reset Lemma in [16], the probability τ_n that at least two different $(\bar{x}, \bar{a}, \bar{e}, \bar{z})$ can be obtained during the phase 2 (for $\mathsf{ctr\text{-}III} = 0$ to $\mathsf{ctr\text{-}III} = 2n^2Q(n) - 1$ for a single value of $\mathsf{ctr\text{-}II}$) satisfies

$$\eta_n \leq \frac{1}{|C_{\bar{R}}|} + {\tau_n}^{1/2}.$$

Then, the probability γ_n that one fails to obtains two different $(\bar{x}, \bar{a}, \bar{e}, \bar{z})$ during the phase 2 (for $\mathsf{ctr\text{-}III} = 0$ to $\mathsf{ctr\text{-}III} = 2n^2Q(n) - 1$ for a single value of $\mathsf{ctr\text{-}II}$) satisfies

$$\begin{aligned}
\gamma_n &= 1 - \tau_n \\
&\leq 1 - \left(\eta_n - \frac{1}{|C_{\bar{R}}|}\right)^2 \\
&= 2\left((1-\epsilon_n)^{n^2Q(n)} + \frac{1}{|C_{\bar{R}}|}\right) - \left((1-\epsilon_n)^{n^2Q(n)} + \frac{1}{|C_{\bar{R}}|}\right)^2
\end{aligned}$$

Then the probability δ_n that S fails to obtain two different $(\bar{x}, \bar{a}, \bar{e}, \bar{z})$ in the phase 2 (simulation with fake random oracles) but S is sent $(\bar{x}, \bar{a}, \bar{e}, \bar{z})$ in the phase 1 for a single value of $\mathsf{ctr\text{-}I}$ (simulation with true random oracles) satisfies

$$\begin{aligned}
\delta_n &= \gamma_n \epsilon_n \\
&\leq \epsilon_n 2\left((1-\epsilon_n)^{n^2Q(n)} + \frac{1}{|C_{\bar{R}}|}\right)
\end{aligned}$$

Now, we estimate an asymptotic behavior of δ_n. For all n such that $\epsilon_n < 1/nQ(n)$, the following relation clearly holds:

$$\delta_n \leq \frac{1}{nQ(n)}. \tag{5.4}$$

On the other hand, for all n such that $\epsilon_n \geq 1/nQ(n)$, the following relation holds:

$$\delta_n \leq 2\left((1-\epsilon_n)^{n^2Q(n)} + \frac{1}{|C_{\bar{R}}|}\right)$$

$$\leq \quad 2\left\{\left(1-\frac{1}{nQ(n)}\right)^{nQ(n)}\right\}^{n}+\frac{2}{|C_{\bar{R}}|}.$$

Hence, the following relation holds.

$$^{\forall}c>0\,^{\exists}N\,^{\forall}n \in \left\{n'|n'>N, \epsilon_{n'} \geq \frac{1}{n'Q(n')}\right\} \qquad \delta_n \leq \left(\frac{2}{e^n}+\frac{1}{n^c}\right)+\frac{2}{|C_{\bar{R}}|}. \tag{5.5}$$

From Eqs. (5.4) and (5.5) and that $\frac{1}{|C_{\bar{R}}|}$ is negligible in n,

$$^{\exists}N\,^{\forall}n>N \qquad \delta_n \leq \frac{1}{nQ(n)}$$

Let θ_n be the probability that the simulation succeeds with out outputting abort in single value of $\mathsf{ctr\text{-}l}$. Since the above equations (5.4) and (5.5) hold for any $(\bar{x},\bar{a})$,

$$\begin{aligned}&^{\exists}N\,^{\forall}n>N\\ &\qquad \theta_n \;\geq\; (1-\delta_n)^{Q(n)} \;\geq\; \left(1-\frac{1}{nQ(n)}\right)^{nQ(n)\cdot\frac{1}{n}} \;\geq\; \frac{1}{e^{\frac{1}{n}}} \;\geq\; \frac{1}{2}\end{aligned}$$

Therefore, by repeating this simulation for n times (for $\mathsf{ctr\text{-}l}=0$ to $n-1$), S succeeds in simulation in overwhelming probability $(1-1/2^n)$ in n. □

Lemma 43 *The view simulated by S is indistinguishable from that of real view, where distinguisher is able to directly access the random oracle.*

Proof. From the construction, it is clear that the simulated view uses no chosen values of random oracles and its distribution is exactly the same to that of real one. □

The theorem follows from Definition 39 and Lemma 41 - 43. □

5.3.4 Addition of Resettable Zero-knowledge Property

We now introduce another variant of the proposed Σ-compiler such that additionally has resettable zero-knowledge property.

We first list the definition of "admissible", and "hybrid cheating verifier" which are given in [11] and [37]. Then, we introduce a construction of deniable rZK protocol from admissible and hybrid deniable zero-knowledge protocol.

Canetti et al. defined a class of admissible protocols and hybrid zero-knowledge in [37]. Then, proposed a generic construction for rZK protocol from an admissible and hybrid zero-knowledge protocol.

Definition 40 *A protocol (P, V) between a prover P and a verifier V is called* admissible *if the following requirements hold:*

1. *The prover P consists of two modules, P_1, P_2. Similarly, the random input r is partitioned into two disjoint parts, $r^{(1)}, r^{(2)}$, where $r^{(i)}$ is given to P_i. The prover initialization message is sent by P_1.*

2. *Each verifier message (other than the first one) is first received by P_1 and is interpreted as consisting of two parts, called* main *and* authenticator. *P_1 decides whether to accept the message or to abort. If P_1 accepts, it forwards the main part of the message to P_2, who generates the next prover message.*

3. *Let V^* be an arbitrary (deterministic) polynomial-size circuit representing a possible strategy for the verifier in the interactive protocol (P, V). Then, except with negligible probability, V^* is unable to generate two different messages for some round ℓ that specify the same session* determining message *in their corresponding prefixes, and such that P_1 accepts both. The determining message is the first message that verifier sends to the prover after the verifier specified that prover as a counterpart.*

Here, an incarnation of the prover is identified via three indices: $P^{(i,j,k)} = P_{x_i,w_i,r_{j,k}}$ where $r_{j,k} = (r_j^{(1)}, r_k^{(2)})$. That is, i specifies the input, j specifies the random input to P_1 and k specifies the random input to P_2.

We define a deniable and random oracle model variant of hybrid zero-knowledge defined in [37].

Definition 41 hybrid cheating verifier V^* *works against admissible protocols as described in the following. That is, V^* proceeds as in Distribution 1 of Definition 32 with the exception that no two sessions started by V^* may interact with incarnations $P^{(i,j,k)}$ and $P^{(i',j',k')}$ such that $k = k'$.*

An admissible protocol is hybrid deniable zero-knowledge in the random oracle model *if it satisfies Definition 32 with respect to hybrid cheating verifiers.*

Here, we present a deniable and random oracle model variant of the construction proposed in [37].

Construction 2 *Given an admissible protocol (P, V), where $P = (P_1, P_2)$, and a collection $\{f\}$ of pseudorandom functions, we define a new protocol $(\mathcal{P}, \mathcal{V})$ as follows.*

1. *The new verifier is identical to V.*

2. *The new prover: The new prover's randomness is viewed as a pair $(r^{(1)}, f)$, where $r^{(1)} \in \{0,1\}^{poly(n)}$ is of length adequate for the random-tape of P_1, and $f : \{0,1\}^{\leq poly(n)} \to \{0,1\}^{poly(n)}$ is a description of a function taken from an ensemble of pseudorandom functions. For convenience we describe the new prover, $\mathcal{P}$, as a pair $\mathcal{P} = (\mathcal{P}_1, \mathcal{P}_2)$. $\mathcal{P}_1$ is identical to P_1 with random-tape $r^{(1)}$; $\mathcal{P}_2$ emulates the actions of P_2 with random tape that is determined by applying f to the determining message and the common input.*

Theorem 23 *If an admissible protocol (P, V) is hybrid deniable zero-knowledge, then $(\mathcal{P}, \mathcal{V})$ is deniable rZK.*

Proof. The proof is similar to that of Theorem 6 in [37]. □

Now we propose another variant of our Σ-compiler.

Construction 3 *A new construction is the same as Construction 1 except the followings:*

1. *P is given a randomly chosen pseudorandom function f. Here, $\{f\}$ is a collection of pseudorandom functions such that $f : (\{0,1\}^n)^4 \to (\{0,1\}^n)^3$.*

2. *P fixes its randomness as follows: Generates its random tapes as $(r_E, r_S, r_P) = f(x, \bar{x}, \bar{a}, \bar{z})$.*

Theorem 24 *The above protocol has rZK property in addition to all other properties that the protocol in Construction 1 has.*

Proof.

Lemma 44 *The protocol of Construction 1 is admissible.*

Proof. It is clear since the protocol is 2-move in which the first message is sent by verifier. □

Lemma 45 *The protocol of Consider 1 is hybrid deniable zero-knowledge.*

Proof. In this protocol, P_1 that appeared in Definition 41 can be considered as procedure in Step 2i. Hence P_1 is deterministic. Therefore, we are only required to consider the simple case when all incarnations of P are different. Then, Hybrid deniable zero-knowledge property follows from Theorem 22. □

Since the proposed protocol in Construction 3 can be constructed from the protocol in Construction 1 according to Construction 4 in [37], the Theorem follows. □

5.4 Efficiency

Our protocol (which is illustrated at the beginning of Sec.5.3) is almost as efficient as the underlying Σ-protocol because Fiat-Shamir transformation and OR-protocol have very small overhead. Moreover, efficient Σ-protocols are known for many useful relations. Hence our construction will find a lot of applications.

On the other hand, the compiler of Pass [129] requires a Cut & Choose method which is very inefficient. Indeed, its overhead is proportion to the security parameter n while ours is only a small constant. Fischlin [62] proposed a straight-line witness extractable proof that has smaller communication complexity than the method of Pass. However, its overhead still depends on the security parameter n. Hence, its communication/computation complexity is still larger than that of ours.

As an example, let's look the deniable identification protocol shown in Section 5.5.2. The communication cost and computational cost are roughly 1/50 and 1/32, respectively, of those of Pass' scheme and are roughly 1/3 and 1/36 [2], respectively, of those of Fischlin's scheme.

5.5 Example Applications

5.5.1 Undeniable Signature

We can obtain a very efficient 2-move undeniable signature scheme as an application of our compiler. Undeniable signature scheme is a variant of signature scheme in which the validity/invalidity of an undeniable signature can be verified only with the signer's consent by engaging interactively or non-interactively in a confirmation/disavowal protocol, as opposed to a digital signature in which its validity/invalidity is universally verifiable. Moreover the result of this confirmation/disavowal have to be non-transferable, i.e., once a verifier is convinced that the signer did or did not sign a message, he should be unable to transmit this conviction to a third party. Because of this property, confirmation/disavowal protocols in these scheme are required have to be deniable. This can be achieved by using our compiler.

Let p, q be primes such that $q|p-1$ and $\mathbb{G}_q$ be the order q subgroup of $(\mathbb{Z}/p\mathbb{Z})^*$. The secret key for the signer is a randomly chosen $w \in \mathbb{Z}/q\mathbb{Z}$ and the public key is $y = g^w$. The signature to the message m is a $h = RO(m)^w$ where RO is a random oracle that maps any string to $\mathbb{G}_q$. r is a valid signature to m if and only if, $(g, y, RO(m), h)$ is a Diffie-Hellman tuple.

It is well known that there exists an efficient Σ-protocol for a relation R such that $a^w = b \wedge c^w = d$, and an efficient Σ-protocol for a relation such that $a^w = b \wedge c^w \neq d$, respectively [34]. Hence by applying our compiler, we can obtain a very efficient 2-move confirmation and a very efficient 2-move disavowal

[2]Operations required in our scheme and those in Fischlin's are very different. Hence this comparison is really rough.

protocol which are both deniable zero-knowledge, where the discrete logarithm relation can be used as $\bar{R}$ in our compiler.

Further, both protocols are deniable concurrent zero-knowledge and resettably-sound argument of knowledge.

On the contrary, very inefficient confirmation/disavowal protocols are obtained if we apply the method of Pass [129]. In the case of confirmation protocol, the communication cost and computational cost of our scheme can be roughly estimated as, respectively, 1/50 and 1/32 of those of Pass' scheme.

(Remark) Recently, Kurosawa et al. showed a 3-move undeniable signature scheme [104]. However, their confirmation/disavowal protocols are not deniable. This fact is also pointed out by Vergnaud Damien [51].

5.5.2 Deniable Identification

Schnorr's identification protocol is illustrated in Table 1. Let p, q be primes such that $q|p-1$ and $\mathbb{G}_q$ be the order q subgroup of $(\mathbb{Z}/p\mathbb{Z})^*$. The secret key for the prover P is a randomly chosen $w \in \mathbb{Z}/q\mathbb{Z}$ and the public key is $x = g^w \bmod p$. Let V denote the verifier.

1. P chooses $r \in \mathbb{Z}/q\mathbb{Z}$ randomly and sends $a = g^r \bmod p$ to V.
2. V sends a random $c \in \mathbb{Z}/q\mathbb{Z}$ to P.
3. P sends $z = r + cw \bmod q$ to V.
4. V accepts iff $g^z = ax^c \bmod p$.

Table 1. Schnorr's identification protocol

Suppose that a malicious verifier V^* sends $c = H(a)$ at step 2, where H is a hash function. Then the transcript (a, c, z) works as an evidence of the fact that P executed the protocol with V^*. Hence the privacy of P is violated. This happens because the protocol is not zero-knowledge.

Now by applying our compiler, we can obtain a very efficient 2-move *deniable* identification protocol as shown in Table 2.

P has a public-key $x(= g^w \bmod p)$ and the secret-key w.

1. V computes $(\bar{x}, \bar{a}, \bar{z})$ as follows, and sends them to P.

   (i) V chooses $\bar{w} \in \mathbb{Z}/q\mathbb{Z}$ randomly and computes $\bar{x} = g^{\bar{w}} \bmod p$.

   (ii) V chooses $\bar{r} \in \mathbb{Z}/q\mathbb{Z}$ randomly and computes $\bar{a} = g^{\bar{r}} \bmod p$.

   (iii) Let $\bar{e} = RO(\bar{x}, \bar{a})$.

   (iv) V computes $\bar{z} = \bar{r} + \bar{e}\bar{w} \bmod q$.

2. P computes $(a, e, z), (\bar{a}', \bar{e}', \bar{z}')$ as follows, and sends them to V.

   (i) P computes $\bar{e} = RO(\bar{x}, \bar{a})$ and verifies that

   $$g^{\bar{z}} = \bar{a}x^{\bar{e}} \bmod p.$$

   (ii) P chooses $\bar{e}', \bar{z}', r \in \mathbb{Z}/q\mathbb{Z}$ randomly and computes

   $$\bar{a}' = g^{\bar{z}'}/x^{\bar{e}'} \bmod p.$$

   (iii) P chooses $r \in \mathbb{Z}/q\mathbb{Z}$ randomly and computes $a = g^r \bmod p$.

   (iv) Let $d = RO(x, a, \bar{x}, \bar{a}')$ and $e = d \oplus \bar{e}'$.

   (v) P computes $z = r + ew \bmod q$.

3. V accepts iff $RO(x, a, \bar{x}, \bar{a}') = e \oplus \bar{e}'$ and

   $$\begin{aligned} g^{\bar{z}} &= \bar{a}x^{\bar{e}} \bmod p \\ g^{z} &= ax^{c} \bmod p \end{aligned}$$

Table 2. Proposed identification protocol

The above protocol is deniable concurrent zero-knowledge and resettably-sound argument of knowledge. If P computes $(\bar{e}', \bar{z}', r)$ as $(\bar{e}', \bar{z}', r) = f(x, \bar{x}, \bar{a}, \bar{z})$, then it is made resettable zero-knowledge, where f is a pseudorandom function.

5.6 Conclusion

We proposed a very efficient compiler that transforms any Σ-protocol to a 2-move deniable zero-knowledge argument scheme in the random oracle model, which is also a resettable zero-knowledge and resettably-sound argument of knowledge. There is no essential loss of efficiency in our transform. We achieved to balance all properties such as efficiency, deniability, resettable zero-knowledge, and resettably-sound argument of knowledge in the random oracle model, which is impossible by black-box reduction in the standard model. Using our compiler, we can obtain a very efficient undeniable signature scheme and a very efficient deniable authentication scheme. We also gave examples of these protocols.

Chapter 6

Black-Box Traitor Revocable Broadcast Encryption with Short Private Keys

Broadcast encryption schemes enable senders to efficiently broadcast ciphertexts to a large set of receivers in a way that only non-revoked receivers can decrypt them. Black-box traitor revocable broadcast encryption schemes are a broadcast encryption schemes that, given a pirate decoder, enable a tracer to identify traitors by black-box accessing the pirated decoder and to revoke traitors so identified. In this chapter, we propose a black-box traitor revocable broadcast encryption scheme in which the size of each of its private keys is constant, the size of the public key is proportional to the number of receivers, and the sizes of its ciphertexts are sub-linear with respect to the number of receivers. The encryption procedure in our scheme requires only a public key. The tracing procedure in it requires a secret key and black-box access to a resettable pirate decoder. The security of our scheme is proved in the generic bilinear group model.

6.1 Introduction

Broadcast encryption schemes are cryptosystems that enable senders to efficiently broadcast ciphertexts to a large set of receivers such that only non-revoked receivers can decrypt them. Their notion was first introduced by Berkovits in [18] and was given formal analysis by Fiat and Naor in [59]. Many schemes, such as [59, 5, 118, 115, 87, 52, 10, 84, 9, 24], have been proposed since then, and their main purpose is to decrease private key size, ciphertext length, public key size, and computational costs for encryption and decryption.

One notable scheme is the one proposed by Boneh et al. in [24]. This is a public key broadcast encryption scheme that features many desirable properties. Particularly notable is the fact that both the receiver's private key size and ciphertexts size are a small constant when broadcasting to any subset of receivers. There is a trade off between the length of the public key and that of the ciphertexts. That is, their product is proportional to the number of receivers.

Black-box traitor revocable broadcast encryption schemes are broadcast encryption schemes that, given a pirate decoder, enable a tracer to identify traitors by black-box accessing the pirated decoder and to revoke traitors so identified. The first such scheme was introduced by D. Naor et al. in [115] as a broadcast encryption scheme with a trace and revoke system. Since the number of receivers in broadcast encryption schemes is potentially large, pirate decoders are always a serious threat, and that is why trace and revoke systems are useful for them.

There also exist schemes that are closely related to black-box traitor revocable broadcast encryption schemes. These are, black-box traitor tracing schemes. Black-box traitor tracing schemes are cryptosystems that enable senders to efficiently send ciphertexts to a large number of receivers such that, given a pirate decoder, at least one receiver who has leaked its key can be traced by black-box accessing the pirate decoder. Here, it is highly desired that the revocation be performed in a black-box manner, so that the tracer will be free from the laborious analysis of obfuscated codes. After the notion of such a scheme was first introduced by Chor et al. in [45], many schemes, [102, 117, 144, 145, 61, 23, 118, 133, 115, 112, 52, 40], including non black-box schemes, have been proposed. One notable scheme is that proposed by Boneh et al. in [25], which is a fully collusion resistant system with constant size private keys and sub-linear size ciphertexts.

It might seem that a simple combination of a black-box traitor tracing scheme and a broadcast encryption scheme might result in a black-box traitor revocable broadcast encryption scheme. Unfortunately, however, this is not true. Consider a simple combination system in which a secret key is distributed as the sum of a secret key distributed using broadcast encryption and one distributed using traitor tracing scheme. In this case, given a pirate decoder, a tracer will be able to identify at least one traitor, but this identification will be limited to the basing traitor tracing scheme. The traitor in the basing broadcast encryption scheme will not necessarily coincide with the traitor identified in the tracing scheme. This means, that the tracer may fail to revoke the traitor in the basing broadcast encryption scheme. Here, the tracer will be able to punish the traitor it has identified, but it will not be able to prevent copies of the same pirate decoders, which could be distributed in the network, from decrypting its ciphertexts. Hence, when we construct a black-box traitor revocable broadcast encryption scheme from a black-box traitor tracing scheme and a broadcast encryption scheme, we are required to exercise some ingenuity in doing so.

In this chapter, we propose a black-box traitor revocable broadcast encryption scheme. Its revocation procedure requires a secret-key, but its encryption procedure requires only a public key. We call such a scheme black-box secret key traitor revocable public key broadcast encryption scheme. We have also formalized security requirements for such a scheme and proved that our scheme satisfies these requirements in the generic bilinear group model.

Our black-box traitor revocable broadcast encryption scheme is a careful combination of a modified version of the broadcast encryption scheme in [24] and the traitor tracing scheme in [25]. The modified broadcast encryption scheme is designed to prevent revoked receivers from being able to check whether ciphertexts have been generated correctly or not. Because of this property, revoked traitors are unable to hinder the tracer from tracing other traitors. This modified scheme was merged with the scheme in [25] to form the proposed scheme.

Here we assume the total number of receivers to be N and the total number of traitor to be $t \leq N$. The traitor tracing scheme in [25] requires its private keys size, ciphertext size, and public key size to be, respectively, $\mathcal{O}(1)$, $\mathcal{O}(\sqrt{N})$, and $\mathcal{O}(\sqrt{N})$. With the original scheme [24], if we were to set the size of ciphertexts to be $\mathcal{O}(\sqrt{N})$, the other size would also have to be the same. On the other hand, with our modified broadcast encryption scheme, private key size, ciphertext size, and public key size can be, respectively, $\mathcal{O}(1)$, $\mathcal{O}(\sqrt{N})$, and $\mathcal{O}(N)$, and with our proposed combination scheme, private key size, ciphertext size, and public key size can be the same, $\mathcal{O}(1)$, $\mathcal{O}(\sqrt{N})$, and $\mathcal{O}(N)$. The cost for revocation in black-box manner is $\mathcal{O}(tN^2)$.

In the standard model, we have only so far been able to prove the security of our modified broadcast encryption scheme only for the situation in which its adversaries are restricted to statically colluding adversaries, which means that our proposed combination scheme would also suffer the same limitation in the standard model. While it is easy to construct a broadcast encryption scheme that is secure against statically colluding adversaries but is totally vulnerable against adaptively colluding adversaries, as a practical matter, adaptively colluding adversaries are of much greater concern. That is,why it would be preferable to be able to prove security with respect to adaptively colluding adversaries even if the proof were done in a model weaker than the standard model. That is why we have chosen to prove its security in the generic bilinear group model. We should also note that the scheme in [24] is already secure against adaptively colluding adversaries in the generic bilinear group model.

Related Work

Boneh and Waters have independently proposed a black-box trace and revoke system in [26]. Although the goal of their work is essentially the same as ours, their results differ from ours in various respects.

The size of private keys in their scheme is $\mathcal{O}(\sqrt{N})$, larger than our $\mathcal{O}(1)$. By way of contrast, the size of public key in their scheme is $\mathcal{O}(\sqrt{N})$, shorter than our $\mathcal{O}(N)$. These differences come from a difference in the respective broadcast encryption portions of the two combination schemes. Boneh's new scheme requires longer private keys but requires only shorter public key. The adaptive security of Boneh's scheme can be proven in the standard model, but ours can only be proven in the generic bilinear model. However, while they say that their proposed scheme supports a public tracing mechanism, it is not secure if does so. Indeed, the adversary will be able to identifies i from $Encrypt_{ABE}(S, PK, (i, j), M)$. This can be done by checking whether

$$e(R_x, \tilde{V}) = e(\tilde{R}_x, V)$$

holds or not. If it does, then $x > i$. Otherwise, $x \leq i$. Hence, the tracing key $(V, \tilde{V})$ must be kept secret. A flaw in their proof appears as d in the 3rd line from the end of the chapter. Although $d = \log_B A$ must be unknown to the simulator, the simulator in their proof uses this value.

They say that the secret key traceability of their scheme can be proven in the same way as the security of the scheme in [25] is proven. This, however, is not precisely correct. Adversaries in a secret-key traitor revocable encryption scheme may obtain data related to this secret key from the output of the revocation procedure, i.e., from the set of revoked receivers. It is necessary, then, to consider adversaries that can access a revocation oracle. In this respect, our formalization of secret-key traitor revocability is novel.

Organization

The chapter is organized as follows: Section 6.2 specifies the model used and defines the security requirements for a black-box traitor revocable encryption scheme. Section 6.3 discusses the revocation mechanism. Section 6.4 reviews a number of basic facts and complexity assumptions that our scheme relies on. Section 6.5 presents our basic scheme. Section 6.6 analyses the efficiency of our basic scheme and describes how we can obtain from the basic scheme a scheme what is fully secure.

6.2 The Model and Security Requirements

6.2.1 The Model

Four types of players participate in a black-box traitor revocable encryption scheme: a membership authority $\mathcal{M}$, a revocation authority $\mathcal{R}$, encrypters $\mathcal{E}$, and receivers. We let the total number of receivers be N and i-th receiver be $\mathcal{R}_i$.

A black-box traitor revocable encryption scheme consists of five algorithms: Gen, Ext, Enc, Dec, and Revoke.

Gen: A probabilistic setup algorithm that, given a security parameter 1^k and the number of receivers N, outputs a public key *pkey*, a registration key *mkey*, and a tracing key *tkey*.

$$(pkey, mkey, tkey) \leftarrow \mathsf{Gen}(1^k, N).$$

mkey and *tkey* are given, respectively, to $\mathcal{M}$ and $\mathcal{T}$. We assume that *pkey* includes N.

Ext: A probabilistic algorithm for $\mathcal{M}$ that, given *mkey*, *pkey*, and an index i of receiver, outputs a private key $skey_i$ for i-th receiver:

$$skey_i \leftarrow \mathsf{Ext}(pkey, mkey, i).$$

Enc: A probabilistic algorithm for encrypters that, given a subset $S \subseteq \{1, \ldots, N\}$ and a public key *pkey*, outputs a shared key *key* and a header *hdr*.

$$(key, hdr) \leftarrow \mathsf{Enc}(S, pkey).$$

Dec: A probabilistic algorithm for receivers that, given a *hdr* for a subset S, $i \in S$, $skey_i$, and the public key *pkey*, outputs *key*.

$$key \leftarrow \mathsf{Dec}(S, i, skey_i, hdr, pkey).$$

Revoke: A probabilistic algorithm for $\mathcal{R}$ that, given a subset $S^\dagger$, *pkey*, *tkey*, a black-box access to a pirate decoder $\mathcal{D}$, and a parameter ϵ, outputs an honest receiver subset $S^\ddagger \subset S^\dagger$.

$$S^\ddagger \leftarrow \mathsf{Revoke}^{\mathcal{D}}(S^\dagger, pkey, tkey, \epsilon).$$

We say the scheme is a black-box secret-key traitor revocable public-key encryption scheme if *tkey* is disclosed only to the revocation authority, and our proposed scheme is of this type.

6.2.2 Security Requirements

Let us now consider formal definitions of security requirements for black-box secret key traitor revocable public key encryption schemes.

Definition 42 (Correctness) *The scheme is* correct *if, for all $N, S, \subset \{1, \ldots, N\}, i \in S$, and random tapes input to* Gen, Ext, Enc, *and* Dec, *the following equation,*

$$key \;=\; key',$$

holds where

$$\begin{aligned} (pkey, mkey, tkey) &\leftarrow \mathsf{Gen}(1^k, N) \\ skey_i &\leftarrow \mathsf{Ext}(pkey, mkey, i) \\ (key, hdr) &\leftarrow \mathsf{Enc}(S, pkey) \\ key' &\leftarrow \mathsf{Dec}(S, i, skey_i, hdr, pkey) \end{aligned}$$

In this section and the next section, we consider several games with respect to which we formalize the rest of the security requirements. As a template for these games, let us introduce Template Game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$.

Definition 43 (Template Game) *Suppose that the number of receivers N, the security parameter κ, and ϵ (where $\epsilon = 1/f(\kappa)$ for some polynomial f) are given to $\mathcal{C}$ and $\mathcal{A}$.*

1. **Key Generation Phase:** *$\mathcal{C}$ runs*

$$(pkey, mkey, tkey) \leftarrow \mathsf{Gen}(1^k, N)$$

and provides pkey to $\mathcal{A}$. $\mathcal{C}$ keeps pkey, mkey, and tkey. $\mathcal{A}$ keeps a set T which is initially set to be $\emptyset$.

2. **Query Phase:** *$\mathcal{A}$ adaptively sends the following queries:*

   - **Extraction:** *$\mathcal{A}$ sends $\mathcal{C}$ an index $i \in \{1, \ldots, N\}$. Then, $\mathcal{C}$ runs*

$$skey_i \leftarrow \mathsf{Ext}(pkey, mkey, i)$$

   and provides $skey_i$ to $\mathcal{A}$.

   - **Decryption:** *$\mathcal{A}$ sends $\mathcal{C}$ a tuple $(S \subset \{1, \ldots, N\}, i \in S, hdr)$. Then, $\mathcal{C}$ runs*

$$(key, hdr) \leftarrow \mathsf{Dec}(S, i, skey_i, hdr, pkey)$$

   and provides key to $\mathcal{A}$. $\mathcal{A}$ replace T with $T \cup i$

- **Revocation:** *$\mathcal{A}$ sends $\mathcal{C}$ a pirate decoder $\mathcal{D}$ and $S^{\dagger}$. Then, $\mathcal{C}$ runs*

$$(S^{\ddagger}) \leftarrow \mathsf{Revoke}^{\mathcal{D}}(S^{\dagger}, tkey, pkey, \epsilon)$$

  and provides $S^{\ddagger}$ to $\mathcal{A}$.

While sending the above three types of queries, $\mathcal{A}$ sends $\mathcal{C}$ **challenge query** *once and is provided a challenge by $\mathcal{C}$ in turn.*

3. **Answer Phase:** *$\mathcal{A}$ sends $\mathcal{C}$ an answer for the challenge. $\mathcal{C}$ evaluates its answer and outputs* 1 *or* 0 *terminating the game.*

The revocation query is no longer required if the traitor revocable encryption is public-key traitor revocable since its responses are simulatable.

We define chosen ciphertext/plaintext attacks by adaptively colluding adversary (ACA) games. In the rest of the chapter, strings `atk` are either "Chosen Ciphertext Attacks (CCA)" or "Chosen Plaintext Attacks (CPA)".

Definition 44 "Chosen ciphertext attacks (CCA) by ACA game" *is the same as Template Game.*

Definition 45 "Chosen plaintext attacks (CPA) by ACA" game *is the same as Chosen ciphertext attacks by ACA game except that $\mathcal{A}$ can send no Decryption Query.*

We defines key distinguishing/pirating game against `atk` by ACA.

Definition 46 "Key Distinguishing against `atk` by ACA" game *is the same as* `atk` *by ACA game except in the following points:*

- **Challenge Query:** *$\mathcal{A}$ sends $\mathcal{C}$ a set $S^* \subset \{1, \ldots, N\}$. Then, $\mathcal{C}$ runs $(key^*, hdr^*) \leftarrow \mathsf{Enc}(S^*, pkey)$, randomly chooses $b \in_R \{0,1\}$ and $key^*_{1-b} \in_R \mathcal{K}$, and assigns $key^*_b = key^*$. $\mathcal{C}$ finally sends to $\mathcal{A}$*

$$(hdr^*, key^*_0, key^*_1).$$

  Here, $\mathcal{C}$ only allows S^* such that $S^* = \{1, \ldots, N\} \setminus T^*$, where T^* be the set T at the end of the game.

- **Answer:** *$\mathcal{A}$ sends $b' \in \{0,1\}$. Then $\mathcal{C}$ outputs* 1 *if $b' = b$, otherwise outputs* 0.

Definition 47 "Pirating against `atk` by ACA" game *is the same as* `atk` *by ACA game except in the following points:*

- **Challenge Query:** *$\mathcal{A}$ sends $\mathcal{C}$ a pirate decoder $\mathcal{D}$ and $S^\dagger \in \{1, \ldots, N\}$, where $\mathcal{D}$ is a probabilistic circuit that takes as input a header S, hdr, and pkey and outputs some key key. Then $\mathcal{C}$ runs $S^\ddagger \leftarrow \mathsf{Revoke}^{\mathcal{D}}(S^\dagger, pkey, tkey, \epsilon)$. $\mathcal{C}$ accepts no Extraction and Decryption queries hereafter.*

- **Answer:** *$\mathcal{A}$ sends nothing. $\mathcal{C}$ outputs 1 if either of the following two conditions hold:*

  - $\Pr[\mathcal{D}(S^\dagger, hdr^\dagger, pkey) = key^\dagger | (key^\dagger, hdr^\dagger) \leftarrow \mathsf{Enc}(S^\dagger, pkey))] \geq \epsilon$.
  - $(S^\dagger \setminus S^\ddagger) \not\subset T$;

  Otherwise outputs 0.

Definition 48 (Key Indistinguishable) *We say a traitor revocable encryption scheme is* key indistinguishable against `atk` by ACA *if, for all N, polynomials f, and $\mathcal{A}$, the difference between* 1/2 *and the probability of $\mathcal{C}$ outputting* 1 *in the key distinguishing against* `atk` *by ACA game is negligible in κ.*

Definition 49 (Black-Box Traitor Revocable) *We say a traitor revocable encryption scheme is* black-box traitor revocable against `atk` by ACA *if, for all N, polynomials f, and $\mathcal{A}$, the probability of $\mathcal{C}$ outputting* 1 *in the pirating against* `atk` *by ACA game is negligible in κ.*

6.3 Black-Box Traitor Revocation and Strong Index Hiding

We define a strong index hiding property of traitor revocable encryption schemes. Later, we will prove that schemes with this property are black-box traitor revocable. In terms of the traitor tracing scheme of [25], this property corresponds to the existence of corresponding linear broadcast encryption. But the property we present here is stronger than the previous one so that we can characterize the revocability.

Suppose that Probe is an algorithm that, given $tkey, pkey, S^\dagger$, and an index $u \in \{1, \ldots, N\}$, outputs a header $(key^\dagger, hdr^\dagger)$.

$$(key^\dagger, hdr^\dagger) \leftarrow \mathsf{Probe}(tkey, pkey, S^\dagger, u).$$

We consider several games by which we formalize strong index hiding property.

Definition 50 "strong-index distinguishing against `atk` by ACA" game *is the same as* `atk` *by ACA game except in the following points:*

- **Challenge:** *$\mathcal{A}$ sends $\mathcal{C}$ a set $S^\dagger \subset \{1, \ldots, N\}$. Then $\mathcal{C}$ chooses arbitrary u such that $u \notin T \cap S^\dagger$, randomly chooses $b \in_R \{0, 1\}$, and runs*

$$(key^\dagger, hdr^\dagger) \leftarrow \mathsf{Probe}(tkey, pkey, S^\dagger, u + b)$$

Then, $\mathcal{C}$ provides $(u, \text{hdr}^{\dagger})$ to $\mathcal{A}$. $\mathcal{C}$ accepts no Extraction and Decryption queries hereafter.

- **Answer:** *$\mathcal{A}$ sends $b' \in \{0,1\}$. Then $\mathcal{C}$ outputs 1 if $b' = b$, otherwise outputs 0.*

Definition 51 "boundary condition checking against `atk` by ACA" game *is the same as strong-index distinguishing against* `atk` *by ACA game except that u is fixed to $N+1$.*

Definition 52 *We say* Probe *of a traitor revocable encryption scheme* satisfies boundary condition *against* `atk` *by ACA if, (1) for all N, polynomials f, and $\mathcal{A}$, the difference between $1/2$ and the probability of $\mathcal{C}$ outputting 1 in the boundary condition checking against* `atk` *by ACA game is negligible in κ and (2) if the distribution of* $\mathsf{Probe}(tkey, pkey, S^{\dagger}, 1)$ *and that of* $\mathsf{Enc}(pkey, S^{\dagger})$ *are identical.*

Definition 53 (Strong Index Hiding) *We say a traitor revocable encryption scheme is* strong index hiding against `atk` by ACA *if, there exists* Probe*, which satisfies boundary condition against* `atk` *by ACA, such that, for all N, polynomials f, and $\mathcal{A}$, the difference between $1/2$ and the probability of $\mathcal{C}$ outputting 1 in the strong-index distinguishing game against* `atk` *by ACA game is negligible in κ.*

The above boundary condition checking game is almost the same to the message hiding game in [25]. The strong-index distinguishing game is almost the same to index hiding game in [25] except that our $S^{\dagger}$ is not necessarily $\{1, \ldots, N\}$. Therefore, the truly new property is defined in the case where $u \notin S^{\dagger}$.

In the revocation procedure, we are always able to find malicious users in $S^{\dagger}$. This is because, informally, according to the new property, only users in $S^{\dagger}$ are relevant for distinguishing $\mathsf{Probe}(tkey, pkey, S^{\dagger}, 1)$ from $\mathsf{Probe}(tkey, pkey, S^{\dagger}, N+1)$. Once we found a malicious user in $S^{\dagger}$, we are able to exclude this detected user from $S^{\dagger}$. Repeating this procedure, we can successfully revoke all and only malicious user relevant to pirating.

Theorem 25 *If the traitor revocable encryption is strong index hiding against* `atk` *by ACA and $\mathcal{D}$ is resettable, then it is black-box traitor revocable against* `atk` *by ACA.*

Proof. We first construct Revoke that uses Probe. In this algorithm, the following procedure Estimate() is repeatedly invoked.

Procedure $p_i \leftarrow \mathsf{Estimate}(i, S^{\dagger}, tkey, pkey, \epsilon, k)$ is as in the following:

1. Do the following $8nk/\epsilon$ times:
   (i) Generate $(key^{\dagger}, hdr^{\dagger}) = \mathsf{Probe}(tkey, pkey, S^{\ddagger}, i)$.
   (ii) Reset $\mathcal{D}$.
   (iii) Run $key^{\ddagger} \leftarrow \mathcal{D}(S^{\ddagger}, hdr^{\dagger}, pkey)$ and compare with $key^{\dagger}$.

2. Based on the above comparison, calculate p_i as the probability that $key^{\ddagger} = key^{\dagger}$ holds.

Now, Revoke is as in the following:

1. Revoke is given $tkey, pkey$, $S^{\dagger}$, and black-box access to a pirate decoder $\mathcal{D}$. It sets $S^{\ddagger} = S^{\dagger}$.
2. Until Revoke stops, do the following:
   - (i) Run $p_1 \leftarrow \mathsf{Estimate}(1, S^{\ddagger}, tkey, pkey, \epsilon, k)$.
   - (ii) For $i = 2$ to $N+1$ do the following:
     - i. If $i \notin S^{\ddagger}$, $p_i = p_{i-1}$.
     - ii. If $i \in S^{\ddagger}$, Revoke runs $p_i \leftarrow \mathsf{Estimate}(i, S^{\ddagger}, tkey, pkey, \epsilon, k)$
   - (iii) For all i such that $|p_i - p_{i+1}| \geq \epsilon/4n$, $S^{\ddagger} \leftarrow S^{\ddagger} \setminus i$. If there is no such i, output $S^{\ddagger}$ and stop, otherwise, go to Step 2i.

Lemma 46 *If a traitor revocable encryption is strong index hiding against* `atk` *by ACA, then by using the above introduced* Revoke*, one can successfully revoke traitors in black-box manner, i.e., it is black-box traitor revocable against* `atk` *by ACA. The running time is $\mathcal{O}(ts^3)$ where t is the number of colluders and s is the size of subset of user who can decrypt ciphertext.*

A proof sketch: Using hybrid argument, the lemma can be proved in the same manner as to prove the reduction from traitor tracing to linear broadcast encryption in [25]. The only difference is that once i is excluded from $S^{\dagger}$, $\mathcal{R}_i$ is irrelevant to distinguishing $\mathsf{Probe}(tkey, pkey, S^{\dagger}, 1)$ from $\mathsf{Probe}(tkey, pkey, S^{\dagger}, N+1)$. Hence, the proof is omitted. □

The theorem follows from Lemma 46 □

The running time can be reduced to $\mathcal{O}(ts^2)$ by using binary search.

6.4 Preliminaries

We describe basic facts and complexity assumptions that our scheme depends on.

Definition 54 Bilinear Maps of composite order

1. *$\mathcal{G}$ and $\mathcal{G}_T$ are two cyclic groups of prime order $n = pq$ where p and q are distinct primes.*
2. *$\hat{\mathcal{G}}$ and $\hat{\mathcal{G}}_T$ are, respectively, order q subgroups of $\mathcal{G}$ and $\mathcal{G}_T$.*
3. *$\check{\mathcal{G}}$ and $\check{\mathcal{G}}_T$ are, respectively, order p subgroups of $\mathcal{G}$ and $\mathcal{G}_T$.*
4. *$G, G_T, \hat{G}, \hat{G}_T, \check{G}$, and $\check{G}_T$ are, respectively, generators of $\mathcal{G}, \mathcal{G}_T, \hat{\mathcal{G}}, \hat{\mathcal{G}}_T, \check{\mathcal{G}}$, and $\check{\mathcal{G}}_T$.*

5. $e : \mathcal{G} \times \mathcal{G} \rightarrow G_1$ is a map such that:

(i) Bilinear: For all $U, V \in \mathcal{G}$ and $\alpha, \beta \in \mathbb{Z}/n\mathbb{Z}$, we have $e([\alpha]U, [\beta]V) = e(U, V)^{\alpha\beta}$.

(ii) Non-degenerate: $e(G, G)$ is G_T a generator of $\mathcal{G}_T$.

We define a "General Bilinear Decision Problem" in the generic bilinear group of composite order, which is a generalization of "General Diffie-Hellman Exponent problem" of [21] in the generic bilinear group model [19].

Definition 55 General Bilinear Decision Problem of Composite Order: *Let n be a composite of two primes and s, m be two positive integer constants. Let $\mathcal{G}$ and $\mathcal{G}_T$ be as defined above. Let $P = (p_1, \dots, p_s), Q = (q_1, \dots, q_s), P' = (p'_1, \dots, p'_s), Q' = (q'_1, \dots, q'_s) \in (\mathbb{Z}/n\mathbb{Z})[X_1, \dots, X_m]^s$ and where $p_1 = q_1 = p'_1 = q'_1 = 1$. Let $P(x_1, \dots, x_n)$ denote $(p_1(x_1, \dots, x_m), \dots, p_s(x_1, \dots, x_m))$ and $[P(x_1, \dots, x_m)]G = [p_1(x_1, \dots, x_m)]G, \dots, [p_s(x_1, \dots, x_m)]G$. We use similar notation for Q, P', Q'.*

We say a algorithm $\mathcal{B}$ solves general bilinear decision problem of composite order with respect to (P, Q) and (P', Q') if

$$\begin{aligned} &|\Pr[\mathcal{B}([P(x_1, \dots, x_n)]G, [Q(x_1, \dots, x_n)]G_T) = 1] \\ &- \Pr[\mathcal{B}([P'(x_1, \dots, x_n)]G, [Q'(x_1, \dots, x_n)]G_T) = 1]| \end{aligned}$$

is not negligible, where $x_0, \dots, x_n \in_R \mathbb{Z}/n\mathbb{Z}$ are randomly chosen.

We say (P, Q) and (P', Q') are dependent if there exists tuple of $s^2 + s$ constants $\{a_{ij}\}_{i=1,\dots,s,j=1,\dots,s}$, $\{b_i\}_{i=1,\dots,s}$ such that either

$$0 \equiv \sum_{i,j=1}^{s} a_{ij} p_i p_j + \sum_{i=1}^{s} b_i q_i \quad , \quad 0 \not\equiv \sum_{i,j=1}^{s} a_{ij} p'_i p'_j + \sum_{i=1}^{s} b_i q'_i$$

or

$$0 \not\equiv \sum_{i,j=1}^{s} a_{ij} p_i p_j + \sum_{i=1}^{s} b_i q_i \quad , \quad 0 \equiv \sum_{i,j=1}^{s} a_{ij} p'_i p'_j + \sum_{i=1}^{s} b_i q'_i$$

holds. Here, each non zero coefficient of $\prod_{i=1}^{n} {x_i}^{t_i}$ must be in $(\mathbb{Z}/n\mathbb{Z})^*$. This is because we assume the factoring of n is hard.

We say that a general bilinear decision problem composite order with respect to (P, Q) and (P', Q') is independent if (P, Q) and (P', Q') are not dependent.

The above definition is straight forward generalization of the general Diffie-Hellman Exponent problem, where the difference between (P, Q) and (P', Q') is only one element in Q and Q. That element is denoted as f in [21]. Note that, if we allow coefficients to be a non zero divisor of $\mathbb{Z}/n\mathbb{Z}$, then, with the corresponding $\{a_{ij}\}$ and $\{b_i\}$, it is easy to distinguish randomly chosen elements of $\hat{\mathcal{G}}$ and $\mathcal{G}$.

Theorem 26 *In the generic bilinear group model of composite order, no polynomial time algorithm solves, in non-negligible probability, any of general bilinear decision problem of composite order with respect to* (P, Q) *and* (P', Q') *which are independent.*

A proof sketch: In the generic bilinear group model, except group operations, the only comparison between two elements that adversaries can do is equality checking. And coincidence of two polynomials, when their variables are fixed to randomly chosen values, occurs only with negligible probability unless these polynomials are equivalent. Hence the theorem holds. Moreover, the lower bound on the advantage of solving the problem can be estimated in the same manner as in the [21]. Hence, the proof detail is omitted. $\square$

The generic bilinear group model of prime order and its general decision problem is defined in the similar way.

6.5 Basic Scheme

We now propose a secret key traitor revocable public-key encryption scheme that is key indistinguishable and black-box traitor revocable against chosen plaintext attacks by adaptively colluding adversaries in the generic bilinear group model.

The key point of our scheme lies in the new broadcast encryption portion. We can say that, in the broadcast encryption scheme proposed in [24], the result of decryption by all non-revoked users is the data of the form $e(G, G)^{\alpha^{\ell+1}}$, but that of all the revoked users is 1^0. Hence, non-revoked users can check that the ciphertext is correctly generated. In our scheme, the result of decryption by all non-revoked users is the data of the form $M \cdot 1^0$, but that of (i, j)-th user who is revoked will be $M \cdot e(G, G)^{\sigma_j \theta_i \alpha^{\ell+1}}$. The fact values $e(G, G)^{\sigma_j \theta_i \alpha^{\ell+1}}$ differ each other prevents revoked users check the ciphertexts.

Let Λ be the product set $\{1, \ldots, \ell\}^2$ such that each element corresponds to each receiver, $\Lambda_j \subset \Lambda$ be a subset $\{(1, \ldots, \ell\}, j)$, and let p, q be sufficiently large primes, $n = pq$, $\mathcal{G}$ and $\mathcal{G}_T$ be order n (elliptic curve) cyclic groups such that bilinear map $e : \mathcal{G} \times \mathcal{G} \to \mathcal{G}_T$ exists, $\hat{\mathcal{G}}$ and $\check{\mathcal{G}}$ be subgroups of $\mathcal{G}$ such that their orders are q and p respectively. Let $\bar{S}$ be the complement of S in $\{1, \ldots, \ell\}^2$. Similar notation is used for other sets such as $S^*, S^\dagger, S^\ddagger$.

Gen: Given 1^k and ℓ, Gen chooses composite order n cyclic groups $\mathcal{G}$ and $\mathcal{G}_T$ with e of size k. Gen symmetrically and randomly chooses

$$\hat{G} \in_R \hat{\mathcal{G}} \quad , \quad \check{G} \in_R \check{\mathcal{G}}$$

$$
\begin{aligned}
\hat{\eta} \in_R \mathbb{Z}/q\mathbb{Z} \quad &, \quad \check{\eta} \in_R (\mathbb{Z}/p\mathbb{Z})^* \\
\hat{\xi} \in_R \mathbb{Z}/q\mathbb{Z} \quad &, \quad \check{\xi} \in_R (\mathbb{Z}/p\mathbb{Z})^* \\
\hat{\zeta} \in_R \mathbb{Z}/q\mathbb{Z} \quad &, \quad \check{\zeta} \in_R (\mathbb{Z}/p\mathbb{Z})^* \\
\hat{\alpha} \in_R \mathbb{Z}/q\mathbb{Z} \quad &, \quad \check{\alpha} \in_R (\mathbb{Z}/p\mathbb{Z})^* \\
(\hat{\beta}_j)_{j=1,\ldots,\ell} \in_R (\mathbb{Z}/q\mathbb{Z})^\ell \quad &, \quad (\check{\beta}_j)_{j=1,\ldots,\ell} \in_R ((\mathbb{Z}/p\mathbb{Z})^*)^\ell \\
(\hat{\delta}_j)_{j=1,\ldots,\ell} \in_R (\mathbb{Z}/q\mathbb{Z})^\ell \quad &, \quad (\check{\delta}_j)_{j=1,\ldots,\ell} \in_R ((\mathbb{Z}/p\mathbb{Z})^*)^\ell \\
(\hat{\gamma}_i)_{i=1,\ldots,\ell} \in_R (\mathbb{Z}/q\mathbb{Z})^\ell \quad &, \quad (\check{\gamma}_i)_{i=1,\ldots,\ell} \in_R ((\mathbb{Z}/p\mathbb{Z})^*)^\ell \\
(\hat{\theta}^{(k)})_{k=1,\ldots,\ell} \in_R (\mathbb{Z}/q\mathbb{Z})^\ell \quad &, \quad (\check{\theta}^{(k)})_{k=1,\ldots,\ell} \in_R ((\mathbb{Z}/p\mathbb{Z})^*)^\ell
\end{aligned}
$$

Next, Gen symmetrically generates

$$
\begin{aligned}
\hat{H} = [\hat{\xi}]\hat{G} \quad &, \quad \check{H} = [\check{\xi}]\check{G} \\
\hat{G}' = [\hat{\eta}]\hat{G} \quad &, \quad \check{G}' = [\check{\eta}]\check{G} \\
\hat{H}' = [\hat{\eta}]\hat{H} \quad & \\
\hat{M} = [\hat{\zeta}]\hat{G} \quad &, \quad \check{M} = [\check{\zeta}]\check{G} \\
(\hat{B}_j)_{j=1,\ldots,\ell} = ([\hat{\beta}_j]\hat{G})_{j=1,\ldots,\ell} \quad &, \quad (\check{B}_j)_{j=1,\ldots,\ell} = ([\check{\beta}_j]\check{G})_{j=1,\ldots,\ell} \\
(\hat{B}'_j)_{j=1,\ldots,\ell} = ([\hat{\beta}_j]\hat{G}')_{j=1,\ldots,\ell} \quad & \\
(\hat{H}_j)_{j=1,\ldots,\ell} = ([\hat{\beta}_j]\hat{H})_{j=1,\ldots,\ell} \quad &, \quad (\check{H}_j)_{j=1,\ldots,\ell} = ([\check{\beta}_j]\check{H})_{j=1,\ldots,\ell} \\
(\hat{H}'_j)_{j=1,\ldots,\ell} = ([\hat{\beta}_j]\hat{H}')_{j=1,\ldots,\ell} \quad & \\
(\hat{G}_i)_{i=1,\ldots,2\ell} = ([\hat{\alpha}^i]\hat{G})_{i=1,\ldots,2\ell} \quad &, \quad (\check{G}_i)_{i=1,\ldots,2\ell} = ([\check{\alpha}^i]\check{G})_{i=1,\ldots,2\ell} \\
(\hat{G}'_i)_{i=1,\ldots,2\ell} = ([\hat{\alpha}^i]\hat{G}')_{i=1,\ldots,2\ell} \quad & \\
(\hat{D}_j)_{j=1,\ldots,\ell} = ([\hat{\delta}_j]\hat{G})_{j=1,\ldots,\ell} \quad &, \quad (\check{D}_j)_{j=1,\ldots,\ell} = ([\check{\delta}_j]\check{G})_{j=1,\ldots,\ell} \\
(\hat{D}'_j)_{j=1,\ldots,\ell} = ([\hat{\delta}_j]\hat{G}')_{j=1,\ldots,\ell} \quad & \\
(\hat{J}_i)_{i=1,\ldots,\ell} = ([\hat{\gamma}_i]\hat{G})_{i=1,\ldots,\ell} \quad &, \quad (\check{J}_i)_{i=1,\ldots,\ell} = ([\check{\gamma}_i]\check{G})_{i=1,\ldots,\ell} \\
(\hat{X}_i^{(k)})_{k=1,\ldots,\ell;i=1,\ldots,2\ell} = ([\hat{\theta}^{(k)}]\hat{G}_i)_{k=1,\ldots,\ell;i=1,\ldots,2\ell} \quad &, \\
(\check{X}_i^{(k)})_{k=1,\ldots,\ell;i=1,\ldots,2\ell} = ([\check{\theta}^{(k)}]\check{G}_i)_{k=1,\ldots,\ell;i=1,\ldots,2\ell} \quad &, \\
\hat{m} = e(\hat{G}, \hat{M}) \quad &, \quad \check{m} = e(\check{G}, \check{M}) \\
\hat{m}' = e(\hat{G}', \hat{M}) \quad &
\end{aligned}
$$

Next, by combining them, Gen generates

$$\begin{aligned} G &= \hat{G} + \check{G} \\ H &= \hat{H} + \check{H} \end{aligned}$$

$$\begin{aligned} (G_i)_{i=1,\ldots,2\ell} &= (\hat{G}_i + \check{G}_i)_{i=1,\ldots,2\ell} \\ (D_j)_{j=1,\ldots,\ell} &= (\hat{D}_j + \check{D}_j)_{j=1,\ldots,\ell} \\ (B_j)_{j=1,\ldots,\ell} &= (\hat{B}_j + \check{B}_j)_{j=1,\ldots,\ell} \\ (H_j)_{j=1,\ldots,\ell} &= (\hat{H}_j + \check{H}_j)_{j=1,\ldots,\ell} \\ (J_i)_{i=1,\ldots,\ell} &= (\hat{J}_i + \check{J}_i)_{i=1,\ldots,\ell} \\ (X_i^{(k)})_{k=1,\ldots,\ell;i=1,\ldots,2\ell} &= (\hat{X}_i^{(k)} + \check{X}_i^{(k)})_{k=1,\ldots,\ell;i=1,\ldots,2\ell} \\ m &= \hat{m} + \check{m} \end{aligned}$$

Finally, Gen outputs

$$\begin{aligned} \mathit{pkey} &= \{m,\ (G,H), (B_j, H_j)_{j=1,\ldots,\ell}, (G_i)_{i=1,\ldots,\ell,\ell+2,\ldots,2\ell}, (D_j)_{j=1,\ldots,\ell}, \\ &\quad \hat{m}', (\hat{G}' \quad\ ), (\hat{B}'_j, \hat{H}'_j)_{j=1,\ldots,\ell}, (\hat{G}'_i)_{i=1,\ldots,\ell,\ell+2,\ldots,2\ell}, (\hat{D}'_j)_{j=1,\ldots,\ell}, \\ &\quad (J_i)_{i=1,\ldots,\ell}, (X_i^{(k)})_{k=1,\ldots,\ell;i=1,\ldots,\ell,\ell+2,\ldots,2\ell}\} \\ \mathit{mkey} &= \{\hat{\zeta}, \check{\zeta}, \hat{\alpha}, \check{\alpha}, (\hat{\delta}_j, \check{\delta}_j, \hat{\beta}_j, \check{\beta}_j)_{j=1,\ldots,\ell}, (\hat{\gamma}_i, \check{\gamma}_i)_{i=1,\ldots,\ell}\} \\ \mathit{tkey} &= \check{G} \end{aligned}$$

Ext: Given a receiver identity $(i,j) \in \{1,\ldots,\ell\}^2$, *mkey*, and *pkey*, Ext generates

$$\hat{K}_{ij} = [\hat{\delta}_j \hat{\alpha}^i \hat{\theta}_i + \hat{\zeta} + \hat{\beta}_j \hat{\gamma}_i]\hat{G} \quad , \quad \check{K}_{ij} = [\check{\delta}_j \check{\alpha}^i \check{\theta}_i + \check{\zeta} + \check{\beta}_j \check{\gamma}_i]\check{G} \quad , \quad K_{ij} = \hat{K}_{ij} + \check{K}_{ij}.$$

Then, Ext outputs $\mathit{skey}_{ij} = K_{ij}$.

Enc: Given $S \subseteq \Lambda$, Enc randomly chooses $k \in \mathcal{G}_T$, $(\sigma_j)_{j=1,\ldots,\ell} \in_R (\mathbb{Z}/n\mathbb{Z})^\ell, (\epsilon_i)_{i=1,\ldots,\ell} \in_R (\mathbb{Z}/n\mathbb{Z})^\ell, \tau \in_R \mathbb{Z}/n\mathbb{Z}$ and generates

$$\begin{aligned} (c_j, \hat{E}_j, \hat{F}_j, \hat{R}_j, \hat{T}_j) &= \left(k \cdot \hat{m}'^{\sigma_j \tau}, [\sigma_j \tau]\hat{G}', [\sigma_j \tau](\hat{D}'_j + \sum_{k \in \bar{S} \cap \Lambda_j} \hat{G}'_{\ell+1-k}), [\sigma_j]\hat{B}'_j, [\sigma_j]\hat{H}'_j \right) \\ (S_i, U_i) &= ([\tau]J_i + [\epsilon_i]H, [\epsilon_i]G) \end{aligned}$$

for $i, j = 1, \ldots, \ell$. Then, Enc outputs, a tuple

$$hdr_S \;=\; ((c_j, \hat{E}_j, \hat{F}_j, \hat{R}_j, \hat{T}_j)_{j=1,\ldots,\ell}, (S_i, U_i)_{i=1,\ldots,\ell})$$

Dec: Suppose that hdr_S, $pkey$, and $skey_{ij}$ such that $(i, j) \in S$ are given. Dec generates

$$\hat{k}_{ij} \;=\; \frac{e(X_i^{(i)}, \hat{F}_j)}{e(\hat{E}_j, K_{ij} + \sum_{k \in \bar{S} \cap \Lambda_j} X_{\ell+1-k+i}^{(i)})} \cdot \frac{e(\hat{R}_j, S_i)}{e(\hat{T}_j, U_i)}$$

Then, Dec outputs $key = k = c_j \cdot \hat{k}_{ij}$.

Revoke: We give the description of Probe. Revoke works by using this algorithm.

Probe: Suppose that $S \subseteq \Lambda$ and $(\bar{i}, \bar{j}) \in S$ are given. Probe randomly chooses $k \in_R \mathcal{G}_T, (\sigma_j)_{j=1,\ldots,\ell} \in_R (\mathbb{Z}/n\mathbb{Z})^\ell, \tau \in_R \mathbb{Z}/n\mathbb{Z}, (k'_j)_{1,\ldots,j=\bar{j}-1} \in_R (\mathcal{G}_T)^{\bar{j}-1}, (\sigma'_j)_{1,\ldots,j=\bar{j}-1} \in_R (\mathbb{Z}/n\mathbb{Z})^{\bar{j}-1}$ and generates $(c_j, \hat{E}_j, \hat{F}_j, \hat{R}_j, \hat{T}_j)_{j=1,\ldots,\ell}$ as in the following:

$$\text{For } j > \bar{j}, \quad \left(k \cdot e(\hat{G}', M)^{\sigma_j \tau}, [\sigma_j \tau]\hat{G}', [\sigma_j \tau](\hat{D}'_j + \sum_{k \in S \cap \Lambda_j} \hat{G}'_{\ell+1-k}), [\sigma_j]\hat{B}'_j, [\sigma_j]\hat{H}'_j \right).$$

$$\text{For } j = \bar{j}, \quad \left(k \cdot e(G, M)^{\sigma_{\bar{j}} \tau}, [\sigma_{\bar{j}} \tau]G, [\sigma_{\bar{j}} \tau](D_{\bar{j}} + \sum_{k \in S \cap \Lambda_{\bar{j}}} G_{\ell+1-k}), [\sigma_{\bar{j}}]B_{\bar{j}}, [\sigma_{\bar{j}}]H_{\bar{j}} \right).$$

$$\text{For } j < \bar{j}, \quad \left(k'_j \cdot e(G, M)^{\sigma_j \tau}, [\sigma_j \tau]G, [\sigma_j \tau](D_j + \sum_{k \in S \cap \Lambda_j} G_{\ell+1-k}), [\sigma'_j]B_j, [\sigma'_j]H_j \right).$$

Next, Probe randomly chooses $(\epsilon_i)_{i=1,\ldots,\ell} \in_R (\mathbb{Z}/n\mathbb{Z})^\ell, (\epsilon'_i)_{i=1,\ldots,\bar{i}} \in_R (\mathbb{Z}/n\mathbb{Z})^\ell$ and generates $(S_i, U_i)_{i=1,\ldots,\ell}$ as in the following:

$$\text{For } i \geq \bar{i}, \quad ([\tau]J_i + [\epsilon_i]H, [\epsilon_i]G).$$
$$\text{For } i < \bar{i}, \quad ([\tau]J_i + [\epsilon_i]H + [\epsilon'_i]\check{G}, [\epsilon_i]G).$$

Then, Enc outputs, a tuple $hdr_S = ((c_j, \hat{E}_j, \hat{F}_j, \hat{R}_j, \hat{T}_j)_{j=1,\ldots,\ell}, (S_i, U_i)_{i=1,\ldots,\ell})$.

Theorem 27 *The scheme is complete.*

Theorem 28 *The basic scheme is key indistinguishable against CPA by ACA in the generic bilinear group model.*

	Our scheme	[26]	[25]	[24]
private key length	$\mathcal{O}(1)$	$\mathcal{O}(\sqrt{N})$	$\mathcal{O}(1)$	$\mathcal{O}(1)$
ciphertext length	$\mathcal{O}(\sqrt{N})$	$\mathcal{O}(\sqrt{N})$	$\mathcal{O}(\sqrt{N})$	$\mathcal{O}(1)$
public key length	$\mathcal{O}(N)$	$\mathcal{O}(\sqrt{N})$	$\mathcal{O}(\sqrt{N})$	$\mathcal{O}(N)$
black-box revoking	secret-key	secret-key	N/A	N/A

Table 6.1: Complexity of schemes

Theorem 29 *The basic scheme is black-box traitor revocable against CPA by ACA in the generic bilinear group model.*

The proof or proof sketch of the above three theorems are given in Section 6.7.

6.6 Efficiency and Security against CCA

Suppose the total number of user is N and the total number of traitor is $t \leq N$. The basic scheme requires the size of private keys to be $\mathcal{O}(1)$, that of ciphertexts to be $\mathcal{O}(\sqrt{N})$, and that of public key to be $\mathcal{O}(N)$. The cost for revocation in black-box manner is $\mathcal{O}(ts^2)$. The proposed scheme is compared with other schemes in Table 6.1 Although the size of public key in our scheme is $\mathcal{O}(N)$. the size of public key that each receiver accesses is $\mathcal{O}(\sqrt{N})$. Receivers are not required to access to the public key as long as the ciphertexts are distributed to the same set of receivers.

The basic scheme is secure only against chosen plaintext attacks. One is able to construct the black-box traitor revocable broadcast encryption scheme that is secure against "adaptive chosen ciphertexts" by adaptively colluding adversaries in the generic bilinear group if one uses the technique of [38]. Since this is the trivial extension of the basic scheme, we omit the detail of its construction here.

6.7 Security of the Basic Scheme

Theorem 3 *The scheme is complete.*

Proof.

$$\begin{aligned}
\hat{k}_{ij} &= \frac{e(X_i^{(i)}, \hat{F}_j)}{e(\hat{E}_j, K_{ij} + \sum_{k \in \bar{S} \cap \Lambda_j} X_{\ell+1-k+i}^{(i)})} \cdot \frac{e(\hat{R}_j, S_i)}{e(\hat{T}_j, U_i)} \\
&= \frac{e([\theta_i]G_i, [\sigma_j \tau](\hat{D}'_j + \sum_{k \in \bar{S} \cap \Lambda_j} \hat{G}'_{\ell+1-k}))}{e([\sigma_j \tau]\hat{G}', [\delta_j \alpha^i \theta_i]G + M + [\beta_j \gamma_i]G + \sum_{k \in \bar{S} \cap \Lambda_j} [\theta_i] G_{\ell+1-k+i})}
\end{aligned}$$

$$\begin{aligned}
& \cdot \frac{e([\sigma_j]\hat{B}'_j, [\tau]J_i + [\epsilon_i]H)}{e([\sigma_j]\hat{H}'_j, [\epsilon_i]G)} \\
= & \frac{e([\theta_i\alpha^i]G, [\sigma_j\tau]([\eta\delta_j]\hat{G} + \sum_{k\in\bar{S}\cap\Lambda_j}[\eta]\hat{G}_{\ell+1-k}))}{e([\sigma_j\tau][\eta]\hat{G}, [\delta_j\alpha^i\theta_i]G + M + [\beta_j\gamma_i]G + \sum_{k\in\bar{S}\cap\Lambda_j}[\theta_i]G_{\ell+1-k+i})} \\
& \cdot \frac{e([\sigma_j][\eta\beta_j]\hat{G}, [\tau][\gamma_i]G + [\epsilon_i]H)}{e([\sigma_j][\eta\beta_j]\hat{H}, [\epsilon_i]G)} \\
= & \frac{e([\theta_i\alpha^i]G, [\sigma_j\tau\eta]([\delta_j]\hat{G} + \sum_{k\in\bar{S}\cap\Lambda_j}\hat{G}_{\ell+1-k}))}{e([\sigma_j\tau\eta]\hat{G}, [\delta_j\alpha^i\theta_i]G + M + [\beta_j\gamma_i]G + \sum_{k\in\bar{S}\cap\Lambda_j}[\theta_i]G_{\ell+1-k+i})} \\
& \cdot \frac{e([\beta_j\sigma_j\eta]\hat{G}, [\tau\gamma_i]G + [\epsilon_i]H)}{e([\beta_j\sigma_j\eta]\hat{H}, [\epsilon_i]G)} \\
= & \frac{e([\theta_i\alpha^i]G, [\sigma_j\tau\eta]([\delta_j]\hat{G} + \sum_{k\in\bar{S}\cap\Lambda_j}\hat{G}_{\ell+1-k}))}{e([\sigma_j\tau\eta]\hat{G}, [\delta_j\alpha^i\theta_i]G + M + \sum_{k\in\bar{S}\cap\Lambda_j}[\theta_i]G_{\ell+1-k+i})} \\
& \cdot \frac{e([\beta_j\sigma_j\eta]\hat{G}, [\tau\gamma_i]G)}{e([\sigma_j\tau\eta]\hat{G}, [\beta_j\gamma_i]G)} \cdot \frac{e([\beta_j\sigma_j\eta]\hat{G}, [\epsilon_i]H)}{e([\beta_j\sigma_j\eta]\hat{H}, [\epsilon_i]G)} \\
= & e(\hat{G}, M)^{-\sigma_j\tau\eta} \\
= & e(\hat{G}', M)^{-\sigma_j\tau}
\end{aligned}$$

Then,

$$\begin{aligned}
k &= c_j\hat{k}_{ij} \\
&= k \cdot e(\hat{G}', M)^{\sigma_j\tau} e(\hat{G}', M)^{-\sigma_j\tau} \\
&= k
\end{aligned}$$

□

Theorem 4 *We assume the generic bilinear group model, the subgroup decision assumption, and the bilinear subgroup decision assumption. Then, the basic scheme is traitor revocable against CPA by ACA.*

A proof sketch:

Lemma 47 *In the generic bilinear group model, no polynomially bounded adversary wins strong-index indistinguishable against CPA by ACA game when $u = (\bar{i}, \bar{j}) \neq (\ell, j)$ for all j.*

Proof.

We consider $\mathcal{C}$ that knows all the secrets and runs Gen, Ext, and Probe in the generic bilinear group model. Operations in the generic bilinear group of order pq are executed by running two generic bilinear groups of order p and q in parallel. That is, each element in groups of order pq are represented as a pair of elements in groups of order p and order q.

We focus on the elements of $\check{\mathcal{G}}$ and $\check{\mathcal{G}}_T$. Hence, we assume $\mathcal{C}$ gives the adversary $\check{\mathcal{G}}$ and $\check{\mathcal{G}}_T$ components of each element in $\mathcal{G}$ and $\mathcal{G}_T$. Moreover, we assume the adversary has $\check{G}$. Hence, we no longer need to consider revocation queries. In the following we removed all " ˇ " for simplicity.

From *pkey* and $(skey_{ij})_{(i,j)\in T}$. The adversary obtains the $[P_1]\check{G}$ and $[Q_1]\check{G}_T$ for the following P_1 and Q_1.

$$\begin{aligned}
P_1 &= (\xi, \\
&\quad (\beta_j)_{j=1,\ldots,\ell}, \\
&\quad (\beta_j\xi)_{j=1,\ldots,\ell}, \\
&\quad (\alpha^i)_{i=1,\ldots,\ell,\ell+2,\ldots,2\ell}, \\
&\quad (\delta_j)_{j=1,\ldots,\ell}, \\
&\quad (\gamma_i)_{i=1,\ldots,\ell}, \\
&\quad (\theta^{(k)}\alpha^i)_{k=1,\ldots,\ell;i=1,\ldots,\ell,\ell+2,\ldots,2\ell} \\
&\quad (\delta_j\alpha^i\theta^{(i)} + \zeta + \gamma_i\beta_j)_{(i,j)\in T}, \\
Q_1 &= (\zeta)
\end{aligned}$$

We assume k are known to the adversary. If we consider the freedom of $k'_j, \check{\sigma}_j, \check{\sigma}'_j$ for $j < \bar{j}$, the unsimulatable data that the adversary obtains from Probe is $[f]\check{G}$, $[P_2]\check{G}$ and $[Q_2]\check{G}_T$ for the following P_2 and Q_2.

$$\begin{aligned}
f &= \tau\gamma_{\bar{i}} + \epsilon_{\bar{i}}\xi, \\
P_2 &= \{f, \sigma_{\bar{j}}\tau, \sigma_{\bar{j}}\tau(\delta_{\bar{j}} + \sum_{k\in \bar{S}^\dagger\cap\Lambda_{\bar{j}}} \alpha^{\ell+1-k}), \sigma_{\bar{j}}\beta_{\bar{j}}, \sigma_{\bar{j}}\beta_{\bar{j}}\xi, \\
&\quad (\tau\gamma_i + \epsilon'_i\xi, \epsilon_i)_{i=1,\ldots,\bar{i}-1}, \epsilon_{\bar{i}}, (\tau\gamma_i + \epsilon_i\xi, \epsilon_i)_{i=\bar{i}+1,\ldots,\ell}, \epsilon_{\bar{i}}\} \\
Q_2 &= \sigma_{\bar{j}}\tau\zeta.
\end{aligned}$$

The distribution of above $P = (P_1, P_2), Q = (Q_1, Q_2)$ is identical to the output of Probe is $\mathsf{Probe}(tkey, pkey, S^\dagger, u = (\bar{i}, \bar{j}))$. Let P' be identical to P except f is independent variable and Q' be identical to Q. Then the distribution of (P', Q') is identical to that of $\mathsf{Probe}(tkey, pkey, S^\dagger, u = (\bar{i}+1, \bar{j}))$. Therefore, deciding whether $u = (\bar{i}, \bar{j})$ or $u = (\bar{i}+1, \bar{j})$ in the generic bilinear group model is equivalent to solving the general bilinear decision problem with respect to the (P, Q) and (P', Q').

With tiresome exhaustive chasing of terms, we can see they are independent unless $((\bar{i}, \bar{j}) \in T) \cap ((\bar{i}, \bar{j}) \in S^\dagger)$. Its rough view is given in the following.

Suppose the problem is dependent, there should be term that contains f in the equivalent equation since this is the only difference between the two distributions. Then quadratic terms that can include $\epsilon_{\bar{i}}\xi$ in f must exists and it is only $\xi \cdot \epsilon_{\bar{i}}$ and $\epsilon_{\bar{i}} \cdot \sigma_{\bar{j}}\beta_{\bar{j}}\xi$. Then, the quadratic terms that can include the remaining $\tau\gamma_{\bar{i}}$ in f are

$$\gamma_{\bar{i}} \cdot \sigma_{\bar{j}}\tau \quad , \quad \gamma_{\bar{i}} \cdot \sigma_{\bar{j}}\tau(\delta_{\bar{j}} + \sum_{k \in \bar{S}^{\dagger} \cap \Lambda_{\bar{j}}} \alpha^{\ell+1-k})$$
$$(\delta_j \alpha^{\bar{i}} \theta^{(\bar{i})} + \zeta + \gamma_{\bar{i}}\beta_j) \cdot \sigma_{\bar{j}}\tau \quad , \quad (\delta_j \alpha^{\bar{i}} \theta^{(\bar{i})} + \zeta + \gamma_{\bar{i}}\beta_j) \cdot \sigma_{\bar{j}}\tau(\delta_{\bar{j}} + \sum_{k \in \bar{S}^{\dagger} \cap \Lambda_{\bar{j}}} \alpha^{\ell+1-k})$$

The first two and 4-th terms can provide f no its counter term. The only possibility is the term $(\delta_{\bar{j}} \alpha^{\bar{i}} \theta^{(\bar{i})} + \zeta + \gamma_{\bar{i}}\beta_{\bar{j}}) \cdot \sigma_{\bar{j}}\tau$ when $(\bar{i}, \bar{j}) \in T$, where the counter term of f is the term $\sigma_{\bar{j}}\beta_{\bar{j}}$. Hence, at least $(\bar{i}, \bar{j}) \in T$ should hold for f to be dependent of P ad Q.

Continuing term chasing, we see that there should be a term that cancels $(\delta_{\bar{j}} \alpha^{\bar{i}} \theta^{(\bar{i})} + \zeta)\sigma_{\bar{j}}\tau$. The only term that can cancel $\delta_{\bar{j}} \alpha^{\bar{i}} \theta^{(\bar{i})} \sigma_{\bar{j}}\tau$ is the term

$$\theta^{(\bar{i})} \alpha^{\bar{i}} \cdot \sigma_{\bar{j}}\tau(\delta_{\bar{j}} + \sum_{k \in \bar{S}^{\dagger} \cap \Lambda_{\bar{j}}} \alpha^{\ell+1-k})$$

Then, the resulting term that we must cancel are

$$\theta^{(\bar{i})} \sigma_{\bar{j}}\tau \sum_{k \in \bar{S}^{\dagger} \cap \Lambda_{\bar{j}}} \alpha^{\ell+1-k+\bar{i}}$$

If $\bar{i} \notin \bar{S}^{\dagger} \cap \Lambda_{\bar{j}}$, all terms can be canceled by quadratic terms $\sigma_{\bar{j}}\tau \cdot \theta^{(\bar{i})} \alpha^{\ell+1-k+\bar{i}}$. However, if $\bar{i} \in \bar{S}^{\dagger} \cap \Lambda_{\bar{j}}$, i.e., $\bar{i} \notin S^{\dagger}$, we see no term that can cancel the term

$$\sigma_{\bar{j}} \theta^{(\bar{i})} \alpha^{\ell+1} \tau.$$

Hence, at least $(\bar{i}, \bar{j}) \in S$ should hold for f to be dependent of P ad Q. Note that the factor $\sigma_{\bar{j}} \theta^{(\bar{i})}$ depends on both $\bar{i}$ and $\bar{j}$. If it depends on only either of indices, such a term can be generated from the data related to other indices such as $(i, \bar{j})$ or $(\bar{i}, j)$. This is why we modified the basing broadcast encryption scheme in [24].

(The term $\zeta\sigma_{\bar{j}}\tau$ can be canceled by chasing the term $(\delta_{\bar{j}} \alpha^{i} \theta^{(i)} + \zeta)\sigma_{\bar{j}}\tau$ for $(i, \bar{j}) \in S^{\dagger}$. This term chasing is almost reverse of the above term chasing with respect to $(i, \bar{j})$.) Therefore, the lemma is proved. □

Lemma 48 *In the generic bilinear group model of composite order, no polynomially bounded adversary* *wins strong-index indistinguishable against CPA by ACA game when* $u = (\bar{i}, \bar{j}) = (\ell, j)$ *for some* j.

Proof. We consider the following 5 distributions. Idea to consider these distributions are introduced in [25].

D1 The output distribution of $\mathsf{Probe}(\mathit{tkey}, \mathit{key}, S^\dagger, u = (\ell, j))$.

D2 The output distribution of $\mathsf{Probe}(\mathit{tkey}, \mathit{key}, S^\dagger, u = (\ell + 1, j))$.

D3 Same as the output distribution of $\mathsf{Probe}(\mathit{tkey}, \mathit{key}, S^\dagger, u = (\ell+1, j))$ except that $[\sigma_{\bar{j}}]B_{\bar{j}}$, and $[\sigma_{\bar{j}}]H_{\bar{j}}$ in $\bar{j}$-th row are replaced with $[\sigma'_{\bar{j}}]B_{\bar{j}}$, and $[\sigma'_{\bar{j}}]H_{\bar{j}}$ where $k'_{\bar{j}} \in \mathcal{G}_T, \sigma'_{\bar{j}} \in_R \mathcal{G}$ are randomly chosen.

D3' Same as the output distribution of $\mathsf{Probe}(\mathit{tkey}, \mathit{key}, S^\dagger, u = (\ell + 1, j))$ except that $k, [\sigma_{\bar{j}}]B_{\bar{j}}$, and $[\sigma_{\bar{j}}]H_{\bar{j}}$ in $\bar{j}$-th row are replaced with $k'_{\bar{j}}, [\sigma'_{\bar{j}}]B_{\bar{j}}$, and $[\sigma'_{\bar{j}}]H_{\bar{j}}$ where $k'_{\bar{j}} \in \mathcal{G}_T, \sigma'_{\bar{j}} \in_R \mathcal{G}$ are randomly chosen.

D4 Same as the output distribution of $\mathsf{Probe}(\mathit{tkey}, \mathit{key}, S^\dagger, u = (1, j + 1))$ except that $\hat{G}', \hat{G}'_i, \hat{D}'_{\bar{j}+1}$, $[\hat{\sigma}_{\bar{j}+1}]\hat{B}'_{\bar{j}+1}$, and $[\hat{\sigma}_{\bar{j}+1}]\hat{H}'_{\bar{j}+1}$ in $(\bar{j} + 1)$-th row are replaced with $G, G_i, D_{\bar{j}+1}, [\sigma_{\bar{j}+1}]B_{\bar{j}+1}$, and $[\sigma_{\bar{j}+1}]H_{\bar{j}+1}$.

D5 $\mathsf{Probe}(\mathit{tkey}, \mathit{key}, S^\dagger, u = (1, j + 1))$.

Here, $\mathsf{Probe}(\mathit{tkey}, \mathit{key}, S^\dagger, u = (\ell + 1, j))$ is defined in natural way as in [25]. Then, we will prove that distributions D1 and D5 are indistinguishable.

Distinguishability between $D1$ and $D2$ can be analyzed exactly in the same way as in Lemma 47.

Indistinguishability between $D2$ and $D3$ in the generic bilinear group model can be proved by exhaustive term chasing as we did in Lemma 47. But this term chasing ends much earlier. Here we may assume $f = \sigma_{\bar{j}}\beta_{\bar{j}}$ and ξ is revealed to the adversary.

Indistinguishability between $D3$ and $D3'$ in the generic bilinear group model can be proved by exhaustive term chasing as we did in Lemma 47. But this term chasing ends much earlier. Here we assume $f = \sigma_{\bar{j}}\tau\zeta$ and prove independence in the sense of the general Diffie-Hellman exponent problem.

Indistinguishability between $D3$ and $D4$ in the generic bilinear group model can be proved by exhaustive term chasing. Here, we use hybrid argument. Indistinguishability of its each step is proved by exhaustive term chasing which is similar to that we do for proving the indistinguishability between $D2$ and $D3$.

Indistinguishability between $D4$ and $D5$ can be proved easily if the adversary asks no revocation queries. This is because it can be reduced to a subgroup decision assumption. But such simplified case is not the case we are considering. Because of the symmetry between elements of order q and elements of order n in $\mathit{pkey}, \mathit{mkey}$, and skey, the two distributions are independent unless outputs of Probe are considered. Hence, non equivalent terms arises from factor $[\epsilon'_i]\hat{G}$. However, as we saw in the proof of Lemma 47, the only term that can couple is the corresponding $\sigma_{\bar{j}}\beta_{\bar{j}}$. Then, since $\epsilon'_i \cdot \sigma_{\bar{j}}\beta_{\bar{j}}$ is already quadratic term, we are unable to generate dependent relations. Thus, the exhaustive term chasing terminates. Therefore, the lemma is proved. $\square$

Lemma 49 *The* Probe *in the basic scheme satisfies boundary condition against CPA by ACA.*

A proof sketch: The only difference between our basic scheme and the scheme in [25] is that our scheme uses $e(X_i^{(i)}, \hat{F}_j)$ and $\sum_{k \in \bar{S} \cap \Lambda_j} X_{\ell+1-k+i}^{(i)}$ instead of ${G_x}^{s_x t}$ and $E^{s_x t}$ in [25]. If we switch these values, the proof that the scheme in [25] is message hiding can be well translated to the proof that our scheme satisfies boundary condition against CPA by ACA. Hence, we omit to present the detail of this proof here. $\square$

From the above lemmas, the theorem follows. $\square$

Theorem 5 *The basic scheme is key indistinguishable against chosen plaintext attacks against CPA by ACA in generic bilinear group model.*

A proof sketch: The theorem is almost proven in Lemma 47. If we consider Q_2 as f in the general Diffie-Hellman exponent problem presented in [21]. We can easily reconstruct the proof of this lemma from the proof of Lemma 47. $\square$

Chapter 7

Group Signature for Separate and Distributed Authorities

7.1 Introduction

A group signature scheme, first proposed by Chaum and van Heyst [44] and followed by [6, 7, 19, 30, 32, 33, 35, 121], allows a group member who has registered with the group to sign messages on behalf of a group without revealing his own identity. One notable feature of the group signature scheme is the existence of an authority that can identify this actual signer. In recent group signature schemes, this authority, known as a *group manager*, also had the ability to add a user to a group. Such centralization of power in a single entity, however, was considered undesirable.

To solve this problem, Kilian and Petrank proposed in [95] to separate these roles into a *membership manager*, who would add new members to a group, and a *tracing manager*, who would identify actual signers. Many similar separate-role group signature schemes [33, 6] followed. We believe that it is even more preferable if the capabilities of the membership manager and the tracing manager can be efficiently distributed, respectively. In this sense, the schemes in [33, 6] do not meet our goal, as the secret keys used by the membership manager in these schemes are based on RSA. Namely, in case the RSA-based scheme is used, authorities should collaboratively generate an RSA modulus in a setup process with its prime factors kept secret to all the authorities, which would not be efficiently computable. Moreover, in a process of member addition, it would take large computation for distributed membership managers every time a new user is added. By way of contrast, in the scheme proposed in [7, 114], the secret key for adding a new member is a discrete logarithm of a public key. Therefore the approach of [7, 114] seems promising because distribution could be efficiently executed by using the method proposed in [130]. Unfortunately, the group signature scheme as is presented in [7] does not achieve separation of the membership manager and the tracing manager since, in the scheme in [7], a resulting group signature contains the value which is the output of a deterministic function. Then, those who knows all membership certificates can identify the actual signer of the group signature from the value even if they do not know secret information for tracing. Thus, the scheme in [7] does not achieve separation of authorities. As to [114], the proposed scheme turns out to be insecure [1]. Therefore, none of the scheme suites for separation and distribution of the authorities, and the scheme that achieves separation of the above two authorities and distribution of both capabilities is desired.

In this chapter, we give a first group signature scheme that achieves both separation and efficient distribution of the membership manager and the tracing manager. In our scheme, the membership manager uses the Nyberg-Rueppel signature scheme [123] to provide a group member with the ability to create a group signature, and the tracing manager uses the ElGamal cryptosystem to trace the actual signer. Both of these primitives are well-suited to distributed computations, and authority-distribution on our scheme carried out efficiently. Moreover, in our proposed scheme, a signer creates a perfectly hiding commitment of a membership certificate instead of a deterministic function in [7]. This prevents the membership manager from identifying the signer from the signature, thus separation is achieved. The price we pay for simultaneously achieving separation and distribution is the increase in signature length and in computational costs, which is about 4.5 times larger than that of [7].

In this chapter, we also formalize the conditions required for security in the group signature scheme in which authorities are both initially separated. The security definitions proposed in Bellare et al. [15] only apply to the case in which all duties of group management are performed by a single manager, and that manager is assumed to be honest. In contrast to this, we formalize our security definition in such a way that (1) neither the tracing manager nor any other entity can add a new member, (2) neither the membership manager nor any other entity can trace an actual signer, and (3) these two managers cannot, even in collusion, create a group signature that would appear to be traceable to an individual group manager. Later, similar security notions are independently proposed in [14, 93]. The security of our proposed scheme can be proved in the random oracle and generic model.

7.2 The Model

In this section, we present a model of our group signature scheme and its security definitions. Note that no previous work has specified security definitions for the case in which a membership manager and a

[1]In [114], finding $(r, x, y) \in Z_P \times Z_q^2$ satisfying $ry^r h^x f^y = 1 \bmod P$ is claimed to be difficult, where $P = pq$ and $h^q = y^q = f^q = 1 \bmod P$. However, $r_q := r \bmod q = 1$ holds since $y = h = f = 1 \bmod q$ and finding $r_p \in Z_p^*$ such that $r_p y^{r_q} h^x f^y = 1 \bmod p$ is clearly easy. Hence, from $r \bmod p = r_p$ and $r \bmod q = 1$, we have desired (r, x, y).

tracing manager act independently of one another.

7.2.1 Model of Group Signature Scheme

Let $b \leftarrow \mathsf{AL}(a)$ denote an algorithm AL, where its input is a and its output is b. Let $\langle c, d\rangle \leftarrow \mathsf{IP}_{A,B}\langle a, b\rangle$ denote an interactive protocol IP between A and B, where private inputs to A and B are, respectively, a and b, and outputs of A and B are, respectively, c and d.

Our model contains three main entity types: a membership manager (MM), a tracing manager (TM), group members U, and a verifier V. The MM is able to add a new member to his group by using a secret key which he generates. More specifically, when a user applies for membership, the MM will conduct a joining procedure together with that user. In completing this procedure, the user will obtain the information which will be required to create a group signature. Such information can only be provided by the MM since the secret key of MM is required to generate it. The MM will subsequently publish these information issued to users. These information would be required for the TM to identify the actual signer of a group signature. The TM is able to identify the actual signer of a given group signature by using a secret generated by himself.

The model of the group signature scheme is defined as follows. In this model, we do not consider revocation for the sake of simplicity.

Definition 56 *Players in the group signature scheme are a membership manager MM, a tracing manager TM, a group member U and a verifier V. $k \in \mathbb{N}$ is a security parameter.*

A group signature scheme $\mathcal{GS}$ *consists of the following five algorithms and one interactive protocol.* (M-KeyGen, T-KeyGen, Join, Sign, Verify, Open),

- *A probabilistic key generation algorithm for MM that, given a security parameter 1^k, outputs a* membership public key *mpk and a* membership secret key *msk.*

$$(msk, mpk) \leftarrow \mathsf{M\text{-}KeyGen}(1^k)$$

- *A probabilistic key generation algorithm for TM that, given mpk, outputs a* tracing public key *tpk and a* tracing secret key *tsk.*

$$(tsk, tpk) \leftarrow \mathsf{T\text{-}KeyGen}(mpk)$$

- *An interactive member registration protocol for the MM and a user U. MM is given mpk, msk,* *the user's identity* U^2*, and a list of all group members* $\mathcal{L}$*.* U *is given* mpk*. If the interaction was successful,* U *outputs a* membership certificate $cert_U$*, a* membership secret sk_U*, and an* identifier ider_U *and* MM *adds a pair* (U, ider_U) *to* $\mathcal{L}$ *and outputs this revised* $\mathcal{L}$*.*

$$\begin{aligned}&\langle(\mathcal{L}), (cert_U, sk_U, ider_U)\rangle \\ &\qquad\qquad \leftarrow \mathsf{Join}_{MM,U}\langle(\mathcal{L}, U, mpk, msk), (mpk)\rangle\end{aligned}$$

- *A probabilistic signature generation algorithm for a* U *that, given* mpk*,* tpk*,* $\mathrm{cert}_U, \mathrm{sk}_U$*, and a* message m*, outputs a group signature* gs *on the message* m*.*

$$gs \leftarrow \mathsf{Sign}(mpk, tpk, cert_U, sk_U, m)$$

- *A deterministic signature verification algorithm for any* V *that, given* mpk*,* tpk*,* m*, and* gs*, returns either* acc *or* rej*. Here,* acc *and* rej *represent, respectively, an acceptance and a rejection of the signature.*

$$acc/rej \leftarrow \mathsf{Verify}(mpk, tpk, m, gs)$$

We say that a group signature gs *on* m *is* valid *if* acc $\leftarrow \mathsf{Verify}(mpk, tpk, m, gs)$*.*

- *A deterministic signer tracing algorithm for the* TM *that, given* mpk*,* tpk*,* tsk*,* m*, and* gs*, outputs* $\perp$ *if* gs *on* m *is not valid. Otherwise, it outputs* $(U, proof)$*, where* proof *assures the validity of the result* U*. If the algorithm cannot find the actual signer in* $\mathcal{L}$*, the algorithm outputs* $\perp'$ *instead of* U*.*

$$\perp/(U/\perp', proof) \leftarrow \mathsf{Open}(mpk, tpk, tsk, m, gs, \mathcal{L})$$

7.2.2 Security Definitions

In this section, we describe the security definitions of a group signature scheme that includes separation of authorities. We define one security notion for each type of entities in the model. With respect to the MM, we require that no one except the MM is able to successfully add a new member to the group. With respect to the TM, we require that no one except the TM is able to successfully identify the actual signer of a signature. With respect to the members, we require that no one except each member is able

[2] *We use the same notation* U *for a user and the identity of this user* U*.*

to successfully create a signature which will be linked to his identity when opened by the TM. These requirements, which is later defined specifically, clarifies the separation of the MM and the TM. We call security properties corresponding the above requirements *MM-invulnerability*, *TM-invulnerability*, *member-invulnerability*, respectively.

MM-invulnerability MM-invulnerability is the property that no colluding subset of the group members, even colluding with the TM, can create a new signature such that the TM can, in any way, prove that the signer of a signature is not in the member list $\mathcal{L}$. Since a creation of a signature whose signer does not belong to $\mathcal{L}$ implies adding a new member, MM-invulnerability implies that only the MM can play the role of the MM. To define MM-invulnerability, we introduce an adversary $\mathcal{A}$ who attempts to break the group signature scheme in terms of MM-invulnerability. Adversary $\mathcal{A}$ is allowed to collude the TM and all group members. Formally, MM-invulnerability is defined as follows.

Definition 57 (**MM-invulnerability**) *Let $\mathcal{GS}$ be a group signature scheme, and let $\mathcal{A}$ be an algorithm. We consider the following experiment that returns* $0/1$. *Here, we assume that* Join *protocols are executed only sequentially.*

```
Experiment $\mathbf{Exp}^{\mathrm{MI}}_{\mathcal{GS},\mathcal{A}}(k)$
    $(mpk, msk) \leftarrow \mathsf{M\text{-}KeyGen}(1^k)$
    $(tpk, State) \leftarrow \mathcal{A}(mpk)$
    $Cont \leftarrow \mathsf{true}$
    While $Cont = \mathsf{true}$ do
        $\langle(\mathcal{L}), (State, Cont)\rangle$
                $\leftarrow \mathsf{Join}_{MM,A}\langle(\mathcal{L}, U, mpk, msk), (mpk, State)\rangle$
    EndWhile
    $(m, gs) \leftarrow \mathcal{A}(State)$
    If $rej \leftarrow \mathsf{Verify}(mpk, tpk, m, gs)$ then return 0
    If $(\perp', proof) \leftarrow \mathsf{Open}(mpk, tpk, tsk, m, gs, \mathcal{L})$,
        then return 1
    Return 0
```

A group signature scheme $\mathcal{GS}$ has MM-invulnerability *property if for all probabilistic, polynomial-time machines $\mathcal{A}$,*

$$\Pr[\mathbf{Exp}^{\mathrm{MI}}_{\mathcal{GS},\mathcal{A}}(k) = 1]$$

is negligible in k.

TM-invulnerability TM-invulnerability is the property that, given a message and a group signature, no colluding subset of the group members, even colluding with the MM, can identify the signer of the signature unless the TM opens that very signature. TM-invulnerability implies that only the TM can play the role of the TM. To define TM-invulnerability, we introduce an adversary $\mathcal{A}$ who attempts to break a group signature scheme in terms of TM-invulnerability. Adversary $\mathcal{A}$ is allowed to collude the MM and the all group members. $\mathcal{A}$ can also add a new member to the group using the colluding MM. Formally, TM-invulnerability is defined as follows.

Definition 58 *(***TM-invulnerability***) Let $\mathcal{GS}$ be a group signature scheme, let $b \in \{0,1\}$, and let $\mathcal{A}$ be an algorithm. We consider the following experiment that returns* $0/1$*.*

```
Experiment $\mathbf{Exp}^{\mathrm{TI}}_{\mathcal{GS},\mathcal{A}}(k, b)$
    $(mpk, State) \leftarrow \mathcal{A}(1^k)$
    $(tpk, tsk) \leftarrow \mathsf{T\text{-}KeyGen}(mpk)$
    $(State, (cert_0, sk_0), (cert_1, sk_1), m)$
        $\leftarrow \mathcal{A}^{\mathsf{Open}(mpk,tpk,tsk,\cdot,\cdot,\cdot)}(State, tpk)$
    $gs \leftarrow \mathsf{Sign}(mpk, tpk, cert_b, sk_b, m)$
    $(b' \in \{0,1\} \leftarrow \mathcal{A}^{\mathsf{Open}(mpk,tpk,tsk,\cdot,\cdot,\cdot)}(State, gs)$
    If $\mathcal{A}$ did not query Open oracle with $(m, gs)$ after
    $gs$ is given, then return $b'$
    Return 0
```

A group signature scheme $\mathcal{GS}$ has TM-invulnerability *property if for all probabilistic polynomial-time machines $\mathcal{A}$,*

$$\Pr[\mathbf{Exp}^{\mathrm{TI}}_{\mathcal{GS},\mathcal{A}}(k,0) = 1] - \Pr[\mathbf{Exp}^{\mathrm{TI}}_{\mathcal{GS},\mathcal{A}}(k,1) = 1]$$

is negligible in k.

Member-invulnerability Member-invulnerability is the property that no colluding subset of the group members, even colluding both the MM and the TM, can create a group signature which is traceable to any non-colluding member. Since to create a group signature of which the member's identity is identified is only allowed for this member, member-invulnerability implies that only each member can play the role of this member. To define member-invulnerability, We introduce an adversary

$\mathcal{A}$ who attempts to break the group signature scheme in terms of member-invulnerability. Adversary $\mathcal{A}$ is allowed to collude with the MM, the TM, and any subset of the group members. Formally, member-invulnerability is defined as follows.

Definition 59 *(***Member-invulnerability***) Let $\mathcal{GS}$ be a group signature scheme, and let $\mathcal{A}$ be an algorithm. We consider the following experiment that returns* $0/1$.

```
Experiment $\mathbf{Exp}^{\mathrm{UI}}_{\mathcal{GS},\mathcal{A}}(k)$
    $(mpk, tpk, State) \leftarrow \mathcal{A}(1^k)$
    $\langle State, (cert_U, sk_U, ider_U)\rangle \leftarrow \mathsf{Join}_{A,U}\langle State, mpk\rangle$
    If the tuple $(cert_U, sk_U, ider_U)$ is not valid
        then return 0
    $(m, gs, \mathcal{L}) \leftarrow \mathcal{A}^{\mathsf{Sign}(mpk,tpk,cert_U,sk_U,\cdot)}(State)$
    $\mathcal{L} \leftarrow \mathcal{L} \cup \{(U, ider_U)\}$
    If $rej \leftarrow \mathsf{Verify}(mpk, tpk, m, gs)$ then return 0
    If $(U, proof) \leftarrow \mathsf{Open}(mpk, tpk, tsk, m, gs, \mathcal{L})$
    and $m$ was not queried by $\mathcal{A}$ to the signing oracle
    Sign, then return 1
    Else return 0
```

A group signature scheme $\mathcal{GS}$ has member-invulnerability *property if for all probabilistic polynomial-time machines $\mathcal{A}$,*

$$\Pr[\mathbf{Exp}^{\mathrm{UI}}_{\mathcal{GS},\mathcal{A}}(k) = 1]$$

is negligible in k.

Separation of the roles of the managers. From MM-invulnerability and TM-invulnerability, we can see that the TM cannot have the ability to add a new member and that the MM cannot have the ability to identify the signer of a signature. Hence, these definitions imply the complete separation of roles of the TM and the MM. With member-invulnerability, other required notions of security for group signatures are satisfied also. Note that the schemes proposed in [7] do not satisfy the notion of TM-invulnerability.

Relation to the previous definitions. Our definition of TM-invulnerability roughly corresponds to the full-anonymity [15]. If the TM and the MM are unified, the sum of MM-invulnerability and member-invulnerability roughly corresponds to full-traceability [15]. As in the case of [7, 94], we are considering the case where the group manager(s) are dishonest and new members are allowed to join the group.

7.3 Building Blocks

The proposed scheme uses signatures of knowledge [35], a verifiable encryption [143], and the Nyberg-Rueppel signatures [123].

Signature of knowledge. A signature of knowledge is a transformation of corresponding special honest-verifier zero-knowledge interactive proof by using Fiat-Shamir heuristics [60]. A signature of knowledge is denoted by $SPK[\alpha_1, \ldots, \alpha_m : f(\alpha_1, \ldots, \alpha_m) = 0]$, where $\alpha_1, \ldots, \alpha_m$ is secret information for a prover, and $f(\alpha_1, \ldots, \alpha_m) = 0$ is an equation that $\alpha_1, \ldots, \alpha_m$ satisfy. A prover can show a verifier that he knows variables $\alpha_1, \ldots, \alpha_m$ such that $f(\alpha_1, \ldots, \alpha_m) = 0$ holds.

The proposed scheme uses two types of signatures of knowledge as building blocks. One is a *signature of knowledge of representations* denoted by

$$SPK\left[\alpha_1, \ldots, \alpha_u : \left(y_1 = \prod_{j \in \mathcal{J}_1} g_j^{\alpha_{e_{1j}}}\right) \wedge \cdots \wedge \left(y_n = \prod_{j \in \mathcal{J}_n} g_j^{\alpha_{e_{nj}}}\right)\right](m),$$

where the indices $e_{ij} \in \{1, \ldots, u\}$ correspond to the secrets $\alpha_1, \ldots, \alpha_u$ and the elements of $\mathcal{J}_i$ are indices corresponding to the base elements $g_1, \ldots, g_l$. The other is a *signature of knowledge for a range* denoted by

$$SPK[\alpha, \beta : E = g^{\alpha} h^{\beta} \wedge \alpha \in [0, b]].$$

The underlying protocol of this signature of knowledge relies on the *interactive proof that a committed value is in an interval* [127]. The scheme proposed in [127] can be constructed only based on the discrete logarithm (DL) problem. In this scheme, a verifier of the signature of knowledge convinces that $\alpha \in [-b, 2b]$, and only when $\alpha \in [0, b]$, the verifier gains no knowledge about α.

There are some more efficient scheme such as [27] in which an RSA modulus is used to make a signature size smaller. It is sufficient to use the scheme in [27] as a proof of the range if authorities are distributed only in the member addition. If distributing authorities require an efficient setup, the DL-based scheme such as [127] is suitable. In this chapter, we adopt the scheme for an efficient setup by the distributing authorities.

Verifiable Encryption. A verifiable encryption scheme is an encryption scheme in which a verifier can confirm that a ciphertext C is an encryption of a value such that $c = g^x$. In our scheme, we use a variant of verifiable encryption that uses perfectly hiding commitment $c' = g^x h^y$ instead of $c = g^x$.

Nyberg-Rueppel Signatures. Originally, the Nyberg-Rueppel signature was proposed as a variant of the ElGamal signature scheme with message recovery. In our scheme, we use a variant of the Nyberg-Rueppel signature called a *modified Nyberg-Rueppel signature* in [8] which removes the property of message recovery. The modified Nyberg-Rueppel signature is existentially unforgeable under the chosen message attack (EUF-CMA) in the random oracle and generic model [8].

7.4 A New Group Signature Scheme

In this section, we give a new group signature scheme that achieves complete separation of the membership manager and the tracing manager. Moreover, the authority of each manager is easily distributed by applying the distributed computation. Our scheme satisfies MM-invulnerability, TM-invulnerability and member-invulnerability defined in Section 7.2.2 in the random oracle and generic model.

7.4.1 Basic Idea

At first, the MM generates domain parameters $(p, q, P, g, h, f, G, H, \mathcal{H})$. Then, the MM and the TM generate their public-key/secret-key pairs by executing M-KeyGen and T-KeyGen respectively. Their secret keys are discrete logarithm of their public keys. A membership certificate $\langle r_U, \xi_U \rangle$ that the MM issues to a member U is the Nyberg-Rueppel signature on the message I_U given by U during the JOIN protocol. The group signing key σ_U for a member U is a discrete logarithm of I_U with respect to g. In SIGN procedure, a member U first generates an ElGamal ciphertext of a part of its membership certificate r_U with the tracing public key of the TM. Next, the member U generates a signature of knowledge [35], which proves that (1) the knowledge of a membership certificate $\langle r_U, \xi_U \rangle$ and a group signing key σ_U, (2) the knowledge of plaintext r'_U of the ElGamal ciphertext (g', e'), and (3) the fact that the r_U and r'_U are the same. The TM identifies the signer of a group signature by decrypting the membership certificate of the signer r_U from the ElGamal ciphertext (g', e') in the signature.

Roughly, to make the correct signature, a member U needs to prove the knowledge of σ_U, which is possible only by the valid member. This implies member-invulnerability. For an adversary to generate a signature that is not linked to any existing member, the adversary needs to knows a new membership certificate, which means forging of the Nyberg-Rueppel signature. This implies MM-invulnerability. In the SIGN procedure, a user creates an ElGamal ciphertext of r_U and the proof that he correctly computes the ciphertext, which is IND-CCA2 secure assuming random oracle and generic model [140]. Hence, breaking TM-invulnerability which is equivalent to guessing r_U will break IND-CCA2 security of the encryption.

7.4.2 Construction

We now present the complete construction of the proposed group signature scheme as in the following. Let $SPK[\alpha_1, \ldots, \alpha_m : f(\alpha_1, \ldots, \alpha_m) = 0]$ denote a signature of knowledge that proves the knowledge of $\alpha_1, \ldots, \alpha_m$ satisfying $f(\alpha_1, \ldots, \alpha_m) = 0$.

M-KeyGen

The membership manager MM first randomly chooses sufficiently large primes p, q, P such that $q|p-1$ and $p|P-1$. Let G_q be an order q subgroup of Z_p^*, and let G_p be an order p subgroup of Z_P^*. The MM chooses $g, h, f \in G_q$ and $G, H \in G_p$ such that neither non-trivial $(\alpha_1, \alpha_2, \alpha_3)$ satisfying $g^{\alpha_1} h^{\alpha_2} f^{\alpha_3} = 1 \pmod p$ nor non-trivial (β_1, β_2) satisfying $G^{\beta_1} H^{\beta_2} = 1 \pmod P$ are not known. The MM initializes a member list $\mathcal{L}$ to $\emptyset$. Let k be a security parameter. Let $\mathcal{H} : \{0,1\}^* \to \{0,1\}^k$ be a collision-resistant hash function. Next, the MM randomly selects $\upsilon \in_U Z_q$ and computes $y = h^\upsilon \bmod p$.

Finally, M-KeyGen outputs

$$\begin{aligned} &(msk, mpk) \\ &:= \quad (\upsilon, (p, q, P, g, h, f, G, H, \mathcal{H}, y)) \end{aligned}$$

T-KeyGen

The tracing manager TM randomly selects $\epsilon \in_U Z_q$ and computes $e = g^\epsilon \bmod p$.

T-KeyGen outputs

$$(tsk, tpk) \quad := \quad ((\epsilon), (e)) \,.$$

Join$_{MM,U}$

A user U runs the following protocol with the membership manager MM to join the group.

1. User U randomly chooses $\sigma_U \in_U Z_q$ and computes

$$I_U = g^{\sigma_U} \bmod p.$$

U generates the signature of knowledge

$$spk_U = SPK[\tilde{\sigma}_U : I_U = g^{\tilde{\sigma}_U} \bmod p]$$

which assures that U knows a discrete logarithm of I_U to the base g. U also computes the digital signature S_U on the message $I_U \| spk_U$. U sends

$$(I_U, spk_U, S_U)$$

to the membership manager MM.

2. The MM checks the validity of spk_U and S_U. If both are valid, the MM randomly selects $\rho \in_U Z_q$ and computes

$$\begin{aligned} r_U &:= I_U h^{\rho} \bmod p \\ \xi_U &:= \rho - r_U \upsilon \bmod q. \end{aligned}$$

The MM sends

$$(r_U, \xi_U)$$

as a membership certificate to U.

3. U checks whether (r_U, ξ_U) sent by the MM satisfies

$$\begin{aligned} r_U &= y^{r_U} g^{\sigma_U} h^{\xi_U} \\ r_U &\in [0, p-1]. \end{aligned}$$

If it holds, U stores $\langle r_U, \xi_U \rangle$ as his membership certificate, and securely stores σ_U as a group signing key.

4. MM adds $\langle I_U, spk_U, r_U, \xi_U, S_U \rangle$ to the member list $\mathcal{L}$.

5.
   - MM outputs the revised $\mathcal{L}$.
   - U outputs
     $(cert_U, sk_U, ider_U) = ((r_U, \xi_U), \sigma_U, I_U)$.

Sign

A member U creates a group signature on a message m as follows.

First, a member U chooses a random number $\tau \in_U Z_q$ and computes

$$(g', e') \quad := \quad (g^{\tau}, {r_U}^{-1} e^{\tau}) \bmod p,$$

which is an ElGamal encryption of r_U with respect to the tracing public key of the TM. Next, U computes the signature of knowledge:

$$\begin{aligned} SPK[\tilde{r}_U, \tilde{\xi}_U, \tilde{\sigma}_U, \tilde{\tau} \quad : \quad & g' = g^{\tilde{\tau}} \bmod p \quad \wedge \quad e' = \tilde{r}_U^{-1} e^{\tilde{\tau}} \bmod p \quad \wedge \\ & \tilde{r}_U = y^{\tilde{r}_U} g^{\tilde{\sigma}_U} h^{\tilde{\xi}_U} \bmod p \quad \wedge \quad \tilde{r}_U \in [0, p-1]](m). \end{aligned}$$

This signature of knowledge can be constructed from two components, a perfectly hiding commitments of r_U and a signature of knowledge: The commitments are computed as

$$h' := y^{r_U} f^{\omega} \bmod p \ , \quad J := G^{r_U} H^{a} \bmod P$$

where $\omega \in_U Z_q$ and $a \in_U Z_p$ are chosen uniformly and randomly, and the signature of knowledge is

$$\begin{aligned} SPK[\tilde{r}_U, \tilde{\xi}_U, \tilde{\sigma}_U, \tilde{\tau}, \tilde{\omega}, \tilde{a} \quad : \quad & g' = g^{\tilde{\tau}} \bmod p \quad \wedge \quad e' = \tilde{r}_U^{-1} e^{\tilde{\tau}} \bmod p \quad \wedge \\ & h' = y^{\tilde{r}_U} f^{\tilde{\omega}} \bmod p \quad \wedge \quad J = G^{\tilde{r}_U} H^{\tilde{a}} \bmod P \quad \wedge \\ & e'h' = f^{\tilde{\omega}} g^{-\tilde{\sigma}_U} h^{-\tilde{\xi}_U} e^{\tilde{\tau}} \bmod p \quad \wedge \quad \tilde{r}_U \in [0, p-1]](m). \end{aligned} \tag{7.1}$$

This signature of knowledge can be computed as in the following:

1. For $1 \leq j \leq k$, generate random numbers $\phi_{2j-1} \in_U [0, p-1]$, and sets $\phi_{2j} := \phi_{2j-1} - p$. If $r_U + \phi_{2j-1} \notin [0, p-1]$ and $r_U + \phi_{2j} \in [0, p-1]$, replace ϕ_{2j-1} with ϕ_{2j} to be $r_U + \phi_{2j-1} \in [0, p-1]$. Also, choose $\psi_{2j-1}, \psi_{2j} \in_U Z_q$, $\eta_{2j-1}, \eta_{2j} \in_U Z_p$ randomly. For $1 \leq j \leq k$, compute

$$V_j := y^{\phi_{2j-1}} f^{\psi_{2j-1}} \| y^{\phi_{2j}} f^{\psi_{2j}} \| G^{\phi_{2j-1}} H^{\eta_{2j-1}} \| G^{\phi_{2j}} H^{\eta_{2j}}.$$

2. Choose t_1, t_2, t_3, t_4, $t_5 \in Z_q$ randomly, and compute

$$\begin{aligned} T_1 \quad &:= \quad y^{t_1} f^{t_2} \bmod p \\ T_2 \quad &:= \quad f^{t_2} g^{-t_3} h^{-t_4} e^{t_5} \bmod p \end{aligned}$$

$$T_3 \quad := \quad g^{t_5} \bmod p.$$

3. For $1 \leq j \leq k$, select $\gamma_j \in_U Z_q$ and $J_j \in_U Z_p$. Set $e_j := e^{\gamma_j} \bmod p$. Then, compute

$$\begin{aligned} g_j &:= g^{\gamma_j} \bmod p \\ J_j &:= G^{e_j} H^{u_j} \bmod P \end{aligned}$$

for $1 \leq j \leq k$.

4. Generate

$$c := \mathcal{H}(g\|h\|f\|G\|H\|y\|e\|V_1\| \cdots \|V_k\|T_1\|T_2\|T_3\|g_1\| \cdots \|g_k\|J_1\| \cdots \|J_k\|m).$$

5. For $1 \leq j \leq k$,

   - If $c[j] = 0$, set

$$\begin{aligned} v_{6j-5} := \phi_{2j-1} \quad &, \quad v_{6j-4} := \phi_{2j}, \\ v_{6j-3} := \psi_{2j-1} \quad &, \quad v_{6j-2} := \psi_{2j}, \\ v_{6j-1} := \eta_{2j-1} \quad &, \quad v_{6j} := \eta_{2j}, \\ s_1 := t_1 \bmod q \quad &, \quad s_2 := t_2 \bmod q, \\ s_3 := t_3 \bmod q \quad &, \quad s_4 := t_4 \bmod q, \\ s_5 := t_5 \bmod q \quad &, \quad w_j := \gamma_j \bmod q, \\ z_j := u_j \bmod p \quad &. \end{aligned}$$

   - If $c[j] = 1$, set

$$\begin{aligned} v_{6j-5} := r_U + \phi_{2j-1} \quad &, \quad v_{6j-4} := y^{\phi_{2j}} f^{\psi_{2j}}, \\ v_{6j-3} := \omega + \psi_{2j-1} \quad &, \quad v_{6j-2} := \psi_0 \in_U Z_q, \\ v_{6j-1} := a + \eta_{2j-1} \quad &, \quad v_{6j} := G^{\phi_{2j}} H^{\eta_{2j}}, \\ s_1 := t_1 - r_U \bmod q \quad &, \quad s_2 := t_2 - \omega \bmod q, \\ s_3 := t_3 - \sigma_U \bmod q \quad &, \quad s_4 := t_4 - \xi_U \bmod q, \\ s_5 := t_5 - \tau \bmod q \quad &, \\ w_j := \gamma_j - \tau \bmod q \quad &, \quad z_j := u_j - a e_j {r_U}^{-1} \bmod p \end{aligned}$$

6. The resulting signature of knowledge is

$$\begin{aligned} SPK_{gs} &= (c, v_1, v_2, v_3, v_4, v_5, v_6, \ldots, v_{6k-5}, v_{6k-4}, v_{6k-3}, v_{6k-2}, v_{6k-1}, v_{6k}, \\ &\quad s_1, s_2, s_3, s_4, s_5, w_1, \ldots, w_k, z_1, \ldots, z_k). \end{aligned}$$

Finally, Sign outputs

$$gs := (g', e', h', J, SPK_{gs})$$

as a signature on message m.

Verify

A verifier V verifies a group signature gs on a message m by checking the validity of the signature of knowledge involved in gs.

For $gs = (g',\ e',\ h',\ J,\ c,\ SPK_{gs})$, V first checks if $v_{6j-5} \in [0, p-1]$ holds at $j \in [1, k]$ such that $c[j] = 1$. If it holds, V checks

$$c = \mathcal{H}(g \| h \| f \| G \| H \| y \| e \| V'_1 \| \cdots \| V'_k \| T'_1 \| T'_2 \| T'_3 \| g'_1 \| \cdots \| g'_k \| J'_1 \| \cdots \| J'_k \| m),$$

where

$$\begin{aligned} V'_j &= \begin{cases} y^{v_{6j-5}} f^{v_{6j-3}} \| y^{v_{6j-4}} f^{v_{6j-2}} \| G^{v_{6j-5}} H^{v_{6j-1}} \| G^{v_{6j-4}} H^{v_{6j}} & c[j] = 0 \\ y^{v_{6j-5}} f^{v_{6j-3}} / h' \| v_{6j-4} \| G^{v_{6j-5}} H^{v_{6j-1}} / J \| v_{6j} & c[j] = 1 \end{cases} \\ T'_1 &= h'^c y^{s_1} f^{s_2}, \quad T'_2 = (e'h')^c f^{s_2} g^{-s_3} h^{-s_4} e^{s_5}, \quad T'_3 = g'^c g^{s_5} \\ g'_j &= g'^{c[j]} g^{w_j} \bmod p \\ J'_j &= \begin{cases} G^{\bar{e}'_j} H^{z_j} \bmod P & c[j] = 0 \\ J^{\bar{e}'_j} H^{z_j} \bmod P & c[j] = 1 \quad (\text{where } \bar{e}'_j := e'^{c[j]} e^{w_j} \bmod p) \end{cases} \end{aligned}$$

V accepts gs if and only if the all above equations are satisfied.

Open

For an accused group signature gs on m, the tracing manager TM decrypts (g', e') with his tracing secret key by $\bar{r} := g'^{\epsilon} / e' \bmod p$. In addition, the TM computes a signature of knowledge $SPK[\tilde{\epsilon} : g'^{\tilde{\epsilon}} = \bar{r}^{-1} e']$ to prove that he surely used his secret key for decryption.

The TM finds the tuple $\langle I_U, spk_U, r_U, \xi_U, S_U \rangle$ from the member list $\mathcal{L}$ such that $r_U = \bar{r}$ holds. The TM concludes that the member U corresponding to $r_U = \bar{r}$ is the actual signer of gs.

7.4.3 Distributed Construction

In the group signature scheme shown in the previous subsection, both the membership secret key and the tracing secret key are discrete logarithms of corresponding public keys. Such secret keys can be easily distributed among multiple entities by using the technique proposed by Pedersen in [130].

More specifically, the membership secret key υ can be distributed into n elements $\upsilon_1, \dots, \upsilon_n$ satisfying $\upsilon = \upsilon_1 + \cdots + \upsilon_n$ for the distributed membership managers $MM_1, \cdots, MM_n$. In the JOIN protocol, each membership manager MM_i computes $r_U = I_U h^t$ and $\xi_i = k_i - r_U \upsilon_i$, where t and k_i is a distributedly generated random number satisfying $t = h^{\Sigma k_i}$. The user U obtains a membership certificate (r_U, ξ_U) by computing $\xi_U = \xi_1 + \cdots + \xi_n$.

Similarly, the tracing secret key ϵ can be distributed into $\epsilon_1, \dots, \epsilon_n$ satisfying $\epsilon = \epsilon_1 + \cdots + \epsilon_n$ for the distributed tracing managers $TM_1, \cdots, TM_n$. In the TRACE algorithm, each tracing manager TM_i computes $g'_i = g'^{\epsilon_i} \bmod p$, and an entire decryption can be executed by computing $\bar{r} = (\prod g'_i)/e' \bmod p$.

7.4.4 Efficiency Analysis

We analyze the efficiency of the proposed scheme. For simplicity, we estimate a basic construction shown in Section 7.4.2. The signature length is estimated by $|P| + (5k+3)|p| + (3k+5)|q|$ bit. If we take $|P| = |p| = 1024$ and $|q| = 160$, the signature length is about 110 KB, which is 4.5 times larger than that of [7]. The computational cost of our scheme is also 4.5 times larger than that of [7]. As mentioned in Section 7.3, we use an DL-based proof of a range due to the efficient setup by the distributing authorities. If we adopt the proof of a range in [27], the signature size becomes 26.2 KB, which is almost as large as that of [7].

7.5 Security Considerations

First of all, we prove that the group signature which a signer creates really serves as a signature of knowledge of Equation (7.1) in the random oracle model. We do prove it by showing that the underlying interactive protocol is honest-verifier zero-knowledge proof of knowledge.

Theorem 6 *The underlying interactive protocol of the proposed group signature scheme is an honest-verifier zero-knowledge proof of knowledge of a membership certificate, corresponding signing key and the* *random number used for encryption of ElGamal ciphertext, where the common input is a set of domain parameter PK, a membership public key, a tracing public key, and ElGamal ciphertext (g', e').*

Proof. Since the proof of this theorem is straight-forward, we omit the proof. □

Next, we prove the entire security of the proposed group signature scheme. We can obtain the following theorem.

Theorem 7 *The proposed group signature scheme $\mathcal{GS}$ is MM-invulnerable in the random oracle and generic model.*

Proof. We show that if there exists an attacker $\mathcal{A}$ that breaks MM- invulnerability of the group signature scheme, then there exists an attacker $\mathcal{A}'$ that breaks the EUF-CMA of the modified Nyberg-Rueppel signature in the random oracle and generic model. The following is the description of $\mathcal{A}'$:

1. Key generation of the target modified Nyberg-Rueppel signature scheme is the same as M-KeyGen procedure of the proposed group signature scheme.

2. When $\mathcal{A}$ asks a joining oracle to join the group by giving g^{σ_U}, $\mathcal{A}'$ asks its signing oracle to sign on g^{σ_U}. Then $\mathcal{A}'$ sends the answer of the oracle to $\mathcal{A}$, which is the membership certificate.

3. Suppose $\mathcal{A}$ generated a group signature, that is linked, by TRACE procedure, to membership certificate that $\mathcal{A}'$ has never sent to $\mathcal{A}$. Then, $\mathcal{A}$ plays a role of a knowledge extractor, namely, rewinds $\mathcal{A}$ and chooses another random oracle to extract the membership certificate $(\bar{r}_U, \bar{\xi}_U)$ and the signing key $\bar{\sigma}_U$. This is possible from Theorem 6. $(\bar{r}_U, \bar{\xi}_U)$ is a modified Nyberg-Rueppel signature on a message $\bar{\sigma}_U$.

It is shown in [8] that the modified Nyberg-Rueppel signature is EUF-CMA in the random oracle and generic model. Thus, the proposed scheme is TM-invulnerable in the random oracle and generic model.

□ □

Theorem 8 *The proposed group signature scheme $\mathcal{GS}$ is TM-invulnerable in the random oracle and generic model.*

Proof. By construction the group signature generated by SIGN includes an ElGamal encryption of r_U with a random number τ, and the signature of knowledge of τ. This encryption-and-signature pair is regarded as a signed ElGamal encryption [140]. Now we can show that if there exists an attacker $\mathcal{A}$ that breaks TM-invulnerability of the group signature scheme, then there exists an attacker $\mathcal{A}'$ that breaks IND-CCA2 of the above signed ElGamal encryption. The following is the description of $\mathcal{A}'$

1. Key generation of the target cryptosystem is the same as T-KeyGen procedure of the proposed group signature scheme.

2. When $\mathcal{A}$ asks a tracing oracle to open a signature gs, $\mathcal{A}'$ picks up the signed ElGamal encryption part and throws it to its decryption oracle. Then $\mathcal{A}'$ sends the result to $\mathcal{A}$.

3. When $\mathcal{A}$ chooses a pair of a membership certificate and a signing key, (r_0, ξ_0, σ_0) and (r_1, ξ_1, σ_1), $\mathcal{A}'$ chooses r_0 and r_1 as target plaintexts.

4. When $\mathcal{A}'$ was given a ciphertext as a challenge, $\mathcal{A}'$ generates a group signature that includes this challenge ciphertext by choosing appropriate random oracle. Then $\mathcal{A}'$ gives it to $\mathcal{A}$ as a challenge.

5. When $\mathcal{A}$ gives the answer $b \in \{0, 1\}$, $\mathcal{A}'$ answers b.

It is shown in [140] that the signed ElGamal encryption is IND-CCA2 secure in the random oracle and generic model. Therefore, $\mathcal{GS}$ is TM-invulnerable in the random oracle and generic model. □

□

Theorem 9 *The proposed group signature scheme $\mathcal{GS}$ is member-invulnerable if the discrete logarithm problem is hard in the random oracle model.*

Proof. It is easy to see from Theorem 6 that each of group signatures includes a signature of knowledge of σ_U.

Now we can show that if there exists an attacker $\mathcal{A}$ that breaks member-invulnerability of the group signature scheme, then there exists an attacker $\mathcal{A}'$ that solves discrete logarithm problem. The following is the description of $\mathcal{A}'$

1. When $\mathcal{A}'$ is given the instance (g, I) as a problem, $\mathcal{A}'$ chooses $h = g^a$ for randomly chosen a.

2. When $\mathcal{A}$ asks a joining oracle to join the group, $\mathcal{A}'$ first generates a simulated signature of knowledge of discrete logarithm of g, I by choosing a random oracle. Next, sends I and this signature of knowledge. Finally, $\mathcal{A}'$ obtains the membership certificate (r_U, ξ_U).

3. When $\mathcal{A}$ asks a signing oracle to sign a message m, $\mathcal{A}'$ generates an ElGamal encryption of r_U and generates a simulated signature of knowledge of a membership certificate, a group signing key, and a random number used for the encryption by choosing a random oracle. Then $\mathcal{A}'$ sends these generated data to $\mathcal{A}$.

4. Suppose $\mathcal{A}$ generated a group signature, that is linked, by TRACE procedure, to a membership certificate r_U. Then, $\mathcal{A}'$ rewinds $\mathcal{A}$ and chooses another random oracle to extract a membership

certificate and a signing key. This is possible from Theorem 6. We denote the extracted data $(r_U, \bar{\xi}_U, \bar{\sigma}_U)$.

5. If $\bar{\xi}_U = \xi_U$, then $g^{\bar{\sigma}_U} = I$, which means a success of the attack.

   If $\bar{\xi}_U \neq \xi_U$, then $g^{a(\xi_U - \bar{\xi}_U) + \bar{\sigma}_U} = I$, which also means a successful attack. □

□

7.6 Nyberg-Rueppel signatures

Originally, the Nyberg-Rueppel signature was proposed as a variant of the ElGamal signature scheme with message recovery. In this signature scheme, a message is converted by using a function called redundancy function. We can easily recover an original message of a signature by using this redundancy function. In our scheme, we use a variant of the Nyberg-Rueppel signature that abandons the property of message recovery so that we can weaken the requirement for the redundancy function. Since the Nyberg-Rueppel signature scheme is based on a discrete logarithm problem, distributing the secret key of the Nyberg-Rueppel signature is easy and its generation of a signature can be done efficiently by distributed signers. These facts make our scheme efficient even if authorities are distributed.

We now describe the construction of the Nyberg-Rueppel signature scheme. The Nyberg-Rueppel signature consists of three algorithms: (NR-KEYGEN, NR-SIGN, NR-VERIFY):

NR-KEYGEN Choose two large primes p, q such that $q|p-1$. Let G_q be the order q subgroup of Z_p^*. Choose generators $g, h \in_U G_q$. Choose a random $x \in_U Z_q$ and compute $y = h^x \bmod p$. The public key of a signer is (p, q, g, h, y) and the secret key is x.

NR-SIGN To sign a message m, first, generate $H(m) = g^m \bmod p$. Next, choose $t \in_U Z_q$ randomly, and compute $r := H(m)h^t \bmod p$ and $s := t - rx \bmod q$. (r, s) is a signature on m.

NR-VERIFY For a message-signature pair $(m, (r, s))$, check if $r \overset{?}{=} H(m)y^r h^s$.

We assume that the above Nyberg-Rueppel signature is existentially unforgeable under adaptively chosen message attack (EUF-CMA).

7.7 Conclusions

We have proposed a new practical group signature scheme with separate and distributed authorities. Our scheme can separate the membership manager and the tracing manager without invading each capability. Moreover, since our scheme is constructed from primitives based on the discrete logarithm problem, our scheme is well suited for distributed authorities. We have also formalized security definitions that describe the complete separation of the capabilities of the two managers and members. We have given the proofs that our scheme is secure under these security definitions.

Chapter 8

Group Signature from Bilinear Mappings

8.1 Introduction

A group signature scheme, first proposed by Chaum and van Heyst [44] and followed by [6, 7, 19, 30, 32, 33, 35, 121], allows each member of a group to sign messages on behalf of the group without revealing his own identity. The scheme also realizes a special authority that can identify actual signers in case of dispute. Group signatures have many applications in which user anonymity is required such as in anonymous credential systems [7], identity escrow [99, 95], voting and bidding [6], and electronic cash systems.

Although earlier group signature schemes required large computational cost and long signatures, recently proposed schemes, such as the one proposed by Ateniese et al. in [6], are very efficient. In particular, Boneh, Boyen, and Shacham [22], Nguyen and Safavi-Naini [121], and Camenisch and Lysyanskaya [32] proposed very efficient group signature schemes based on bilinear maps. Currently, the most efficient construction is the one proposed in [22]. The signature length of the scheme in [22] is 42% and 38% of those of [121] and [32] respectively. The computational cost for the scheme in [22] is also smaller than those of [121] and [32][1].

This chapter proposes a novel group signature scheme based on bilinear maps. Our scheme is more efficient than any of the previous schemes. Moreover, our scheme requires fewer assumptions than the scheme in [22], which is the most efficient among the previous schemes.

[1]The heaviest computation in these schemes is computation of a bilinear map such as Tate pairing. As shown in Table 10 in [90], its computational cost is smaller than that of computation of full-exponent RSA.

Our approach to the construction of a group signature scheme is similar to that adopted by Boneh et al. in [22]. They used a set of three groups $\mathcal{G}_1, \mathcal{G}_2$, and $\mathcal{G}_T$ of the same prime order p such that there exists a bilinear map from $\mathcal{G}_1 \times \mathcal{G}_2$ to $\mathcal{G}_T$. Each group member has a pair comprising a membership certificate and a membership secret with which he signs on behalf of the group. The membership certificate and membership secret are elements of $\mathcal{G}_1$ and $(\mathbb{Z}/p\mathbb{Z})^*$. For a special authority to identify actual signers from group signatures in their scheme, signers are required to attach an encryption of a part of the membership certificate which is an element of $\mathcal{G}_1$. Because of the existence of the bilinear map, their scheme is not able to simply use ElGamal encryption scheme for this purpose. Hence, they introduced a new encryption scheme called "linear encryption scheme" based on a new assumption called the Decision Linear Diffie-Hellman (DLDH) assumption. This encryption scheme is more complex than the ordinary ElGamal type encryption scheme.

The main difference between our approach and that in [22] is that we use a group $\mathcal{G}$ of the same order p in addition to the three groups $\mathcal{G}_1, \mathcal{G}_2$, and $\mathcal{G}_T$ such that the Decision Diffie-Hellman (DDH) problem on $\mathcal{G}$ is difficult to solve. For a special authority to identify actual signers from group signatures in our scheme, signers are required to attach an encryption of the exponentiation of the membership secret in $\mathcal{G}$. Because this exponentiation to be encrypted is in $\mathcal{G}$, we can apply a simple ElGamal type encryption scheme. This makes our scheme more efficient and requires fewer assumptions than the scheme in [22].

For the groups $\mathcal{G}_1, \mathcal{G}_2$, and $\mathcal{G}_T$ and their associated bilinear map, we can use, for example, the elliptic curve proposed by [113] (MNT curve) and Tate pairing. The choice of such a curve makes it possible to express elements in $\mathcal{G}_1$ by a short string. Although the number of such curves are found in [113] is small, more MNT curves are found in [138]. Therefore, since we can easily find an elliptic curve of the same given order p as $\mathcal{G}$ with practically high probability by exploiting a complex multiplication method [29, 4], finding a desired set of $(\mathcal{G}_1, \mathcal{G}_2, \mathcal{G})$ is practical. Indeed, we have found several of such curves [101].

As a result, our signature lengths are, respectively, 83%, 36%, and 32% of those of signatures in [22], [121], and [32] if we choose groups so that elements of $\mathcal{G}_1$, $\mathcal{G}_T$, and $\mathcal{G}$ can be expressed in 171, 1020, and 171 bit strings respectively. Although we cannot present a precise estimation of the computational cost since it depends on the choice of groups, our scheme requires less computational cost than any of the schemes in [22, 121, 32]. The security of our scheme depends on the DDH assumption, the Strong Diffie-Hellman (SDH) assumption, and the existence of random oracles. We do not present how to revoke group members. However, the revocation mechanisms described in [22] can be also applied to our system. In our scheme, group members are able to determine their secret key when they join the group, which enables them to join many groups using the same secret key. This property may reduce operational cost when there are many groups. The scheme in [121] does not have such a property. (The scheme in [32] does.)

This chapter is organized as follows. Section 8.2 describes the model and security requirements of the group signature scheme and notation and complexity assumptions. Section 8.3 proposes our group signature scheme, and Section 8.4 discusses its security. Section 8.5 compares our scheme with the previous schemes.

8.2 Background

The model and security requirements are the same as presented in Chapter 7

8.2.1 Notation and Complexity Assumption

Let $\mathcal{G}_{1k}, \mathcal{G}_{2k}, \mathcal{G}_{Tk}$, and $\mathcal{G}_k$ be a cyclic group of length k prime order p. We omit index k if not confusing. Let G_1, G_2, and G be, respectively, generators of $\mathcal{G}_1$, $\mathcal{G}_2$, and $\mathcal{G}$. Let ψ be an isomorphism from $\mathcal{G}_2$ to $\mathcal{G}_1$, with $\psi(G_2) = G_1$. Let e be a non-degenerate bilinear map $e : \mathcal{G}_1 \times \mathcal{G}_2 \to \mathcal{G}_T$. Let $\mathcal{H}$ be a hash function that maps string to $(\mathbb{Z}/p\mathbb{Z})^*$.

Definition 60 (Decision Diffie-Hellman assumption) *Let the Decision Diffie-Hellman problem in $\mathcal{G}_k$ be defined as follows: given 4-tuple $(G, [a]G, [b]G, [c]G) \in (\mathcal{G}_k)^4$ as input, output 1 if $c = ab$ and 0 otherwise. An algorithm $\mathcal{A}$ has advantage $\epsilon(k)$ in solving the Decision Diffie-Hellman problem in $\mathcal{G}_k$ if*

$$\begin{aligned}|\Pr[A(G, [a]G, [b]G, [ab]G) = 1] \\ - \Pr[A(G, [a]G, [b]G, [c]G) = 1]| \geq \epsilon(k)\end{aligned}$$

where the probability is taken over the random choice of generator G in $\mathcal{G}_k$, of $(a, b, c) \in ((\mathbb{Z}/p\mathbb{Z})^*)^3$, and of the random tape of $\mathcal{A}$.

We say that the Decision Diffie-Hellman assumption holds in $\{\mathcal{G}_k\}_{k\in\mathbb{N}}$ if no polynomial-time algorithm has advantage $\epsilon(k)$ non-negligible in k in solving the Decision Diffie-Hellman problem in $\mathcal{G}_k$.

Definition 61 (Strong Diffie-Hellman Assumption) *Let the q-Strong Diffie-Hellman Problem (q-SDH) in $(\mathcal{G}_{1k}, \mathcal{G}_{2k})$ be defined as follows:*
Given a $(q+2)$-tuple $(G_1, G_2, [\gamma]G_2, [\gamma^2]G_2, \ldots, [\gamma^q]G_2) \in \mathcal{G}_{1k} \times (\mathcal{G}_{2k})^{q+1}$ as input, output a pair $([1/(x+\gamma)]G_1, x)$ where $x \in (\mathbb{Z}/p\mathbb{Z})^*$. An algorithm $\mathcal{A}$ has advantage $\epsilon(k)$ in solving the q-SDH problem in $(\mathcal{G}_{1k}, \mathcal{G}_{2k})$ if

$$\begin{aligned}&\Pr[A(G_1, G_2, [\gamma]G_2, \ldots, [\gamma^q]G_2) \\ &= ([1/(x+\gamma)]G_1, x)] \geq \epsilon(k),\end{aligned}$$

where the probability is taken over the random choice of generator G_2 in $\mathcal{G}_{2k}$ (with $G_1 = \psi(G_2)$), of $\gamma \in (\mathbb{Z}/p\mathbb{Z})^*$, and of the random tape of $\mathcal{A}$.

We say that the Strong Diffie-Hellman (SDH) assumption holds in $\{(\mathcal{G}_{1k}, \mathcal{G}_{2k})\}_{k\in\mathbb{N}}$ if no polynomial-time algorithm has advantage $\epsilon(k)$ non-negligible in k in solving the q-SDH problem in $(\mathcal{G}_{1k}, \mathcal{G}_{2k})$ for q polynomial of k.

The SDH assumption is proposed and proved to hold in generic bilinear groups in [19]. This assumption is a variant of an assumption proposed by Mitsunari et al. in [112].

8.3 Proposed Group Signature Scheme

Now we will present our efficient group signature scheme.

M-KeyGen

Given 1^k, M-KeyGen chooses $\mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_T$ such that its order p is of length k and then randomly chooses $w \in_R \mathbb{Z}/p\mathbb{Z}$ and $(H, K) \in_R (\mathcal{G}_1)^2$ and generates $Y = [w]G_2$. Then, M-KeyGen outputs

$$\begin{aligned}&(msk, mpk)\\ &:= \quad (w, (p, \mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_T, e, \mathcal{G}, G_1, G_2, G, \psi, \mathcal{H}, Y, H, K))\end{aligned}$$

Here, some of the symbols are interpreted as binary strings that describe those symbols. For example, $\mathcal{G}$ expresses the string of the document that specifies group $\mathcal{G}$.

T-KeyGen

Given mpk, T-KeyGen first randomly chooses $(s, t) \in_R (\mathbb{Z}/p\mathbb{Z})^2$. Next, T-KeyGen generates $(S, T) = ([s]G, [t]G)$. Finally, T-KeyGen outputs

$$(tsk, tpk) \quad := \quad ((s, t), (S, T)).$$

Join$_{MM,U}$

1. • MM is given group member list $\mathcal{L}$, an identity of a user U, mpk, and msk.

   • A user U is given mpk.

2. U randomly chooses $sk_U := x_U \in_R \mathbb{Z}/p\mathbb{Z}$ and $z'_U \in_R \mathbb{Z}/p\mathbb{Z}$ and generates

$$ider_U := Q_U = [x_U]G \quad , \quad H_U = [x_U]H + [z'_U]K$$

and sends $(Q_U, H_U)$to MM[2].

Then, U proves in zero-knowledge to MM the knowledge of x_U and z'_U as follows. Although the protocol given here is only honest verifier zero-knowledge, from this we can construct a black-box zero-knowledge protocol using the technique presented in [110]. We still assume that Join protocols are executed in a sequential manner (or concurrently but with an appropriate timing-constraint [54]).

(i) U randomly chooses $(x'_U, z') \in_R ((\mathbb{Z}/p\mathbb{Z})^*)^2$ and generates

$$Q'_U = [x'_U]G \quad , \quad H'_U = [x'_U]H + [z']K$$

and sends them to MM.

(ii) MM sends U randomly chosen $c_U \in_R (\mathbb{Z}/p\mathbb{Z})^*$.

(iii) U generates

$$r_U = c_U x_U + x'_U \quad , \quad s_U = c_U z'_U + z'$$

and sends (r_U, s_U) to MM.

(iv) MM checks that the following equations hold:

$$\begin{aligned} [r_U]G &= [c_U]Q_U + Q'_U \\ [r_U]H + [s_U]K &= [c_U]H_U + H'_U \end{aligned}$$

3. The MM randomly chooses $(y_U, z''_U) \in_R (\mathbb{Z}/p\mathbb{Z})^2$ and generates

$$A_U = [1/(w + y_U)](G_1 - H_U - [z''_U]K)$$

and sends (A_U, y_U, z''_U) to U. The MM adds an entry $(U, ider_U) = (U, Q_U)$ to its group member list $\mathcal{L}$.

[2]U needs to sign on Q_U to prove that U agreed to be a group member; we omit this process for the sake of simplicity.

4. U generates its membership certificate as

$$cert_U := (A_U, y_U, z_U) = (A_U, y_U, z'_U + z''_U).$$

U checks that the following equation holds:

$$\begin{aligned} & e(A_U, Y + [y_U]G_2) \cdot e([x_U]H, G_2) \cdot e([z_U]K, G_2) \\ = \;& e(G_1, G_2). \end{aligned}$$

5. • MM outputs the revised $\mathcal{L}$.
   • U outputs
   $(cert_U, sk_U, ider_U) = ((A_U, y_U, z_U), x_U, Q_U)$.

Remark 2 *Publishing* $(cert_U, ider_U)$ *which* MM *is able to obtain does not compromise the security of the system.*

Sign

1. Sign is given $mpk, tpk, cert_U, sk_U$, and m.
2. Sign randomly chooses $(r, q) \in_R ((\mathbb{Z}/p\mathbb{Z})^*)^2$ and generates

$$B = A_U + [q]K, R = [x_U + r]G, V = [r]S, W = [r]T \tag{8.1}$$

Here, the following equation holds.

$$\begin{aligned} e(G_1, G_2) = & \; e(B, Y) \cdot e(H, G_2)^{x_U} \\ & \cdot e(B, G_2)^{y_U} \cdot e(K, G_2)^{z_U - q\, y_U} \cdot e(K, Y)^{-q} \end{aligned} \tag{8.2}$$

The data generated hereafter is a Fiat-Shamir transformation of a zero-knowledge proof of knowledge of x_U, y_U, z_U, and q, r that satisfies Eqs. (8.1) and (8.2). Since B is a perfect hiding commitment of A_U, the only knowledge that the receiver of the signature can obtain is (R, V, W) which is an ElGamal type double encryption of $[x_U]G$

(i) Sign randomly chooses $(t, u, v, f, o) \in_R ((\mathbb{Z}/p\mathbb{Z})^*)^5$ and generates

$$\begin{aligned} X' &= e(H, G_2)^t e(B, G_2)^u e(K, G_2)^v e(K, Y)^f \\ R' &= [t + o]G \;,\;\; V' = [o]S \;,\;\; W' = [o]T \end{aligned}$$

(ii) Sign generates

$$
\begin{aligned}
c &= \mathcal{H}(p, G_1, G_2, G_T, G, \psi, Y, S, T, H, K, \\
&\quad B, R, V, W, X', V', W', R', m)
\end{aligned}
$$

(iii) Sign generates

$$
\begin{aligned}
x' &= cx_U + t, y' = cy_U + u, \\
z' &= c(z_U - qy_U) + v, q' = -cq + f, \\
r' &= cr + o
\end{aligned}
$$

3. Sign outputs

$$gs := (B, R, V, W, c, x', y', z', q', r')$$

as a signature on message m.

Verify

1. Verify is given mpk, tpk, m, and gs.

2. Verify generates

$$
\begin{aligned}
X' &= e(H, G_2)^{x'} e(B, [y']G_2 + [c]Y) \\
&\quad \cdot e(K, G_2)^{z'} e(K, Y)^{q'} e(G_1, G_2)^{-c} \\
R' &= [x' + r']G - [c]R, V' = [r']S - [c]V, \\
W' &= [r']T - [c]W.
\end{aligned}
$$

3. Verify outputs acc if equation

$$
\begin{aligned}
c &= \mathcal{H}(p, G_1, G_2, G_T, G, \psi, Y, S, T, H, K, \\
&\quad B, R, V, W, X', V', W', R', m)
\end{aligned}
$$

holds. Otherwise, it outputs rej.

Open

1. Open is given mpk, tpk, tsk, m, gs, and $\mathcal{L}$.
2. If Verify$(mpk, tpk, m, gs) = rej$, it outputs $\perp$ and stops.
3. Open generates and outputs

$$Q \quad = \quad R - [1/s]V \qquad (= R - [1/t]W)$$

   Then, Open generates and outputs a non-interactive proof of knowledge of either s or t that satisfies either of the above equations and Q as a *proof*.

4. Open searches Q_U that coincides with the Q in $\mathcal{L}$. If there is such a Q_U, it outputs the corresponding U. Otherwise, it outputs $\perp'$.

8.4 Security

Theorem 10 *The proposed scheme has MM-invulnerability if the SDH assumption holds.*

Proof.

Lemma 50 *Given two sets of a signature and a random oracle for a message m*

$$\{(B, R, V, W, X', V', W', R', x'_1, y'_1, z'_1, q'_1, r'_1), \mathcal{H}_1\}$$
$$\{(B, R, V, W, X', V', W', R', x'_2, y'_2, z'_2, q'_2, r'_2), \mathcal{H}_2\}$$

such that

$$\begin{aligned}
& (x'_1, y'_1, z'_1, q'_1, r'_1) \neq (x'_2, y'_2, z'_2, q'_2, r'_2) \\
& c_1 := \mathcal{H}_1(p, G_1, G_2, G_T, G, \psi, Y, S, T, H, K, \\
& B, R, V, W, X', V', W', R', m) \\
& \neq c_2 := \mathcal{H}_2(p, G_1, G_2, G_T, G, \psi, Y, S, T, H, K, \\
& B, R, V, W, X', V', W', R', m), \\
X' \quad = \quad & e(H, G_2)^{x'_1} e(B, G_2)^{y'_1} e(K, G_2)^{z'_1} \\
& e(K, Y)^{q'_1} \left(\frac{e(G_1, G_2)}{e(B_1, Y)}\right)^{-c_1} \\
X' \quad = \quad & e(H, G_2)^{x'_2} e(B, G_2)^{y'_2} e(K, G_2)^{z'_2}
\end{aligned}$$

$$\begin{array}{rcl} & & e(K,Y)^{q_2'}\left(\dfrac{e(G_1,G_2)}{e(B_1,Y)}\right)^{-c_2} \\ R' & = & [x_1'+r_1']G-[c_1]R \\ R' & = & [x_2'+r_2']G-[c_2]R \\ V' & = & [r_1']S-[c_1]V \\ V' & = & [r_2']S-[c_2]V \\ W' & = & [r_1']T-[c_1]W \\ W' & = & [r_2']T-[c_2]W \end{array}$$

then one can compute $(A_U, x_U, y_U, z_U, q, r) \in \mathcal{G}_1 \times ((\mathbb{Z}/p\mathbb{Z})^*)^5$ *that satisfies Eqs. (8.1) and (8.2).*

Proof. The following $(A_U, x_U, y_U, z_U, q, r)$ proves the lemma.

$$\begin{array}{rcl} A_U & = & B-\left[\dfrac{q_1'-q_2'}{c_1-c_2}\right]K \\ x_U & = & \dfrac{x_1'-x_2'}{c_1-c_2} \\ y_U & = & \dfrac{y_1'-y_2'}{c_1-c_2} \\ z_U & = & \dfrac{(z_1'-z_2')(c_1-c_2)-(y_1'-y_2')(q_1'-q_2')}{(c_1-c_2)^2} \\ q & = & \dfrac{-q_1'+q_2'}{c_1-c_2} \\ r & = & \dfrac{r_1'-r_2'}{c_1-c_2} \end{array}$$

□

Lemma 51 *If there exists an attacker* $\mathcal{A}$ *that obeys in* Join *protocol less than* $q-1$ *times and breaks MM-invulnerability, then there exists an attacker* $\mathcal{A}'$ *that breaks the q-SDH problem or solves the discrete logarithm problem using* $\mathcal{A}$ *as a black-box.*

Proof. Suppose that $\mathcal{A}'$ is given a tuple $(q+2)$-tuple $(Q_1, Q, [\gamma]Q, [\gamma^2]Q, \ldots, [\gamma^q]Q)$ where $Q \in \mathcal{G}_2$ and $Q_1 = \psi(Q)$.

1. $\mathcal{A}'$ randomly chooses

$$\alpha \in_R (\mathbb{Z}/p\mathbb{Z})^*, \{(a_i, b_i) \in_R ((\mathbb{Z}/p\mathbb{Z})^*)^2\}_{i\in[q-1]}, m \in_R [q-1].$$

2. Suppose $w = \gamma - a_m$. $\mathcal{A}'$ randomly chooses $\theta \in_R (\mathbb{Z}/p\mathbb{Z})^*$ and generates, from the given tuple,

$$\begin{aligned}
G_2 &= [b_m \prod_{i=1, i\neq m}^{q-1} (\gamma + a_i - a_m)]Q \\
&\quad +[\alpha \prod_{i=1}^{q-1} (\gamma + a_i - a_m)]Q \\
G_1 &= \psi(G_2) \\
H &= [\prod_{i=1, i\neq m}^{q-1} (\gamma + a_i - a_m)]\psi(Q) \\
K &= [\theta]H \\
Y &= [w]G_2 = [(\gamma - a_m)b_m \prod_{i=1, i\neq m}^{q-1} (\gamma + a_i - a_m)]Q \\
&\quad +[(\gamma - a_m)\alpha \prod_{i=1}^{q-1} (\gamma + a_i - a_m)]Q
\end{aligned}$$

3. $\mathcal{A}'$ outputs

$$mpk = (p, \mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_T, e, \mathcal{G}, G_1, G_2, G, \psi, \mathcal{H}, Y, H, K).$$

and invokes $\mathcal{A}$ by giving it.

4. $\mathcal{A}'$ receives $tpk = (S, T)$ from $\mathcal{A}$.

5. Whenever $\mathcal{A}$ runs and successfully completes the first two steps of i-th Join protocol as a user U with $\mathcal{A}'$, do the following:

    (i) $\mathcal{A}'$ obtains x_U and z'_U by rewinding $\mathcal{A}$ and choosing other random oracles.

    (ii) $\mathcal{A}'$ generates A_U, y_U, z''_U as

$$\begin{aligned}
z''_U &= \frac{b_i - x_U}{\theta} - z'_U \\
y_U &= a_i \\
A_U &= [1/(y_U + w)](G_1 - [x_U]H - [z'_U + z''_U]K) \\
&= [1/(a_i + w)](G_1 - [b_i]H) \\
&= [\alpha \prod_{j=1, j\neq i}^{q-1} (\gamma + a_j - a_m)]\psi(Q) \\
&\quad +[(b_m - b_i) \prod_{j=1, j\neq m,i}^{q-1} (\gamma + a_j - a_m)]\psi(Q)
\end{aligned}$$

    using the $(q+2)$-tuple. Note that $b_i - b_m = 0$ when $i = m$. Then $\mathcal{A}'$ sends (A_U, y_U, z''_U) to $\mathcal{A}$.

(iii) $\mathcal{A}$ adds $(R, Q_U = [x_U]G)$ to $\mathcal{L}$.

6. $\mathcal{A}$ outputs a pair (m, gs) and wins the game with non-negligible ϵ_1. That is, $acc \leftarrow \mathsf{Verify}(mpk, tpk, m, gs)$ and $(\perp', proof) \leftarrow \mathsf{Open}(mpk, tpk, tsk, m, gs, \mathcal{L})$.

   Then, from the Forking Lemma [131] and Lemma 50, $\mathcal{A}'$ is able to obtain $(A_U, x_U, y_U, z_U, q, r) \in \mathcal{G}_1 \times ((\mathbb{Z}/p\mathbb{Z})^*)^5$ that satisfies Eqs. (8.1) and (8.2) and $[x_U]G \notin \mathcal{L}$.

   Here,

$$\begin{aligned} A_U &= [1/(y_U + w)](G_1 - [x_U]H - [z'_U + z''_U]K) \\ &= [1/(y_U + w)](G_1 - [x_U + \theta\, z_U]H) \\ &= \left[\frac{\alpha\gamma - (x_U + \theta\, z_U) + b_m}{\gamma + y_U - a_m} \prod_{j=1, j\neq m}^{q-1} (\gamma + a_j - a_m)\right]\psi(Q) \end{aligned}$$

7. We will see later that

$$(\gamma + y_U - a_m)|(\alpha\gamma - (x_U + \theta\, z_U) + b_m) \tag{8.3}$$

   happens only with negligible probability as long as the discrete logarithm problem is difficult to solve.

   Since the probability that $y_U \notin \{a_i\}_{i=1,\ldots,q-1}$ holds is smaller than ϵ_1, the probability that $y_U \notin \{a_i\}_{i=1,\ldots,q-1} \setminus a_m$ is greater than $\epsilon_1/(q-1)$, which is non-negligible. Hence, at least with this non-negligible probability, we have $\gamma + y_U - a_m$ as the denominator of A_U. Thus, in this case, with the formal division of

$$(\alpha\gamma - (x_U + \theta\, z_U) + b_m) \prod_{j=1, j\neq m}^{q-1} (\gamma + a_j - a_m)$$

   by $(\gamma + y_U - a_m)$, we are able to compute $\{c_i \in (\mathbb{Z}/p\mathbb{Z})^*\}_{i=0,\ldots,q}$ such that

$$A_U = \sum_{i=0}^{q-1} [c_i \gamma^i]\psi(Q) + [c_q/(\gamma + y_U - a_m)]\psi(Q).$$

   From $A_U, \{c_i \in (\mathbb{Z}/p\mathbb{Z})^*\}_{i=0,\ldots,q}, \theta, x_u, y_U, z_U,$ $\{a_i\}_{i=1,\ldots,q-1}$, and b_m, the solution for the given q-SDH follows. That is,

$$\left([1/c_q](A_U - \sum_{i=0}^{q-1}[c_i\gamma^i]\psi(Q)), y_U - a_m\right).$$

Now we consider the case where Eq. (8.3) and $a_m \neq y_U$ hold with non-negligible probability. From Eq. (8.3),

$$\alpha(y_U - a_m) + x_U + \theta\, z_U - b_m = 0$$

holds.

We consider another attacker $\mathcal{A}''$ that is given a pair $(D, L) \in (\mathcal{G}_2)^2$. $\mathcal{A}''$ randomly chooses $(b_m, \gamma, a_m, \theta) \in ((\mathbb{Z}/p\mathbb{Z})^*)^4$ and generates

$$\begin{aligned} G_2 &= [b_m]D + [\gamma]L \\ G_1 &= \psi(G_2) \\ H &= \psi(L) \\ K &= [\theta]H \\ Y &= [w]G_2 = [\gamma - a_m]G_2. \end{aligned}$$

Since $\mathcal{A}''$ knows $w = \gamma - a_m$, it is able to perfectly play the role of membership manager. Considering relations

$$\begin{aligned} D &\Leftrightarrow [\prod_{i=1, i\neq m}^{q-1} (\gamma + a_i - a_m)]Q \\ L &\Leftrightarrow [\alpha \prod_{i=1, i\neq m}^{q-1} (\gamma + a_i - a_m)]Q, \end{aligned}$$

we can see that $\mathcal{A}''$ is able to generate $\alpha = \log_D L$ from Eq. (8.3).

Now we consider the case where Eq. (8.3) and $a_m = y_U$ hold with non-negligible probability. From Eq. (8.3),

$$\theta\, z_U - b_m = 0$$

holds.

We consider another attacker $\mathcal{A}'''$ that is given a pair $(D, K) \in (\mathcal{G}_2)^2$. $\mathcal{A}''$ is the same to $\mathcal{A}''$ except in the following:

1. $\mathcal{A}'''$ generates $\alpha \in (\mathbb{Z}/p\mathbb{Z})^*$ but does not generate $\theta \in (\mathbb{Z}/p\mathbb{Z})^*$.

2. $\mathcal{A}'''$ generates $L = [\alpha]D$ and uses the given K in the K in *mpk*.

we can see that $\mathcal{A}''$ is able to generate $\theta = \log_D K = b_m/z_U$.

□

The theorem follows from Lemma 51. □

Theorem 11 *The proposed scheme has TM-invulnerability property if the DDH assumption holds.*

Proof.

Lemma 52 *The underlying proof of knowledge of x_U, y_U, z_U, q, r in* Sign *protocol is zero-knowledge.*

Proof. The following simulator $\mathcal{S}$ proves the lemma:

1. $\mathcal{S}$ randomly chooses x', y', z', q', r', c from $(\mathbb{Z}/p\mathbb{Z})^*$.

2. $\mathcal{S}$ generates

$$\begin{aligned} X' &= e(H, G_2)^{x'} e(B, [y']G_2 + [c]Y) e(K, G_2)^{z'} \\ &\quad e(K, Y)^{q'} e(G_1, G_2)^{-c} \\ R' &= [x' + r']G - [c]R \\ V' &= [r']S - [c]V \\ W' &= [r']T - [c]W. \end{aligned}$$

$\mathcal{S}$ outputs $X', R', V', W', c, x', y', z', q', r'$ as the view of the protocol. □

Lemma 53 *If there exists an attacker $\mathcal{A}$ that breaks TM-invulnerability, then there exists an attacker $\mathcal{A}'$ that breaks the Decision Diffie-Hellman problem by using $\mathcal{A}$ as a black-box.*

Proof. The data (R, V, W) in a group signature is a double encryption of $Q_U = [x_U]G$, which is semantically secure if we assume the Decision Diffie-Hellman assumption. And, from Lemma 50 and 50, the rest of the data in the group signature is a simulation sound [134] and non-interactive zero-knowledge proof of knowledge of $r = \log_S V = \log_T W$ in the random oracle model if *msk* is public. Therefore, from [119, 134], the group signature can be considered as an IND-CCA2 encryption of Q_U if *msk* is public.

Now it is enough to show that one can break the above IND-CCA2 secure cryptosystem (target cryptosystem) by using an attacker $\mathcal{A}$ that breaks TM-invulnerability of the group signature scheme. This can be shown if we consider the following correspondence.

1. Key generation of the target cryptosystem corresponds to T-KEYGEN.

2. The decryption oracle of the target cryptosystem corresponds to Open.

3. The choice of challenge plaintexts by $\mathcal{A}'$ in IND-CCA2 game of the target cryptosystem corresponds to the choice of a pair comprising a membership certificate and a signing key by $\mathcal{A}$.

4. The response of $\mathcal{A}'$ at the end of the IND-CCA2 game of the target cryptosystem corresponds to the response of $\mathcal{A}$ at the end of the experiment $\mathbf{Exp}^{\mathrm{An}}_{\mathcal{GS},\mathcal{A}}(k)$.

□

The theorem follows from Lemma 53. □

Theorem 12 *The proposed scheme has Member-Invulnerability property if we assume the discrete logarithm problem is difficult to solve.*

Proof. We can show that if there exists an attacker $\mathcal{A}$ that breaks Member-Invulnerability of the group signature scheme, then there exists an attacker $\mathcal{A}'$ that solves the discrete logarithm problem by using $\mathcal{A}$ as a black-box.

The following is the description of $\mathcal{A}'$

1. Suppose $\mathcal{A}'$ is given the instance (Q, G) and asked to compute $\log_G Q$, Then, $\mathcal{A}'$ engages in the Join protocol with $\mathcal{A}$ as U. In this protocol, $\mathcal{A}'$ randomly choose $z'_U \in_R (\mathbb{Z}/p\mathbb{Z})^*$, generates $Q_U = Q$ and $H_U = Q + [z'_U]K$, and sends Q_U and H_U to $\mathcal{A}$. Although $\mathcal{A}'$ does not know $\log_G Q_U$, it is able to complete the rest of the protocol by rewinding $\mathcal{A}$. This is possible since the rest of the protocol is zero-knowledge.

2. Suppose $\mathcal{A}$ asks $\mathcal{A}'$ to generate a group signature gs on message m such that Open outputs Q if this (gs, m) is given. Then, $\mathcal{A}'$ randomly chooses $r \in_R (\mathbb{Z}/p\mathbb{Z})^*$ and generates $(R, V, W) = (Q + [r]G, [r]S, [r]T)$. Although $\mathcal{A}'$ does not know $\log_G Q_U$, $\mathcal{A}'$ is able to generate the rest of the data for the signature by choosing random oracles. This is possible since the rest of the protocol is zero-knowledge (Lemma 52).

3. Since $\mathcal{A}$ breaks Member-Invulnerability of the proposed group signature scheme, $\mathcal{A}$ outputs a group signature gs' on message m' such that Open outputs Q if this (gs', m') is given with non-negligible probability.

   Then, from the Forking Lemma [131] and Lemma 50, $\mathcal{A}'$ is able to extract, by rewinding $\mathcal{A}$ and choosing other random oracles, x_U such that $Q = [x_U]G$. This solves the discrete logarithm problem.

□

	A variant of [22] Sign/Verify	Scheme in [121] Sign/Verify
# of SMul in $\mathcal{G}$	-	-
# of SMul in $\mathcal{G}_1$	11/12	20/13
# of SMul in $\mathcal{G}_2$	0/2	-
# of MExp in $\mathcal{G}_T$	3/3	6/2
# of pairings	0/1	0/3
Sig. Len. (bits)	2057	4782
Assumptions	SDH,DLDH	SDH,DBDH
	Scheme in [32] Sign/Verify	Our Scheme Sign/Verify
# of SMul in $\mathcal{G}$	-	6/6
# of SMul in $\mathcal{G}_1$	3/0	1/0
# of SMul in $\mathcal{G}_2$	-	0/2
# of MExp in $\mathcal{G}_T$	13/13	4/4
# of pairings	0/5	0/1
Sig. Len. (bits)	5296	1711
Assumptions	LRSW,DDH	SDH,DDH

Table 8.1: Complexity & Assumptions

8.5 Comparison with Previous Schemes

We compare the signature length and computational complexity of the proposed scheme to those of the previous schemes [121, 32] and those of a variant of the scheme in [22]. This variant protocol is given in Section 8.7.

The variant scheme of [22] differs from the original one in two points. The first point is that it provides a joining protocol, whose construction is already presented in Section 7 of [19]. The second point is that it uses a double encryption scheme [119] variant of the linear encryption scheme instead of the simple linear encryption scheme used in the original scheme. Since the *Open* oracle in group signature plays a role similar to that of the role of the decryption oracle in the IND-CCA2 game of public key cryptosystems, the encryption scheme used in group signature needs to be IND-CCA2 secure. However, the signed ElGamal encryption is IND-CCA2 secure only in the generic model [140], in the same way that the linear encryption scheme adopted in [22] is. Hence, the use of a double encryption variant is a legitimate solution to avoid dependence on the generic group model. [3]

Although the above variant scheme is less efficient than the original scheme, comparing our scheme with this variant scheme is appropriate. This is because our scheme and the schemes in [121] and [32]

[3]In [22], a scheme that is more efficient is proposed under the assumption that the discrete logarithm problem in $\mathcal{G}_1$ is difficult to solve. However, we believe this is still non-standard assumption.

all provide a Join protocol and their security is proved in a non-generic group model.

We compare the group signature lengths of our scheme and those of the previous schemes. We assume that $\mathcal{G}_1 \neq \mathcal{G}_2$ such that the representation of G_1 can be a 172 bit string when $|p| = 171$ by using the elliptic curve defined by [113]. The choice of such a curve makes it possible to express B by a short string. When such a curve is not available, the signature length of our scheme is much shorter than those of the other previous schemes. We also assume that the representations of G_T and G are 1020 bits and 172 bits. A group signature of the variant of the scheme in [22] is composed of seven $(\mathbb{Z}/p\mathbb{Z})^*$ and five $\mathcal{G}_1$ elements. That of the scheme in [121] is composed of ten $(\mathbb{Z}/p\mathbb{Z})^*$, six $\mathcal{G}_1$, and two $\mathcal{G}_T$ elements, and that of the scheme in [32] is composed of four $(\mathbb{Z}/p\mathbb{Z})^*$, three $\mathcal{G}_1$, and four $\mathcal{G}_T$ elements. In contrast, that of the proposed scheme is composed of six $(\mathbb{Z}/p\mathbb{Z})^*$, one $\mathcal{G}_1$, and three $\mathcal{G}$ elements, and thus its signature length is the shortest among the other previous schemes.

We also estimate the computational cost of our scheme and that of the previous schemes by the number of scalar multiplications/modular exponentiations in $\mathcal{G}, \mathcal{G}_1, \mathcal{G}_2$, and $\mathcal{G}_T$ and the number of pairing operations e required for Sign and Verify, since these are the most costly computations. Here, we assume that the signer has precomputed values $e(H, G_2), e(K, G_2), e(K, Y)$, and $e(A_U, G_2)$. Although we cannot present a precise estimation of the computational cost of each operation since it depends on the choice of the groups $\mathcal{G}, \mathcal{G}_1, \mathcal{G}_2$, and $\mathcal{G}_T$, these computations can be done quite efficiently if we choose Tate pairing for e and adopt the computation tools described in [107].

We also list the assumptions required in our scheme and the previous schemes [121, 32], and the variant of the scheme in [22]. From Theorems 10, 11, and 12, our scheme requires the SDH assumption, the Decision Diffie-Hellman assumption, and the existence of random oracles. The scheme in [22] requires the SDH assumption, the DLDH assumption, and the existence of random oracles. That in [121] requires the SDH assumption, the Decision Bilinear Diffie-Hellman (DBDH) assumption, and the existence of random oracles. That in [32] requires the Lysyanskaya-Rivest-Sahai-Wolf (LRSW) assumption, the Decision Diffie-Hellman assumption, and the existence of random oracles. The DLDH assumption is proposed in [22] which is proved to hold in generic bilinear groups. The LRSW assumption is proposed in [105] and is proved to hold in generic groups. The LRSW assumption is also proved to hold in generic bilinear groups in [32]. The SDH assumption and the LRSW assumption cannot be compares to each other.

These results of estimation and required assumptions are given in Table 8.1 [4], where "# of SMul" , "# of MExp", "# of pairings", and "Sig. Len." are abbreviations of "the number of scalar multiplications" , "the number of modular exponentiations", "the number of pairings", and "signature length".

Revocation: Installing the revocation mechanism proposed in [22] has no effect on this estima-

[4] The table is given by [88], which is better than the original one in [67].

tion. This can be seen as in the following: Given $(G_1, G_2, A_U, y_U, x_U, z_U, y_{\bar{U}}, \bar{H}, \bar{K}, Y)$ such that $Y = [w]G_2, [w + y_U]A_U + [x_U]H + [z_U]K = G_1, [w + y_{\bar{U}}]\bar{G}_1 = G_1, [w + y_{\bar{U}}]\bar{H} = H, [w + y_{\bar{U}}]\bar{K} = K$ for some $w \in (\mathbb{Z}/p\mathbb{Z})^*$, $\bar{A}_U$ that satisfies $[w + y_U]\bar{A}_U + [x_U]\bar{H} + [z_U]\bar{K} = \bar{G}_1$ can be computed as $\bar{A}_U = [1/(y_{\bar{U}} - y_U)](A_U - \bar{G}_1 - [x_U]\bar{H} - [z_U]\bar{K})$. .

8.6 Generation of Curves

The proposed scheme is most efficient among previously known group signature schemes in signature length and in computational complexity under q-strong Diffie-Hellman assumption in the random oracle model. This group signature scheme requires a pair of elliptic curves of the same order such that one provides efficient bilinear maps but the other has no efficient bilinear map. However, a generation of such a pair has not been reported. In this section, we report such a generation of pair of elliptic curves, which demonstrates the feasibility of our group signature scheme.

Roughly, we have constructed the pair of curves as in the following:

1. Choose a prime order elliptic curve with bilinear maps.

2. Generate an elliptic curve such that its order is the same to the above curve and that it has no efficiently computable bilinear maps.

The first step is easy if we use known algorithms proposed in [113, 138]. Although the second step cannot be completed in a time polynomial of the length of curve order, several method such as one in [4] are proposed. The question is that whether we can successfully generate a curve pair of required security parameter in realistic time.

The curves with bilinear maps that we choose is

$$\begin{aligned} y^2 &= x^3 + ax + b \\ \text{with} \\ a &= -3 \\ b &= 34563027717274042184109525861787336385553847212297605300752115677 0 \end{aligned}$$

defined over $\mathbb{F}_p$ of 224 bit prime

$$p = 15028799613985034465755506450771565229282832217860390155996483840017.$$

Its order N is prime and

$$N = 15028799613985034465755506450771561352583254744125520639296541195021.$$

Its embedding degree is 6. The curve was found in [138] using the algorithm proposed in that paper. Here, discriminant $-D$ of complex quadratic field for this curve is 496659.

We first briefly introduce the complex multiplication method on which our curve generation depend.

1. Choose complex quadratic field $K = \mathbb{Q}(\sqrt{-D})$ such that its corresponding class number h_D is small.
2. Find integers x and y that satisfy
$$4p = x^2 + Dy^2 \tag{8.4}$$
for some prime p.
3. Find roots of Hilbert class polynomial H_D with respect to $K = \mathbb{Q}(\sqrt{-D})$.
4. Obtain an elliptic curve defined over $\mathbb{F}_p$ whose j-invariant is a root of H_D.
5. The order of resulting curve is $p + 1 \pm x$.

We need to modifies Eq. (8.4) since the given parameter is the order of the elliptic curve but is not the field over which the curve is defined. Hence, we find a prime p and integers x, y that satisfy

$$\begin{aligned} 4N &= (x \pm 2)^2 + Dy^2 \\ p &= N - 1 \pm x \end{aligned}$$

for D with small class number h_D. 5 hours of search using Magma [106] on Pentium 4 3.4GHz machine successfully ends in several tuples of (D, x, y, p). From these tuple, with several minutes, we are able to construct an elliptic curve of order N.

The following is one example if the curve.

$$\begin{aligned} y^2 &= x^3 + ax + b \\ &\text{with} \\ a &= 4807863678177631534209903490596038497378606311932775891579012483613 \\ b &= 5413072257629771397335699469579480283163369916360461536284359662565 \end{aligned}$$

defined over $\mathbb{F}_p$ of 224 bit prime

$$p = 15028799613985034465755506450771557277920582540226013319442384629791$$

Here, discriminant $-D$ of complex quadratic field for this curve is 33307, whose class number is 28.

8.7 A Variant of Scheme in [22]

The following is a variant of the short signature scheme in [22].

M-KeyGen

Given 1^k, M-KeyGen chooses prime p of size k, bilinear groups $(\mathcal{G}_1, \mathcal{G}_2)$ of order p with a bilinear map e such that $e : \mathcal{G}_1 \times \mathcal{G}_2 \to \mathcal{G}_T$, generators G_1 and G_2 of $\mathcal{G}_1$ and $\mathcal{G}_2$, an isomorphism ψ from $\mathcal{G}_2$ to $\mathcal{G}_1$ with $\psi(G_2) = G_1$, and a hash function $\mathcal{H}$ that maps strings to $(\mathbb{Z}/p\mathbb{Z})^*$.

Next, M-KeyGen randomly chooses $w \in_R \mathbb{Z}/p\mathbb{Z}$ and $H \in_R \mathcal{G}_1$ and generates $W = [w]G_2$. Finally, M-KeyGen outputs

$$\begin{aligned} msk &= w \\ mpk &= (p, \mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_T, e, G_1, G_2, \psi, \mathcal{H}, W, H) \end{aligned}$$

T-KeyGen

Given mpk, T-KeyGen first randomly chooses $\xi_1, \xi_2, \bar{\xi}_1, \bar{\xi}_2 \in_R \mathbb{Z}/p\mathbb{Z}$. Next, T-KeyGen generates $R, V, \bar{R}$, and $\bar{V}$ such that $H = [\xi_1]R = [\xi_2]V = [\bar{\xi}_1]\bar{R} = [\bar{\xi}_2]\bar{V}$. Finally, T-KeyGen outputs

$$\begin{aligned} tsk &= (\xi_1, \xi_2, \bar{\xi}_1, \bar{\xi}_2) \\ tpk &= (R, V, \bar{R}, \bar{V}) \end{aligned}$$

Join$_{MM,U}$

1. - MM is given group member list $\mathcal{L}$, an identity of a user U, mpk, and msk.
   - A user U is given mpk.
2. U randomly chooses $(x'_U, z) \in_R (\mathbb{Z}/p\mathbb{Z})^2$, generates $H' = [x_U]H + [z]G_1$ and sends H_U to MM.
3. MM randomly chooses $(x''_U, z') \in_R ((\mathbb{Z}/p\mathbb{Z})^*)^2$ and send them to U.

4. U generates

$$\begin{aligned} x_U &= x''_U x'_U + z' \\ H_U &= [x_U]H \end{aligned}$$

and sends H_U to MM. Then, U proves in zero-knowledge to MM the knowledge of x_U and $r'(= x''_U z)$ satisfying

$$\begin{aligned} H_U &= [x_U]H \\ [x''_U]H' + [z']H - H_U &= [r']H. \end{aligned}$$

5. The MM randomly chooses $y_U \in_R \mathbb{Z}/p\mathbb{Z}$, generates

$$A_U = [1/(w + y_U)](G_1 - H_U)$$

and sends (A_U, y_U) to U as a membership certificate. The MM adds an entry $(U, \mathit{ider}_U := A_U)$ to its group member list $\mathcal{L}$.

6.
   - MM outputs the revised $\mathcal{L}$.
   - U outputs

$$\begin{aligned} \mathit{cert}_U &= (A_U, y_U) \\ \mathit{sk}_U &= x_U \\ \mathit{ider}_U &= A_U \end{aligned}$$

Sign

1. Sign is given $\mathit{mpk}, \mathit{tpk}, \mathit{cert}_U, \mathit{sk}_U, m$

2. Sign randomly chooses $(\alpha, \beta) \in_R ((\mathbb{Z}/p\mathbb{Z})^*)^2$ and generates

$$T_1 = [\alpha]R \tag{8.5}$$

$$T_2 = [\beta]V \tag{8.6}$$

$$\bar{T}_1 = [\alpha]\bar{R} \tag{8.7}$$

$$\bar{T}_2 = [\beta]\bar{V} \tag{8.8}$$

$$T_3 = [\alpha + \beta]H + A_U \tag{8.9}$$

3. Sign randomly chooses $(\alpha', \beta', x', \delta'_1, \delta'_2, y') \in_R ((\mathbb{Z}/p\mathbb{Z})^*)^6$ and generates

$$\begin{aligned}
R_1 &= [\alpha']R \\
R_2 &= [\beta']V \\
\bar{R}_1 &= [\alpha']\bar{R} \\
\bar{R}_2 &= [\beta']\bar{V} \\
R_3 &= e(T_3, G_2)^{y'} e(H, W)^{-(\alpha'+\beta')} \\
&\quad e(H, G_2)^{-(\delta'_1+\delta'_2)+x'} \\
R_4 &= [y']T_1 - [\delta'_1]R \\
R_5 &= [y']T_2 - [\delta'_2]V
\end{aligned}$$

4. Sign generates

$$\begin{aligned}
c = {} & \mathcal{H}(p, G_1, G_2, G_T, \psi, W, H, R, V, \bar{R}, \bar{V}, T_1, T_2, \\
& \bar{T}_1, \bar{T}_2, T_3, R_1, R_2, \bar{R}_1, \bar{R}_2, R_3, R_4, R_5, m)
\end{aligned}$$

5. Sign generates

$$\begin{aligned}
s_\alpha &= \alpha' + c\alpha \\
s_\beta &= \beta' + c\beta \\
s_x &= x' + cx_U \\
s_{\delta_1} &= \delta'_1 + cy_U\alpha \\
s_{\delta_2} &= \delta'_2 + cy_U\beta \\
s_y &= y' + cy_U
\end{aligned}$$

6. Sign outputs

$$gs = (T_1, T_2, \bar{T}_1, \bar{T}_2, T_3, c, s_\alpha, s_\beta, s_x, s_{\delta_1}, s_{\delta_2}, s_y)$$

as a signature on message m.

Verify

1. Verify is given mpk, tpk, m, and gs.

2. Verify generates

$$\begin{aligned}
R_1 &= [s_\alpha]R - [c]T_1 \\
R_2 &= [s_\beta]V - [c]T_2 \\
\bar{R}_1 &= [s_\alpha]\bar{R} - [c]\bar{T}_1 \\
\bar{R}_2 &= [s_\beta]\bar{V} - [c]\bar{T}_2 \\
R_3 &= e(T_3, [s_y]G_2 + [c]W)e(H, W)^{-(s_\alpha + s_\beta)} \\
&\quad e(H, G_2)^{-(s_{\delta_1} + s_{\delta_2}) + s_x} e(G_1, G_2)^{-c} \\
R_4 &= [s_y]T_1 - [s_{\delta_1}]R \\
R_5 &= [s_y]T_2 - [s_{\delta_2}]R
\end{aligned}$$

3. Verify outputs acc if equation

$$\begin{aligned}
c = {} & \mathcal{H}(p, G_1, G_2, G_T, \psi, W, H, R, V, \bar{R}, \bar{V}, T_1, T_2, \\
& \bar{T}_1, \bar{T}_2, T_3, R_1, R_2, \bar{R}_1, \bar{R}_2, R_3, R_4, R_5, m)
\end{aligned}$$

   holds. Otherwise, it outputs rej.

Open

1. Open is given $mpk, tpk, tsk, m, gs, \mathcal{L}$.

2. If Verify$(mpk, tpk, m, gs) = rej$, it outputs $\perp$ and stops.

3. Open generates and outputs

$$\begin{aligned}
Q &= T_3 - [\xi_1]T_1 - [\xi_2]T_2 \\
&= T_3 - [\bar{\xi}_1]\bar{T}_1 - [\bar{\xi}_2]\bar{T}_2
\end{aligned}$$

   Then, Open generates and outputs a non-interactive proof of knowledge of either (ξ_1, ξ_2) or $(\bar{\xi}_1, \bar{\xi}_2)$ that satisfies either of the above equations and Q as a $proof$.

4. Open searches A_U that coincides with the Q in $\mathcal{L}$. If there is such a A_U, it outputs the corresponding U. Otherwise, it outputs $\perp'$.

8.8 Conclusion

We proposed a new group signature scheme which is secure if we assume the Decision Diffie-Hellman assumption, the q-Strong Diffie-Hellman assumption, and the existence of random oracles. The proposed scheme is the most efficient among the all previous group signature schemes in signature length and in computational complexity. We also generated a pair of curves that our scheme requires.

Chapter 9

Conclusion

In this dissertation, we have proposed several efficiency enhanced multi-party protocols. They are:

1. six protocols that are useful for electronic voting systems:
   (i) An efficient publicly verifiable shuffle that is perfect zero-knowledge argument.
   (ii) An efficient publicly verifiable shuffle and decryption that is complete permutation hiding argument.
   (iii) An efficient publicly verifiable anonymous hybrid mix-net.
   (iv) An efficient publicly verifiable shuffle and decryption that is perfect zero-knowledge argument.
   (v) An aggregate shuffle argument scheme.
   (vi) An efficient compiler from Σ to deniable zero-knowledge argument.
2. A black-box traitor revocable broadcast encryption scheme.
3. Two group signature schemes:
   (i) A group signature scheme for separate and distributed authorities.
   (ii) An efficient group signature based on bilinear mappings.

Since a multiple number of players engage in these protocols, they have complex security requirements. Such circumstance, besides the fact that the number of players itself is large, makes it hard to make the protocols efficient. We have succeeded to enhance efficient of these protocols, which has been a major and crucial interest in designing of these protocols.

Publication

Journal

1. Jun Furukawa: Secure Detection of Watermarks. IEICE Transactions 87-A(1): 212-220 (2004)
2. J. Furukawa: Efficient and Verifiable Shuffling and Shuffle-Decryption. IEICE Trans. Fundamentals, Vol.E88-A, No.1, pp.172-188, 2005.
3. Jun Furukawa, Hideki Imai: An Efficient Group Signature Scheme from Bilinear Maps. IEICE Trans. Fundamentals, Vol.E89-A, No.5, pp.1328-1338, 2006.
4. Rie Shigetomi, Akira Otsuka, Jun Furukawa, Keith Martin, Hideki Imai: A Provably Secure Refreshable Partially Anonymous Token and Its Applications. IEICE Trans. Fundamentals, Vol.E89-A, No.5, pp.1396-1406, 2006.
5. J. Furukawa, K. Sako: Efficient Publicly Verifiable Mix-net for Long Inputs. IEICE Trans. Fundamentals, 2007.

International Conferences

1. Jun Furukawa, Kazue Sako: An Efficient Scheme for Proving a Shuffle. CRYPTO 2001: 368-387
2. J. Furukawa, K. Mori, S. Obana, and K. Sako: An Implementation of a Universally Verifiable Electronic Voting Scheme based on Shuffling. Financial Cryptography 2002.
3. Jun Furukawa: Efficient, Verifiable Shuffle Decryption and Its Requirement of Unlinkability. Public Key Cryptography 2004: 319-332
4. Jun Furukawa, Shoko Yonezawa: Group Signatures with Separate and Distributed Authorities. Fourth Conference on Security in Communication Networks '04 (SCN04), 2004.

5. Isamu Teranishi, Jun Furukawa, Kazue Sako: k-Times Anonymous Authentication (Extended Abstract). ASIACRYPT 2004: 308-322

6. Rui Zhang, Jun Furukawa, Hideki Imai: Short Signature and Universal Designated Verifier Signature Without Random Oracles. ACNS 2005: pp.483-498.

7. Jun Furukawa, Hideki Imai: An Efficient Group Signature Scheme from Bilinear Maps. ACISP 2005: 455-467

8. J. Furukawa, K. Sako: Efficient Publicly Verifiable Mix-net for Long Inputs. Financial Cryptography 2006.

Bibliography

[1] Masayuki Abe: Universally Verifiable Mix-net with Verification Work Independent of the Number of Mix-servers. EUROCRYPT 1998: 437-447

[2] Masayuki Abe: Mix-Networks on Permutation Networks. Advances in Cryptology — ASIACRYPT '99, LNCS 1716, pp. 258-273, Springer-Verlag, (1999).

[3] Masayuki Abe and Hideki Imai: Flaws in Some Robust Optimistic Mix-Nets. ACISP 2003: 39-50.

[4] Amod Agashe and Kristin Lauter and Ramarathnam Venkatesan: Constructing elliptic curves with a given number of points over a finite field. Cryptology ePrint Archive, Report 2001/096

[5] te Jun Anzai, Natsume Matsuzaki, Tsutomu Matsumoto: A Quick Group Key Distribution Scheme with "Entity Revocation". ASIACRYPT 1999: 333-347

[6] G. Ateniese, J. Camenisch, M. Joye, G. Tsudik: A Practical and Provable Secure Coalition-Resistant Group Signature Scheme. CRYPTO 2000, LNCS 1880, pp.255–270.

[7] G. Ateniese, B. de Medeiros: Efficient Group Signatures without Trapdoors. ASIACRYPT 2003, LNCS 2894, pp.246–268.

[8] G. Ateniese and B. de Medeiros: A Provably Secure Nyberg-Rueppel Signature Variant with Applications, Cryptographic ePrint Archive, 2004/093.

[9] Nuttapong Attrapadung, Hideki Imai: Graph-Decomposition-Based Frameworks for Subset-Cover Broadcast Encryption and Efficient Instantiations. ASIACRYPT 2005: 100-120

[10] Nuttapong Attrapadung, Kazukuni Kobara, Hideki Imai: Sequential Key Derivation Patterns for Broadcast Encryption and Key Predistribution Schemes. ASIACRYPT 2003: 374-391

[11] Boaz Barak, Oded Goldreich, Shafi Goldwasser, Yehuda Lindell: Resettably-Sound Zero-Knowledge and its Applications. FOCS 2001: 116-125

[12] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, Ke Yang: On the (Im)possibility of Obfuscating Programs. CRYPTO 2001: 1-18

[13] Niko Bari, Birgit Pfitzmann: Collision-Free Accumulators and Fail-Stop Signature Schemes without Trees. EUROCRYPT 1997: 480-494.

[14] Mihir Bellare, Haixia Shi, Chong Zhang: Foundations of Group Signatures: The Case of Dynamic Groups. CT-RSA 2005: 136-153

[15] M. Bellare, D. Micciancio, B. Warinschi: Foundations of Group Signatures: Formal Definitions, Simplified Requirements, and a Construction Based on General Assumptions. EUROCRYPT 2003, LNCS 2656, pp. 614-629.

[16] Mihir Bellare, Adriana Palacio: GQ and Schnorr Identification Schemes: Proofs of Security against Impersonation under Active and Concurrent Attacks. CRYPTO 2002: 162-177

[17] Michael Ben-Or, Oded Goldreich, Shafi Goldwasser, Johan Håstad, Joe Kilian, Silvio Micali, and Phillip Rogaway: Everything Provable is Provable in Zero-Knowledge. CRYPTO 1988: 37-56.

[18] Shimshon Berkovits: How To Broadcast A Secret. EUROCRYPT 1991: 535-541

[19] Dan Boneh, Xavier Boyen: Short Signatures Without Random Oracles. EUROCRYPT 2004: 56-73.

[20] Dan Boneh, Xavier Boyen: Efficient Selective-ID Secure Identity-Based Encryption Without Random Oracles. EUROCRYPT 2004: 223-238

[21] Dan Boneh, Xavier Boyen, Eu-Jin Goh: Hierarchical Identity Based Encryption with Constant Size Ciphertext. EUROCRYPT 2005: 440-456

[22] Dan Boneh, Xavier Boyen, Hovav Shacham: Short Group Signature. CRYPTO 2004, Lecture Notes in Computer Science 3152, pp. 41-55, 2004, Springer.

[23] Dan Boneh, Matthew K. Franklin: An Efficient Public Key Traitor Tracing Scheme. CRYPTO 1999: 338-353

[24] Dan Boneh, Craig Gentry, Brent Waters: Collusion Resistant Broadcast Encryption With Short Ciphertexts and Private Keys, eprint.

[25] Dan Boneh, Amit Sahai, Brent Waters: Fully Collusion Resistant Traitor Tracing. To appear in Eurocrypt 2006.

[26] Dan Boneh, Brent Waters: Fully Collusion Resistant Broadcast, Trace and Revoke System. At the author's web page: http://crypto.stanford.edu/ dabo/papers/tr.pdf

[27] F. Boudot: Efficient Proofs that a Committed Number Lies in an Interval, EUROCRYPT 2000, LNCS 1807, pp. 431–444.

[28] S. Brands: An Efficient Off-line Electronic Cash System Based On The Representation Problem. CWI Technical Report CS-R9323, (1993).

[29] Reinier Bröker, Peter Stevenhagen: Elliptic Curves with a Given Number of Points. ANTS 2004: 117-131

[30] Jan Camenisch, Jens Groth: Group Signatures: Better Efficiency and New Theoretical Aspects. Security in Communication Networks - SCN 2004, LNCS series.

[31] Jan Camenisch, Anna Lysyanskaya: A Signature Scheme with Efficient Protocols. SCN 2002: 268-289.

[32] Jan Camenisch, Anna Lysyanskaya: Signature Schemes and Anonymous Credentials from Bilinear Maps. Crypto 2004, Springer Verlag, 2004.

[33] J. Camenisch, M. Michels: A group signature scheme based on an RSA-variant. Technical Report RS-98-27, BRICS, University of Aarhus, November 1998. An earlier version appears in ASIACRYPT '98.

[34] Jan Camenisch, Victor Shoup: Practical Verifiable Encryption and Decryption of Discrete Logarithms. CRYPTO 2003: 126-144 2003

[35] J. Camenisch, M. Stadler: Efficient Group Signature Schemes for Large Groups. CRYPTO '97, LNCS 1296, pp. 410–424.

[36] Ran Canetti: Universally Composable Security: A New Paradigm for Cryptographic Protocols. FOCS 2001, pp. 136-145

[37] R. Canetti, O. Goldreich, S. Goldwasser, and S. Micali: Resettable Zero-Knowledge. Proc. of STOC 2000.

[38] Ran Canetti, Shai Halevi, Jonathan Katz: Chosen-Ciphertext Security from Identity-Based Encryption. EUROCRYPT 2004: 207-222

[39] R. Canetti, J. Kilian, E. Petrank, and A. Rosen: Black-box concurrent zero-knowledge requires Omega (log n) rounds. ACM Symposium on Theory of Computing, pp. 570-579, 2001.

[40] Hervé Chabanne, Duong Hieu Phan, David Pointcheval: Public Traceability in Traitor Tracing Schemes. EUROCRYPT 2005: 542-558

[41] D. Chaum: Zero-Knowledge undeniable signatures. LNCS 473, Proc. Eurocrypt 1990, pp. 458-464 1991.

[42] D. Chaum: Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms, Communications of the ACM, Vol.24, No.2, pp. 84-88, (1981).

[43] David Chaum, Hans Van Antwerpen: Undeniable Signatures. CRYPTO 1989: 212-216

[44] D. Chaum, E. van Heyst: Group Signatures. EUROCRYPT '91, LNCS 547, pp. 257–265.

[45] Benny Chor, Amos Fiat, Moni Naor: Tracing Traitors. CRYPTO 1994: 257-270

[46] Ronald Cramer, Ivan Damgård, Berry Schoenmakers: Proofs of Partial Knowledge and Simplified Design of Witness Hiding Protocols. CRYPTO 1994: 174-187

[47] R. Cramer and V. Shoup: A practical key cryptosystem provably secure against adaptive chosen ciphertext attack, Advances in Cryptology — Crypto '98, LNCS 1462, pp. 13-25, (1998).

[48] R.Cramer and V. Shoup: Design and analysis of practical public-key encryption scheme secure against adaptive chosen ciphertext attack. SIAM Journal on Computing, Vol. 33, No. 1, pp. 167-226, 2003.

[49] Giovanni Di Crescenzo, Giuseppe Persiano, Ivan Visconti: Constant-Round Resettable Zero Knowledge with Concurrent Soundness in the Bare Public-Key Model. CRYPTO 2004: 237-253

[50] I. Damgård: Efficient Concurrent Zero-Knowledge in the Auxiliary String Model. LNCS 1807, Proc. Eurocrypt 2000, pp.419-430

[51] Vergnaud Damien: Private communication.

[52] Yevgeniy Dodis, Nelly Fazio: Public Key Trace and Revoke Scheme Secure against Adaptive Chosen Ciphertext Attack. Public Key Cryptography 2003: 100-115

[53] Cynthia Dwork, Moni Naor: Zaps and Their Applications. Electronic Colloquium on Computational Complexity (ECCC)(001): (2002)

[54] Cynthia Dwork, Moni Naor, Amit Sahai: Concurrent Zero-Knowledge. STOC 1998: 409-418.

[55] C. Dwork, A. Sahai: Concurrent Zero-Knowledge: Reducing the Need for Timing Constraints. LNCS 1462, pp.442- 1998.

[56] Cynthia Dwork, Larry J. Stockmeyer: 2-round zero knowledge and proof auditors. STOC 2002: 322-331

[57] U. Feige and A. Shamir: Zero Knowledge Proofs of Knowledge in Two Rounds. Crypto 89, pp.526-544, 1990

[58] U. Feige and A. Shamir: Witness Indistinguishable and Witness Hiding Protocols. Proc. of STOC, pp. 416-426 1990.

[59] Amos Fiat, Moni Naor: Broadcast Encryption. CRYPTO 1993: 480-491

[60] Amos Fiat, Adi Shamir: How to Prove Yourself: Practical Solutions to Identification and Signature Problems. CRYPTO 1986: 186-194.

[61] Amos Fiat, Tamir Tassa: Dynamic Traitor Training. CRYPTO 1999: 354-371

[62] Marc Fischlin: Communication-Efficient Non-Interactive Proofs of Knowledge with Online Extractors. CRYPTO 2005:

[63] Pierre-Alain Fouque and David Pointcheval: Threshold Cryptosystems Secure against Chosen-Ciphertext Attacks. ASIACRYPT 2001: 351-368.

[64] Jun Furukawa: Secure Detection of Watermarks. IEICE Transactions 87-A(1): 212-220 (2004)

[65] Jun Furukawa: Efficient, Verifiable Shuffle Decryption and Its Requirement of Unlinkability. Public Key Cryptography 2004: 319-332

[66] J. Furukawa: Efficient and Verifiable Shuffling and Shuffle-Decryption. IEICE Trans. Fundamentals, Vol.E88-A, No.1, pp.172-188, 2005.

[67] Jun Furukawa, Hideki Imai: An Efficient Group Signature Scheme from Bilinear Maps. ACISP 2005: 455-467

[68] Jun Furukawa, Hideki Imai: An Efficient Group Signature Scheme from Bilinear Maps. IEICE Trans. Fundamentals, Vol.E89-A, No.5, pp.1328-1338, 2006.

[69] J. Furukawa, K. Mori, S. Obana, and K. Sako: An Implementation of a Universally Verifiable Electronic Voting Scheme based on Shuffling. Financial Cryptography 2002.

[70] Jun Furukawa, Kazue Sako: An Efficient Scheme for Proving a Shuffle. CRYPTO 2001: 368-387

[71] J. Furukawa, K. Sako: Efficient Publicly Verifiable Mix-net for Long Inputs. Financial Cryptography 2006.

[72] J. Furukawa, K. Sako: Efficient Publicly Verifiable Mix-net for Long Inputs. IEICE TRANS. FUNDAMENTALS, 2007.

[73] Jun Furukawa, Shoko Yonezawa: Group Signatures with Separate and Distributed Authorities. Fourth Conference on Security in Communication Networks '04 (SCN04), 2004.

[74] Rosario Gennaro and Victor Shoup: A Note on an Encryption Scheme of Kurosawa and Desmedt. Cryptology ePrint Archive, Report 2004/194

[75] L. Goldenberg, L. Vaidman, and S. Wiesner: Quantum Gambling, Phys. Rev. Lett., 82, pp.3356-3359, 1999.

[76] O. Goldreich: A Uniform-Complexity Treatment of Encryption and Zero-Knowledge, Journal of Cryptology, Vol. 6, pp. 21-53, (1993).

[77] Oded Goldreich, Ariel Kahan: How to Construct Constant-Round Zero-Knowledge Proof Systems for NP. J. Cryptology 9(3): 167-190 (1996)

[78] O. Goldreich and H. Krawczyk: On the Composition of Zero Knowledge Proof Systems. SIAM J. on Computing, Vol.25, No.1, pp.169-192,1996

[79] O. Goldreich, S. Micali, and A. Wigderson: A Proof that Yields Nothing but Their Validity or All Languages in NP Have Zero-Knowledge Proof System. ACM, 38, 1, pp. 691-729 1991.

[80] O. Goldreich and Y. Oren: Definitions and properties of Zero-Knowledge proof systems. Journal of Cryptology, Vol. 7, No. 1 pp. 1-32 1994.

[81] S. Goldwasser and S. Micali: Probabilistic Encryption, JCSS, Vol. 28, No. 2, pp. 270-299, (1984).

[82] S. Goldwasser, S. Micali, and C. Rackoff: The Knowledge Complexity of Interactive Proof Systems. SIAM J. Comput. Vol. 18, No. 1, pp. 186-208 1989.

[83] P. Golle, S. Zhong, D. Boneh, M. Jakobsson, and A. Juels: Optimistic mixing for exit-polls, Asiacrypt 2002, LNCS 2501, pp. 451-465 (2002)

[84] Michael T. Goodrich, Jonathan Z. Sun, Roberto Tamassia: Efficient Tree-Based Revocation in Groups of Low-State Devices. CRYPTO 2004: 511-527

[85] Jens Groth: A verifiable Secret Shuffle of Holomorphic Encryptions. Public Key Cryptography 2003 pp. 145-160 (2003).

[86] Jens Groth: A Verifiable Secret Shuffle of Homomorphic Encryptions. Cryptology ePrint Archive, Report 2005/246

[87] Dani Halevy, Adi Shamir: The LSD Broadcast Encryption Scheme. CRYPTO 2002: 47-60

[88] Henrik S. Hansen, Kristoffer K. Pagels: Private communication.

[89] R. Impagliazzo, L.Levin and M.Luby: Pseudo-random Generation from one-way functions. STOC 1989, pp.12–24 (1989)

[90] Tetsuya Izu, Tsuyoshi Takagi: Efficient Computations of the Tate Pairing for the Large MOV Degrees. ICISC 2002: 283-297.

[91] M. Jakobsson: A practical mix, Eurocrypt '98, LNCS 1403, pp. 448-461 (1998)

[92] A. Juels and M. Jakobsson, An optimally robust hybrid mix network. Proc. of the 20th annual ACM Symposium on Principles of Distributed Computation, 2001.

[93] Aggelos Kiayias, Moti Yung: Group Signatures: Provable Security, Efficient Constructions and Anonymity from Trapdoor-Holders. MYCRYPT 2005, LNCS 3175, pp.151-170.

[94] A. Kiayias, Y. Tsiounis, M. Yung: Traceable Signatures. EUROCRYPT 2004, LNCS 3027, pp. 571–589.

[95] Joe Kilian, Erez Petrank: Identity Escrow. CRYPTO 1998: 169-185.

[96] J. Kilian and E. Petrank: Concurrent zero-knowledge in poly-logarithmic rounds. STOC 2001.

[97] J. Kilian, E. Petrank, and C. Rackoff: Lower Bounds for Zero Knowledge on the Internet. FOCS 1998: 484-492.

[98] J. Kilian, E. Petrank, R. Richardson: On Concurrent and Resettable Zero-Knowledge Proofs for NP.

[99] Seungjoo Kim, Sung Jun Park, Dongho Won: Convertible Group Signatures. ASIACRYPT 1996: 311-321.

[100] Tetsutaro Kobayashi, Kazumaro Aoki, Fumitaka Hoshino, and Hiroaki Oguro: Software Implementation of Parallel Elliptic Curve Cryptosystem. The 2001 Symposium on Cryptography and Information Security, Oiso, Japan, Vol 1, pp.299-303, 2001.

[101] Norikazu Kubotera, Jun Furukawa, Kazue Sako: A Method for Generating Elliptic Curves suitable for Implementing an Efficient Group Signature Scheme. ISEC 2006.

[102] Kaoru Kurosawa, Yvo Desmedt: Optimum Traitor Tracing and Asymmetric Schemes. EUROCRYPT 1998

[103] Kaoru Kurosawa and Yvo Desmedt: A New Paradigm of Hybrid Encryption Scheme. Crypto 2004, pp. 426-442.

[104] Kaoru Kurosawa, Swee-Huay Heng: 3-Move Undeniable Signature Scheme. EUROCRYPT 2005: 181-197

[105] Anna Lysyanskaya, Ronald L. Rivest, Amit Sahai, Stefan Wolf: Pseudonym Systems. Selected Areas in Cryptography 1999: 184-199.

[106] MAGMA, The Magma Computational Algebra System for Algebra, Number Theory and Geometry, http://magma.maths.usyd.edu.au/magma/

[107] A. Menezes, C. van Oorschot, S. Vanstone: Handbook of Applied Cryptography. CRC Press, pp. 617-627, (1997).

[108] S. Micali1 and L. Reyzin: Min-Round Resettable Zero-Knowledge in the Public-Key Model. Eurocrypt 2001, LNCS 2045, pp.373-393, 2001

[109] Silvio Micali, Leonid Reyzin: Soundness in the Public-Key Model. CRYPTO 2001: 542-565

[110] Daniele Micciancio, Erez Petrank: Efficient and Concurrent Zero-Knowledge from any public coin HVZK protocol. Electronic Colloquium on Computational Complexity (ECCC)(045):(2002).

[111] M. Michels and M. Stadler: Efficient Convertible Undeniable Signature Schemes. Proc. SAC 1997 pp. 231-244 1997.

[112] S. Mitsunari, R. Sakai, M. Kasahara: A new Traitor tracing. IEICE Trans. Fundamentals, E85-A(2), pp.481-484, Feb. 2002.

[113] Atsuko Miyaji, Masaki Nakabayashi, Shunzou Takano: New explicit conditions of elliptic curve traces for FR-reduction. IEICE Trans. E85-A(2), pp. 481-484, 2002.

[114] A. Miyaji and K. Umeda: A Fully-Functional Group Signature Scheme over Only Known-Order Group, ACNS 2004, LNCS 3089, pp. 164–179.

[115] Dalit Naor, Moni Naor, Jeffery Lotspiech: Revocation and Tracing Schemes for Stateless Receivers. CRYPTO 2001: 41-62

[116] M. Naor: Bit Commitment Using Pseudo-Randomness. Journal of Cryptology, vol.4, 1991,pp.151-158

[117] Moni Naor, Benny Pinkas: Threshold Traitor Tracing. CRYPTO 1998: 502-517

[118] Moni Naor, Benny Pinkas: Efficient Trace and Revoke Schemes. Financial Cryptography 2000: 1-20

[119] Moni Naor, Moti Yung: Public-key Cryptosystems Provably Secure against Chosen Ciphertext Attacks. STOC 1990: 427-437.

[120] C.A. Neff: A Verifiable Secret Shuffle and its Application to E-Voting, ACMCCS 01 pp. 116-125 (2001).

[121] Lan Nguyen, Rei Safavi-Naini: Efficient and Provably Secure Trapdoor-free Group Signature Schemes from Bilinear Pairings. Asiacrypt 2004. pp. 372-386.

[122] Lan Nguyen, Reihaneh Safavi-Naini, and Kaoru Kurosawa: Verifiable Shuffles: A Formal Model and a Paillier-Based Efficient Construction with Provable Security. ACNS 2004: 61-75

[123] K. Nyberg and R. A. Rueppel: Message Recovery for Signature Schemes Based on the Discrete Logarithm Problem, EUROCRYPT '94, LNCS 950, pp. 182–193.

[124] W. Ogata, K. Kurosawa and S.H. Heng, The Security of the FDH Variant of Chaum's Undeniable Signature Scheme, Accepted by IEEE Trans. on IT.

[125] W. Ogata, K. Kurosawa, K. Sako, and K. Takatani: Fault tolerant anonymous channel, ICICS, LNCS 1334, pp. 440-444 (1997).

[126] M. Ohkubo and M .Abe: A length-invariant hybrid mix, Asiacrypt 2000, LNCS 1976, pp. 178-191 (2000)

[127] T. Okamoto: An Efficient Divisible Electronic Cash Scheme, CRYPTO '95, LNCS 963, pp. 438–451.

[128] C. Park, K. Itoh, and K. Kurosawa: Efficient Anonymous Channel and All/Nothing Election Protocol, Eurocrypt '93, pp.248-259, (1993)

[129] R. Pass: On Deniability in the Common Reference String and Random Oracle Model. CRYPTO 2003, pp. 316-337.

[130] T. Pedersen: A Threshold Cryptosystem without a Trusted Party, EUROCRYPT '91, LNCS 547, pp. 522–526.

[131] D. Pointcheval, J. Stern: Security Arguments for Digital Signatures and Blind Signatures. J. Cryptology 13(3): 361-396 (2000).

[132] R. Richardson and J. Kilian: On the Concurrent Composition of Zero-Knowledge Proofs. EUROCRYPT 1999, pp. 415-431, 1999.

[133] Reihaneh Safavi-Naini, Yejing Wang: Sequential Traitor Tracing. CRYPTO 2000: 316-332

[134] Amit Sahai: Non-Malleable Non-Interactive Zero Knowledge and Adaptive Chosen-Ciphertext Security. FOCS 1999: 543-553

[135] K. Sako: Electronic voting protocols allowing open objection to the tally, Transactions of IEICE, Vol. E77-A No. 1, Jan. (1994).

[136] K. Sako, A Network Voting System Using a Mix-net in a Japanese Private Organization. DIMACS Workshop on Electronic Voting – Theory and Practice 2004.

[137] K. Sako and J. Kilian: Receipt-free mix-type voting scheme –A practical solution to the implementation of voting booth. Eurocrypt '95, pp. 393-403 (1995).

[138] Michael Scott, Paulo S.L.M Barreto: Generating more MNT elliptic curves. Cryptology ePrint Archive, Report 2004/058.

[139] C. P. Schnorr: Efficient signature generation by smart cards, Journal of Cryptology, 4, pp. 161–174, (1991).

[140] Claus-Peter Schnorr, Markus Jakobsson: Security of Signed ElGamal Encryption. ASIACRYPT 2000: 73-89.

[141] Rie Shigetomi, Akira Otsuka, Jun Furukawa, Keith Martin, Hideki Imai: A Provably Secure Refreshable Partially Anonymous Token and Its Applications. IEICE Trans. Fundamentals, Vol.E89-A, No.5, pp.1396-1406, 2006.

[142] V. Shoup and R. Gennaro: Securing Threshold Cryptosystems against Chosen Ciphertext Attack, EUROCRYPT 1998: 1-16

[143] M. Stadler: Publicly Verifiable Secret Sharing, EUROCRYPT '96, LNCS 1070, pp. 190–199.

[144] Douglas R. Stinson, Ruizhong Wei: Combinatorial Properties and Constructions of Traceability Schemes and Frameproof Codes. SIAM J. Discrete Math. 11(1): 41-53 (1998)

[145] Douglas R. Stinson, Ruizhong Wei: Key Preassigned Traceability Schemes for Broadcast Encryption. Selected Areas in Cryptography 1998: 144-156

[146] Isamu Teranishi, Jun Furukawa, Kazue Sako: k-Times Anonymous Authentication (Extended Abstract). ASIACRYPT 2004: 308-322

[147] Y. Tsiounis and M. Yung, On the Security of ElGamal Based Encryption. Public Key Cryptography 1998: 117-134.

[148] Douglas Wikström: A Universally Composable Mix-Net. TCC 2004: 317-335.

[149] Douglas Wikström: A Universally Composable Mix-Net. TCC 2004: 317-335

[150] Douglas Wikström: A Sender Verifiable Mix-Net and a New Proof of a Shuffle. Asiacrypt 2005.

[151] Rui Zhang, Jun Furukawa, Hideki Imai: Short Signature and Universal Designated Verifier Signature Without Random Oracles. ACNS 2005: pp.483-498.

[152] R. Zhang, G. Hanaoka, J. Shikata, and H. Imai, On the Security of Multiple Encryption or CCA-security+CCA-security=CCA-security?. Public Key Cryptography 2004, pp.360-374.

[153] Yunlei ZHAO: Concurrent/Resettable Zero-Knowledge With Concurrent Soundness in the Bare Public-Key Model and Its Applications. Cryptology ePrint Archive, Report 2003/265.