2018

# New Conditional Privacy-preserving Encryption Schemes in Communication Network

Zhongyuan Yao
*University of Wollongong*

UNIVERSITY
OF WOLLONGONG
AUSTRALIA

# New Conditional Privacy-preserving Encryption Schemes in Communication Network

Zhongyuan Yao

Supervisor:
Prof. Yi Mu
Co-supervisor:
Dr. Guomin Yang

*This thesis is presented as required for the conferral of the degree:*

Doctor of Philosophy

The University of Wollongong
School of Computing and Information Technology

August 2018

## Declaration

*I, Zhongyuan Yao, declare that this thesis submitted in fulfilment of the requirements for the conferral of the degree Doctor of Philosophy, from the University of Wollongong, is wholly my own work unless otherwise referenced or acknowledged. This document has not been submitted for qualifications at any other academic institution.*

---

***Zhongyuan Yao***
*July 7, 2019*

# Abstract

Nowadays the communication networks have acted as nearly the most important fundamental infrastructure in our human society. The basic service provided by the communication networks are like that provided by the ubiquitous public utilities. For example, the cable television network provides the distribution of information to its subscribers, which is much like the water or gas supply systems which distribute the commodities to citizens. The communication network also facilitates the development of many network-based applications such as industrial pipeline controlling in the industrial network, voice over long-term evolution (VoLTE) in the mobile network and mixture reality (MR) in the computer network, etc. Since the communication network plays such a vital role in almost every aspect of our life, undoubtedly, the information transmitted over it should be guarded properly. Roughly, such information can be categorized into either the communicated message or the sensitive information related to the users. Since we already got cryptographical tools, such as encryption schemes, to ensure the confidentiality of communicated messages, it is the sensitive personal information which should be paid special attentions to. Moreover, for the benefit of reducing the network burden in some instances, it may require that only communication information among legitimated users, such as streaming media service subscribers, can be stored and then relayed in the network. In this case, the network should be empowered with the capability to verify whether the transmitted message is exchanged between legitimated users without leaking the privacy of those users. Meanwhile, the intended receiver of a transmitted message should be able to identify the exact message sender for future communication. In order to cater to those requirements, we re-define a notion named conditional user privacy preservation.

In this thesis, we investigate the problem how to preserve user conditional privacy in pubic key encryption schemes, which are used to secure the transmitted information in the communication networks. In fact, even the term conditional privacy preservation has appeared in existing works before, there still have great differences between our conditional privacy preservation definition and the one proposed before. For example, in our definition, we do not need a trusted third party (TTP) to help tracing the sender of a message. Besides, the verification of a given

encrypted message can be done without any secret.

In this thesis, we also introduce more desirable features to our redefined notion user conditional privacy preservation. In our second work, we consider not only the conditional privacy of the message sender but also that of the intended message receiver. This work presents a new encryption scheme which can be implemented in communication networks where there exists a blacklist containing a list of blocked communication channels, and each of them is established by a pair of sender and receiver. With this encryption scheme, a verifier can confirm whether one ciphertext is belonging to a legitimated communication channel without knowing the exact sender and receiver of that ciphertext. With our two previous works, for a given ciphertext, we ensure that no one except its intended receiver can identify the sender. However, the receiver of one message may behave dishonest when it tries to retrieve the real message sender, which incurs the problem that the receiver of a message might manipulate the origin of the message successfully for its own benefit. To tackle this problem, we present a novel encryption scheme in our third work. Apart from preserving user conditional privacy, this work also enforces the receiver to give a publicly verifiable proof so as to convince others that it is honest during the process of identifying the actual message sender. In our forth work, we show our special interest in the access control encryption, or ACE for short, and find this primitive can inherently achieve user conditional privacy preservation to some extent. we present a newly constructed ACE scheme in this work, and our scheme has advantages over existing ACE schemes in two aspects. Firstly, our ACE scheme is more reliable than existing ones since we utilize a distributed sanitizing algorithm and thus avoid the so called single point failure happened in ACE systems with only one sanitizer. Then, since the ciphertext and key size of our scheme is more compact than that of the existing ACE schemes, our scheme enjoys better scalability.

# Acknowledgments

I would like to express my sincere thanks to my supervisors, Professor Yi Mu and Dr. Guomin Yang, for their careful guidance, helpful suggestions, and great patience. I still remember the scene of our first meeting at Professor Mu's office as if it happened to me days before. Prof. Mu and Dr. Yang spent so much time and effort on training me how to do research. I would treat what they have taught me as the most invaluable present I have ever had. Every time when I feel frustrated during my research, they encourage me and give me hints about the problem encountered, I can hardly complete my PhD study and finish this thesis without their encouragement and help.

I would also want to give thanks to my colleagues during my PhD candidacy, it is my great pleasure to work and live with them. The non-exhaustive list of whom includes: *Dr. Fuchun Guo, Dr. Peng Jiang, Dr. Nan Li, Dr. Kefeng Wang, Dr. Hui Cui, Dr. Rongmao Chen, Dr. Yinhao Jiang, Dr.Weiwei Liu, Dr. Jiannan Wei, Dr. Fatemeh Rezaeibagha, Dr. Yangguang Tian, Dr. Shiwei Zhang, Dr. Yanwei Zhou, Dr. Shengmin Xu, Dr. Jianchang Lai, Ms.Tong Wu, Ms. Xueqiao Liu, Ms. Zhen Zhao, Ms. Yannan Li, Mr. Ge Wu, Mr. Ke Huang, Mr. Xiaoguo Li and Mr. Binrui Zhu.* I really enjoy the bush walking, short-distance travel and festival parties we have joined before, you all and the activities will remain as precious memories in my life forever.

It is my honour to be a member of Institute of Cybersecurity and Cryptology(IC2), School of Computing and Information Technology, University of Wollongong. I would like to thank our University and the Chinese Scholarship Council(CSC) for providing me the scholarships to cover my tuition fee and living expenses respectively during my PhD candidacy.

Finally, I am sincerely grateful to my family, my father, mother, wife and my little son, for their generous love and support. It is very regretful that I cannot accompany them during my PhD candidacy. I would like to present this thesis as a gift to my family because it would have been impossible without their encouragement.

*Zhongyuan Yao*
September 2018
Wollongong, NSW, Australia

# Publications

This thesis is based on the following published or presented papers, which were finished when I was in pursuit of the PhD degree in University of Wollongong.

1. Zhongyuan Yao, Yi Mu, and Guomin Yang. A Privacy Preserving Source Verifiable Encryption Scheme. In Feng Bao and Liqun Chen and Robert H. Deng and Guojun Wang, editors, *Information Security Practice and Experience - 12th International Conference, ISPEC 2016, Zhangjiajie, China, November 16-18, 2016, Proceedings*, pages 182-193, Cham, 2016. Springer International Publishing.

2. Zhongyuan Yao, Yi Mu and Guomin Yang. Group-Based Source-Destination Verifiable Encryption with Blacklist Checking. In Joseph K. Liu and Ron Steinfeld, editors, *Information Security Practice and Experience - 13th International Conference, ISPEC 2017, Melbourne, VIC, Australia, December 13-15, 2017, Proceedings*, pages 186–203, Cham, 2017. Springer International Publishing.

3. Zhongyuan Yao and Yi Mu. ACE with Compact Ciphertext Size and Decentralized Sanitizers. *International Journal of Foundations of Computer Science*,pages 531-549, vol. 30, No.04,2019.

4. Zhongyuan Yao and Yi Mu. Publicly Verifiable Secure Communication with User and Data Privacy in Public Surveillance System. *Personal and Ubiquitous Computing*, pages 1-17, vol.24,No.01,2019.

# List of Notations

Some notations used throughout this thesis are listed below, other special ones will be defined when they are first appeared.

| | |
|---|---|
| $\ell$ | A security parameter; |
| $1^\ell$ | The string of $\ell$ ones; |
| $\forall$ | For all; |
| $\exists$ | There exists; |
| $\mathbb{Z}$ | The set of all integers; |
| $\mathbb{Z}^+$ | The set of all positive integers; |
| $\mathbb{Z}_p$ | The set consists of the integers modulo $p$; |
| $\mathbb{Z}_p^*$ | The multiple group of integers modulo $p$; |
| $\mathbb{G}$ | A set $\mathcal{G}$ or a group $\mathbb{G}$ when the binary operation is specified in the context; |
| $\|\mathbb{G}\|$ | The cardinality of the set $\mathbb{G}$ or the order of the group $\mathbb{G}$; |
| $\{u\}$ | a set with elements including $u$; |
| $\epsilon(\ell)$ | A negligible function on $\ell$; |
| $ord(a)$ | The order of an element $a$ in a group specified in the context; |
| $a\|\|b$ | The concatenation of the string $a$ and the string $b$; |
| $a \xleftarrow{R} \mathbb{G}, a \in_R \mathbb{G}$ | $a$ is selected from $\mathbb{G}$ uniformly at random; |
| $\mathsf{A}(x) \rightarrow y$ | $y$ is computed by running the algorithm $\mathbf{A}$ on input $x$; |
| $a \in \mathbf{A}$ ($a \notin \mathbf{A}$) | $a$ is (not) in the set $\mathbf{A}$. |
| $\mathbf{A} \bigcup \mathbf{B}$ | The union of sets $\mathbf{A}$ and $\mathbf{B}$; |
| $\mathcal{S}$ | The simulator in the security model; |
| $\mathcal{C}$ | The challenger in the security proof; |
| $\mathcal{A}$ | The adversary in the security model; |
| $\mathcal{O}$ | One oracle in the security proof; |
| $\Pr[\mathsf{E}]$ | The probability when event $\mathsf{E}$ occurs; |
| $\mathsf{Adv}_{\mathcal{A}}$ | The advantage of $\mathcal{A}$ when it wins the game. |

# List of Abbreviations

The following abbreviations are used throughout this thesis, other special ones will be defined when they are first used.

| | |
|---|---|
| PPT | Probabilistic polynomial time; |
| DL | Discrete Logarithm; |
| CDH | Computational Diffie-Hellman; |
| DDH | Decisional Diffie-Hellman; |
| GDH | Generalized Diffie-Hellman; |
| DHD | Diffie-Hellman Decision; |
| GDDHE | General Decision Diffie-Hellman Exponent; |
| $k$-CCA | $k$-Collision Attack Assumption |
| s-RSA | Strong RSA; |
| ZPK | Zero-knowledge Proof; |
| ZKPK | Zerok-knowledge proof of Knowledge; |
| PKE | Public Key Encryption; |
| PKG | Private Key Generator; |
| IND-CCA2 | Indistinguishability against Adaptive Chosen Ciphertext Attacks; |
| IND-CPA | Indistinguishability against Adaptive Chose Plaintext Attacks; |
| EU-CMA | Existentially Unforgeable under Chosen-message Attacks; |
| SEU-CMA | Strong Existentially Unforgeable under Chosen-message Attacks; |

# Contents

# Chapter 1

# Introduction

## 1.1 Background

**Communication Network.** Basically, a communication network is a set of nodes, electrical devices mainly, connected by various types of transmission medium such as twisted pair, fiber optic and even the air. Examples of such communication networks include telephone networks, computer networks, and also the Internet. Since the nodes in a communication network are interconnected with great flexibility and reasonable redundancy, it allows information to be transmitted successfully among nodes located in various geographical points and thus facilitates the flow of information. The communication network provides two primary functionalities for users, one is to gather a large volume of information efficiently, and another is to share their information easily. Those two functionalities form the basis of many existed network-based services, such as e-mail and FTP, and therefore nourish an unlimited number of future network-based applications such as mixture reality (MR), tele-surgery, etc.

**Public Key Encryption.** Introduced by Diffie and Hellman in [DH76], the notion of public key encryption, PKE for short, provides us with new direction when we find tools to keep the transmitted information confidential in the communication network. Unlike the symmetric key encryption, or secret key encryption, the two keys, the public key and secret key respectively, in the PKE are not the same. The public key can be published publicly and the secret key should be kept private. Furthermore, it is infeasible to compute the secret key from its corresponding public key. In a PKE scheme, a message sender encrypts a message under its communicator's public key , which is received from and authenticated by the PKI. Then, the generated ciphertext is sent to the receiver. Latter, the receiver can recover the plaintext from the given ciphertext using its own secret key. PKE relieves the key distribution and management problems existed in symmetric key encryption.

However, the execution of a PKE algorithm is more costly comparing to that of a symmetric key encryption algorithm, and it takes immerse expense to maintain a robust PKI in the public key encryption scheme. Thus, in practical applications, the PKE is usually used to transmit secret keys confidentially between parties and the symmetric key encryption is latter applied to exchange a bulk of data, such kind of encryption mode is thus called hybrid encryption.

Some standard PKE schemes, such as the RSA encryption [RSA83], the ElGamal encryption [Gam85] and the Cramer-Shoup encryption [CS98a], can only provide the functionality of message confidentiality. There also exists some PKE schemes which preserve extra functionalities. For example, the broadcast encryption scheme [FN93] enables a message sender to send a encrypted information to multiple recipients. The identity-based encryption scheme [Sha84, BF01] allows users to use unique information about the identity of the receiver as the pubic key rather than to receive authenticated public key from the PKI. The public-key encryption with keyword search scheme [BCOP04] empowers one user with the capability to search encrypted keywords from the original encrypted data without compromising its confidentiality. The signcryption scheme [Zhe97] combines the functionalities of both the public key encryption and digital signature in a logical single step, it enjoys extra benefits with respect to computational costs and communication overheads comparing to the traditional sign-then-encrypt approach.

**Research Problem.** In many network applications, keeping the exchanged message confidential may be the primary concern but is not the only concern. For example, in e-voting or e-cash systems, the user privacy of the participants should also be protected. We introduce a new notion named conditional privacy preservation in this thesis. Unlike the conventional user privacy definition which only focuses on protecting users' personal information from being leaked, our conditional one requires that not only one user's privacy is preserved but also its legitimation can still be publicly verified. Also, the anonymity of this user can only be revoked by its corresponding communicator. One obvious benefit of our new notion is that it enables some entities in the communication network to act as a cryptographic "firewall" to filtrate information not sent from legitimated users, which further reduces the communicational and computational cost in the network. Thus, we think our conditional privacy preservation definition is more desirable than the conventional one in practice.

Our conditional privacy preservation property is different from that mentioned in [RH07, LLZ+08, ELO13, HBCC13] even they are all named conditional privacy preservation and they all require anonymous message authentication. The main difference is that the ones in previous papers all need a trusted third party (TTP) to

trace the private information, e.g., position, of one certain user. However, in our definition, it is the corresponding receiver of a message who has the capability to trace the personal information of the message sender. Thus, comparing to the conditional privacy preservation property defined before, our one should be more realistic since it is usually hardly and costly to maintain a TTP. Moreover, we extend our conditional privacy preservation property to capture more desirable features. For example, in our paper [YMY17], the communication blacklist checking is included in that property, it enables a node to check whether the communication channel between the sender and intended receiver of a given message is blocked without breaching the users' conditional privacy. In [YM19], the receiver's capability is enhanced to not only trace the exact sender of a message but also convince anyone that it is well-behaved during the process of finding that user without leaking its privacy. In [YM18], we gave an unique ACE construction which provides message confidentiality and user conditional privacy to some extent. In this work, the intended receivers of an ACE ciphertext can ascertain that the sender of that ciphertext is coming from a group but cannot identify it exactly. Besides, our construction also enjoys high reliability and scalability comparing to existing ACE schemes because of the distributed sanitizing algorithm and hierarchical structure of user keys respectively.

Communication schemes which preserve the two security properties, message confidentiality and user conditional privacy preservation, simultaneously are useful in many real-life network applications. For example, in the camera surveillance system, the location of one specific camera should be kept private to resist certain location-based network attacks. Additionally, this camera's legitimation should be verifiable to enable it to send encrypted information to the server. Besides, the server should be empowered with the capability to retrieve the locations of the cameras to index all the received encrypted information. And also in the DNA database system, one searcher may not want to leak either any of its personal information or its searched content to anyone except the database manager during searching the database. Also, it should be able to authenticate itself to the database manager to be ascertained that it has the privilege to access the database. Furthermore, since the database manager may charge the searcher for this database service, it should have the capability to trace the exact sender of each encrypted query.

In this thesis, we are aiming at proposing encryption schemes implemented in communication networks which provide the functionality of conditional privacy preservation as explained before, and we find there exists no PKE schemes with such functionality before our works.

**The Research Gap.** We studied some existed primitives and found they all have deficiencies in solving our research problem. For instance, the primitive group sig-

nature [CvH91] provides the user with the capability to authenticate its legitimation among a group of users including itself, that is, it guarantees the user privacy during the authentication. However, the group manager in the scheme is the only authority which can revoke the anonymity of the actual signer of a signature, and the revocation phase needs the participation of the trusted authority. The ring signcryption [HSMZ05a] provides properties such as message confidentiality, user authentication and user privacy, but it is infeasible for the receiver to retrieve the actual sender of a ciphertext in this primitive. Latter, some techniques [FS07, LASZ14, LLM$^+$07] have been proposed to address the issue how to trace the signer in the ring signature. However, those techniques can hardly be applied in the ring signcryption schemes. Besides, in most existing ring signcryption schemes, the legitimation of the signer of a given ciphertext is not publicly verified and can only be checked by its intended receiver, which further makes this primitive inadequate to be a promising solution to our research problem.

The Access Control Encryption(ACE) scheme, first proposed in [DHO16], provides another approach to maintain the two properties, message confidentiality and user conditional privacy preservation, simultaneously. In one ACE scheme, the information flow is controlled in such a manner that only specified parties are allowed to communicate freely according to the given access control policy, even when some of them are misbehaving. With this primitive, the message sent from one party is confidential to parities who is not its communicators, and the intended receivers of a message can only confirm that the sender of that message is eligible to communicate with them, but can hardly identify that party. The cryptographic primitive ACE seems like a promising solution to our research problem, while existing ACE schemes have deficiencies in the aspects of system reliability and scalability.

Explicitly, there must exist a sanitizer in existing ACE schemes [DHO16, FGKO17, TZMT17] to ensure that the access control strategies can be enforced successfully, thus the security of the sanitizer should be considered carefully and thoroughly. In previous works, the minimum security requirement of the sanitizer is that it should be semi-trusted. We argue that the sanitizer should also be extremely reliable, otherwise, none of the access control policies of the ACE scheme would be guaranteed further. Furthermore, existing ACE constructions can hardly be implemented directly in large-scale networks since none of their ciphertext size is compact. For example, the complexity of the ciphertext size of the ACE schemes in [DHO16, FGKO17] are exponential and polylogarithmic respectively in terms of the numbers of identities $n$ in the system under standard cryptographic assumptions.

## 1.2 Contribution of This Thesis

The main contributions of our thesis are made to the following aspects.

1. Communication Schemes with User Conditional Privacy. In our first work, we formalize the notion of user conditional privacy preservation and propose a privacy preserving source-verifiable encryption scheme which maintains message confidentiality and user conditional privacy concurrently. We also give a server-aided variant of the construction of our scheme.

   In the second work, apart from discussing user conditional privacy, we consider a more complex scenario where a user should be able to prove the legitimation of the communication channel between it and its communicator without leaking their privacy. We find such scenario is realistic when there exists a authority in the system which maintains a publicly published blacklist to block communication channels between specific message senders and receivers. We present a group-based source-destination verifiable encryption scheme with blacklist checking which can be used in the scenario. Our construction utilizes the zero-knowledge proof of membership and also zero-knowledge of inequality techniques to handle the two aforementioned issues.

   Our two aforementioned works all give answers to the problem how to preserve the user conditional privacy, however, those two solutions have the same insufficiency. Namely, since the receiver in the proposed schemes is the only parity which can revoke the anonymity of the sender of a given ciphertext, and no one else in the system has the capability to verify whether the receiver behaves honestly during revealing the identity of the sender, it can manipulate the origination of one ciphertext successfully. We address this issue in our third work. We develop a secure communication scheme applied between the surveillance camera and the server. With our scheme, the server can give a proof to convince others the origination of a ciphertext without leaking its content. Such property enables the server to build a searchable database using the camera's identifier as index and also enables the message auditor to check the ciphertext and its origination stored in the database without any dispute.

2. Access Control Encryption. The newly proposed primitive Access Control Encryption, ACE for short, provides data confidentiality and also user conditional privacy to some extent. Namely, the sanitizer, which is an indispensable party in the ACE system, is in charge of the communication channel and ensures that only a legitimated user from one layer can send encrypted messages to users in the upper layer. Moreover, it cannot identify the exact identity of the sender but find the specific layer this user lays in.

In our fourth work, we give the first ACE scheme construction where the ciphertext size is compact. Moreover, the resulted scheme keeps not only the ciphertext size but also the key size of each users in the ACE compact. One more significant contribution of this work is giving a decentralized implementation of the sanitizer in ACE system to restrain its capability and also increase its reliability. Unlike previous scheme with only one sanitizer, our construction distributes the sanitizing functionality of the origin ACE among $n$ sanitizers, it is impossible for one of the sanitizers in our construction to produce a new access policy, so our construction imposes restriction on the capability of the sanitizer. Besides, as one message sender in our ACE construction can choose $t$ out of $n$ sanitizers itself to collaboratively produce a valid sanitized ciphertext, even some of the $n$ sanitizers cannot provide service or off-line, the whole ACE system can never encounter the single-point-failure and still work as normal. So, our construction improves the reliability of the sanitizer and even the robustness of the whole ACE system.

## 1.3 Thesis Organization

The remainder of this thesis is organized as follows.

In Chapter 2, we first summarize the miscellaneous notations used throughout this thesis. Then, we introduce some basic knowledge about the abstract algebra such as group, field and bilinear maps. Furthermore, we review the computational complexity theory and some well studied complexity assumptions. Finally, we describe some basic cryptographic tools such as hash functions, zero-knowledge proofs used as building blocks in our proposed schemes presented in the thesis.

In Chapter 3, we propose a privacy-preserving source verifiable encryption scheme guaranteeing message confidentiality and user conditional privacy simultaneously in communication networks. Furthermore, we define three models to capture the desirable properties preserved by our scheme and prove its security in the random oracles model.

In Chapter 4, we consider user conditional privacy preservation in a more complex scenario. Namely, apart from keeping user conditional privacy, we also try to address the issue of how to prove the legitimation of the communication channel between a message sender and its intended communicator. We present a encryption scheme and define security models for it, latter we prove its security with the help of random oracle.

In Chapter 5, we explore the security issues existed in security surveillance systems, we propose important and desirable security and privacy features that should be achieved by such systems. Latter, to achieve all the security goals, we

present a secure communication scheme applied between the surveillance camera and the server. We present formal security models to define these security requirements and give formal security proofs in the random oracle model.

In Chapter 6, we present an access control encryption (ACE) scheme which enjoys advantages over previous works in several aspects such as key and ciphertext size. Our scheme is also believed to be the first implementation of ACE with decentralized sanitizers. We also define our extended no-write rule model to allow the corruption of some sanitizers in our ACE system. We prove the security of our scheme under models define in this chapter.

We conclude our thesis in Chapter 7.

# Chapter 2

# Preliminaries

To make our thesis self-contained, we introduce miscellaneous notations, basic notions including abstract algebra, computational complexity and assumptions, and cryptographic tools which will be used throughout this thesis in this chapter. For more detailed cryptography theory, readers are recommended to the books [Mao03, LK14].

## 2.1 Miscellaneous Notations

In this thesis, a set, usually denoted by a capital, is a collection of distinct elements, if the number of elements in that set is finite, we call it a finite set, otherwise, it is an infinite set. For a finite set $X$, the two notations $x \xleftarrow{R} X$, $x \in_R X$ can all be used to denote that $x$ is selected randomly and uniformly from the set $X$, so we can use them interchangeably. We use the notation $|X|$ to represent the cardinality of set $X$, which equals to the number of elements in $X$. Conventionally, the blackboard bold is used to represent some special sets of numbers. For example, the set of all prime numbers, all natural numbers, all integers and all real numbers are denoted by $\mathbb{P}, \mathbb{N}, \mathbb{Z}$ and $\mathbb{R}$ respectively. Given a positive integer $p$, the notation $\mathbb{Z}_p$ denotes a set consisting of integers modulo $p$, that is $\mathbb{Z}_p = \{0, 1, 2, \cdots, p-1\}$, by $[p]$, we denote a set of integers $\{1, 2, \cdots, p\}$. Given two strings $a, b$, we denote by $a||b$ the concatenation of the string $a$ and $b$.

We denote by $\ell$ the security parameter and $1^\ell$ the string of $\ell$ ones. A function $\epsilon : \mathbb{Z} \to \mathbb{R}$ is said to be negligible if for all $k \in \mathbb{Z}$, there exists one element $z \in \mathbb{Z}$ such that $\epsilon(x) \leq \frac{1}{x^k}$ for all $x > z$. Unless otherwise specified, by $\epsilon$, we always denote a negligible function. By $p(x) \xleftarrow{R} \mathbb{Z}_p[x]$, we denote the polynomial $p(x)$ is randomly selected from the polynomial ring $\mathbb{Z}_p[x]$ consisting of the polynomials that coefficients are from the finite field $\mathbb{Z}_p$.

## 2.2 Abstract Algebra

In this section, some basic abstract algebra structures including groups, rings and bilinear maps are reviewed.

### 2.2.1 Groups

**Definition 2.1 (Group)** *A group, usually denoted by* $(\mathbb{G}, \circ)$*, consists of a non-empty set* $\mathbb{G}$ *and a binary operation* $\circ$ *over elements in* $\mathbb{G}$*, it should satisfy the following four properties.*

- **Closure**: $\forall a, b \in \mathbb{G},\ a \circ b \in \mathbb{G}$.
- **Associativity**: $\forall a, b, c \in \mathbb{G},\ (a \circ b) \circ c = a \circ (b \circ c)$.
- **Identity**: $\exists e \in \mathbb{G},\ \forall a \in \mathbb{G},\ a \circ e = e \circ a = a \in \mathbb{G}$.
- **Inverse**: $\forall a \in \mathbb{G},\ \exists a' \in \mathbb{G},\ a \circ a' = a' \circ a = e$.

The order of one group $(\mathbb{G}, \circ)$ is defined as the number of elements in $\mathbb{G}$ and thus denoted by $|\mathbb{G}|$. $(\mathbb{G}, \circ)$ is a finite group if its order is finite, otherwise, it is infinite. For simplicity, a group $(\mathbb{G}, \circ)$ can be denoted as $\mathbb{G}$ when the binary operation $\circ$ is specified in the context. Namely, we say $\mathbb{G}$ is an additive group if the binary operation $\circ$ is specified as addition $+$, an example of the additive group is $\mathbb{Z}_p$. While $\mathbb{G}$ is a multiplicative group when the binary operation $\circ$ is specified as multiplication $\cdot$, a simple multiplicative group is $\mathbb{Z}_p^*$.

For the benefit of differentiating the two types of groups, we use different notations to represent the identity and inverses in them. When $\mathbb{G}$ is an additive group, the additive identity $e$ is denoted as 0 and the inverse of $a \in \mathbb{G}$ is denoted as $-a$. When $\mathbb{G}$ is a multiplicative groups, the multiplicative identity $e$ is denoted as 1 and the inverse of $a \in \mathbb{G}$ is denoted as $a^{-1}$.

**Definition 2.2 (Abelian Group)** *A group* $(\mathbb{G}, \circ)$ *is an abelian group if the following property is satisfied.*

- **Commutativity:** $\forall a, b \in \mathbb{G},\ a \circ b = b \circ a$**.**

**Definition 2.3 (Order of A Group Element)** *By ord*$(a)$*, we denote the order of* $a$ *in* $(\mathbb{G}, \circ)$*. Given a group* $(\mathbb{G}, \circ)$ *with identity* $e$ *and an element* $a \in \mathbb{G}$*, let* $a^i = \underbrace{a \circ a \cdots a}_{i} \circ \cdots a$*, if there exists a positive integer* $j$ *satisfying* $a^j = e$ *and* $a^l \neq 1$ *for all positive integers* $l < j$*, then we say* $j$ *is the order of* $a$ *in* $(\mathbb{G}, \circ)$*, that is,* ord$(a) = j$*. If there exists no positive integers* $j$ *such that* $a^j = e$*, then we say the order of* $a$ *in* $(\mathbb{G}, \circ)$ *is infinite.*

**Definition 2.4 (Cyclic Group)** *A group $(\mathbb{G}, \circ)$ is a cyclic group if there exists an element $g \in \mathbb{G}$, for every $h \in \mathbb{G}$, there exists a positive integer $i$ such that $g^i = h$. The element $g$ can then be called as a generator of $(\mathbb{G}, \circ)$, and the group $(\mathbb{G}, \circ)$ can be represented as $\langle g \rangle$, or we say the group $(\mathbb{G}, \circ)$ is generated by $g$.*

Notice that definition 2.4 also applies to groups with other operations, such as addition.

### 2.2.2 Rings and Fields

**Definition 2.5 (Ring)** *A ring, usually denoted by $(\mathbb{G}, +, \cdot)$, consists of an abelian group $(\mathbb{G}, +)$ and a binary operation $\cdot$ over elements in $\mathbb{R}$, it satisfies the following four properties.*

- **Closure under multiplication**: $\forall a, b \in \mathbb{G},\ a \cdot b \in \mathbb{G}$.
- **Associativity of multiplication**: $\forall a, b, c \in \mathbb{G},\ (a \cdot b) \cdot c = a \cdot (b \cdot c)$.
- **Multiplicative identity**: $\exists e \in \mathbb{G},\ \forall a \in \mathbb{G},\ a \cdot e = e \cdot a = a \in \mathbb{G}$.
- **Distributivity**: $\forall a, b, c \in \mathbb{G},\ a \cdot (b+c) = (a \cdot b) + (a \cdot c),\ (a+b) \cdot c = (a \cdot c) + (b \cdot c)$.

**Definition 2.6 (Communicative Ring)** *A ring $(\mathbb{G}, +, \cdot)$ is communicative if the following property is satisfied.*

- **Commutativity of multiplication**: $\forall a, b \in \mathbb{G},\ a \cdot b = b \cdot a$.

**Definition 2.7 (Field)** *A communicative ring $(\mathbb{G}, +, \cdot)$ is a field if $(\mathbb{G} \setminus \{0\}, \cdot)$ forms a group.*

### 2.2.3 Bilinear Maps

Let $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_\tau$ be three multiplicative cyclic groups with the same large prime order $p$. Let $g, h$ be the generators of $\mathbb{G}_1, \mathbb{G}_2$ respectively. A bilinear map is a map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_\tau$ satisfying the following properties.

- **Bilinearity**: $\forall a, b \in \mathbb{Z}_p, \forall x \in \mathbb{G}_1, \forall y \in \mathbb{G}_2, e(x^a, y^b) = e(x, y)^{ab}$
- **Non-Degeneracy**: $e(g, h) \neq 1_{\mathbb{G}_\tau}$, where $1_{\mathbb{G}_\tau}$ represents the identity of $\mathbb{G}_\tau$.
- **Computability**: $\forall x \in \mathbb{G}_1, \forall y \in \mathbb{G}_2,$, the element $e(x, y)$ in $\mathbb{G}_\tau$ can always be efficiently computed.

**Definition 2.8 (Bilinear Groups)** *A bilinear group system, denoted by $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_\tau, e)$, consists of three multiplicative cyclic groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_\tau$ and a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_\tau$. This group also requires the group action in $\mathbb{G}_1$, $\mathbb{G}_2$ can be computed efficiently and $|\mathbb{G}_1| = |\mathbb{G}_2| = |\mathbb{G}_\tau| = p$.*

## 2.3 Computational Complexity and Complexity Assumptions

Since the security of cryptographic schemes constructed in modern cryptography are mainly based on the computational complexity, in this section, we first introduce some important notions in this area and then present some intractable assumptions. More details about the complexity theory can be found in book [Gol03].

### 2.3.1 Turing Machine and Algorithm

In complexity theory, a Turing machine, first presented by Alan Turing in 1936, is a hypothetical machine which can be used to simulate any computer algorithm no matter how complicate it is. An algorithm works like a manual which clearly specifies the instructions needed to solve a class of problems. Once executed, it starts from an initial state and input, proceeds through a number of successive states and finally terminates at the end state and produces an output. An algorithm is deterministic if its outputs are always the same when the same particular input is given in multiple executions. If the algorithm employs a certain degree of randomness such as using random bits as its auxiliary input during the execution, it is called a randomized algorithm or a probabilistic algorithm. In this case, either the running time or the output of this algorithm are random variables defined by the involved random bits. A Turing machine is deterministic if it simulates a deterministic algorithm, otherwise, it is a probabilistic Turing machine.

A probabilistic algorithm is efficient if its running time is bounded by a polynomial $p(\cdot)$ defined by the length of its input. In this case, it is called a probabilistic polynomial-time (PPT) algorithm. In our thesis, all algorithms involved in the schemes should be PPT algorithms. Besides, the security models defined in our thesis also model the adversaries and oracles as PPT algorithms.

### 2.3.2 Problem Complexity Classes

**Definition 2.9 ($\mathcal{P}$ Problem)** *$L$ is a $\mathcal{P}$ problem if it is solvable in polynomial time. That is, there exists a deterministic Turing machine $M$ and a polynomial $p(\cdot)$ such that*

- *on input a problem instance $x$, $M$ halts after at most $p(|x|)$ steps, and*
- *$M(x) = 1$ if and only if $x \in L$.*

*$\mathcal{P}$ is the collection of all problems that can be solved in polynomial time.*

**Definition 2.10 ($\mathcal{NP}$ problem)** *$L$ is a $\mathcal{NP}$ problem if $L$ has solutions which can be efficiently tested for validity in polynomial time. That is, given a Boolean relation $R_L \subset \{0,1\}^* \times \{0,1\}^*$ in which each of its elements is a pair consisting of a problem instance of $L$ and its solution, there exists a deterministic Turning machine $M$ and a polynomial $p(\cdot)$ such that*

- *on input a problem instance and its solution pair $(x, y)$ such that $|y| \leq p(|x|)$, $M$ halts after at most $p(|x|)$ steps, and*
- *$M(x, y) = 1$ if and only if $x \in L, (x, y) \in R_L$. Here $y$ is also called a witness of membership of $x \in L$.*

*$\mathcal{NP}$ is a problem set consisting of all problems whose solutions can be tested for validity in polynomial time.*

**Definition 2.11 ($\mathcal{NP}$-complete problem)** *A problem is said to be $\mathcal{NP}$-complete if it is in $\mathcal{NP}$ and every problem in $\mathcal{NP}$ is polynomially reducible to it. A problem $L$ is polynomial reducible to another problem $L'$ if there exists a polynomial-time computable function $f$ such that $x \in L$ if and only if $f(x) \in L'$.*

Even the answer to the problem whether the two problem sets, $\mathcal{N}$ and $\mathcal{NP}$, are equal is still uncertain at present, it is widely believed that $\mathcal{P} \neq \mathcal{NP}$. In that case, every algorithm trying to solve a problem $L \in \mathcal{NP}$ will have a super-polynomial running time in the worst case. Since $\mathcal{NP}$-complete problems are harder than problems in $\mathcal{NP}$, so $\mathcal{NP}$-complete problems have definitely super-polynomial-time complexity in the worst case. That is to say, there exists no PPT algorithm which can solve a $\mathcal{NP}$ or $\mathcal{NP}$-complete problem. For simplicity, we say the $\mathcal{NP}$ or $\mathcal{NP}$-complete problems are hard to any existed algorithms when $\mathcal{P} \neq \mathcal{NP}$.

### 2.3.3 Computational and statistical indistinguishability

One important notion in complexity theory is the indistinguishability of two probability distributions. Given two probability distributions $\Omega_1(\ell)$ and $\Omega_2(\ell)$ which are defined over the same finite set $\Omega$ and the same security parameter $\ell$, this two distributions are said to be computational indistinguishable if there exist no PPT distinguisher which can tell the difference between them. We say they are statistically indistinguishable if no distinguisher, even with infinite computational power, can tell them apart. In this part, we give formal definitions of this two notions.

**Definition 2.12 (computational indistinguishability)** *We say that two probability distributions $\Omega_1(\ell)$ and $\Omega_2(\ell)$ defined above are computationally indistinguish-*

*able if, for all PPT algorithms $\mathcal{A}$,*

$$\left| \Pr_{x \in \Omega_1(\ell)}[\mathcal{A}(x) = 1] - \Pr_{x \in \Omega_2(\ell)}[\mathcal{A}(x) = 1] \right| \leq \epsilon(\ell).$$

**Definition 2.13 (statistical indistinguishability)** *We say that two probability distributions $\Omega_1(\ell)$ and $\Omega_2(\ell)$ defined above are statistically indistinguishable if*

$$\sum_z \left| \Pr_{x \in \Omega_1(\ell)}[x = z] - \Pr_{x \in \Omega_2(\ell)}[x = z] \right| \leq \epsilon(\ell).$$

Conventionally, if two probability distributions are statistically distinguishable, they are computationally distinguishable. Unless otherwise specified, by indistinguishability, we mean that it is computationally indistinguishable.

### 2.3.4 Hard Problems and Complexity Assumptions

In this part, we summarize some hard problems and their related complexity assumptions which are pervasively used in the security proofs given in this thesis.

**Definition 2.14 ( Discrete Logarithm(DL) Assumption [Odl84])** *Let $\mathbb{G} = \langle g \rangle$ and $\mathcal{G}(1^\ell) \to (p, \mathbb{G})$ where $\ell$ is the security parameter. Given $(g, y) \in \mathbb{G}^2$, we say that the discrete logarithm assumption holds on $\mathbb{G}$ if there exists no PPT adversary $\mathcal{A}$ which can compute a $x \in \mathbb{Z}_p$ such that $y = g^x$ with non-negligible advantage, that is*

$$\mathsf{Adv}_{\mathcal{A}}^{DL} = \Pr\left[y = g^x, x =: \mathcal{A}(p, g, y, \mathbb{G})\right] \leq \epsilon(\ell),$$

*where $\epsilon$ is a negligible function defined before, the probability is taken over the random choice of $y \in \mathbb{G}$ and the bits consumed by the adversary $\mathcal{A}$.*

**Definition 2.15 (Computational Diffie-Hellman(CDH) Assumption [DH76])** *Let $\mathbb{G}$ be a cyclic multiplicative group of prime order $p$ where $|p| = \ell$ and $\ell$ is the security parameter. Let $a, b \in_R \mathbb{Z}_p$, and $g \in_R \mathbb{G}$. Given $g$, $g^a$ and $g^b$, we say the CDH assumption holds on $\mathbb{G}$ if there exists not PPT algorithm $\mathcal{A}$ which can compute $g^{ab}$ with non-negligible advantage, that is*

$$\mathsf{Adv}_{\mathcal{A}}^{CDH} = \Pr[g^{ab} =: \mathcal{A}(g, g^a, g^b)] \leq \epsilon(\ell),$$

*where $\epsilon$ is a negligible function defined before, the probability is taken over the random choices of $a, b \in \mathbb{Z}_p$ and the bits consumed by the adversary $\mathcal{A}$.*

In [Mau94], the relationship between the DL assumption and CDH assumption is discussed in detail. There also exists a decisional variant of the CDH problem

named DDH problem, and many encryption schemes such as the ElGamal encryption scheme [Gam85] and Cramer-Shoup encryption schemes [CS98a] are based on this problem.

**Definition 2.16 (Decisional Diffie-Hellman(DDH) problem [Bon98])** *Let $\mathbb{G}$ be a cyclic multiplicative group of prime order $p$ where $|p| = \ell$ and $\ell$ is the security parameter. Let $a, b \in_R \mathbb{Z}_p$, and $g, Z \in_R \mathbb{G}$. Given two probability distributions $\mathcal{D}_{DDH} = \{(g, g^a, g^b, g^{ab})\}$ and $\mathcal{D}_R = \{(g, g^a, g^b, Z)\}$, we say the CDH assumption holds on $\mathbb{G}$ if there exists not PPT algorithm $\mathcal{A}$ which can distinguish the two distributions $\mathcal{D}_{DDH}$ and $\mathcal{D}_R$ with non-negligible advantage, that is*

$$\mathsf{Adv}_{\mathcal{A}}^{DDH} = |\Pr[1 =: \mathcal{A}(D \in_R \mathcal{D}_{DDH})] - \Pr[1 =: \mathcal{A}(D \in_R \mathcal{D}_R)]| \leq \epsilon(\ell),$$

*where $\epsilon$ is a negligible function defined before, the probability is taken over the random choices of $a, b \in_R \mathbb{Z}_p, g, Z \in_R \mathbb{G}$ and the bits consumed by the adversary $\mathcal{A}$.*

**Definition 2.17 (k-Collision Attack Assumption(K-CAA)[MSK02])** *Let $\mathbb{G}$ be a cyclic multiplicative group of prime order $p$ where $|p| = \ell$ and $\ell$ is the security parameter. For an integer $k$, and one element $x$ randomly chosen from $\mathbb{Z}_p$, $g^x \in \mathbb{G}$, given $g, g^x \in \mathcal{G}, h_1, h_2, \cdots, h_k \in \mathbb{Z}_p, g^{\frac{1}{x+h_1}}, g^{\frac{1}{x+h_2}}, \cdots, g^{\frac{1}{x+h_k}} \in \mathbb{G}$ as inputs, the problem solver needs to output element $g^{\frac{1}{x+h}}$ for some $h \notin \{h_1, h_2, \cdots, h_k\}$. We say the k-CCA holds in $\mathbb{G}$ if there exists no PPT algorithm $\mathcal{A}$ which can solve the k-CCA problem with non-negligible probability, that is*

$$\mathsf{Adv}_{\mathcal{A}}^{k-CCA} = \Pr[g^{\frac{1}{x+h}} =: \mathcal{A}(g, g^x, h_1, h_2, \cdots, h_k \in \mathbb{Z}_q, g^{\frac{1}{x+h_1}}, g^{\frac{1}{x+h_2}}, \cdots, g^{\frac{1}{x+h_k}})] \leq \epsilon,$$

*where $h \notin \{h_1, h_2, \cdots, h_k\}$ and the probability is over the random choice of the generator $g$ in $\mathbb{G}$, the random choice of $h_1, h_2, \cdots, h_k \in \mathbb{Z}_q$ and the random bits consumed by $\mathcal{A}$.*

**Definition 2.18 ($(f, g, F)$-GDDHE Assumption [Del07])** *Given a bilinear group system $\mathcal{B} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e(\cdot, \cdot))$ and let $f, g$ be two co-prime polynomials with pairwise distinct roots, of respective orders $t$ and $n$. Let $g_0, h_0$ be one generator of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively. the $(f, g, F)$-GDDHE problem is, given the tuple*

$$\begin{array}{llll} ( & g_0 & , g_0^{\gamma}, g_0^{\gamma^2}, \cdots, g_0^{\gamma^{t-1}}, & g_0^{\gamma \cdot f(\gamma)}, \quad g_0^{k \cdot \gamma \cdot f(\gamma)}) \in \mathbb{G}_1, \\ ( & h_0 & , h_0^{\gamma}, h_0^{\gamma^2}, \cdots, h_0^{\gamma^{2n}}, & h_0^{k \cdot g(\gamma)}) \in \mathbb{G}_2 \quad \textit{and} \\ & T & \in \mathbb{G}_T & \end{array}$$

*to decide whether $T$ is equal to $e(g_0, h_0)^{k \cdot f(\gamma)} \in \mathbb{G}_T$ or is a random element in $\mathbb{G}_T$. We say the $(f, g, F)$-GDDHE Assumption holds if there exists no PPT algorithm $\mathcal{A}$*

*which can solve the $(f, g, F)$-GDDHE problem with non-negligible probability, that is*

$$\mathsf{Adv}_{\mathcal{A}}^{GDDHE}(f, g, F) = |\Pr[e(g_0, h_0)^{k \cdot f(\gamma)} =: \mathcal{A}(f, g, F)] - \Pr[T \in \mathbb{G}_T / \{e(g_0, h_0)^{k \cdot f(\gamma)}\}, T =: \mathcal{A}(f, g, F)]| \leq \epsilon(\ell)$$

*The probability is taken over the random choice of all the group elements in $\mathbb{G}_1, \mathbb{G}_2$ and all the random bits consumed by $\mathcal{A}$.*

**Definition 2.19 (Strong RSA(s-RSA) Assumption [FO97])** *Let $\ell$ be the security parameter and $\mathcal{G}(1^\ell)$ be a group generator whose output is a group with composite order which has length $\ell$ and consists of two prime factors of length $(\ell - 2)/2$. The strong RSA problem is, given $G \leftarrow \mathcal{G}(1^\ell), z \in G/\{\pm 1\}$, to find a pair $(u, e) \in G \times \mathbb{Z}$ such that $u^e = z$ and $e > 1$. We say the strong RSA assumption holds if there exists no PPT algorithm $\mathcal{A}$ which can find a solution to the problem with non-negligible probability, that is*

$$\mathsf{Adv}_{\mathcal{A}}^{s-RSA} = \Pr[z = u^e \wedge e > 1 : \mathcal{A}(G, z) = (u, e)] \leq \epsilon(\ell)$$

*The probability is taken oven all the random bits using by $\mathcal{A}, \mathcal{G}_\ell$ and the uniformly random choice of $z$.*

**Definition 2.20 (Generalized Diffie-Hellman(GDH) Problem [BBR99])** *Let $\ell$ be the security parameter and $\mathcal{FG}$ be a probabilistic polynomial-time algorithm which takes $1^\ell$ as input and outputs $N = PQ$ where $P, Q$ are two n-bit prime numbers and $P \equiv Q \equiv 3 (mod\ 4)$, let $\vec{a} =< a_1, a_2, \cdots, a_k >$ be any sequence of $k \geq 2$ elements of $[n]$. Given $N \leftarrow \mathcal{FG}(1^\ell)$, one quadratic-residue $g \in \mathbb{Z}_N^*$ and a k-bit input $x = x_1 x_2 \cdots x_k$, to compute*

$$h_{N,g,\vec{a}}(x) = g^{\prod_{x_i=1} a_i} (mod\ N).$$

*The restriction is that the value*

$$h_{N,g,\vec{a}}^R(x') = g^{\prod_{x'_i=1} a_i} (mod\ N).$$

*for any $x' = x'_1 x'_2 \cdots x'_k \in \{0,1\}^k / \{x\}$ is known.*

**Definition 2.21 (GDH Assumption [BBR99])** *Given a generalized Diffie-Hellman problem instance $(N, g, x, \{h_{N,g,\vec{a}}^R\})$ where $N, g, x, \vec{a}$ are predefined above and $\{h_{N,g,\vec{a}}^R\}$ is the collection of all elements $h_{N,g,\vec{a}}^R(x') = g^{\prod_{x'_i=1} a_i} (mod\ N)$ for any $x' = x'_1 x'_2 \cdots x'_k \in \{0,1\}^k / \{x\}$, there exists no polynomial time algorithm $\mathcal{A}$ which can find a solution*

*to such instance with non-negligible probability, that is*

$$\mathsf{Adv}_{\mathcal{A}}^{GDH} = \Pr[\mathcal{A}(N, g, x, \{h_{N,g,\vec{a}}^{R}\}) = g^{\prod_{x_i=1} a_i}(mod\ N)] \leq \epsilon(\ell)$$

*The probability is taken over the random choice of all the group elements and the random bits using by* $\mathcal{A}, \mathcal{FG}$

**Definition 2.22 (CDH in Composite Group (CDHCG))** *When* $k = 2, \vec{a} = (a, b), x = 11$*, the GDH assumption can be valued as the Computational Diffie-Hellman assumption in composite group(CDHCG). That is,*

$$\mathsf{Adv}_{\mathcal{A}}^{CDHCG} = \Pr[\mathcal{A}(N, g, g^a, g^b) = g^{ab}(mod\ N)] \leq \epsilon(\ell).$$

*The probability is taken over the random choice of the group elements and the random bits using by* $\mathcal{A}, \mathcal{FG}$*.*

**Definition 2.23 (Diffie-Hellman Decision Assumption(DHD)[ACJT00a])** *Let* $\ell$ *be*
*the security parameter and* $\mathcal{G}(1^\ell)$ *be a group generator which generates groups with composite order* $n$ *such that* $|n| = \ell$*. Given one group* $G \in \mathcal{G}(1^\ell)$*,* $n'$ *be the divisor of* $G$*'s order of length* $\ell - 2$*. Define the following two sets*

$$\begin{aligned}
&\mathcal{DH}(G) := \{(g_1, y_1, g_2, y_2) \in G^4 | \\
&\mathsf{ord}(g_1) = \mathsf{ord}(g_2) = n', \log_{g_1} y_1 = \log_{g_2} y_2\} \\
&\mathcal{Q}(G) := \{(g_1, y_1, g_2, y_2) \in G^4 | \\
&\mathsf{ord}(g_1) = \mathsf{ord}(g_2) = \mathsf{ord}(y_1) = \mathsf{ord}(y_2) = n'\}
\end{aligned}$$

*of Diffie-Hellman and arbitary 4-tuples, respectively.*
*The DHD assumption states that given a specific* $G =< g >$ *where* $(g|n) = 1$*, a 4-tuples* $T \in G^4$*, there exists no probabilistic polynomial time algorithms* $\mathcal{A}$ *which can discover whether* $T \in \mathcal{DH}(G)$ *or* $T \in \mathcal{Q}(G)$ *with non-negligible advantage over random guess for sufficiently large* $\ell$*. Namely,*

$$\mathsf{Adv}_{\mathcal{A}}^{DHD} = \Pr[1 := \mathcal{A}(T \in_R \mathcal{DH}(G))] - \Pr[1 := \mathcal{A}(T \in_R \mathcal{Q}(G))] \leq \epsilon(\ell).$$

*The probability is taken over the random choice of all the group elements and the randomness used by* $\mathcal{A}$*.*

## 2.4 Cryptographic Tools

In this section, we review some basic cryptographic tools used in the constructed scheme and the security proofs presented in our thesis.

### 2.4.1 One-way Function and Cryptographic Hash Function

When we say a function $f$ is a one-way function, it means that $f$ is computationally easy to compute but computationally hard to invert. Explicitly, for the one-way function $f$, there exists a PPT algorithm which can compute $f(x)$ efficiently when given one input $x$ chosen from the co-domain of $f$. However, when given a image $y$ chosen from the domain of $f$, there exists no PPT algorithm which can find a $x$ such that $f(x) = y$ successfully with non-negligible probability, where the probability is taken over the choices of $y$ and the randomness consumed by the algorithm. Since the existence of one-way functions always imply the existence of efficient processes which is hard to inverse, and the security of many cryptographic schemes are basing on those processes, the one-way function plays an important role in modern cryptography. One promising candidate of the one-way function is the hash function.

Introduced by Carter and Wegman in [CW79], the universal classes of hash functions can be divided into tree types. Basically, a hash function $\mathcal{H} : \{0,1\}^* \rightarrow \{0,1\}^\lambda$ is a deterministic function which maps a bit string with any length to another bit string with fixed length $\lambda$. According to [Mao03], a hash function should preserve the following three properties:

1. Mixing Transformation. Given any input, the output of $\mathcal{H}$ should be computationally indistinguishable from any uniform binary string in $\{0,1\}^\lambda$;

2. Pre-image Resistance. Given an output $y \in \{0,1\}^\lambda$, it is computationally infeasible to find its corresponding pre-image $x$ such that $y = \mathcal{H}(x)$;

3. Collusion Resistance. It is computationally infeasible to find $x_1 \neq x_2$ such that $\mathcal{H}(x_1) = \mathcal{H}(x_2)$.

The cryptographic primitive hash function works as a indispensable ingredient in modern cryptography and has been used as a building block in many cryptographic tools such as encryption scheme [FOPS01], digital signature scheme [BR93], message authentication code (MAC) scheme [BCK96].

### 2.4.2 Random Oracle Model

The main topics of modern cryptography are about defining security notions and proposing schemes which are proven to be secure regarding to those notions. In some

instances, it is more desirable to construct schemes and then prove their security in more idealized computation models. The well-known random oracle is one of such models.

A *random oracle*, usually denoted by $\mathcal{O}$, can be viewed as a powerful and idealized hash function which preserves three properties: efficient, deterministic and uniform output. Explicitly, given any input to a *random oracle*, its corresponding output is efficiently computable. When provided with the same input multiple times, the *random oracle* would always output the same result. Furthermore, the output of a *random oracle* is uniformly distributed over its output space.

Formalized by Bellare and Rogaway in [BR93], the notion of random oracle model is an idealized model of cryptography hash functions where all parties have the privilege to access to it. In this model, the hash function $\mathcal{H}$ is modeled as a magic box where the hash value $\mathcal{H}(x)$ is completely random and unknown before querying the oracle when given an input $x$. More precisely, in a security proof, the simulator $\mathcal{S}$ maintains a hash list $\mathcal{H}$ of pairs $\{(w, h)\}$, which is initially empty, to answer hash queries made by the adversary $\mathcal{A}$. When $\mathcal{A}$ wants to "see" the hash value of $x$, it has to ask $\mathcal{S}$ with this query. When the simulator receives $x$, it first searches the list to find whether there already exists such an entry $(w, h)$ that $x = w$. If yes, $\mathcal{S}$ responds $\mathcal{A}$ with the hash value $h$, otherwise, $\mathcal{S}$ randomly uniformly chooses a value $h$ as the hash value $H(x)$ of $x$ and sends it back to $\mathcal{A}$, then it appends the newly created pair $(x, h)$ to the list $\mathcal{H}$ for later use.

When we are proving the security of a scheme in the random oracle, we still need to construct a so-called security reduction. It shows you how to construct an algorithm to break one certain well-studied computational assumption by interacting with an adversary $\mathcal{A}$ which can break the scheme. During the proof, the random oracle can be simulated by $\mathcal{S}$ as part of the reduction. Since the random oracle is fully controlled by $\mathcal{S}$, it can be programmed to answer $\mathcal{A}$ with any input as long as the outputs are uniformly randomly distributed. Such simulation enables the simulator to embed the hard problem into the random oracle and thus makes the security proof more simple. As a result, the schemes designed in the random oracle model are much more efficient than those designed in the standard model.

Notably, once the random oracle is initialized with a concrete hash function, the aforementioned advantages would not be preserved further. Canetti et al. [CGH04] have shown that cryptosystems proven secure in the random oracle model are not necessarily secure in the stand model.

### 2.4.3 Zero-knowledge Proofs

In general, a zero-knowledge proof (ZPK), introduced by Goldwasser, Micali and Rackoff [GMW86], is a two-party interactive protocol which is applied between one prover $\mathcal{P}$ and one verifier $\mathcal{V}$. It can be used by a prover to convince a verifier that a statement is true without leaking any extra information except the validity of the statement. Similarly, when a prover holds a secret and wants to prove its ownership of the secret without revealing it, the prover can use a zero-knowledge proof of knowledge (ZKPK) to convince a verifier of the truth without leaking any extra information about the secret. Zero-knowledge proofs have been introduced and defined formally in [FFS88].

**Definition 2.24** *Let $(\mathcal{P}, \mathcal{V})$ be an interactive proof system where the interactive machine $\mathcal{P}$ and $\mathcal{V}$ model the prover and verifier respectively. $(\mathcal{P}, \mathcal{V})$ is said to be a zero-knowledge protocol for proof of membership in a language $L$ if it satisfies the following conditions:*

1. *Completeness: For all language elements $x \in L$, the honest prover $\mathcal{P}$ will convince the honest verifier $\mathcal{V}$ to accept, except with negligible probability.*

2. *Soundness: For all $x \notin L$, any cheating prover $\mathcal{P}^*$ will be unable to convince the honest verifier to accept, except with negligible probability known as the soundness error. Depending on the types of cheating for which this guarantee is made, we have different notions of soundness.*

3. *Zero-Knowledge: For all $x \in L$, for every PPT interactive verifier $\mathcal{V}^*$, there exists a PPT algorithm $\mathcal{S}$ known as the simulator, such that for every $x \in L$ the two variables $< \mathcal{P}, \mathcal{V}^* > (x)$ and $\mathcal{S}(x)$ are indistinguishable to a distinguisher or the environment. Where $< \mathcal{P}, \mathcal{V}^* > (x)$ denotes the output of the interactive machine $\mathcal{V}^*$ after interacting with $\mathcal{P}$ on common input $x$ and $\mathcal{S}(x)$ the output of the machine $\mathcal{S}$ on input $x$. Depending on the classes of environments against whom these random variables remain indistinguishable, we have different notions of zero-knowledge.*

According to [GMW86], all languages in $\mathcal{NP}$ should have zero-knowledge proofs assuming the existence of one-way functions.

In this part, we also describe some zero-knowledge proof systems of the discrete logarithm. Our description follows the $\sum$-protocol [CDS94] manner. That is, when the prover wants to show its knowledge on some discrete logarithm statements to the verifier, the two parties involve in the following 3-move interaction:

1. The prover commits itself to a *commitment* $t$ and sends it to a verifier.

2. Upon receiving $t$, the verifier sends back a *challenge* $c$ to the prover.

3. The prover finally responds a *response* $s$ to the verifier .

To empower the prover with the capability of producing such proof system itself or, literally, make such proof system non-interactive, the Fiat-Shamir transformation [FS86] can be applied in the second step of the proof system. Namely, the prover can produce the *challenge* $c$ itself with the only constraint that the *commitment* $t$ should be properly produced in advance.

Conveniently, we use the notation ZPK$\{(x_1, x_2, \cdots, x_n) : st\}$, introduced by [JM97], to denote a zero-knowledge proof system. This notation shows that the prover has the knowledge of a tuple of values $(x_1, x_2, \cdots, x_n)$ such that the statement $st$ holds. Obviously, here the elements listed in the round bracket are those only known to the prover and being proved, while other parameters in $st$ are known to both the prover and the verifier. By using such notation, we give the following several zero-knowledge proof systems of some discrete logarithms statements:

* Proof system 1. ZPK$\{(a, b) : \mathcal{H}_1, (g, h, Z_1, Z_2) \in \mathbb{G}^4; Z_1 = g^a \wedge Z_2 = h^b\}$ [CS03a]
  For the prover, to prove the knowledge of two integers $a, b$ such that $Z_1 = g^a$ and $Z_2 = h^b$, it computes the following values:

  1. $w_1, w_2 \overset{R}{\leftarrow} \mathbb{Z}_p, t_1 = g^{w_1}, t_2 = h^{w_2}$,
  2. $c = \mathcal{H}_1(g||h||t_1||t_2)$,
  3. $s_1 = w_1 - c \cdot a, s_2 = w_2 - c \cdot b$.

  Finally, ZPK$\{(a, b) : \mathcal{H}_1, (g, h, Z_1, Z_2) \in \mathbb{G}^4; Z_1 = g^a \wedge Z_2 = h^b\} = (c, s_1, s_2)$.
  A verifier computes $t'_1 = g^{s_1} Z_1^c$, $t'_2 = h^{s_2} Z_2^c$ and accepts the given proof if and only if $c = \mathcal{H}_1(g||h||t'_1||t'_2)$.

* Proof system 2. ZPK$\{(a) : \mathcal{H}_2, l_1, l_2, (g, y) \in \mathbb{G}^2; y = g^a \wedge (a \in \{2^{l_1}, \cdots, 2^{l_1} + 2^{l_2}\})\}$ [CFT98, FO97, Bou00]
  For the prover, to prove the knowledge of $a$ such that $y = g^a$ and also $a$ lies in the interval $\{2^{l_1}, \cdots, 2^{l_1} + 2^{l_2}\}$, it computes the following values:

  1. $w \overset{R}{\leftarrow} \{0, 1\}^{l_2+k}$, $t = g^w$
  2. $c = \mathcal{H}_2(g||y||t)$,
  3. $s = w - c(a - 2^{l_1})$.

  Finally, ZPK$\{(a) : \mathcal{H}_2, l_1, l_2, (g, y) \in \mathbb{G}^2; y = g^a \wedge (a \in \{2^{l_1}, \cdots, 2^{l_1} + 2^{l_2}\})\} = (c, s)$.
  For a verifier, it computes $t' = g^{s - 2^{l_1} c} y^c$ and accepts the given proof if and only if $c = \mathcal{H}_2(g||h||t')$.

* Proof system 3. ZPK$\{(x) : \mathcal{H}_3, (g,h,y,z) \in \mathbb{G}^4; y = g^x, \log_g y \neq \log_h z\}$ [CS03a]
  For the prover, to prove the knowledge of $x$ such that $y = g^x$ and $\log_g y \neq \log_h z$, it does the following procedures;

  1. It chooses $a \xleftarrow{R} \mathbb{Z}_p$ and then sets $\alpha = a, \beta = ax$.
  2. It computes $st_1 = \frac{y^\alpha}{g^\beta}, st_2 = \frac{z^\alpha}{h^\beta}$.
  3. It chooses $w_1, w_2 \xleftarrow{R} \mathbb{Z}_p$ and sets the commitment $t_1 = y^{w_1}\frac{1}{g^{w_2}}, t_2 = z^{w_1}\frac{1}{h^{w_2}}$.
  4. It computes the challenge $c = \mathcal{H}_3(g||h||y||z||t_1||t_2)$.
  5. It generates the corresponding responses as $s_1 = w_1 - c\alpha, s_2 = w_2 - c\beta$.

  ZPK$\{(x) : \mathcal{H}, (g,h,y,z) \in \mathbb{G}^4; y = g^x, \log_g y \neq \log_h z\} = (st_1, st_2, c, s_1, s_2)$
  For a verifier, to verify such a given proof, it first checks whether $st_1 = 1$ and $st_2 \neq 1$, then it accepts the given proof if and only if

$$c = \mathcal{H}_3(g||h||y||z||\frac{y^{s_1}(st_1)^c}{g^{s_2}}||\frac{z^{s_1}(st_2)^c}{h^{s_2}}).$$

Notably, the group $\mathbb{G}$ used in aforementioned proof systems is generated by a group generator $\mathcal{G}(\cdot)$ on input the security parameter $\ell$. It is not specified and its group order $p$ can be either a prime or a composite number, the only restriction is that the discrete logarithm problem should be hard in $\mathbb{G}$. Also, for simplifying the description, the hash functions used in above proof systems are not clearly defined, in case of any dispute happened to it, we state that three hash functions used in the above proof systems can be $\mathcal{H}_1 : \{0,1\}^* \to \mathbb{Z}_P$,$\mathcal{H}_2 : \{0,1\}^* \to \{0,1\}^{l_2+k}$,$\mathcal{H}_3 : \{0,1\}^* \to \mathbb{Z}_P$ respectively.

### 2.4.4 Shamir's Secret Sharing

Given a secret $s$, Shamir's secret sharing scheme [Sha79] enables the secret holder to divide it into $n$ pieces $(s_1, \ldots, s_n)$ using a $k-1$ degree polynomial, the secret holder can then distribute the $n$ pieces among $n$ users so that each of the users only has a unique secret piece. The Shamir's secret sharing scheme ensures that the origin secret $s$ can only be recovered using polynomial interpolation when at least $k$ users join together, and it is infeasible to reveal any information about $x$ when knowing at most $k-1$ secret pieces. This is the reason why this scheme is also called a $(k,n)$ *Threshold Secret Sharing Scheme*. Mathematical details about how to recover $s$ are given as follows.

Given a user group $S = \{U_1, \ldots, U_n\}$, let $\mathbb{F}_q$ be a finite field of order $q$ where $q > n$ and $n$ is the number of users in $S$. Assuming each user in $S$ is associated with a public unique element $u_i \in \mathbb{F}_q$. To share a secret $s$ among users in $S$, the secret holder first chooses a random $k-1$ degree polynomial $p(x) = s + \sum_{j=1}^{k-1} a_j x^j$ where $a_j \in_R \mathbb{F}_q$. Then each user in $S$ is given a secret share $s_i = p(u_i)$. When $k$ users form a user set $A \subset S$, then we reconstruct the $k-1$ degree polynomial as $p(x) = \sum_{U_i \in A} \Delta_i^A s_i$ where $\Delta_i^A = \prod_{U_\ell \in A \wedge i \neq \ell} \frac{x - u_\ell}{u_i - u_\ell}$, then we can recover the secret $s = p(0)$.

### 2.4.5 Public-Key Encryption

Similar to the symmetric encryption scheme, the public-key, or asymmetric, encryption (PKE) scheme can also be used to ensure message confidentiality in unreliable communication channels. Unlike the symmetric encryption scheme where the encryption and decryption key are the same, the two keys in the PKE are different and usually relative. However, it is infeasible to compute the decryption key from the public encryption key.

**Definition 2.25 (PKE)** *Formalized by [DH76], a public key encryption (PKE) scheme can be defined by the following four algorithms.*

- $\mathsf{Setup}(1^\ell)$. *On input* $1^\ell$*, this setup algorithm outputs the public parameters* $params$.
- $\mathsf{KeyGen}(1^\ell)$. *Taking* $1^\ell$ *as input, the key generation algorithm outputs a secret-public pair* $(sk, pk)$.
- $\mathsf{Enc}(params, pk, m)$. *On input* $params$*,* $pk$ *and a message* $m$ *chosen from the message space specified in* $params$*, the encryption algorithm outputs a ciphertext* $CT$.
- $\mathsf{Dec}(params, sk, CT)$. *The decryption algorithm takes* $params$*,* $sk$ *and the ciphertext* $CT$ *as input and outputs the original message* $m$.

**Definition 2.26 (Correctness)** *The correctness of one public key encryption is ascertained if*

$$\Pr\left[\mathsf{Dec}(params, sk, CT) \rightarrow m \;\middle|\; \begin{array}{l} \mathsf{Setup}(1^\ell) \rightarrow params; \\ \mathsf{KeyGen}(1^\ell) \rightarrow (sk, pk); \\ \mathsf{Enc}(params, pk, m) \rightarrow CT \end{array}\right] = 1,$$

*where the probability is taken over the randomness consumed by all algorithms in the scheme.*

**Definition 2.27 (IND-CCA2)** *The indistinguishability against adaptive chosen ciphertext attacks (IND-CCA2) [RS91] is a strong standard security notion for the PKE scheme. It can defined by the following game executed between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$.*

- Setup. *Taking the security parameter $\ell$ as input, $\mathcal{C}$ executes* $\mathsf{Setup}(1^\ell)$ *to generate the public parameters params, then it sends parames to $\mathcal{A}$.*
- KeyGen. *$\mathcal{C}$ also executes* $\mathsf{KeyGen}(1^\ell)$ *to generate a secret-public key pair $(sk, pk)$, then it sends the public key $pk$ to $\mathcal{A}$.*
- Phase 1. *In this phase, $\mathcal{A}$ can query the decryption oracle adaptively multiple times. When $\mathcal{A}$ submits a ciphertext $CT$ to $\mathcal{C}$, $\mathcal{C}$ responds $\mathcal{A}$ with the origin message $m$ when $CT = Enc(param, pk, m)$, otherwise, it responds $\mathcal{A}$ with nothing.*
- Challenger. *When $\mathcal{A}$ decides to complete the* Phase 1*, it randomly chooses two messages $m_0$ and $m_1$ from the message space such that $|m_0| = |m_1|$, and then submits them to $\mathcal{C}$. Upon receiving the two messages, $\mathcal{C}$ randomly selects $b \in \{0, 1\}$ and computes $CT^* = \mathsf{Enc}(params, pk, m_b)$. Finally, $\mathcal{C}$ sends the challenge ciphertext $CT^*$ to $\mathcal{A}$.*
- Phase 2. *In this phase, $\mathcal{A}$ can still query the decryption oracle adaptively. While the only restriction is that $\mathcal{A}$ cannot query the challenge ciphertext $CT^*$ in this phase.*
- Guess. *$\mathcal{A}$ outputs its guess $b'$ on $b$ and wins the game if $b' = b$.*

*A PKE scheme is $(T, q, \epsilon(\ell))$-indistinguishable against adaptive chosen ciphertext attacks, or IND-CCA2 secure, if there exists no PPT adversary $\mathcal{A}$ making $q$ decryption queries which can win the aforementioned game with non-negligible advantage, that is,*

$$\mathsf{Adv}_{\mathcal{A}}^{IND-CCA2} = \left| \Pr[b' = b] - \frac{1}{2} \right| \leq \epsilon(\ell),$$

*where $\epsilon(\ell)$ is a negligible function with input $\ell$ and the advantage is taken over all the randomness consumed in the game.*

The *indistinguishability against adaptive chosen plaintex attacks* (IND-CPA) is another security notion for PKE which is weaker than IND-CCA2. Those two models are similar and the only difference between them is that $\mathcal{A}$ is not allowed to query the decryption oracle in the IND-CPA game. Here, we omit the description of the detail of this model and only give the conclusion.

**Definition 2.28 (IND-CPA secure)** *A PKE scheme is $(T, \epsilon(\ell))$-indistinguishable against adaptive chosen plaintext attacks, or IND-CPA secure, if there exists no PPT adversary $\mathcal{A}$ which can win the aforementioned IND-CPA game with non-negligible advantage, that is,*

$$\mathsf{Adv}_{\mathcal{A}}^{IND-CPA} = \left| \Pr[b' = b] - \frac{1}{2} \right| \leq \epsilon(\ell).$$

*Where $\mathcal{A}$'s advantage is taken over all the random bits consumed in the aforementioned game.*

### 2.4.6 Digital Signature

Similar to the handwritten signature which provides user authentication and non-repudiation in our daily life, the digital signature, first proposed by Diffie and Hellman in [DH76], can provide those properties in the network communication. Explicitly, one user can authenticate itself to anyone by issuing a valid digital signature on certain public message using its own signing key. Besides, when a signature on one message is generated already, the signer of that signature cannot deny this behavior forever.

**Definition 2.29 (Digital Signature)** *Formalized by [GMR88], a digital signature can be defined by the following four algorithms.*

- $\mathsf{Setup}(1^\ell)$. *On input $1^\ell$, this setup algorithm outputs the public parameters $params$.*
- $\mathsf{KeyGen}(1^\ell)$. *Taking $1^\ell$ as input, the key generation algorithm outputs a secret-public pair $(sk, pk)$.*
- $\mathsf{Sign}(params, sk, m)$. *The signature algorithm takes $params$, $sk$ and a message $m$ chosen from the message space as input, and outputs a signature $\sigma$ on $m$.*
- $\mathsf{Verify}(params, m, pk, \sigma)$. *On inputs $params, m, pk, \sigma$, the verification algorithm outputs $True$ if $\mathsf{Sign}(params, m, sk) \to \sigma$, otherwise, it outputs $False$.*

**Definition 2.30 (Correctness)** *We say that a digital signature is correct if*

$$\Pr\left[ \mathsf{Verify}(params, m, pk, \sigma) \to True \;\middle|\; \begin{array}{l} \mathsf{Setup}(1^\ell) \to params; \\ \mathsf{KeyGen}(1^\ell) \to (sk, pk); \\ \mathsf{Sign}(params, sk, m) \to \sigma. \end{array} \right] \geq 1 - \epsilon(\ell)$$

*and*

$$\Pr\left[\begin{array}{l|l} \mathsf{Verify}(params, m, pk, \sigma) \rightarrow False & \begin{array}{l}\mathsf{Setup}(1^\ell) \rightarrow params;\\ \mathsf{KeyGen}(1^\ell) \rightarrow (sk, pk);\\ \mathsf{Sign}(params, sk, m) \rightarrow \sigma.\end{array}\end{array}\right] < \epsilon(\ell),$$

*where $\epsilon(\ell)$ is a negligible function with input the security parameter $\ell$ and the probability is taken over all the random bits consumed in the scheme.*

**Definition 2.31 (EU-CMA)** *One basic security notion for the digital signature is existential unforgeability under adaptive chosen message attacks (EU-CMA) [GMR88]. This security model can be formally defined by the following game executed between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$.*

- Setup. *Taking the security parameter $\ell$ as input, $\mathcal{C}$ executes* $\mathsf{Setup}(1^\ell)$ *to generate the public parameters params, then it sends parames to $\mathcal{A}$.*
- KeyGen. *$\mathcal{C}$ also executes* $\mathsf{KeyGen}(1^\ell)$ *to generate a secret-public key pair $(sk, pk)$, then it sends the public key $pk$ to $\mathcal{A}$.*
- Query. *In this phase, $\mathcal{A}$ can query the signing oracle adaptively. When $\mathcal{A}$ queries $\mathcal{C}$ with the message $m$, $\mathcal{C}$ executes* $\mathsf{Sign}(params, sk, m)$ *to generate a signature $\sigma$ on $m$ and responds $\mathcal{A}$ with $\sigma$.*
- Output. *When $\mathcal{A}$ decides to complete the* Query *phase, it outputs a message-signature pair $(m^*, \sigma^*)$. $\mathcal{A}$ wins the game if* $\mathsf{Verify}(params, m^*, pk, \sigma^*) \rightarrow True$ *and $m^*$ has never appeared as queried message in the previous* Query *phase.*

*A digital signature scheme is $(T, q, \epsilon(\ell))$-existentially unforgeable against adaptive chosen message attacks, or EU-CMA secure, if there eixsts no PPT adversary $\mathcal{A}$ which can win the aforementioned game with non-negligible advantage, that is,*

$$\mathsf{Adv}_{\mathcal{A}}^{EU-CMA} = \Pr\left[\mathsf{Verify}(params, m^*, pk, \sigma^*) \rightarrow True\right] \leq \epsilon(\ell),$$

*where the advantage of $\mathcal{A}$ is taken over all the random bits consumed in the game.*

**Definition 2.32 (SEU-CMA)** *There also exists a more strong security notion named strongly existential unforgeability under an adaptive chosen message attack (SEU-CMA) for digital signature scheme. It is defined by the following game executed between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$.*

- Setup. *Taking the security parameter $\ell$ as input, $\mathcal{C}$ executes* $\mathsf{Setup}(1^\ell)$ *to generate the public parameters params, then it sends parames to $\mathcal{A}$.*

- KeyGen. *$\mathcal{C}$ also executes* $\mathsf{KeyGen}(1^\ell)$ *to generate a secret-public key pair* $(sk, pk)$*, then it sends the public key* $pk$ *to* $\mathcal{A}$*.*

- Query. *In this phase,* $\mathcal{A}$ *can still query the signing oracle adaptively. Assuming* $\mathcal{A}$ *queries the signing oracle* $q$ *times, each time when it sends a message* $m_i$ *chosen randomly from the message space to* $\mathcal{C}$*, it gets a signature* $\sigma_i$ *on* $m_i$ *as response for* $i = 1, 2, \cdots, q$*.*

- Output. $\mathcal{A}$ *outputs a message-signature pair* $(m^*, \sigma^*)$*.* $\mathcal{A}$ *wins the game if* $(m^*, \sigma^*) \notin \{(m_1, \sigma_1), (m_2, \sigma_2), \cdots, (m_q, \sigma_q)\}$ *and* $\mathsf{Verify}(params, m^*, pk, \sigma^*) \rightarrow True$*.*

*A digital signature scheme is* $(T, q, \epsilon(\ell))$*-strongly existentially unforeable against adaptive chosen message attacks, or SEU-CMA secure, if there exists no PPT adversary* $\mathcal{A}$ *which can win the above game with non-negligible advantage, that is,*

$$\mathsf{Adv}_{\mathcal{A}}^{SEU-CMA} = \Pr\left[\mathsf{Verify}(params, m^*, pk, \sigma^*) \rightarrow True\right] \leq \epsilon(\ell),$$

*where the advantage of* $\mathcal{A}$ *is taken over all the random bits consumed in the game.*

## 2.5 Chapter Summary

In this chapter, we introduced the fundamental knowledge in cryptography including basic notions, miscellaneous notations and general cryptographic tools which are widely used throughout this thesis. For readers who still feel confusing with the meaning of one certain notation or abbreviation when it is encountered in the following chapters, they are more recommended to refer to the List of Notations or List of Abbreviations part for a quick review. We will present our published works one by by from the next chapter.

# Part I

# Communication Schemes with User and Data Privacy

# Chapter 3

# A Privacy-Preserving Source Verifiable Encryption Scheme

It is critical to guarantee message confidentiality and user privacy in communication networks, especially for group communications. We find previous works seldom consider the two aspects at the same time and some trivial solutions cannot remain secure under strong security models. In order to address the aforementioned problem properly, we propose a privacy-preserving source-verifiable encryption scheme. With our scheme, the sender can convince anyone of its legitimation among a set of users chosen by itself without leaking its privacy. Moreover, only the intended receiver can retrieve the original message and the identity of the sender from a given ciphertext. Considering the security of our scheme, we define three security models which capture the message confidentiality, the user privacy and the user impersonation resistance respectively. We prove that our scheme maintains all the three aforementioned properties under the random oracles model.

## 3.1 Introduction

There are many practical network scenarios where content of messages and privacy of users should be protected concurrently during the communication. For example, in mobile ad hoc networks (MANETs) [RLL09], due to the mobility of communication nodes and the nature of wireless communications, user privacy and message confidentiality are essential requirements for mission critical communications. Another mobile scenario, where the above two security considerations should be taken into account, is the mobile phone sensing application [ZCZ16]. In order to provide customized services, a typical mobile sensing application may need to aggregate sensitive information from users for analysis. A simple example is the health-care sensing application which collects information including physical location, health in-

dices such as weight, heart rate and blood pressure from users. Obviously, protecting user privacy is the most important task for that application. Message confidentiality and user privacy issues also exist in data mining systems [Zha08] and the on-line navigation systems [CYHL14] during the user data collection stage. As shown above, a solution which can address the message confidentiality and user privacy simultaneously is desirable in many real-life applications.

Preventing content of messages from being eavesdropped or modified can be achieved using cryptographic tools such as encryption and digital signature. Also, there are cryptographic primitives that can provide user privacy properly, such as the ring signature [RST01], the group signature [ACJT00b], etc. It seems that our problem can be solved by simply combining two cryptographic primitives which provide the message confidentiality and user privacy respectively. However, below we present an example to illustrate that maintaining the message confidentiality and the user privacy at the same time is not a trivial task.

Assuming there is a ring signature scheme $\mathcal{RIN}$ and an IND-CCA2 secure encryption scheme $\mathcal{EN}$, where the signing and verification algorithms of the $\mathcal{RIN}$ are denoted by Sig and Ver respectively, the encryption and decryption algorithms of the $\mathcal{EN}$ are denoted by Enc and Dec respectively. Let the public key of the receiver be $pk$ and the signing key of user $U_i$ as $sk_i$, then user $U_i$ computes $c_1 = Enc_{pk}(m), c_2 = Sig_{sk_i}(c_1)$, and sends the message tuple $(c_1, c_2)$ to the receiver. According to the properties provided by $\mathcal{RIN}$ and $\mathcal{EN}$, any party within the group can compute a ring signature and anyone can check the validity of this ring signature without knowing the actual signer. In addition, it is hard for anyone to create a valid ring signature on any message for any group without knowing a secret key which belongs to a user of that group. It seems that this solution maintains the message confidentiality and user privacy properties. However, such a scheme cannot achieve message confidentiality in the IND-CCA2 model [BDPR98a]. When the challenge ciphertext $(c_1, c_2)$ is sent to the adversary, it can use another signing key $sk_j$ of user $U_j$ in the ring to sign $c_1$, which is the first component of the given challenge. That is, the adversary generates ${c_2}' = Sig_{sk_j}(c_1)$. The adversary then gets a new tuple $(c_1, {c_2}')$. When it provides this tuple to the decryption oracle, it can definitely guess which message is encrypted with probability 1 in the IND-CCA2 game. Hence, this solution cannot achieve IND-CCA2 security towards the message confidentiality. From the above example, we can say that simply combining two schemes with message confidentiality and user privacy cannot work.

### 3.1.1 Related Work

To solve user privacy problems in ad hoc groups, Dodis, Kiavias, Nocolosi and Shoup [DKNS04] proposed anonymous identification schemes in multi-user setting. Their schemes allow participants from a user population to form ad-hoc groups, and then prove membership anonymously in such groups. They also provided a formal model for their scheme and designed a generic scheme based on any accumulator with one-way domain as well as an efficient implementation of such accumulator based on the Strong RSA Assumption. Their anonymous identification schemes have some salient features, one of them is that their schemes can be generally and efficiently amended in order to allow the recovery of the signer's identity by an authority, if it is desired. Besides, by using the Fiat-Shamir transformation, they also obtained constant-size, signer-ambiguous group and ring signatures (provably secure in the random oracle model) from their identification schemes.

In Eurocrypt 2015, Groth and Kohlweiss [GK15] constructed one-out-of-mangy proofs to address the user privacy problem in multi-user environment. Their proof is actually a 3-move public coin special honest verifier zero-knowledge proof, or $\sum$-protocol, for a list of commitments having at least one commitment that opens to 0. It is not required for the prover to know openings of the other commitments. The proof system is efficient, particularly, in terms of communication requiring only the transmission of a logarithmic number of commitments. The authors used their proof system, by applying the Fiat-Shamir transformation, instantiate both ring signatures and zerocoin, a novel mechanism for bitcoin privacy. They used the proposed $\sum$-protocol as a linkable ad-hoc group identification scheme where the users have public keys where are indeed commitments and demonstrate knowledge of an opening for one of the commitments to unlinkably identify themselves (once).

Some more concrete solutions to the user privacy problem can be found in [RLL09, ZCZ16, Zha08, CYHL14]. In [RLL09], Ren et al. proposed a novel unconditionally secure source anonymous message authentication scheme (SAMAS) that enables messages to be released without relying on any trusted third parties. While providing source privacy, the proposed scheme also provided message content authenticity. The author then proposed a novel communication protocol for MANET that can ensure communication privacy of both communication parties and their end-to-end routing. For solving user privacy issues in mobile phone sensing, Zhang, Chen and Zhong [ZCZ16] presented an efficient protocol that allows an untrusted data aggregator to periodically collect sensed data from a group of mobile phone users without knowing which data belongs to which user. Assuming there are $n$ users in the group, their protocol achieved $n$-source anonymity in the sense that the aggregator only learns that the source of a piece of data is one of the $n$ users.

Besides, they also considered a practical scenario where users may have different source anonymity requirements and provided a solution based on dividing users into groups. Zhan [Zha08] provided solutions for privacy-preserving collaborative data mining problems, in particular, the author illustrated how to conduct privacy-preserving naive Bayesian classification which is one of the data mining tasks. In [CYHL14], Chim et al. made use of the idea of the anonymous credential to ensure that all driver's privacy cannot be breached.

We find the above works towards the user privacy problem seldom consider keeping the message confidentiality property at the same time. Besides, almost all the proposed solutions ensure no one in the system can compromise the users' privacy. Privacy-preserving solutions of this kind would incur problems in reality. One of the problems is that users can deny their previous behavior during the communication for nobody can identify them, moreover, as the message receiver cannot ascertain who is the actual sender, it is inconvenient for him to directly send his message back to the sender securely when a response is needed. From what we have discussed, we consider that the conditional user privacy-preserving property should be more realistic in real-life applications, which means that a message sender's privacy can only be revealed by the intended message receiver.

The cryptographic primitive verification encryption is often used to deal with privacy problems. After the notion of verifiable encryption was invented by Stadler [Sta96], many concrete schemes have been constructed [BFPV11, Ate04, CS03b, CD00, Bao00]. The verifiable encryption scheme can be used as a building block to solve many problems, such as [HM12, Fuc10], where the realization of practical revocable anonymous credentials using verifiable encryption was discussed. Also in [GDM02, PCS03, TV09], the authors used verifiable encryption to solve variants of the fair-exchange problem, and in [KPW97, CD00], verifiable encryption was applied to build separable group signatures and signature sharing schemes. The verifiable encryption can also be used in key escrow systems [Mao97] and file-sharing systems [HP10] to provide desirable properties.

However, we cannot derive a solution from a verifiable encryption scheme for the reason that, in a verifiable encryption scheme, we encrypt the identity of the user rather than the message which we want to keep absolutely confidential. Besides, when we extend the verifiable encryption into group setting by applying the one-out-of-many proof system [GK15], we need to consider the impersonation attack where an unauthorized user may masquerade as one member of the legitimated group.

### 3.1.2 Contribution

In this chapter, we make the following contributions.

1. To maintain message confidentiality and user privacy concurrently, we propose a privacy preserving source-verifiable encryption scheme. Our scheme provides conditional privacy for message encryptors, which means that the message encryptor's identity cannot be disclosed by other users except the intended receiver. We find this kind of user privacy is more practical in many real applications. Besides, a prover can prove its legitimation in a set of users chosen by itself. Our scheme is flexible and efficient when the size of the chosen set is small.

2. Further, we analyze the security of our scheme in detail. For message confidentiality, we prove our scheme is IND-CCA2 secure under the random oracle model. Besides, we also define the security models for the user privacy and impersonation resistance respectively, and prove that our scheme maintains all the aforementioned security properties under our models.

### 3.1.3 Chapter Organization

The rest of the chapter is organized as follows: In Section 3.2, we give the formal definition of our privacy-preserving source-verifiable encryption scheme, we also define three security models in this section for the purpose of proving the security of our scheme. Our concrete construction of the scheme is presented in detail in Section 3.3. In Section 3.4, we prove the security of our scheme under the previously defined models respectively. We also give a short discussion of a server-aided variant of our scheme in Section 3.5. At the end of this chapter, we make our conclusion and point out our future work.

## 3.2 Definitions and Security Models

**Definition 3.1 (Privacy-preserving Source-verifiable Encryption)** *Our privacy preserving source-verifiable encryption scheme, consisting of a list of polynomial time algorithms* (Setup, Gen, Enc, Ver, Dec)*, is described as follows.*

- Setup($1^k$): *On input* $1^k$*, it outputs a system parameters* PM*. As* PM *is regarded as default input to all the following algorithms, we omit it.*

- Gen($\cdot$)*: For a user* $U_i$*, he runs the key generation algorithm, on input* PM*, to get his unique identity* $ID_i$*, a secret* $s_i$ *and a public-private key pair* $(pk_i, sk_i)$*. Assuming all users' identities and public keys can be distributed properly among others in the group,* $U_i$ *would finally get a user identity set* $\mathcal{ID}$ *and a public key set* $\mathcal{PK}$*.*

- $\mathsf{Enc}(m, ID_i, s_i, pk_j, \mathcal{ID}_i)$*: For an encryptor who holds his own identity* $ID_i$ *and an identity set* $\mathcal{ID}$*, if he wants to send a message securely to* $U_j$*, he first chooses a subset* $\mathcal{ID}_i$ *from* $\mathcal{ID}$*, note that* $\mathcal{ID}_i$ *should include* $ID_i$ *and* $|\mathcal{ID}_i| \geq 2$*.* $U_i$ *encrypts a message* $m$ *chosen from the message space* $\mathcal{M}$ *by executing the* Enc *algorithm, which takes* $(m, ID_i, pk_j, \mathcal{ID}_i)$ *and* $ID_i$*'s secret* $s_i$ *as inputs. Finally, the encryptor gets the ciphertext* $ct$*.*

- $\mathsf{Ver}(ct)$*: Everyone can be a verifier in our scheme upon knowing* PM *and receiving a ciphertext* $ct$*. The verification algorithm* Ver *is deterministic, after the execution of it, a verifier outputs accept if* $ct$ *satisfies certain rules, otherwise, it outputs reject.*

- $\mathsf{Dec}(ct, sk_j)$*: The decryption algorithm should only be executed by the decrytor and is also deterministic. Before the decryptor retrieves* $m$ *and the encryptor's identity* $ID_i$ *from a given ciphertext* $ct$*, he first executes* Ver *to verify the validity of it, and only when* Ver *outputs accept, the decryptor then continues to decrypt* $ct$*.*

We require that a privacy-preserving source-verifiable encryption scheme should have the following three security properties: message confidentiality, user privacy and user impersonation resistance. In order to capture those requirements, we define the following three security models.

**Definition 3.2 (Modified IND-CCA2)** *Setting the security parameter as* $k$*, given our privacy-preserving source-verifiable encryption scheme, a polynomial* $n(\cdot)$*, a PPT adversary* $\mathcal{A}$ *and a challenger* $\mathcal{S}$*, let's consider the following game played between* $\mathcal{A}$ *and* $\mathcal{S}$*:*

- Setup*: First,* Setup*, which takes* $1^k$ *as input, is run by* $\mathcal{S}$ *to produce the system parameter* PM*. Given a polynomial* $n(\cdot)$*,* $\mathcal{S}$ *runs* Gen*, with* PM *as input,* $n(k)$ *times. After all executions are properly finished,* $\mathcal{S}$ *gets a public key set* $\mathcal{PK}$*, a private key set* $\mathcal{SK}$*, a user secret set* S *and an identity set* $\mathcal{ID}$*, where* $|\mathcal{PK}| = |\mathcal{SK}| = |\mathcal{ID}| = |\mathsf{S}| = n(k)$*. The adversary* $\mathcal{A}$ *is given* PM*,* $\mathcal{ID}$ *and* $\mathcal{PK}$*.*

- Corruption phase*: In order to make* $\mathcal{A}$ *more powerful, he is permitted to corrupt users from the identity set* $\mathcal{ID}$*. Namely,* $\mathcal{A}$ *can get the secret of a user after taking the identity of that user as the queried message.*

- Decryption phase 1*:* $\mathcal{A}$ *can also ask decryption queries adaptively to* $\mathcal{S}$*, when* $\mathcal{A}$ *provides* $\mathcal{S}$ *a valid ciphertext,* $\mathcal{S}$ *needs to return the corresponding plaintext of this ciphertext to* $\mathcal{A}$*.*

- Challenge phase: *$\mathcal{A}$ chooses two messages $m_0, m_1$ from $\mathcal{M}$, two identities $ID_i$, $ID_j$ from $\mathcal{ID}$ as the sender and receiver's identity respectively and a subset $\mathcal{ID}_i$ from $\mathcal{ID}$ such that $ID_i \in \mathcal{ID}_i, |\mathcal{ID}_i| \geq 2$. $\mathcal{A}$ then sends them to $\mathcal{S}$. Upon receiving those information, $\mathcal{S}$ randomly chooses a bit $b$ from $\{0,1\}$ and encrypts $m_b$ using the encryption algorithm of our scheme, which takes $m$, $ID_i$, secret $s_i$ of $ID_i$, $pk_j$, $\mathcal{ID}_i$ as inputs. The corresponding ciphertext is given to $\mathcal{A}$ as the challenge ciphertext.*

- Decryption phase 2: *After receiving the challenge ciphertext, $\mathcal{A}$ can still query the decryption oracle with the only restriction that the queried ciphertext must be different from the challenge one.*

- Guess phase: *At the end of the game, $\mathcal{A}$ outputs the guess $b'$ from $\{0,1\}$ about $b$. If $b' = b$, then $\mathcal{A}$ succeeds in the game, otherwise $\mathcal{A}$ fails.*

*Remark:* $\mathcal{A}$ is allowed to ask hash queries under the random oracle model.
According to the defined model, let $\mathsf{Adv}$ denote the probability that $\mathcal{A}$ wins the above game over random guess, then $\mathsf{Adv} = \left|\Pr\left[b' = b\right] - \frac{1}{2}\right|$.

**Definition 3.3 (User Privacy)** *Setting the security parameter as $k$, then given our privacy-preserving source-verifiable encryption scheme, a PPT adversary $\mathcal{A}$ and a challenger $\mathcal{S}$, let's consider the following game played by $\mathcal{A}$ and $\mathcal{S}$:*

- Setup phase: *First, the algorithm* Setup*, which takes $1^k$ as input, is run by $\mathcal{S}$ to produce a system parameter* PM*. Given a polynomial $n(\cdot)$, $\mathcal{S}$ runs* Gen*, with* PM *as input, $n(k)$ times. After all executions are properly finished, $\mathcal{S}$ gets a public key set $\mathcal{PK}$, a private key set $\mathcal{SK}$, a user secret set* s *and an identity set $\mathcal{ID}$, where $|\mathcal{PK}| = |\mathcal{SK}| = |\mathcal{ID}| = |\mathsf{s}| = n(k)$. The adversary $\mathcal{A}$ is given* PM*, $\mathcal{ID}$ and $\mathcal{PK}$.*

- Corruption phase: *In order to make $\mathcal{A}$ more powerful, he is permitted to corrupt users from the identity set $\mathcal{ID}$. Namely, $\mathcal{A}$ can get the secret of a user after taking the identity of that user as the queried message.*

- ID extraction phase 1: *When $\mathcal{A}$ makes such kind of query, he submits a ciphertext to $\mathcal{S}$, then he gets the identity of the original encryptor of the submitted ciphertext when the queried ciphertext is valid, otherwise, he gets nothing.*

- Challenge phase: *$\mathcal{A}$ chooses one message $m$, a subset $\mathcal{ID}_i$, an identity $ID_j \notin \mathcal{ID}_i$ as the receiver's identity and sends them to $\mathcal{S}$, $\mathcal{S}$ randomly chooses a index $inx$ from the indexes of the chosen subset $\mathcal{ID}_i$, and encrypts $m$ by taking $ID_{inx}$, $s_{inx}$, $\mathcal{PK}_j$ of $ID_j$ and $\mathcal{ID}_i$ as inputs. The corresponding ciphertext is given to $\mathcal{A}$.*

- ID extraction phase 2*: After receiving the challenge ciphertext, $\mathcal{A}$ can still ask ID extraction queries adaptively with the constraint that the queried ciphertext must not be identical to the challenge one.*

- Guess phase*: At the end of the game, $\mathcal{A}$ outputs his guess $inx'$ from the indexes of the chosen subset $\mathcal{ID}_i$ about $inx$. If $inx' = inx$, then $\mathcal{A}$ succeeds in the game, otherwise $\mathcal{A}$ fails.*

*Remark:* Under the random oracle model, $\mathcal{A}$ is allowed to ask hash queries.
According to the defined model, let Adv denote the probability that $\mathcal{A}$ wins the above game over random guess, then $\mathsf{Adv} = \left|\Pr\left[inx' = inx\right] - \frac{1}{|\mathcal{ID}_i|}\right|$.

**Definition 3.4 (User Impersonation Resistance)** *Setting the security parameter as $k$, then given our privacy-preserving source-verifiable encryption scheme, a polynomial $n(\cdot)$, a polynomial probabilistic time (PPT) adversary $\mathcal{A}$ and a challenger $\mathcal{S}$, let's consider the following impersonation game played by $\mathcal{A}$ and $\mathcal{S}$:*

- Setup phase*: First, the algorithm* Setup*, which takes $1^k$ as input, is run by $\mathcal{S}$ to produce a system parameter* PM*. Given a polynomial $n(\cdot)$, $\mathcal{S}$ runs* Gen*, with* PM *as input, $n(k)$ times. After all executions are properly finished, $\mathcal{S}$ gets a public key set $\mathcal{PK}$, a private key set $\mathcal{SK}$, a user secret set* s *and an identity set $\mathcal{ID}$, where $|\mathcal{PK}| = |\mathcal{SK}| = |\mathcal{ID}| = |\mathsf{s}| = n(k)$. The adversary $\mathcal{A}$ is given* PM*, $\mathcal{ID}$ and $\mathcal{PK}$.*

- Corruption phase*: In order to make $\mathcal{A}$ more powerful, he is permitted to corrupt users from the identity set $\mathcal{ID}$. Namely, $\mathcal{A}$ can get the secret of a user after taking the identity of that user as the queried message. Here let $\mathcal{CID}$ denote the corruption set.*

- Encryption query phase*: In this phase, we denote the uncorrupted user set as $\mathcal{UID}$, while $\mathcal{UID} = \mathcal{ID} - \mathcal{CID}$. The adversary $\mathcal{A}$ chooses a message $m$ from $\mathcal{M}$, two identities $ID_i$, $ID_j$ from $\mathcal{UID}$ as the sender and receiver's identity respectively and a subset $\mathcal{UID}'$ from $\mathcal{UID}$ such that $ID_i \in \mathcal{UID}', ID_j \notin \mathcal{UID}', |\mathcal{UID}'| \geq 2$, and then sends them to $\mathcal{S}$. After receiving those information, $\mathcal{S}$ takes $m, ID_i, s_i, \mathcal{PK}_j, \mathcal{UID}'$ as inputs of the* Enc *algorithm and sends the generated ciphertext $ct$ to $\mathcal{A}$.*

- Forgery phase*: In this phase, $\mathcal{A}$ chooses a message $m^*$, an identity $ID_j^*$ as the receiver and a subset $\mathcal{UID}^*$ of $\mathcal{UID}$, then it tries to forge a corresponding valid ciphertext $ct^*$. It is required that $(m^*, \mathcal{UID}^*)$ cannot appear in any previous encryption query.*

If the forgery produced by $\mathcal{A}$ in the forgery phase can be accepted by the verification algorithm of our scheme, then $\mathcal{A}$ wins this game. Let $\mathsf{Adv}$ denote the probability that $\mathcal{A}$ wins the predefined game, then $\mathsf{Adv} = \Pr[\mathsf{Ver}(ct^*) = 1]$.

## 3.3 Our Privacy-preserving Source-verifiable Encryption Scheme

With our scheme, only a group of legitimated users can encrypt the message taking the receiver's public key, its own secret and a chosen identity subset as inputs. Also this encryptor can prove his legitimation to others. Upon receiving the ciphertext, which includes a proof of the encrytor's identity, a verifier can verify the legitimation of the source of this ciphertext without decrypting it. Only the decryptor can retrieve the origin message and the identity of the user who encrypts this message from the ciphertext.

Setting the security parameter as $k$, we give a concrete construction of our privacy-preserving source-verifiable encryption scheme as follows:

- $\mathsf{Setup}(1^k)$: On input $1^k$, it produces a cyclic group $\mathbb{G}$ of large prime order $p$ with generator $g$. This algorithm also outputs a description of the message space $\mathcal{M} = \{0,1\}^q$ and a ciphertext space $\mathcal{C}$. $\mathbb{G}, p, g, \mathcal{M}, \mathcal{C}$ are considered as the system parameter $\mathsf{PM}$ and default inputs to all the following algorithms.

- $\mathsf{Gen}(\cdot)$: For one user $U_i$, when executing $\mathsf{Gen}(\cdot)$ which takes $1^k$ as input, he himself randomly chooses his own secret $s_i$ and private key $SK_i = x_i$ from $\mathbb{Z}_p$ respectively and keeps them unknown to others, $U_i$ then calculates $ID_i = g^{s_i}$ and $PK_i = y_i = g^{x_i}$. Assuming the identity and public key of each user can be distributed properly to all other users. Finally, $U_i$ gets an identity set $\mathcal{ID} = \{ID_1, \ldots, ID_n\}$ and a public key set $\mathcal{PK} = \{y_1, \ldots y_n\}$, where $n$ is the number of members in the legitimated group. Each time when a new member joins the group, $\mathcal{ID}$, $\mathcal{PK}$ would be updated. Our scheme also applies three collision-resistance hash functions: $H_1 : \{0,1\}^q \times \mathbb{G}^3 \to \mathbb{Z}_p$, $H_2 : \mathbb{G} \to \{0,1\}^q$, $H_3 : \{0,1\}^* \to \mathbb{Z}_p$, where q denotes the length of the message.

- $\mathsf{Enc}(m, s_i, y_j, \mathcal{ID}_i)$: When $U_i$ wants to send a message $m \in \mathcal{M}$ to $U_j$, he first chooses a identity subset $\mathcal{ID}_i$ from $\mathcal{ID}$. Note that $ID_i \in \mathcal{ID}_i, ID_j \notin \mathcal{ID}_i, |\mathcal{ID}_i| \geq 2$ should include his own identity. $U_i$ takes $m$, $s_i$, $PK_j = y_j$, $\mathcal{ID}_i$ as inputs and does the following calculations:

$$r_1 \xleftarrow{R} \mathbb{Z}_p, r_2 = H_1(m, g^{r_1}, g^{s_i}, y_j^{r_1}),$$
$$C_1 = g^{r_1}, C_2 = g^{r_2}, C_3 = y_j^{s_i} y_j^{r_2}, C_4 = m \oplus H_2(y_j^{r_1} y_j^{r_2}).$$

After $(C_1, C_2, C_3, C_4)$ is generated, $U_i$ executes the following procedures to generate a proof:

- $U_i$ chooses $w_i$ randomly from $\mathbb{Z}_p$ and sets $a_i = g^{w_i}$,$b_i = y_j^{w_i}$.
- For each identity, say $g^{s_t}$, in $\mathcal{ID}_i$ except $g^{s_i}$, $U_i$ chooses $c_t$, $z_t$ randomly from $\mathbb{Z}_p$ and sets $a_t = g^{z_t}(g^{s_t}C_2)^{c_t}$, $b_t = y_j^{z_t}(C_3)^{c_t}$.
- $U_i$ sets $c = H_3(\alpha_i, \beta_i, C_1, C_2, C_3, C_4)$, where $\alpha_i = (\dots, a_i, \dots, a_t, \dots)$, $\beta_i = (\dots, b_i, \dots, b_t, \dots)$ and $|\alpha_i| = |\beta_i| = |\mathcal{ID}_i|$.
- $U_i$ sets $c_i = c - \sum_{g^{s_t} \in \mathcal{ID}_i \text{ except } g^{s_i}} c_t$ and $z_i = w_i - c_i(s_i + r_2)$. $U_i$ keeps the tuple $(\{c_i\}, \{z_i\})$ where

$$\{c_i\} = (\dots, c_i, \dots, c_t, \dots), \{z_i\} = (\dots, z_i, \dots, z_t, \dots).$$

$U_i$ appends the identity of the receiver, $ID_j$, to $\mathcal{ID}_i$ as its last element, and then gets a new identity set $\mathcal{ID}_{ij}$. Eventually, $U_i$ gets the ciphertext $ct = (C_1, C_2, C_3, C_4, \{c_i\}, \{z_i\}, \mathcal{ID}_{ij})$.

- $\mathsf{Ver}(ct)$: A verifier executes the following verification algorithm to check the validity of a received ciphertext. In fact, everyone who holds the system parameter $\mathsf{PM}$ can be a verifier. Upon receiving a ciphertext $ct = (C_1, C_2, C_3, C_4, \{c_i\}, \{z_i\}, \mathcal{ID}_{ij})$, a verifier $\mathcal{V}$ does as follows:
  - $\mathcal{V}$ first gets the subset $\mathcal{ID}_i$ and the receiver's identity $ID_j$ from $\mathcal{ID}_{ij}$. As $\mathcal{V}$ knows the public key set $\mathcal{PK}$ and user identity set $\mathcal{ID}$, obviously, he knows the corresponding public key $y_j$ of $ID_j$, so he can re-compute $a_i = g^{z_i}(g^{s_i}C_2)^{c_i}$ as well as $b_i = y_j^{z_i}(C_3)^{c_i}$ from $\{c_i\}, \{z_i\}, C_2, C_3$ for each identity $g^{s_i} \in \mathcal{ID}_i$ to get the two sets $\alpha_i$, $\beta_i$.
  - $\mathcal{V}$ checks whether the equation $H(\alpha_i, \beta_i, C_1, C_2, C_3, C_4) = \sum_{c_i \in \{c_i\}} c_i$ holds.
  - If all the above checks are successfully completed, then $\mathcal{V}$ can make sure that the encryptor of the received ciphertext is a legitimated user. Otherwise, the verifier rejects the received ciphertext.
- $\mathsf{Dec}(ct, x_j)$: When given a ciphertext $ct = (C_1, C_2, C_3, C_4, \{c_i\}, \{z_i\}, \mathcal{ID}_{ij})$, one user can easily find out whether he is the intended receiver by checking the last identity in $\mathcal{ID}_{ij}$. $U_j$, after finding out he is the decryptor, would do as follows:
  - $U_j$ first executes the verification algorithm $\mathsf{Ver}$ to check whether the given ciphertext is generated by a legitimated user, if not, $U_j$ rejects it, otherwise $U_j$ continues.

- $U_j$ computes $w = C_3{}^{\frac{1}{x_j}}/C_2$ and checks whether $w$ is listed in $\mathcal{ID}_i$. If not, $U_j$ rejects the ciphertext, otherwise he continues.
- $U_j$ calculates $m' = C_4 \oplus H_2((C_1C_2)^{x_j})$ and then checks whether the equation $C_2 = g^{h_1(m', C_1, w, (C_1)^{x_j})}$ holds, it not, $U_j$ rejects the given ciphertext.

When all the above checks are successfully finished, $U_j$ finally outputs $w$ and $m'$ as the sender's identity and original message respectively.

## 3.4 The Security Proofs of Our Scheme

**Theorem 3.1** *Our privacy-preserving source-verifiable encryption scheme maintains message confidentiality under the previously defined modified IND-CCA2 model assuming the DDH problem is hard in $\mathbb{G}$ when hash functions $H_1, H_2, H_3$ are modeled as random oracles. Concretely, if there is an adversary $\mathcal{A}$ which can break our scheme with non-negligible probability $\epsilon$, supposing $\mathcal{A}$ makes at most $q_{H_1}$,$q_{H_2}$,$q_{H_3}$ queries to the $H_1, H_2, H_3$ hash oracles respectively, and $q_D$ queries to the decryption oracle, then we can construct another algorithm $\mathcal{B}$ that solves the DDH problem in $\mathbb{G}$ with advantage at least $\frac{1}{n}(1 - \frac{q_D}{2^k})\epsilon$, where $k$ is the security parameter and $n$ is a constant.*

*Proof.* We show how to construct an algorithm $\mathcal{B}$ that solves the DDH problem by interacting with an adversary $\mathcal{A}$ of our scheme under our predefined model.

- Setup phase: On input $1^k$, $\mathcal{B}$ runs the Setup algorithm of our scheme to produce system parameters PM which includes $\mathbb{G}, p, g, \mathcal{M}, \mathcal{C}$. $\mathcal{B}$ is given a DDH tuple $(g^a, g^b, Z)$. For a given polynomial $n(\cdot)$, set $n = n(k)$. $\mathcal{B}$ runs the key generation algorithm $Gen(\cdot)$ $n$ times, except that $\mathcal{B}$ sets $PK_j = g^a$ for a randomly chosen $j \in [1, n]$ and does not have the corresponding private key $x_j$. Namely, $\mathcal{B}$ gets an identity set $\mathcal{ID} = \{ID_1, \ldots, ID_n\}$, a user secret set $\mathsf{s} = \{s_1, \ldots, s_n\}$, a public key set $\mathcal{PK} = \{PK_1, \ldots, PK_n\}$ and a private key set $\mathcal{SK} = \{x_1, \ldots, x_{j-1}, x_{j+1}, \ldots, x_n\}$. $\mathcal{B}$ chooses three collision-resistance hash functions: $H_1 : \{0,1\}^q \times \mathbb{G}^3 \to \mathbb{Z}_p$, $H_2 : \mathbb{G} \to \{0,1\}^q$, $H_3 : \{0,1\}^* \to \mathbb{Z}_p$ and sends them to $\mathcal{A}$, $H_1, H_2, H_3$ are fully controlled by $\mathcal{B}$ and are modeled as random oracles. Finally, $\mathcal{B}$ gives $\mathcal{A}$ PM, $\mathcal{ID}$ and $\mathcal{PK}$.

- Corruption queries: To make $\mathcal{A}$ more powerful, we first allow $\mathcal{A}$ to corrupt some users of $\mathcal{ID}$. Namely, when $\mathcal{A}$ wants to corrupt a user $U_i$ of $\mathcal{ID}$ and sends $ID_i$ to $\mathcal{B}$, $\mathcal{B}$ needs to respond $\mathcal{A}$ with the corresponding secret $s_i$.

- $H_1$-queries: $\mathcal{A}$ can issue queries to the hash function $H_1$. In order to respond those queries, $\mathcal{B}$ keeps a hash list $H_1^{list}$.

When $\mathcal{A}$ asks the hash value of the $v$-th message tuple $< m_v, g^{r_{v1}}, g^{s_{vx}}, y_i^{r_{v1}} >$, where $g^{s_{vx}} \in \mathcal{ID}$ and $y_i \in \mathcal{PK}$. $\mathcal{B}$ checks whether this tuple has appeared before, :

- If yes, $\mathcal{B}$ responds $\mathcal{A}$ with the record $h_{1v} = H_1(m_v, g^{r_{v1}}, g^{s_{vx}}, y_i^{r_{v1}})$.
- Otherwise, $\mathcal{B}$ chooses a random value $h_{1v}$ from $\mathbb{Z}_p$, and sets this value as the hash of the queried message tuple, $\mathcal{B}$ responds $\mathcal{A}$ with $h_{1v}$ and adds this message and hash value pair to the $H_1^{list}$. Further, B fills the third column of this query row with value $y_i^{h_{1v}} y_i^{r_{v1}}$, B fills the forth column of this query row with the symbol $\Phi$, which denotes empty.

One row of $H_1^{list}$ should be like:

| $(m_v, g^{r_{v1}}, g^{s_{vx}}, y_i^{r_{v1}})$ | $h_{1v}$ | $y_i^{r_{v1}} y_i^{h_{1v}}$ | $\Phi$ |
|---|---|---|---|

where $v \leq q_{H_1}$.

- $H_2$-queries: $\mathcal{A}$ can also ask $H_2$ hash queries. To answer this kind of query, the algorithm $\mathcal{B}$ maintains a $H_2^{list}$ table which has two columns.
  When the $u$-th query $y_j^{r_{u2}} y_j^{r_{u1}}$ is made, $\mathcal{B}$ checks whether this tuple has appeared before:
  - If yes, $\mathcal{B}$ responds $\mathcal{A}$ with the corresponding record $h_{2j} = H_2(y_j^{r_{u2}} y_j^{s_{ux}})$.
  - Otherwise, $\mathcal{B}$ chooses a random value $h_{2u}$ from $\{0,1\}^q$, where $q$ is the length of the message in $\mathcal{M}$, and sets this value as the hash of the queried message tuple, $\mathcal{B}$ responds with $h_{2u}$ and adds this message and hash value pair to the $H_2^{list}$.

  One row of $H_2^{list}$ shows in Table 2:

| $y_j^{r_{u2}} y_j^{r_{u1}}$ | $h_{2u}$ |
|---|---|

  Where $u \leq q_{H_2}$.

  B updates another table $T_3$ from the $H_1^{list}$ and $H_2^{list}$ tables whenever $H_1^{list}$ or $H_2^{list}$ is updated. The process of updating $T_3$ is as follows:

  If there is a new row added to the $H_1^{list}$, we check the value $y_j^{r_{u2}} y_j^{r_{u1}}$ in the first column of the $H_2^{list}$ table from the first to the last row.
  - If the value $y_i^{r_{v1}} y_i^{h_{1v}}$ in the third column of this new row is not empty and there exists such a row in the $H_2^{list}$ that making the aforementioned two values equal, then we replace the corresponding $H_2$ hash value $\Phi$ in the $H_1^{list}$ with the corresponding $H_2$ value $h_{2u}$ in the $H_2^{list}$. Further, we add

such a row to the $T_3$ table, and then delete the corresponding row from the $H_1^{list}$ and also the $H_2^{list}$.

– Otherwise, the three tables keep unchanged.

If there is a new row added to the $H_2^{list}$, we check the value $y_i^{r_{v1}} y_i^{h_{1v}}$ in the third column of the $H_1^{list}$ from the first to the last row.

– If $y_j^{r_{u2}} y_j^{r_{u1}}$ in the first column of the new row is not empty, and there exists such a row in $H_1^{list}$ that making the aforementioned two values equal, then we replace the corresponding $H_2$ hash value $\Phi$ in the $H_1^{list}$ with the corresponding $H_2$ value $h_{2u}$ in the $H_2^{list}$. Further, we add such a row to the $T_3$ table, and then delete the corresponding row from the $H_1^{list}$ and also the $H_2^{list}$.

– Otherwise, the three tables keep unchanged.

| ..... | ..... | ..... | ..... |
|---|---|---|---|
| $(m_u, g^{r_{u1}}, g^{s_{ux}}, y_i^{r_{u1}})$ | $h_{1u}$ | $y_i^{r_{u1}} y_i^{h_{1u}}$ | $h_{2u}$ |
| ..... | ..... | ..... | ..... |
| $(m_f, g^{r_{f1}}, g^{s_{fx}}, y_j^{r_{f1}})$ | $h_{1f}$ | $y_j^{r_{f1}} y_j^{h_{1f}}$ | $h_{2f}$ |

We can maintain such a table $T_3$ as below, where $y_i, y_j \in \mathcal{PK}$ and $g^{s_{ux}}, g^{s_{fx}} \in \mathcal{ID}$. We denote the size of $T_3$ as $|T_3|$
As the $H_1$ query and $H_2$ query can be asked in an interleaving manner, so, each time when $H_1^{list}$ or $H_2^{list}$ is updated, $T_3$ would be updated by executing the previous procedures.

- $H_3$- queries: $\mathcal{A}$ can ask $H_3$ queries. $\mathcal{B}$ creates a hash table $H_3^{list}$ to respond this kind of query. For the t-th query tuple $(\alpha_t, \beta_t, C_{1t}, C_{2t}, C_{3t}, C_{4t})$, $\mathcal{B}$ acts as following:

  – $\mathcal{B}$ first check whether this tuple has appeared before, if yes, $\mathcal{B}$ responds with the existing value $h_{3t} = H_3(\alpha_t, \beta_t, C_{1t}, C_{2t}, C_{3t}, C_{4t})$ to $\mathcal{A}$.

  – Otherwise, $\mathcal{B}$ chooses a random value $h_{3t}$ from $\mathbb{Z}_p$ , and sets it as the hash value of the queried message tuple. $\mathcal{B}$ responds $\mathcal{A}$ with $h_{3t}$ and adds this message and hash value pair to the $H_3^{list}$.

- Decryption queries phase 1: $\mathcal{A}$ can make decryption queries adaptively to $\mathcal{B}$. If a queried cihpertext received by $\mathcal{B}$ is $(C_{1w}, C_{2w}, C_{3w}, C_{4w}, \{c_i\}_w, \{z_i\}_w, \mathcal{ID}_{ij})$, $\mathcal{B}$ would first check whether the intended receiver is the user of which $\mathcal{B}$ does not know the private key, here the user with identity $ID_j$, by extracting the receiver's identity from $\mathcal{ID}_{ij}$:

– If no, $\mathcal{B}$ knows the receiver's private key, he can decrypt the given ciphertext by using the Dec algorithm and return the corresponding plaintext to $\mathcal{A}$.

– If yes, as $\mathcal{B}$ does not know the receiver's private key, $\mathcal{B}$ would use $T_3$ and $H_3^{list}$ to simulate the decryption process. $\mathcal{B}$ first executes the verification algorithm Ver of our scheme to recompute the sets $(\alpha_w, \beta_w)$. Then $\mathcal{B}$ does the following steps:
*Step 1.*
$\mathcal{B}$ checks:

∗ whether the tuple $(\alpha_w, \beta_w, C_{1w}, C_{2w}, C_{3w}, C_{4w})$ has been asked in the $H_3$ phase;

∗ whether the $H_3$ hash value of $(\alpha_w, \beta_w, C_{1w}, C_{2w}, C_{3w}, C_{4w})$ equals to the sum of the set $\{c_i\}_w$, that is $h_{3w} = \sum_{c_i \in \{c_i\}} c_i$

If either of the above two checks fails, $\mathcal{B}$ rejects. Otherwise, $\mathcal{B}$ turns to the following step,
*Step 2.*
$\mathcal{B}$ checks each row of the $T_3$ table and tests whether there exists an i-th row that can make the following four equations hold.

$$C_{1w} = g^{r_{i1}}, C_{3w} = y_j^{s_{ix}} y_j^{h_{1i}}, C_{4w} \oplus m_i = h_{2i}, C_{2w} = g^{h_{1i}}.$$

if $\mathcal{B}$ can find such a row satisfying the above equations, $\mathcal{B}$ outputs the message $m_i$ which can be found in the first column of the $T_3$ table. Otherwise, $\mathcal{B}$ rejects this query.

*Remark:* We use the component $(C_{2w}, C_{3w}, \{c_i\}_w, \{z_i\}_w)$ of the ciphertext in *step 1* checking, and the component $(C_{1w}, C_{2w}, C_{3w}, C_{4w})$ in *step 2* checking, so we use all parts of the given ciphertest to simulate the decryption process when we do not know the intended receiver's private key.

*Remark:* We should note that, when the intended receiver's private key is unknown to us, there may be some valid ciphertexts which would be rejected by our aforementioned decryption simulator. Here a valid ciphertext means, when it appears, the decryptor of our scheme can decrypt it correctly and return a valid message, while our decryption simulator cannot.

Assuming there are $q_D$ decryption queries asked during the decryption phase 1 and decryption phase 2. According to the encryption algorithm of our scheme, a ciphertext is not valid until it is generated after querying all the three hash

functions, and obviously, this kind of ciphertext can definitely be decrypted by our decryption simulator correctly. However, if at least one of the three hash functions is not asked when producing a ciphertext, this ciphertext may still have probability to be a valid one while our decryption simulator would reject it, the probability that it is still a valid ciphertext is at most $\frac{1}{2^k}$, where $k$ is the security parameter. Let us consider the event that at least one of the $q_D$ queried ciphertexts is valid but is rejected by our decryption simulator. Let symbol $fail$ denotes this event and symbol $r(i)$ denote the event that the i-th queried ciphertext is rejected but actually is a valid one, where $1 \leq i \leq q_D$. Then the probability of this event, $Pr[fail]$, is:

$$\begin{aligned} \Pr[fail] &= \Pr[r(1) \cup r(2) \cup, \ldots, \cup r(q_D)] \\ &= \leq \Pr[r(1)] + \Pr[r(2)] + \Pr[r(3)] + .... + \Pr[r(q_D)] \\ &= \frac{q_D}{2^k} \end{aligned}$$

Namely, with probability at most $\frac{q_D}{2^k}$, the decryption simulator would reject valid ciphertext(s). That is, with probability at least $1 - \frac{q_D}{2^k}$, $\mathcal{B}$ would do a perfection simulation in the decryption phases.

- Challenge phase: After the decryption queries are properly answered, $\mathcal{A}$ chooses two messages $m_0, m_1$ from $\mathcal{M}$, two identities $ID_i$, $ID_j$ from $\mathcal{ID}$ as the sender and receiver's identity respectively and a subset $\mathcal{ID}_i$ from $\mathcal{ID}$ such that $ID_i \in \mathcal{ID}_i, |\mathcal{ID}_i| \geq 2$, then sends them to $\mathcal{B}$. Upon receiving those information, $\mathcal{B}$ would first check whether the receiver's identity is $ID_j$, that is, the one $\mathcal{B}$ does not know the corresponding private key. If not, $\mathcal{B}$ aborts the game and outputs a random bit, otherwise $\mathcal{B}$ randomly chooses $c \in \{0, 1\}$ and encrypts $m_c$ using the Enc algorithm of our scheme. Namely, $\mathcal{B}$ asks $H_1$ query about the message tuple $(m_c, g^b, g^{s_i}, Z)$ to get $r_2^*$ , asks $H_2$ query about the message tuple $Zy_j^{r_2^*}$ to get $h_2$, asks $H_3$ query for the purpose of generating the proof tuple $(\{c_i\}^*, \{z_i\}^*)$, and sets the ciphertext as $(g^b, g^{r_2^*}, y_j^{s_i} y_j^{r_2^*}, m_c \oplus h_2, \{c_i\}^*, \{z_i\}^*, \mathcal{ID}_{ij}^*)$, where $h_2 = H_2(Zy_j^{r_2^*}), r_2^* = H_1(m_c, g^b, g^{s_i}, Z)$. $\mathcal{B}$ sends the generated ciphertext to $\mathcal{A}$ as the challenge ciphertext. Then $\mathcal{B}$ adds the message tuple $((m_c, g^b, y^A, Z), r_2^*, Zy^{r_2^*}, h_{2*})$ as a row to the $T_3$ table.

- Decryption queries phase 2: In this phase, $\mathcal{A}$ can still ask decryption queries with the only constraint that $\mathcal{A}$ cannot use the challenge ciphertext as one of his queried messages. $\mathcal{B}$ answers the decryption queries using the same procedures stated in the previous decryption queries phase 1.

- Guess phase: After the decryption queries phase 2 is finished, $\mathcal{A}$ would make

a guess $c' \in \{0,1\}$ about $c$, and sends his guess to $\mathcal{B}$. $\mathcal{B}$ outputs 1 if and only if $c = c'$, otherwise, it outputs 0.

Let's consider the probability that our algorithm $\mathcal{B}$ would output 1, that is, $\mathcal{A}$ succeeds in the previous game. We analyze this probability under the following two different cases:

1. When $Z \neq g^{ab}$, easily, the probability $\Pr[c = c']$ in this case should be $\frac{1}{2}$ for the reason that the challenger ciphertext is a random element from the view of $\mathcal{A}$, $\mathcal{A}$ cannot get any useful information from it.

2. When $Z = g^{ab}$ and also the intended receiver's identity is $ID_j$, we find that the challenge ciphertext produced by $\mathcal{B}$ is valid. According to our previous analysis, our simulator would reject valid given ciphertext(s) with probability at most $\frac{q_D}{2^k}$ during simulating the decryption process, which means that with probability at least $1-\frac{q_D}{2^k}$, $\mathcal{B}$ would do a perfect simulation during the previous game played by $\mathcal{A}$ and $\mathcal{B}$. Also, with probability $\frac{1}{n}$, $\mathcal{A}$ would choose $ID_j$ as the receiver's identity. As $\mathcal{A}$ can break our scheme with non-negligible probability $\epsilon$, the challenge ciphertext is valid when the given tuple is a DDH tuple and the receiver's identity is $ID_j$. In this case, the probability $\Pr[c = c']$ should be at least $\frac{1}{2} + \frac{1}{n}(1 - \frac{q_D}{2^k})\epsilon$.

As $\epsilon$ is non-negligible, $\mathcal{B}$ can solve the DDH problem with advantage at least $\frac{1}{n}(1 - \frac{q_D}{2^k})\epsilon$.

**Theorem 3.2** *Our privacy-preserving source-verifiable encryption scheme holds user privacy under the previously defined model assuming the DDH problem is hard in $\mathbb{G}$ when hash functions $H_1, H_2, H_3$ are modeled as random oracles. Concretely, if there exists such an adversary $\mathcal{A}$ which can break our scheme with non-negligible probability $\epsilon$, supposing $\mathcal{A}$ makes at most $q_{H_1}$,$q_{H_2}$,$q_{H_3}$ queries to the $H_1, H_2, H_3$ hash oracles respectively, and $q_{ID}$ ID extraction queries, then we can construct another algorithm $\mathcal{B}$ that can solve the DDH problem in $\mathbb{G}$ with probability at least $\frac{1}{n}(1-\frac{q_{ID}}{2^k})\epsilon$, where n is a constant.*

*Proof.* We show how to construct an algorithm $\mathcal{B}$ that solves the DDH problem by interacting with $\mathcal{A}$ under our predefined model. For the sake of simplifying the description of this proof, we omit the procedures identical to those in the previous security proof.

- Setup phase: This phase is the same as that in the previous security proof.

- Corruption queries: This phase is also identical to the corruption phase in the previous security proof, let $\mathcal{CID}$ denote the set including all the identities of the corrupted users.

- $H_1$,$H_2$,$H_3$-queries: When answering those three hash queries, $\mathcal{B}$ does the same as in the previous message confidentiality proof, so we omit them.

- ID extraction queries phase 1: This phase is almost the same as the decryption queries phase 1 described in the previous message confidentiality security proof except that $\mathcal{B}$ returns the identity of the user who generates the given ciphertext. From what we have discussed before, if $\mathcal{B}$ knows the private key of the receiver of a given ciphertext, he can definitely return the encryptor's identity to $\mathcal{A}$, when $\mathcal{B}$ does not know the receiver's private key, he can still find the encryptor's identity because our simulated decryption process can return plaintext and identity of the encryptor of a given ciphertext. In this phase, we still need to calculate the probability that $\mathcal{B}$ would make a perfect decryption simulation. From the previous calculation in the proof of the message confidentiality, we know that this probability is at least $1 - \frac{q_{ID}}{2^k}$.

- Challenge phase: After the ID extraction queries are properly answered, $\mathcal{A}$ chooses one message $m$, an identity $U_j$ as the receiver's identity and a subset $\mathcal{SID}$ of the set $\mathcal{ID}$, assuming the number of elements in $\mathcal{SID}$ is $u$. $\mathcal{A}$ sends them to $\mathcal{B}$, $\mathcal{B}$ randomly chooses a index $inx$ from the indexes of the chosen subset $\mathcal{SID}$, and encrypts $m$ using the public key of $U_j$ and the secret of the sender who has the chosen index $inx$. $\mathcal{B}$ asks $H_1$ query about the message tuple $(m, g^b, g^{s_{inx}}, Z)$ to get $r_{2^*}$, asks $H_2$ query about the message tuple $Zy_j^{r_{2^*}}$ to get $h_{2^*}$. and encrypts $m$ as $(g^b, g^{r_{2^*}}, y_j^{s_{inx}} y_j^{r_{2^*}}, m \oplus h_{2^*})$, $\mathcal{B}$ asks the $H_3$ function for the purpose of generating the proving tuple $(\{c_i\}^*, \{z_i\}^*)$, $\mathcal{B}$ appends $ID_j$ to $\mathcal{SID}$ as its last element and gets a new set $\mathcal{SID}^*$. We represent the generated tuple as $(C_1^*, C_2^*, C_3^*, C_4^*, \{c_i\}^*, \{z_i\}^*, \mathcal{SID}^*)$ for $\mathcal{A}$. Then $\mathcal{B}$ adds the message tuple $((m, g^b, g^{s_{inx}}, Z), r_{2^*}, Zy^{r_{2^*}}, h_{2^*})$ as a row to the $T_3$ table.

- ID extraction queries phase 2: In this phase, the algorithm $\mathcal{B}$ interacts with $\mathcal{A}$ in the same way as we described in the decryption queries phase 2 of the previous security proof.

- Guess phase: After finishing the ID extraction phase 2, $\mathcal{A}$ would make a random index guess $inx'$ from the indexes of the chosen subset about the challenge index $inx$, and sends his guess to $\mathcal{B}$. Algorithm $\mathcal{B}$ outputs 1 if and only if $inx = inx'$, otherwise, it outputs 0.

Let's consider the probability that our algorithm $\mathcal{B}$ would output 1, that is, $\mathcal{A}$ succeeds in the previous game. We assume the chosen subset $\mathcal{SID}$ has $u$ identities, we analyze this probability under the following two different cases:

1. When $Z \neq g^{ab}$, the probability $\Pr[inx = inx']$ in this case should be $\frac{1}{u}$ for the reason that the challenger ciphertext is random from the view of $\mathcal{A}$, $\mathcal{A}$ cannot

get any useful information from it, the best choice for him is to make a random guess.

2. When $Z = g^{ab}$ and also the intended receiver's identity is $ID_j$, we can find that the challenge ciphertext produced by $\mathcal{B}$ is valid. According to our previous analysis, our simulator would reject valid given ciphertext(s) with probability at most $\frac{q_{ID}}{2^k}$ during simulating the ID extraction process, which means that with probability at least $1 - \frac{q_{ID}}{2^k}$, $\mathcal{B}$ would do a perfect simulation during the previous game played by $\mathcal{A}$ and $\mathcal{B}$. Also, with probability $\frac{1}{n}$, $\mathcal{A}$ would choose $ID_j$ as the receiver's identity. As $\mathcal{A}$ can break our scheme with non-negligible probability $\epsilon$, the challenge ciphertext is valid when the given tuple is a DDH tuple and the receiver's identity is $ID_j$, in this condition, the probability $\Pr[inx = inx']$ should be at least $\frac{1}{u} + \frac{1}{n}(1 - \frac{q_{ID}}{2^k})\epsilon$

As $\epsilon$ is non-negligible, $\mathcal{B}$ can determine whether the given tuple is a valid Diffie–Hellman tuple with advantage at least $\frac{1}{n}(1 - \frac{q_{ID}}{2^k})\epsilon$.

**Theorem 3.3** *Our privacy-preserving source-verifiable encryption scheme has user impersonation resistance under the previously defined security model assuming the DL problem is hard in $\mathbb{G}$. That is, if there is an adversary $\mathcal{A}$ which can break our scheme with non-negligible probability $\epsilon$, then we can construct another algorithm $\mathcal{B}$ to break the DL problem successfully with non-negligible probability $(\varepsilon - \frac{1}{p})^2 \cdot \frac{1}{n}$, where $p$ is the order of group $\mathbb{G}$ and $n$ is a constant.*

*Proof.* We show how to construct an algorithms $\mathcal{B}$ that solves the DL problem in $\mathbb{G}$ by interacting with the adversary $\mathcal{A}$ under our previously defined model.

- Setup phase: On input $1^k$, $\mathcal{B}$ runs the Setup algorithm of our scheme to produce system parameters PM which includes $\mathbb{G}, p, g, \mathcal{M}, \mathcal{C}$. $\mathcal{B}$ is given a DL problem instance $(g, g^a)$. For a given polynomial $n(\cdot)$, set $n = n(k)$. $\mathcal{B}$ runs the key generation algorithm $Gen(\cdot)$ $n$ times, except that $\mathcal{B}$ sets $ID_j = g^a$ for a randomly chosen $j \in [1, n]$ and does not have the corresponding user secret $s_j$. Namely, $\mathcal{B}$ gets an identity set $\mathcal{ID} = \{ID_1, \ldots, ID_n\}$, a user secret set $\mathsf{s} = \{s_1, \ldots, s_{j-1}, s_j, \ldots, s_n\}$, a public key set $\mathcal{PK} = \{PK_1, \ldots, PK_n\}$ and a private key set $\mathcal{SK} = \{x_1, \ldots, x_n\}$. $\mathcal{B}$ chooses three collision-resistance hash functions: $H_1 : \{0,1\}^q \times \mathbb{G}^3 \to \mathbb{Z}_p$, $H_2 : \mathbb{G} \to \{0,1\}^q$, $H_3 : \{0,1\}^* \to \mathbb{Z}_p$ and sends them to $\mathcal{A}$, $H_1, H_2, H_3$ are fully controlled by $\mathcal{B}$ and are modeled as random oracles. Finally, $\mathcal{B}$ gives $\mathcal{A}$ PM, $\mathcal{ID}$ and $\mathcal{PK}$.

- Corruption phase: In order to make $\mathcal{A}$ more powerful, $\mathcal{A}$ is permitted to corrupt some users of the given set $\mathcal{ID}$ with the only restriction that $\mathcal{A}$ cannot corrupt the user with the identity $g^a$. Namely, $\mathcal{A}$ can get the secret of certain user

after submitting the identity of that user as the queried message. However, when the queried message is $g^a$, $\mathcal{B}$ abort. Let $\mathcal{CID}$ denote the set containing all the identities of the corrupted users, the uncorrupted user set $\mathcal{UID}$ can be expressed as $\mathcal{UID} = \mathcal{ID} - \mathcal{CID}$.

- Encryption queries phase: In this phase, $\mathcal{A}$ can issue encryption queries for multiple, say $q_e$, times. For each query, $\mathcal{A}$ randomly chooses a message $m$, a subset $\mathcal{SID}$ of the $\mathcal{UID}$ and a identity $g^{s_u}$ from $\mathcal{SID}$, and submits them to $\mathcal{B}$, $\mathcal{B}$ can then do the encryption for $\mathcal{A}$ and return the corresponding ciphertext to $\mathcal{A}$. We should notice that when $\mathcal{B}$ do the encryption, the three hash functions $H_1, H_2, H_3$ act the same as what we have described in our message confidentiality proof. During this phase, we consider the following two cases:

  - When $g^{s_u} \neq g^a$, as $\mathcal{B}$ knows the corresponding secret $s_x$ of the chosen identity, he can choose $ID_v$ satisfying $ID_v \in \mathcal{UID}, ID_v \notin \mathcal{SID}$ as the receiver's identity and easily generate a ciphertext $(C_1, C_2, C_3, C_4, \{c_i\}, \{z_i\}, \mathcal{SID}')$ and send it to the adversary $\mathcal{A}$ as the response.
  - When $g^{s_u} = g^a$, the given message is $m$, in order to respond this query, $\mathcal{B}$ chooses an identity $ID_v$ satisfying $ID_v \in \mathcal{UID}, ID_v \notin \mathcal{SID}$ as the receiver's identity, and then does the following simulation:

$$
\begin{aligned}
&r_1 \xleftarrow{R} G,\\
&r_2 = H_1(m, g^{s_u}, g^{r_1}, y^{r_1}),\\
&C_1 = g^{r_1},\\
&C_2 = g^{r_2},\\
&C_3 = (g^{s_u} g^{r_2})^{x_v},\\
&C_4 = m \oplus H_2((g^{r_1} g^{r_2})^{x_v}).
\end{aligned}
$$

    After $(C_1, C_2, C_3, C_4)$ is generated, $\mathcal{B}$ executes the following procedures with the inputs $(C_1, C_2, C_3, C_4)$ and the system parameters PM to simulate a proof for this ciphertext.

    * For each identity $g^{s_i}$ in $\mathcal{SID}$, $\mathcal{B}$ chooses $c_i$,$z_i$ randomly from $\mathbb{Z}_p$ and sets $a_i = g^{z_i}(g^{A_i} C_2)^{c_i}$, $b_i = y^{z_i}(C_3)^{c_i}$, $\mathcal{B}$ gets four sets, $\{a_i\} = (\ldots, a_i, \ldots, a_u, \ldots)$,$\{b_i\} = (\ldots, b_i, \ldots, b_u, \ldots)$, $\{c_i = (\ldots, c_i, \ldots, c_u, \ldots)$, $\{z_i\} = (\ldots, z_i, \ldots, z_u, \ldots)$
    * $\mathcal{B}$ sets $H_3(\{a_i\}, \{b_i\}, C_1, C_2, C_3, C_4) = \sum_{c_i \in \{c_i\}} c_i$.
    * $\mathcal{B}$ returns $(\{c_i\}, \{z_i\})$ as the proof value.

    $\mathcal{B}$ appends $ID_v$ to $\mathcal{SID}$ as its last element and still denotes this new user set as $\mathcal{SID}$. Eventually, $\mathcal{B}$ sends $(C_1, C_2, C_3, C_4, \{c_i\}, \{z_i\}, \mathcal{SID})$ to $\mathcal{A}$ as

the response. We can easily find that such a cipher-text can pass the verification algorithm of our scheme.

- Forgery phase: In this phase, $\mathcal{A}$ chooses a message $m'$ and a subset $\mathcal{UID}'$ of the set $\mathcal{UID}$, and try to forge a ciphertext of the chosen message and chosen subset. Let $(C_1', C_2', C_3', C_4', \{c_i\}', \{z_i\}', \mathcal{UID}')$ be the forgery, then $\mathcal{A}$ sends this forgery, $m'$ and the subset $\mathcal{UIDS}'$ to $\mathcal{B}$.

We assume this forgery can be accepted by the verification algorithm of our scheme with probability $\epsilon$. From the construction of our scheme, we can find that the corresponding proof value $(\{c_i\}', \{z_i\}')$ can be regarded as the signature of the tuple $(C_1', C_2', C_3', C_4')$.
We consider the adversary algorithm $\mathcal{A}$ as a turning machine with random tape $R'$. Now, for algorithm $\mathcal{B}$, $\mathcal{B}$ rewinds the algorithm $\mathcal{A}$ with the same random tape $R'$, that is, in the second time, $\mathcal{B}$ would do the same as $\mathcal{A}$ until the $H_3$ query is asked by $\mathcal{A}$ to generate the proof value in the forgery phase. According to the *Forking Lemma* [PS96], if we give another different $H_3$ response to $\mathcal{A}$, $\mathcal{A}$ would generate another valid tuple of proof values $(\{c_i\}^*, \{z_i\}^*)$ with non-negligible probability. As the same random tape $R'$ is used in the two rounds until the $H_3$ response, the generated ciphertexts should be the same after the execution of the two rounds. That is, in the first round, the final forgery is $(C_1', C_2', C_3', C_4', \{c_i\}', \{z_i\}', \mathcal{UID}')$, and in the second round, $(C_1', C_2', C_3', C_4', \{c_i\}^*,$
$\{z_i\}^*, \mathcal{UID}')$. Let $|\{c_i\}'| = |\{c_i\}^*| = |\{z_i\}'| = |\{z_i\}^*| = |\mathcal{UID}'| = l$
From the conclusion given in [PS96], if the probability that $\mathcal{A}$ could make a correct forgery is $\epsilon$, then the probability that the *forking lemma* executes successfully should be large than $(\varepsilon - \frac{1}{p})^2$, where $p$ is the order of the chosen group $\mathbb{G}$.
If the *Forking Lemma* executes successfully, that is, the two proof tuples $(\{c_i\}', \{z_i\}')$ and $(\{c_i\}^*, \{z_i\}^*)$ are both valid toward a certain identity in the chosen subset $\mathcal{UID}'$. As $H_3' = \sum c_i', H_3^* = \sum c_i^*, H_3' \neq H_3^*$, definitely, there exists at least one index $i$ such that $c_i' \neq c_i^*$.
With probability $\frac{1}{n}$, we have $ID_j \in \mathcal{UID}'$ and $c_j' \neq c_j^*$, then we have

$$z_j' = w - c_j'(a + r_{2j})$$
$$z_j^* = w - c_j^*(a + r_{2j})$$

In this case, $\mathcal{B}$ computes $a = (z_j^* - z_j')(c_j' - c_j^*)^{-1} - r_{2j}$, where $r_{2j}$ can be found in the table $H_1^{list}$. So the probability that $\mathcal{B}$ can solve the $DL$ problem is at least $(\varepsilon - \frac{1}{p})^2 \frac{1}{n}$.

## 3.5 The Server-aided Variant of Our Scheme

According to our scheme, every involved parties needs to handle amount of modular exponentiation computations when executing the specified algorithms, which restrains the scheme from being used by nodes with low capability. In order to make our scheme more applicable to resource-constraint devices to which costly computations such as modular exponentiations are unaffordable, we utilize the idea of *cut-and-choose technique* and modify the scheme to a server-aided one. For the ease of describing our server-aided variant, we first propose a server-aided modular exponentiation algorithm, our server-aided scheme simply takes the server-aided modular exponentiation algorithm as subroutine.

### 3.5.1 The Server-aided Modular Exponentiation Scheme

Before describing our scheme, we assume the server is honest but curious. That is, the sever behaves honest when do what the client tells it to do, but it also wants to get more information than what the client gives it. Our server-aided modular exponentiation scheme involves two parties, a client and server precisely, and contains the following four polynomial time algorithms;

- SysSetup: Given a security parameter $k$ as input, this algorithm outputs a cyclic group $\mathbb{G}$ with large prime order $q$, where $\mathbb{G}$ is a subgroup of $\mathbb{Z}_p^*$, $p$ is a secure prime and $q|p-1$.

- CPreCom: When a client is asked to compute $U^a$, where $U \in \mathbb{G}$ and $a \in \mathbb{Z}$, he first randomly chooses $\lambda$ elements $r_1, r_2, \ldots, r_\lambda \in \mathbb{Z}_p$. For each element $r_i$ in set $\{r_i\}$, the client chooses $l \to \{0,1,2,3,4,5\}$, if $l = 0$, he copies $r_i$ to a new set named CheckingSet, otherwise $r_i$ is kept intact. Additionally, the client computes two elements $r_{\lambda+1} = 1 - \sum_{r_i \in \text{CheckingSet}} r_i \bmod p$ , $r_{\lambda+2} = a - \sum_{r_i \in \text{CheckingSet}} r_i \bmod p$ and then copies them to sets $\{r_i\}$, CheckingSet respectively. Finally, the client permutes $\{r_i\}$ randomly and gets another set $\{r_i'\}$

- SerCom: Upon receiving the message tuple $(U, \{r_i'\})$, the server computes $R_i = U^{r_i'}$ for each $r_i' \in \{r_i'\}$ and then sends sets $\{r_i'\}, \{R_i\}$ back to the client.

- CpostCom: For the two sets $\{r_i'\}, \{R_i\}$, the client first computes $U' = \prod_{r_i' \in (\text{CheckingSet except } r_{\lambda+2})} R_i$ and checks whether $U = U'$. If yes, the client outputs $U^a = \prod_{r_i' \in (\text{CheckingSet except } r_{\lambda+1})} R_i$, otherwise, the clients outputs a symbol of false and drops the received values.

### 3.5.2 Implementation Considerations of The Server-aided Modular Exponentiation Scheme

Our server-aided modular exponentiation scheme embeds a subset sum problem, which is defined previously, to make the result computing from the received values correct and checkable. That is, it enforces attackers to first solve the subset sum problem when they intend to get more information from the transcript of the interaction between the client and the server.

Form what we mentioned above, the first problem we need to consider is the average number of times an attacker needed to solve a modified-Knapsack instance, which is determined by parameter $\lambda$. Here we denote the average number of times with the symbol $\overline{\mathsf{NUM}}$ and assume the attacker only executes brute-force attack. Assuming there are $\lambda + 2$ elements in set $\{r_i\}$ of our scheme and there are at least 2 elements in the CheckingSet set, then:

$$\begin{aligned}
\overline{\mathrm{NUM}} &= (\sum_{i=2}^{\lambda+2} \frac{1 + C_{\lambda+2}^{i}}{2})/(\lambda+1) \\
&= \frac{(\lambda + 1 + 2^{\lambda+2} - (\lambda + 2 + 1))}{2(\lambda+1)} \\
&= \frac{2^{\lambda+2} - 2}{2(\lambda+1)} \\
&= \frac{2^{\lambda+1} - 1}{\lambda+1}.
\end{aligned}$$

Clearly, when the security parameter is $k$, to make the brute-force attack infeasible, the parameter $\lambda$ should be chosen reasonably large, for example, as close as $k$.

After considering the choice of $\lambda$, we further evaluate the computational efficiency of our scheme comparing to modular exponentiation computation without server-aided technique. Assuming a client is trying to compute $U^a$ using our server-aided technique, the client still has to calculate $(|\mathsf{CheckingSet}|+1)$ times of multiplication utilizing our scheme, form the previous scheme we can find the average value of $|\mathsf{CheckingSet}|$ should be $\frac{\lambda}{6}$. When a user uses the fast multiplication technique, the range of the number of times of multiplication needed by the client should be $[|a| - 1, 2(|a| - 1)]$ where $|a|$ denotes the length of $a$'s binary representation. Here, the average value should be $\frac{3(|a|-1)}{2}$. Through simple comparison, we find our server-aided modular exponentiation scheme should be more efficient as long as $|a| > \frac{\lambda}{9}$, and we think this condition is easy to satisfy in the implementation.

### 3.5.3 The Security of The Server-aided Modular Exponentiation Scheme

Our server-aided modular exponentiation scheme holds two security properties, correctness and checkability respectively. We declare that the security of our server-aided modular exponentiation scheme is based on the subset sum problem. Here, we give a sketch of the description of the proof.

*Proof. Our proposed scheme can do checkable and correct modular exponentiation assuming the subset sub problem is hard. Namely, if there is an adversary $\mathcal{A}$ which can break our scheme with non-negligible probability $\epsilon$ in polynomial time, then it can itself compute $U^1$ from the two sets $\{r'_i\}, R_i$ with $\epsilon$ in polynomial time. Then, It can further choose correctly a subset $R_{sub}$ of $\{r'_i\}$ such that $\sum_{r'_i \in R_{sub}} r'_i = 1$ with probability $\epsilon$ in polynomial time. So given the set $\{r'_i\}$, the target 1 and an adversary $\mathcal{A}$ to our scheme, we can easily construct another PPT algorithm to solve an instance of the subset sum problem with the same non-negligible probability $\epsilon$ as that of $\mathcal{A}$ to our scheme. As the subset sum problem is NP-complete and there is no PPT solution that can solve it, so our scheme is also secure.*

## 3.6 Summary

In this chapter, we consider the problem of maintaining message confidentiality and user privacy in communication networks. We show that achieving both security properties simultaneously is not a trivial task if we aim to maintain a strong security level for both properties. Moreover, we propose a new notion named conditional privacy which requires that the intended receiver is able to recover the sender's identity. We argue that it is important in network communications when the receiver wants to send a response to the sender. We define three security models to capture the three security properties, message confidentiality, user conditional privacy and user impersonation resistance respectively, and propose a concrete scheme that is proven secure under the random oracle model.

We only considered the sender's conditional privacy in this work. A natural extension is to also consider the receiver's conditional privacy. Here, by saying receiver's conditional privacy, we mean a message sender can convince others that it is eligible to communicate with its corresponding communicator without leaking the privacy of that user. We argue that this property is reasonably helpful when where there may exist some access policies to prohibit some communication channels, where each channel is established between two users. We leave it as our next work.

# Chapter 4

# Group-based Source-destination Verifiable Encryption with Blacklist Checking

In this chapter, we first consider the message sender and receiver's conditional privacy in a more complex scenario where there exists a communication blacklist. Unlike the full privacy preservation problem, our conditional one ensures that the message sender's as well as the intended receiver's privacy are well preserved while their legitimation can still be publicly verified. Besides, the actual sender of a encrypted message can only be identified by the intended receiver. Furthermore, when numbers of communication channels are blocked by the authority, we also address the issue of proving the legitimation of the communication channel between a sender and its intended communicator. To the best of our knowledge, previous works only solve partial of our former problem and there exists no thorough solution capturing our aforementioned two problems simultaneously. With this chapter, we present a encryption scheme which keeps not only the transmitted message confidential but also the user conditional privacy preserved, our scheme also empowers the message sender the capability to give a proof of the legitimation of the communication channel. We provide several security models for our scheme and prove its security with the help of the random oracle.

## 4.1 Introduction

**Background.** The security concerns of the public key encryption are mainly focused on the secrecy of the encrypted data. Some well studied security models, indistinguishably, or non-malleability, under chosen plaintext, or ciphertext attacks (IND/NM-CPA/CCA) [DDN91, GM84, RS91], are examples catering to different

security requirements of the encrypted data. However, since encryption schemes are deployed in various hostile environments, the user privacy preservation problem should be considered seriously when attackers are more interesting in the exact parties participated in the communication.

In fact, the user privacy preservation problem have been the subject of formal studies in cryptographic literature, for example, the primitives ring signature [RST01] and group signature [CvH91] are ideal tools to protect one message sender's privacy while still keeps it authenticated. In the area of basic public key encryption (PKE), since the sender privacy preservation is thought to be an inherent property, literature related to user privacy preservation are mainly about key-privacy [BBDP01], or anonymity, which are notions the same as the receiver privacy preservation property mentioned here. In this chapter, we particularly show our interest on the user conditional privacy preservation property in PKE, which is different from the conventional one. The conditional privacy preservation notion keeps not only the privacy of the message sender but also its communicator well preserved. Furthermore, it also ensures that the legitimation of both the message sender and the receiver can be publicly verified. Besides, it also requires that only the intended message receiver of a ciphertext can discover the actual message sender.

Apart from that, We take one step further by considering a more complex scenario where the authority is allowed to block communication channels between specific message senders and receivers, and those blocked channels are publicly published as blacklist by the authority. Under such condition, the message sender should be empowered with capability to prove the legitimation of the communication channel between it and its communicator without leaking their privacy.

There exists primitive which can solve partial of our former problem. For example, the ring signcryption [HSMZ05b] can keep the transmitted message confidential and the legitimation of the message sender publicly verified but cannot maintain the conditional privacy preservation property of the message receiver. However, there is no thorough solution tackling all the two aforementioned problem properly.

**Our Contribution.** In this chapter, we present a group-based source-destination verifiable encryption scheme with blacklist checking. Our solution utilize the zero-knowledge proof of membership and also zero-knowledge of inequality technique to handle the two previously mentioned problem respectively.

Considering the security concerns of our scheme, we define three security models, which capture the message confidentiality, the sender privacy preservation, the receiver privacy preservation accordingly. We then give security proofs under our predefined models with the help of the random oracle.

**Related Work.** Among all the existed primitives, the most promising one related to our problem is the ring signcryption, which was first proposed by Huang et al. [HSMZ05b]. As it inherits properties from both the ring signature [RST01] and public key encryption, this primitive provides anonymity, authenticity of the sender along with the message confidentiality. Following works [DC06] of this primitive also consider protecting the receiver's privacy in the multi-recipient setting. Although some ring signcrypiton schemes have been proven to be insecure latter, this primitive remains to be a potential candidate when deals with problems about maintaining message confidentiality and user privacy simultaneously. However, because of the inherent property of the ring-based construction, this primitive always considers the complete anonymous of the message sender rather than the user conditional privacy preservation.

The user conditional privacy preservation is a more practical and attractive research problem in real world applications comparing to the complete privacy preservation and many existing works have considered it in some concrete scenarios. In [LLZ+08], the authors addressed the issue on anonymous authentication for messages with traceability between the on-board-units (OBUs) and roadside units (RSUs) in vehicular ad hoc networks (VANETs), this conditional privacy preservation protocol relies on the authority to trace the origin of the authenticated messages. Another similar authentication with conditional privacy example can be found in [HBCC13], where the authors considered not only user conditional privacy but forward user revocation in wireless networks. In [ELO13], pseudonym techniques are used to construct conditional privacy preservation methods protecting the privacy of users in the NFC electronic payment environment.

The receiver privacy preservation, or key-privacy, problem was first formalized by M. Bellare et al. in [BBDP01] and latter extended in [ABC+08]. In their paper, the receiver's privacy means that an eavesdropper, even in possession of a given ciphertext and a list of public keys, can not tell which specific key is the one used to generate the given ciphertext, this is the reason why they call this property key-privacy or anonymity. The authors define practical security models about the key-privacy and further state that although some classical encryption schemes, such as the El Gamal scheme [Gam85] and the Cramer-Shoup scheme [CS98b], have already provided such key-privacy property, encryption schemes with careless construction, especially for those schemes with ciphertexts including the receivers' public keys such as broadcast encryption [GSY99], still cannot hold this requirement. Key-privacy requirement is always considered in multi-receiver settings where multiple intended receivers are conventionally included in the generated ciphertext for the benefit that they can be easily identified by the message sender. In [HCW13] and also [ZM15], the authors discussed key-privacy in multi-receiver encryption scheme and

used extended receiver sets which include users who are not the intended receivers to hide the real receiver set. The anonymous broadcast encryption in [BBW06] is the first work considering receiver's privacy in broadcast encryption schemes, where a broadcast encryption scheme was constructed achieving anonymity and IND-CCA security against static adversaries from a key-private, IND-CCA secure PKE scheme, however, their technique is only analyzed in Random Oracle Model. Latter, Libert et at. in [LPQ12] propose an anonymous broadcast encryption scheme with adaptive security in the Standard Model.

**Chapter Organization.** The rest of Chapter 4 is organized as follows: Section 4.2 presents the formal definition of our source verifiable conditional privacy preserving encryption scheme, this part also defines four security models for the purpose of proving the security of our scheme. Our concrete construction of the scheme is presented in detail in Section 4.3. In Section 4.4, we prove the security of our scheme under the previously defined models respectively. At the end of this chapter, we give a conclusion and describe our future work.

## 4.2 Definitions and Security Model

### 4.2.1 Definition of the GSVEBC

In a group-based source-destination verifiable encryption with blacklist checking (GSVEBC) scheme, there are three types of participants, the message sender, verifier and receiver, involved. The sender creates and sends encrypted messages, or ciphertexts, to its intended receivers, the verifier is a party which holds some block rules, and it is the verifier that can verify whether a given ciphertext comes from a given legitimated sender set and goes to a given legitimated receiver set without knowing the exact sender and receiver of that ciphertext. Besides, the verifier can also check whether the communication channel between the sender and receiver of a given ciphertext is blocked basing on the its block rules, also without knowing both of them. The receiver of a ciphertext is the only party who can decrypt that ciphertext and recover the original message. We give a definition of our GSVEBC scheme as follows;

**Definition 4.1 (GSVEBC)** *A group-based source-destination verifiable encryption scheme with blacklist checking (GSVEBC) scheme consists of the following polynomial time algorithms.*

- $\mathsf{Setup}(1^k)$*: Taking* $1^k$ *as input, this algorithm outputs the public parameter* $\mathsf{PM}$.

- KeyGen(PM)*: For each user, this algorithm, on input* PM*, outputs a public key pair* $(pk, sk)$*. In order to make the notation more clear, let* $(pk_s, sk_s)$ *denotes a sender's key pair and* $(pk_r, sk_r)$ *be a receiver's key pair.*

- Enc(PM, $m, sk_s, pk_r, \mathcal{PK}_S, \mathcal{PK}_R$)*: This polynomial time algorithm can be executed by every message sender. Given a message* $m$,PM*, two users' public key sets* $\mathcal{PK}_S, \mathcal{PK}_R$*, the message sender's private key* $sk_s$ *and the receiver's public key* $pk_r$*, this algorithm outputs a GSVEBC ciphertext* $C$*.*

- Ver(PM, $C$)*: The verification algorithm is deterministic. Taking* PM *and a given PKESDVBRC ciphertext* $C$ *as inputs, that algorithm would first check whether the ciphertext comes from a given legitimated sender set and is sent to a given legitimated receiver set. Note that the given legitimated sender and receiver set can be included in the ciphertext* $C$*. After that, this algorithm can also check whether the communication between the sender and receiver of that given ciphertext* $C$ *is permitted according to the block rules, which may be included in* PM *or can be given by an authority to the verifier separately. This algorithm returns a symbol "True" if and only if all the above checks are successfully complete, otherwise, it returns a symbol "False". For privacy consideration, this algorithm is executed without the knowledge of the exact sender and receiver of the ciphertext* $C$*.*

- Dec(PM, $C, sk_r$)*: The decryption algorithm* Dec *is deterministic and executed by the intended receiver. When a receiver gets* $C$*, he would first execute the previous verification algorithm* Ver*, if* Ver *returns "False", he just drop this message. Otherwise, the receiver executes* Dec*, which takes* PM, $C$ *and the receiver's private key* $sk_R$ *as inputs, and recovers the original message* $m$*.*

**Definition 4.2 (Message Confidentiality)** *To capture the message confidentiality property, considering the following game between a simulator* $\mathcal{S}$ *and an adversary* $\mathcal{A}$*:*

- *Setup phase: At the setup phase, the* Setup *algorithm of the scheme, which takes* $1^k$ *as input, is first run by* $\mathcal{S}$ *to produce the system parameter* PM*. Given a polynomial* $n(\cdot)$*,* $\mathcal{S}$ *runs* KeyGen*, with* PM *as input,* $n(k)$ *times. After all executions are properly finished,* $\mathcal{S}$ *gets a public key set* $\mathcal{PK}$*, a private key set* $\mathcal{SK}$*, where* $|\mathcal{PK}| = |\mathcal{SK}| = n(k)$*. The adversary* $\mathcal{A}$ *is given* PM *and* $\mathcal{PK}$*.*

- *Decryption phase 1:* $\mathcal{A}$ *can also ask decryption queries adaptively to* $\mathcal{S}$*. That is, when* $\mathcal{A}$ *provides* $\mathcal{S}$ *a valid ciphertext,* $\mathcal{S}$ *needs to return the corresponding plaintext of this ciphertext to* $\mathcal{A}$*.*

- *Challenge phase: $\mathcal{A}$ chooses two messages $m_0, m_1$ from $\mathcal{M}$, two public keys $pk_s$, $pk_r$ from $\mathcal{PK}$ as the sender and receiver's public key respectively, two subsets $\mathcal{PK}_S, \mathcal{PK}_R$ from $\mathcal{PK}$ such that $pk_s \in \mathcal{PK}_S, pk_r \in \mathcal{PK}_R, |\mathcal{PK}_S| \geq 2, |\mathcal{PK}_R| \geq 2$, and then sends them to the simulator. Upon receiving those information, $\mathcal{S}$ randomly chooses a bit $b$ from $\{0,1\}$ and encrypts $m_b$ using the encryption algorithm of our scheme, which takes $m_b$, $sk_s$,$pk_r$,$\mathcal{PK}_S$,$\mathcal{PK}_R$ as inputs. After that, the generated ciphertext is given to $\mathcal{A}$ as the challenge ciphertext.*

- *Decryption phase 2: After receiving the challenge ciphertext, $\mathcal{A}$ can still query the decryption oracle adaptively with the only restriction that the queried ciphertext must be different from the challenge one.*

- *Guess phase: At the end of the game, $\mathcal{A}$ outputs the guess $b'$ from $\{0,1\}$ about $b$. If $b' = b$, then $\mathcal{A}$ succeeds in the game, otherwise $\mathcal{A}$ fails.*

*Remark:* $\mathcal{A}$ is allowed to ask hash queries under the random oracle model.
According to the defined model, let $\mathsf{Adv}_{\mathcal{A}}^{\text{IND-CCA}}$ denote the probability that $\mathcal{A}$ wins the above game over random guess, then

$$\mathsf{Adv}_{\mathcal{A}}^{\text{IND-CCA}} = \left| \Pr\left[b' = b\right] - \frac{1}{2} \right|$$

**Definition 4.3 (Sender Anonymity)** *Setting the security parameter as $k$, then given our scheme PKESDVBRC=(Setup, KeyGen, Enc, Ver, Dec), a polynomial $n(\cdot)$, a polynomial probabilistic time (PPT) adversary $\mathcal{A}$ and a simulator $\mathcal{S}$, let's consider the following game, which captures the sender privacy property, played by $\mathcal{A}$ and $\mathcal{S}$:*

- *Setup phase: At the setup phase, the Setup algorithm of the scheme, which takes $1^k$ as input, is first run by $\mathcal{S}$ to produce the system parameter PM. Given a polynomial $n(\cdot)$, the simulator runs KeyGen, with PM as input, $n(k)$ times. After all executions are properly finished, $\mathcal{S}$ gets a public key set $\mathcal{PK}$, a private key set $\mathcal{SK}$, where $|\mathcal{PK}| = |\mathcal{SK}| = n(k)$. The adversary $\mathcal{A}$ is given PM and $\mathcal{PK}$.*

- *Sender extraction phase 1: When $\mathcal{A}$ makes such kind of query, he submits a ciphertext to $\mathcal{S}$, then he gets the public key of the original encryptor of that ciphertext when it is valid, otherwise, he gets nothing.*

- *Challenge phase: $\mathcal{A}$ chooses one message $m$ from $\mathcal{M}$, $pk_r$ from $\mathcal{PK}$ as the receiver's public key and two subsets $\mathcal{PK}_S, \mathcal{PK}_R$ from $\mathcal{PK}$ such that $pk_r \in \mathcal{PK}_R, |\mathcal{PK}_S| \geq 2, |\mathcal{PK}_R| \geq 2$, then sends them to $\mathcal{S}$, $\mathcal{S}$ randomly chooses a*

*public key $pk_s$ from the chosen subset $\mathcal{PK}_S$, and encrypts $m$ by taking $pk_s$, $sk_s$,$pk_r$, $\mathcal{PK}_S, \mathcal{PK}_R$ as inputs. The corresponding ciphertext is given to $\mathcal{A}$ as challenge ciphertext.*

- *Sender extraction phase 2: After receiving the challenge ciphertext, $\mathcal{A}$ can still ask sender extraction queries with the only constraint that the queried ciphertext must not be identical to the challenge one, and the simulator behaves the same as in the sender extraction phase 1.*

- *Guess phase: At the end of the game, $\mathcal{A}$ outputs his guess $pk'_s$ about the public key of the sender from the chosen subset $\mathcal{PK}_S$. If $pk'_s = pk_s$, then $\mathcal{A}$ succeeds in the game, otherwise $\mathcal{A}$ fails.*

*Remark:* Under the random oracle model, $\mathcal{A}$ is allowed to ask hash queries.
According to the defined model, let $\mathsf{Adv}_{\mathcal{A}}^{\text{Sender-Anonymity}}$ denote the probability that $\mathcal{A}$ wins the above game over random guess, then

$$\mathsf{Adv}_{\mathcal{A}}^{\text{Sender-Anonymity}} = \left| \Pr\left[pk'_s = pk_s\right] - \frac{1}{|\mathcal{PK}_S|} \right|,$$

where $|\mathcal{PK}_S|$ represents the size of the subset $\mathcal{PK}_S$

**Definition 4.4 (Receiver Anonymity)** *Setting the security parameter as $k$, then given our scheme PKESDVBRC=(Setup, KeyGen, Enc, Ver, Dec), a polynomial $n(\cdot)$, a PPT (polynomial probabilistic time) adversary $\mathcal{A}$ and a simulator $\mathcal{S}$, let's consider the following game, which captures the receiver privacy property, played by $\mathcal{A}$ and $\mathcal{S}$:*

- *Setup phase: At the setup phase, the Setup algorithm of the scheme, which takes $1^k$ as input, is first run by $\mathcal{S}$ to produce the system parameter* PM. *Given a polynomial $n(\cdot)$, the simulator runs KeyGen, with* PM *as input, $n(k)$ times. After all executions are properly finished, $\mathcal{S}$ gets a public key set $\mathcal{PK}$, a private key set $\mathcal{SK}$, where $|\mathcal{PK}| = |\mathcal{SK}| = n(k)$. The adversary $\mathcal{A}$ is given* PM *and $\mathcal{PK}$.*

- *Receiver extraction phase 1: In this phase, when $\mathcal{A}$ submits ciphertext to $\mathcal{S}$, $\mathcal{S}$ needs to send back the public key of the receiver of that ciphertext to $\mathcal{A}$ as response when it is valid. Otherwise, $\mathcal{A}$ gets nothing.*

- *Challenge Phase: In the phase, $\mathcal{A}$ randomly chooses a message $m$ from $\mathcal{M}$, $pk_s$ as the sender's public key and two public key sets $\mathcal{PK}_S$, $\mathcal{PK}_R$ such that $pk_s \in \mathcal{PK}_S, |\mathcal{PK}_S| \geq 2, |\mathcal{PK}_R| \geq 2$. $\mathcal{A}$ then sends those information to $\mathcal{S}$. $\mathcal{S}$ randomly chooses $pk_r \in \mathcal{PK}_R$ as the receiver's public key and encrypts message*

*$m$ using algorithm* Enc*, which takes $m, sk_s, pk_s, pk_r, \mathcal{PK}_S, \mathcal{PK}_R$ as inputs. $\mathcal{S}$ sends the generated ciphertext as response to $\mathcal{A}$.*

- *Receiver extraction phase 2: After the challenge phase, $\mathcal{A}$ can still ask $\mathcal{S}$ to extract the public key of the receiver of a valid ciphertext for him adaptively, the only restriction is that $\mathcal{A}$ cannot use the challenge ciphertext as a queried message in this phase.*

- *Guess phase: At the end of the game, $\mathcal{A}$ would make a guess $pk'_r$ about the receiver's possible public key from the subset $\mathcal{PK}_R$. If $pk'_r = pk_r$, then $\mathcal{A}$ succeeds in the game, otherwise $\mathcal{A}$ fails.*

*Remark:*$\mathcal{A}$ is allowed to ask hash queries under the random oracle model.
According to the defined model, let $\mathsf{Adv}_{\mathcal{A}}^{\text{Receiver-Anonymity}}$ denote the probability that $\mathcal{A}$ wins the above game over random guess, then

$$\mathsf{Adv}_{\mathcal{A}}^{\text{Receiver-Anonymity}} = \left| \Pr\left[pk'_r = pk_r\right] - \frac{1}{|\mathcal{PK}_R|} \right|,$$

where $|\mathcal{PK}_R|$ represents the size of the chosen subset $\mathcal{PK}_R$.

## 4.3 Our Concrete Construction

We first give a group-based source-destination verifiable encryption scheme but without blacklist checking, then we extend this scheme to one with blacklist checking.

### 4.3.1 A Simple Construction without Blacklist Checking

Setting the security parameter as $k$, our scheme works as follows;

- $\mathsf{Setup}(1^k)$: On input $1^k$, it produces a cyclic group $\mathbb{G}$ of large prime order $q$ with generator $g$, where $\mathbb{G}$ is a subgroup of $\mathbb{Z}_p^*$ and $q|p-1$. This algorithm also outputs a description of the message space $\mathcal{M} = \{0,1\}^q$ and a ciphertext space $\mathcal{C}$. $\mathbb{G}, q, g, \mathcal{M}, \mathcal{C}$ are considered as the system parameter PM and default inputs to all the following algorithms. PM also includes two collision resistance hash functions: $H_1 : \{0,1\}^q \times \mathbb{G}^3 \to \mathbb{Z}_q, H_2 : \mathbb{G} \to \{0,1\}^q$.

- $\mathsf{KeyGen}(\cdot)$: For one user, $U_i$ for example, he randomly chooses $x_i \in \mathbb{Z}_q$ as his private key and computes $y_i = g^{x_i} \in \mathbb{G}$ as his corresponding public key. Assuming the public key set $\mathcal{PK}$ contains all user's public key.

- $\mathsf{Enc}(m, sk_s, pk_r, \mathcal{PK}_S, \mathcal{PK}_R)$: When a sender, $U_i$, wants to send a message to a receiver, $U_j$, for the purpose of illustrating our scheme more clear, lets

$S_i$,$R_i$ denotes the sender $U_i$ and receiver $U_j$'s secret key $sk_s, sk_r$ respectively, accordingly, the sender and receiver's public key should be $pk_s = g^{S_i}$ and $pk_r = g^{R_j}$. Given a message $m \in \mathcal{M}$, the sender encrypts $m$ as follows;

$$
\begin{aligned}
&r_1 \xleftarrow{R} \mathbb{Z}_p, \\
&C_1 = g^{r_1}, C_2 = g^{S_i \cdot r_1}, C_3 = g^{R_j \cdot r_1}, C_4 = g^{R_j \cdot S_i} \\
&r_2 = H_1(m, g^{S_i}, g^{R_j}, C_1, C_2, C_3, C_4), C_5 = g^{r_2}, \\
&C_6 = m \oplus H_2(g^{R_j \cdot (r_1 + r_2)}).
\end{aligned}
$$

After that, the sender chooses a subgroup $\mathcal{PK}_S \subset \mathcal{PK}$, which includes the sender's public key $g^{S_i}$, and then prove its legitimation in that group. Here, we utilize the zero-knowledge proof to deal with the group membership issue. That is, the sender needs to do a proof like:

$$
pf(S_i : \log_g g^{S_i} = \log_{C_1} C_2 = \log_{g^{r_1}} (g^{S_i})^{r_1} \wedge g^{S_i} \in \mathcal{PK}_S).
$$

To do such a proof, the sender does as follows;

- For each public key $g^{x_l} \in \mathcal{PK}_S$ except $g^{S_i}$, the sender chooses challenge and response $c_l, z_l$ randomly from $\mathbb{Z}_q$ respectively, then it computes two commitments
$$
\alpha_l = g^{z_l}(g^{x_l})^{c_l}, \beta_l = (C_1)^{z_l}(C_2)^{c_l}.
$$

- For the sender's own pubic key $g^{S_i}$, it chooses $w_i \in \mathbb{Z}_q$ and sets the commitments as
$$
\alpha_i = g^{w_i}, \beta_i = (C_1)^{w_i}.
$$
Let $\{\alpha\}$ denote commitments set $\{\dots \alpha_l \dots \alpha_i \dots\}$ and $\{\beta\}$ denote commitments set $\{\dots \beta_l \dots \beta_i \dots\}$, where $|\alpha| = |\beta| = |\mathcal{PK}_S|$. The sender computes its challenge and response as:
$$
\begin{aligned}
&h = H_3(\{\alpha\}, \{\beta\}, C_1, C_2, C_3, C_4, C_5, C_6), \\
&c_i = h - \sum_{g^{x_l} \in \mathcal{PK}_S} c_l, \qquad z_i = w_i - c_i S_i.
\end{aligned}
$$

- The sender sets the challenges set as $\{c\} = \{\dots c_i \dots c_l \dots\}$ the responses set as $\{z\} = \{\dots z_i \dots z_l \dots\}$, and value this two sets $\{c\}, \{z\}$ as the proof value.

The sender needs still to prove to the verifier that the generated ciphertext goes to a legitimated receiver. To do this, the sender chooses a receiver subset $\mathcal{PK}_R \subset \mathcal{PK}$, which includes the receiver's public key and have to do a proof

like:

$$pf(r_1 : \log_g C_1 = \log_{g^{R_j}} C_3 \wedge g^{R_j} \in \mathcal{PK}_R),$$

the sender generates the proof as follows;

- For each public key $g^{x_t} \in \mathcal{PK}_R$ except the intended receiver's public key $g^{R_j}$, the sender chooses challenge and response $\widehat{c}_t, \widehat{z}_t$ randomly from $\mathbb{Z}_q$ respectively, then it computes two commitments

$$\widehat{\alpha}_t = g^{\widehat{z}_t}(C_1)^{\widehat{c}_t}, \widehat{\beta}_t = (g^{x_t})^{\widehat{z}_t}(C_3)^{\widehat{c}_t}.$$

- For the intended receiver's pubic key $g^{R_j}$, it chooses $\widehat{w}_j \in \mathbb{Z}_q$ and sets the commitments as

$$\widehat{\alpha}_j = g^{\widehat{w}_j}, \widehat{\beta}_j = (g^{R_j})^{\widehat{w}_j}.$$

  Let $\widehat{\{\alpha\}}$ denote commitments set $\{\dots \widehat{\alpha}_t \dots \widehat{\alpha}_j \dots\}$ and $\widehat{\{\beta\}}$ denote commitments set $\{\dots \widehat{\beta}_t \dots \widehat{\beta}_j \dots\}$, where $|\widehat{\{\alpha\}}| = |\widehat{\{\beta\}}| = |\mathcal{PK}_R|$. The sender computes its challenge and response as:

$$\widehat{h} = H_3(\widehat{\{\alpha\}}, \widehat{\{\beta\}}, C_1, C_2, C_3, C_4, C_5, C_6), \widehat{c}_j = \widehat{h} - \sum_{g^{x_t} \in \mathcal{PK}_R} \widehat{c}_t, \widehat{z}_j = \widehat{w}_j - \widehat{c}_j r_1$$

- The sender sets the challenges set as $\widehat{\{c\}} = \{\dots \widehat{c}_j \dots \widehat{c}_t \dots\}$ the responses set as $\widehat{\{z\}} = \{\dots \widehat{z}_j \dots \widehat{z}_t \dots\}$, and value this two sets $\widehat{\{c\}}, \widehat{\{z\}}$ as the proof value.

After the two proofs are finished, the final ciphertext should be $CT = (C_1, C_2, C_3, C_4, C_5, C_6, \mathcal{PK}_S, \{c\}, \{z\}, \mathcal{PK}_R, \widehat{\{c\}}, \widehat{\{z\}})$.

- $\mathsf{Ver}(CT)$: Every user can act as the verifier. Upon receiving a given ciphertext like the above format $CT = (C_1, C_2, C_3, C_4, C_5, C_6, \mathcal{PK}_S, \{c\}, \{z\}, \mathcal{PK}_R, \{\widehat{c}\}, \{\widehat{z}\})$, a verifier does the following steps to verify the validity of the ciphertext:

  - For the ciphertext components $(C_1, C_2, C_3, C_4, C_5, C_6, \mathcal{PK}_S, \{c\}, \{z\})$, the verifier recomputes

$$\alpha_l^{'} = g^{z_l}(g^{x_l})^{c_l}, \beta_l^{'} = (C_1)^{z_l}(C_2)^{c_l} \text{ for each } g^{x_l} \in \mathcal{PK}_S$$

    and gets two sets $\{\alpha^{'}\} = \{\dots \alpha_l^{'} \dots\}, \{\beta^{'}\} = \{\dots \beta_l^{'} \dots\}$, then it checks whether the equation

$$H_3(\{\alpha^{'}\}, \{\beta^{'}\}, C_1, C_2, C_3, C_4, C_5, C_6) = \sum_{c_l \in \{c\}} c_l$$

holds. If no, it returns a symbol of false and drops this ciphertext, otherwise it continues to the next step.

– For the ciphertext components $(C_1, C_2, C_3, C_4, C_5, C_6, \mathcal{PK}_R, \widehat{\{c\}}, \widehat{\{z\}})$, the verifier further computes

$$\widehat{\alpha'}_t = g^{\widehat{z}_t}(C_1)^{\widehat{c}_t}, \widehat{\beta'}_t = (g^{x_t})^{\widehat{z}_t}(C_3)^{\widehat{c}_t} \text{ for each } g^{x_t} \in \mathcal{PK}_R.$$

Then it gets two sets

$$\widehat{\{\alpha'\}} = \{\dots \widehat{\alpha'}_t \dots \widehat{\alpha'}_j \dots\}, \widehat{\{\beta'\}} = \{\dots \widehat{\beta'}_t \dots \widehat{\beta'}_j \dots\}.$$

The verifier finally checks whether the equation

$$H_3(\widehat{\{\alpha'\}}, \widehat{\{\beta'\}}, C_1, C_2, C_3, C_4, C_5, C_6) = \sum_{\widehat{c}_t \in \widehat{\{c\}}} \widehat{c}_t$$

holds. If no, the verifier returns a symbol of false and drops this ciphertext, otherwise it returns a symbol of true relay this ciphertext to the receiver set.

- $\mathsf{Dec}(CT, R_x)$: This decryption algorithm are executed by all the possible receivers of a given ciphertext. When given a copy of the ciphertext $CT = (C_1, C_2, C_3, C_4, C_5, C_6, \mathcal{PK}_S, \{c\}, \{z\}, \mathcal{PK}_R, \widehat{\{c\}}, \widehat{\{z\}})$, all possible receivers in set $\mathcal{PK}_R$ do as following:

  – All possible receivers in $\mathcal{PK}_R$ would first execute the verification algorithm $\mathsf{Ver}$ of our scheme as a subroutine. If $\mathsf{Ver}$ returns false, they drop $CT$ and returns a symbol of failure, otherwise they continue to the next step.

  – Each user $U_x$ in $\mathcal{PK}_R$ uses its secret key $R_x$ to check whether equation $C_1^{R_x} = C_3$ holds. If not, it drops $CT$ and returns a symbol of failure, otherwise, this user goes to the next step.

  – For each of the users whose secret key satisfying the above equation, it first gets the public key, which is denoted by $g^{s'}$, of the original sender of the given $CT$ by computing

$$g^{s'} = (C_5)^{{R_x}^{-1}},$$

then it recover the encrypted message, denoted by $m'$, as

$$m' = C_6 \oplus H_2((C_1 C_5)^{R_x}).$$

**Table 4.1:** The Blocklist

| $< . \ , \ . >$ |
|---|
| $< g^S, g^R >$ |
| $< . \ , \ . >$ |

After getting $g^{s'}$ and $m'$, it would check whether the equation

$$C_5 = g^{H_1(m', g^{s'}, g^{R_x}, C_1, C_2, C_3, C_4)}$$

holds, if yes, this user outputs $g^{s'}$ as the public key of the message sender and $m'$ as the original message. Otherwise, this user drops $CT$ and returns a symbol of failure.

### 4.3.2 Our Concrete Construction with Blacklist Checking

Basing on the former scheme, We give another construction to empower our scheme with blacklist checking capability. Here, for simplicity, a block rule can be expressed as $< pk_s, pk_r >$, where the former one is one specific sender's and the other is one specific receiver's public key respectively, and is used to disable the communication from one message sender to one receiver. Assuming the blacklist, BL for short, includes several block rules and is publicly accessible, our scheme assures that a verifier can check whether a given ciphertext should be rejected according to the BL.

By applying the technique of zero-knowledge proof of inequality of two discrete logarithms, which was proposed in [CS03a], we find a way to extend our original scheme to a scheme with blacklist checking, which only add a set of proof values to the original one. Because those two schemes are pretty similar, we only give explicit description of the most different part between them.

Our public key encryption scheme with source-destination verifiability and block rules checking (PKESDVBRC) consists of the following polynomial time algorithms.

- $\mathsf{Setup}(1^k)$: This algorithm acts the same as the previous scheme.
- $\mathsf{KeyGen}(\cdot)$:This algorithm is also identical to the aforementioned scheme.
- $\mathsf{Enc}(m, sk_s, pk_r, \mathcal{PK}_S, \mathcal{PK}_R)$: Apart from the encryption process of the encryption scheme of the previous scheme, here the sender also needs to generate a proof to convince the verifier that the generated ciphertext should not be blocked according to the block rules. Assuming there is a blacklist like follows;

  For each block rule, $< g^S, g^R >$ for example in Table 4.1, in the blacklist, the

message sender needs to prove that a ciphertext does not come form a user with identity $g^S$ or go to a user with identity $g^R$. That is, according to our scheme, the message sender should produce a proof

$$pf((S_i, r_1) : \log_{C_1} C_2 \neq \log_g g^S \vee \log_g C_1 \neq \log_{g^R} C_3)$$

for this rule. Assuming there is a message sender with identity $g^{S_i}$ and one ciphertext generated by that sender is sent to a receiver with identity $g^{R_j}$, the message sender generates such a proof $pf$ for that ciphertext basing on the following different types of ciphertext;

- If $\log_{C_1} C_2 = \log_g g^S$ and $\log_g C_1 \neq \log_{g^R} C_3$, that is $g^{S_i} = g^S$ and $g^{R_j} \neq g^R$:
  - The message sender chooses $\delta$ randomly from $\mathbb{Z}_p$ and sets $\gamma = S_i \cdot \delta$, then it tries to give a proof like

    $$pf((\gamma, \delta) : St_0 = g^\gamma/(g^S)^\delta \neq 1 \vee St_1 = (g^{r_1})^\gamma/(g^{S_i r_1})^\delta = 1).$$

    As the above proof is to release the truth that $g^{S_i} \neq g^S$, so the message sender needs to simulate such a proof. That is, the message sender first chooses a challenge $CH \in \mathbb{Z}_q$ and two responses $e_0, e_1 \in \mathbb{Z}_q$ respectively, and sets the two commitments

    $$COM_0 = {St_0}^{CH}(g)^{e_0}/(g^S)^{e_1}, COM_1 = {St_1}^{CH}(g^{r_1})^{e_0}/(g^{S_i r_1})^{e_1}.$$

  - The message sender chooses $\widehat{\delta}$ randomly from $\mathbb{Z}_p$ and sets $\widehat{\gamma} = r_1 \cdot \widehat{\delta}$, then gives a real proof

    $$pf((\widehat{\gamma}, \widehat{\delta}) : \widehat{St_0} = (g^R)^{\widehat{\gamma}}/(g^{R_j r_1})^{\widehat{\delta}} \neq 1 \vee \widehat{St_1} = (g)^{\widehat{\gamma}}/(g^{r_1})^{\widehat{\delta}} = 1).$$

    That is, the message sender first chooses two elements $\widehat{w_0}, \widehat{w_1} \in \mathbb{Z}_q$ and computes the two commitments

    $$\widehat{COM_0} = (g^R)^{\widehat{w_0}}/(g^{R_j r_1})^{\widehat{w_1}}, \widehat{COM_1} = (g)^{\widehat{w_0}}/(g^{r_1})^{\widehat{w_1}}.$$

    The sender then computes a hash value

    $$X = H_3(COM_0, COM_1, \widehat{COM_0}, \widehat{COM_1})$$

    and sets the challenge of this proof as $\widehat{CH} = X - CH$, the two

responses should be

$$\widehat{e_0} = \widehat{w_0} - \widehat{CH} \times \widehat{\gamma}, \widehat{e_1} = \widehat{w_1} - \widehat{CH} \times \widehat{\delta}$$

respectively.

* After all the required values are properly computed, let $pf$ denote the proof values, then

$$pf = (St_0, St_1, CH, e_0, e_1, \widehat{St_0}, \widehat{St_1}, \widehat{CH}, \widehat{e_0}, \widehat{e_1}).$$

– If $\log_{C_1} C_2 \neq \log_g g^S$ and $\log_g C_1 = \log_{g^R} C_3$, that is $g^{S_i} \neq g^S$ and $g^{R_j} = g^R$:

* The message sender chooses $\widehat{\delta}$ randomly from $\mathbb{Z}_p$ and sets $\widehat{\gamma} = r_1 \cdot \widehat{\delta}$, and it tries to give a proof like

$$pf((\widehat{\gamma}, \widehat{\delta}) : \widehat{St_0} = (g^R)^{\widehat{\gamma}}/(g^{R_j r_1})^{\widehat{\delta}} \neq 1 \vee \widehat{St_1} = (g)^{\widehat{\gamma}}/(g^{r_1})^{\widehat{\delta}} = 1).$$

As the above proof is to release the truth that $g^{R_j} \neq g^R$, so the message sender needs to simulate such a proof. That is, the message sender first chooses a challenge $\widehat{CH} \in \mathbb{Z}_q$ and two responses $\widehat{e_0}, \widehat{e_1} \in \mathbb{Z}_q$ respectively, and sets the two commitments

$$\widehat{COM_0} = \widehat{St_0}^{\widehat{CH}}(g^R)^{\widehat{e_0}}/(g^{R_j r_1})^{\widehat{e_1}}, \widehat{COM_1} = \widehat{St_1}^{\widehat{CH}}(g)^{\widehat{e_0}}/(g^{r_1})^{\widehat{e_1}}.$$

* The message sender chooses $\delta$ randomly from $\mathbb{Z}_p$ and sets $\gamma = S_i \cdot \delta$, then it gives a proof

$$pf((\gamma, \delta) : St_0 = (g)^{\gamma}/(g^S)^{\delta} \neq 1 \vee St_1 = (g^{r_1})^{\gamma}/(g^{S_i r_1})^{\delta} = 1).$$

That is, the message sender first chooses two elements $w_0, w_1 \in \mathbb{Z}_q$ and computes the two commitments

$$COM_0 = (g)^{w_0}/(g^S)^{w_1}, COM_1 = (g^{r_1})^{w_0}/(g^{S_i r_1})^{w_1}.$$

The sender then computes a hash value

$$X = H_3(COM_0, COM_1, \widehat{COM_0}, \widehat{COM_1})$$

and sets the challenge of this proof as $CH = X - \widehat{CH}$, the two

responses should be

$$e_0 = w_0 - CH \times \gamma, e_1 = w_1 - CH \times \delta$$

respectively.

* After all the required values are properly computed, let $pf$ denote the proof values, then

$$pf = (St_0, St_1, CH, e_0, e_1, \widehat{St_0}, \widehat{St_1}, \widehat{CH}, \widehat{e_0}, \widehat{e_1})$$

– If $\log_{C_1} C_2 \neq \log_g g^S$ and $\log_g C_1 \neq \log_{g^R} C_3$, that is $g^{S_i} \neq g^S$ and $g^{R_j} \neq g^R$:

* The message sender chooses $\delta$ randomly from $\mathbb{Z}_p$ and sets $\gamma = S_i \cdot \delta$, then it gives a proof like

$$pf((\gamma, \delta) : St_0 = (g)^\gamma/(g^S)^\delta \neq 1 \vee St_1 = (g^{r_1})^\gamma/(g^{S_i r_1})^\delta = 1).$$

That is, the message sender first chooses two elements $w_0, w_1 \in \mathbb{Z}_q$ and computes the two commitments

$$COM_0 = (g)^{w_0}/(g^S)^{w_1}, COM_1 = (g^{r_1})^{w_0}/(g^{S_i r_1})^{w_1}.$$

The sender then chooses a challenge of this proof $CH \in \mathbb{Z}_q$, the two responses should be

$$e_0 = w_0 - CH \times \gamma, e_1 = w_1 - CH \times \delta$$

respectively.

* The message sender chooses $\widehat{\delta}$ randomly from $\mathbb{Z}_p$ and sets $\widehat{\gamma} = r_1 \cdot \widehat{\delta}$, then it gives a proof like

$$pf((\widehat{\gamma}, \widehat{\delta}) : \widehat{St_0} = (g^R)^{\widehat{\gamma}}/(g^{R_j r_1})^{\widehat{\delta}} \neq 1 \vee \widehat{St_1} = (g)^{\widehat{\gamma}}/(g^{r_1})^{\widehat{\delta}} = 1).$$

That is, the message sender first chooses two elements $\widehat{w_0}, \widehat{w_1} \in \mathbb{Z}_q$ and computes the two commitments

$$\widehat{COM_0} = (g^R)^{\widehat{w_0}}/(g^{R_j r_1})^{\widehat{w_1}}, \widehat{COM_1} = (g)^{\widehat{w_0}}/(g^{r_1})^{\widehat{w_1}}.$$

The sender then computes a hash value

$$X = H_3(COM_0, COM_1, \widehat{COM_0}, \widehat{COM_1})$$

and sets the challenge of this proof as $\widehat{CH} = X - CH$, the two responses should be

$$\widehat{e_0} = \widehat{w_0} - \widehat{CH} \times \widehat{\gamma}, \widehat{e_1} = \widehat{w_1} - \widehat{CH} \times \widehat{\delta}$$

respectively.

* After all the required values are properly computed, let $pf$ denote the proof values, then

$$pf = (St_0, St_1, CH, e_0, e_1, \widehat{St_0}, \widehat{St_1}, \widehat{CH}, \widehat{e_0}, \widehat{e_1}).$$

Assuming there are $n$ rules in the blacklist, the message sender needs to generate $n$ proofs accordingly using the technique we described above. Let $\{pf\}$ denote the collection of all the $n$ proofs ,then the full ciphertext $CT$ should be $(C_1, C_2, C_3, C_4, C_5, C_6, \mathcal{PK}_S, \{c\}, \{z\}, \mathcal{PK}_R, \widehat{\{c\}}, \widehat{\{z\}}, \{pf\})$

- $\mathsf{Ver}(CT)$: During the execution of this algorithm, a verifier would first do the same as what in the verification algorithm of the previous scheme. Furthermore, to check the block rules, for each proof $(St_0, St_1, CH, e_0, e_1, \widehat{St_0}, \widehat{St_1}, \widehat{CH}, \widehat{e_0}, \widehat{e_1})$ in $\{pf\}$ and its corresponding rule $< g^S, g^R >$, the verifier computes

$$COM_0' = {St_0}^{CH}(g)^{e_0}/(g^S)^{e_1}, COM_1' = {St_1}^{CH}(g^{r_1})^{e_0}/(g^{S_i r_1})^{e_1},$$
$$\widehat{COM}_0' = {\widehat{St_0}}^{\widehat{CH}}(g^R)^{\widehat{e_0}}/(g^{R_j r_1})^{\widehat{e_1}}, \widehat{COM}_1' = {\widehat{St_1}}^{\widehat{CH}}(g)^{\widehat{e_0}}/(g^{r_1})^{\widehat{e_1}})$$

and then checks whether the equation

$$CH + \widehat{CH} = COM_0' + COM_1' + \widehat{COM}_0' + \widehat{COM}_1'$$

holds. If yes, the verifier turns to the next proof in the list $\{pf\}$, otherwise it drops this ciphertext. The verifier would relay the ciphertext if all the proofs in $\{pf\}$ are successfully checked.

- $\mathsf{Dec}(CT, R_j)$: This algorithm shares no difference from that in the the previous scheme.

## 4.4 Security Proofs

**Theorem 4.1** *Our scheme maintains message confidentiality under the previously defined message confidentiality model assuming the DDH problem is hard in* $\mathbb{G}$ *when*

*hash functions $H_1, H_2, H_3$ are modeled as random oracles. Concretely, if there is an adversary $\mathcal{A}$ which can break our scheme with non-negligible probability $\epsilon$, supposing $\mathcal{A}$ makes at most $q_{H_1}$,$q_{H_2}$,$q_{H_3}$ queries to the $H_1, H_2, H_3$ hash oracles respectively, and $q_D$ queries to the decryption oracle, then we can construct another algorithm $\mathcal{B}$ that solves the DDH problem in $\mathbb{G}$ with advantage at least $\frac{1}{n}(1 - \frac{q_D}{2^k})\epsilon$, where $k$ is the security parameter and $n$ is a constant.*

**Proof.** We show how to construct an algorithm $\mathcal{B}$ that solves the DDH problem by interacting with an adversary $\mathcal{A}$ of our scheme under our predefined model.

- Setup phase: On input $1^k$, $\mathcal{B}$ runs the Setup algorithm of our scheme to produce system parameters PM which includes $\mathbb{G}, p, g, \mathcal{M}, \mathcal{C}$ and the block rule list BRL. $\mathcal{B}$ is then given a DDH tuple $(g^a, g^b, Z)$. For a given polynomial $n(\cdot)$, set $n = n(k)$. $\mathcal{B}$ runs the key generation algorithm $Gen(\cdot)$ $n$ times, except that $\mathcal{B}$ sets $pk_j = g^a$ for a randomly chosen $j \in [1, n]$ and does not have the corresponding private key $x_j$. Namely, $\mathcal{B}$ gets a public key set $\mathcal{PK} = \{pk_1, \ldots, pk_n\}$ and a private key set $\mathcal{SK} = \{x_1, \ldots, x_{j-1}, x_{j+1}, \ldots, x_n\}$. $\mathcal{B}$ chooses three collision-resistance hash functions: $H_1 : \mathcal{M} \times \mathbb{G}^5 \to \mathbb{Z}_p$, $H_2 : \mathbb{G} \to \{0,1\}^{|\mathcal{M}|}$, $H_3 : \{0,1\}^* \to \mathbb{Z}_p$ and sends them to $\mathcal{A}$, $H_1, H_2, H_3$ are fully controlled by $\mathcal{B}$ and are modeled as random oracles. Finally, $\mathcal{B}$ gives $\mathcal{A}$ PM and $\mathcal{PK}$.

- $H_1$-query phase: $\mathcal{A}$ can issue queries to the hash function $H_1$. In order to respond those queries, $\mathcal{B}$ keeps a hash table $H_1^{table}$. When $\mathcal{A}$ asks the hash value of the $v$-th message tuple $(m, g^{S_i}, g^{R_j}, g^{r_1}, (g^{S_i})^{r_1}, (g^{R_j})^{r_1}, g^{S_i \cdot R_j})_v$, $\mathcal{B}$ checks whether this tuple has appeared before, :
  - If yes, $\mathcal{B}$ responds $\mathcal{A}$ with the record $h_{1v}$.
  - Otherwise, $\mathcal{B}$ chooses a random value $h_{1v}$ from $\mathbb{Z}_p$, and sets this value as the hash value of the queried message tuple, $\mathcal{B}$ responds $\mathcal{A}$ with $h_{1v}$ and adds this message and hash value pair to the $H_1^{table}$.

- $H_2$-query phase: $\mathcal{A}$ can also ask $H_2$ hash queries. To answer this kind of query, the algorithm $\mathcal{B}$ maintains a $H_2^{table}$ table which has two columns.
  When the $u$-th query $(g^{R_j \cdot (r_1 + r_2)})_u$ is made, $\mathcal{B}$ checks whether this tuple has appeared before:
  - If yes, $\mathcal{B}$ responds $\mathcal{A}$ with the corresponding record $h_{2j}$.
  - Otherwise, $\mathcal{B}$ chooses a random value $h_{2u}$ from $\{0,1\}^{|\mathcal{M}|}$, where $|\mathcal{M}|$ denotes the length of the message in $\mathcal{M}$, and sets this value as the hash of the queried message tuple, $\mathcal{B}$ responds with $h_{2u}$ and adds this message and hash value pair to the $H_2^{table}$.

- $H_3$-query phase: $\mathcal{A}$ can ask $H_3$ queries. $\mathcal{B}$ creates a hash table $H_3^{table}$ to respond this kind of query. For the t-th query tuple $< \{0,1\}^* >_t$, $\mathcal{B}$ acts as following:
  - $\mathcal{B}$ first check whether this tuple has appeared before, if yes, $\mathcal{B}$ responds with the existing value $h_{3t}$ to $\mathcal{A}$.
  - Otherwise, $\mathcal{B}$ chooses a random value $h_{3t}$ from $\mathbb{Z}_p$ , and sets it as the hash value of the queried message tuple. $\mathcal{B}$ responds $\mathcal{A}$ with $h_{3t}$ and adds this message and hash value pair to the $H_3^{table}$.

  *Remark:* The three hash oracles can be asked interchangeably.

- Decryption query phase 1: $\mathcal{A}$ can make decryption queries adaptively to $\mathcal{B}$. To respond such kind of query, $\mathcal{B}$ itself first constructs a table $H^{table}$ with four columns from $H_1^{table}$ table and $H_2^{table}$ table. For each queried message and hash value pair, denoted by $((m, g^{S_i}, g^{R_j}, g^{r_1}, g^{S_i \cdot r_1}, g^{R_j \cdot r_1}, g^{S_i \cdot R_j}), h_{1v})$, in the table $H_1^{table}$, $\mathcal{B}$ would process it as below until every pair in $H_1^{table}$ table is searched:
  - $\mathcal{B}$ first computes $g^{R_j \cdot r_1} \cdot g^{R_j \cdot h_{1v}}$, and searches table $H_2^{table}$ to find whether there exist a queried message and hash value pair in table $H_2^{table}$ such that the value $g^{R_j \cdot r_1} \cdot g^{R_j \cdot h_{1v}}$ computed form $H_1^{table}$ and the queried message value in $H_2^{table}$ are equal.
  - If $\mathcal{B}$ can find such a queried message and hash value pair, then this queried message and hash value pair in $H_1^{table}$ and the corresponding queried message and hash value pair in $H_2^{table}$ are first concatenated as a new row with four elements and this new row is added to the table $H^{table}$. After that, the two queried message and hash value pairs are deleted from their corresponding hash table respectively, then $\mathcal{B}$ jumps to the next row of $H_1^{table}$ table and acts as above.
  - If $\mathcal{B}$ cannot find such a queried message and hash value pair, $\mathcal{B}$ just jumps to the next row of $H_1^{table}$ table and acts as above.

  Notice that $H^{table}$ should be updated timely when either the $H_1^{table}$ or $H_2^{table}$ is changed. The updating procedure is the same as the procedure the $H^{table}$ is constructed. Assuming each row of the $H^{table}$ has the format like $((m, g^{S_i}, g^{R_j}, g^{r_1}, g^{S_i \cdot r_1}, g^{R_j \cdot r_1}, g^{S_i \cdot R_j}), h_{1v}, g^{R_j \cdot r_1} \cdot g^{R_j \cdot h_{1v}}, h_{2v})$.

  When a queried cihpertext received is $(C_1, C_2, C_3, C_4, C_5, C_6, \mathcal{PK}_S, \{c\}, \{z\}, \mathcal{PK}_R, \widehat{\{c\}}, \widehat{\{z\}}, \{pf\})$, $\mathcal{B}$ responds this decryption query using $H_3^{table}$ and $H^{table}$:
  - $\mathcal{B}$ would execute the Ver algorithm first. According to the scheme, $\mathcal{B}$ needs to search the $H_3$ list several times during the execution of the Ver algorithm. If there exists not such values in the $H_3$ list that making

the received ciphertext acceptable, which means the ciphertext can pass the Ver algorithm executed by $\mathcal{B}$, $\mathcal{B}$ drops it and returns nothing to $\mathcal{A}$. Otherwise, $\mathcal{B}$ continues to the next step.

- Assuming $H^{table}$ has $t$ rows, $\mathcal{B}$ first checks whether there exist rows in $H^{table}$ such that queried messages in the first column of that row including $(C_1, C_2, C_3, C_4)$ part of the given ciphertext, if not, $\mathcal{B}$ drops it and returns nothing to $\mathcal{A}$.
- If there have rows in $H^{table}$ satisfying the above checking, for each of them, $\mathcal{B}$ further checks whether all the following equations hold;

$$C_5 = g^{h_{1v}}, m = C_6 \oplus h_{2v}.$$

- If there exists a row that fulfilling all the above checking, then $\mathcal{B}$ checks the first column of that row and sends $m$ in this column to $\mathcal{A}$. Otherwise, $\mathcal{B}$ drops it and returns nothing to $\mathcal{A}$.

*Remark:* We should note that there may be some valid ciphertexts which would be rejected by our aforementioned decryption simulator, especially when the intended receiver of a given ciphertext is the user with identity $ID_j$. Here a valid ciphertext means, when it appears, the intended receiver can decrypt it correctly and return the corresponding plaintext, while our decryption simulator cannot.

Assuming there are $q_D$ decryption queries asked during the decryption phases. According to the encryption algorithm of our scheme, a ciphertext is not valid until it is generated after querying all the three hash functions, and obviously, this kind of ciphertext can definitely be decrypted by our decryption simulator correctly. However, if at least one of the three hash functions is not asked when producing a ciphertext, this ciphertext may still have probability to be a valid one while our decryption simulator would reject it. Clearly, the probability of this case should be at most $\frac{1}{2^k}$, where $k$ is the security parameter. Let us consider the event that at least one of the $q_D$ queried ciphertexts is valid but is rejected by our decryption simulator. Let symbol $fail$ denote this event and symbol $r(i)$ denote the event that the i-th queried ciphertext is rejected but actually is a valid one, where $1 \leq i \leq q_D$. Then the probability of this event, $\Pr[fail]$, is:

$$\begin{aligned} \Pr[fail] &= \Pr[r(1) \cup r(2) \cup, \ldots, \cup r(q_D)] \\ &\leq \Pr[r(1)] + \Pr[r(2)] + \Pr[r(3)] + .... + \Pr[r(q_D)] \\ &= \frac{q_D}{2^k} \end{aligned}$$

Namely, with probability at most $\frac{q_D}{2^k}$, the decryption simulator would reject valid ciphertext(s). That is, with probability at least $1 - \frac{q_D}{2^k}$, $\mathcal{B}$ would do a perfect simulation in the decryption phases.

- Challenge phase: After the decryption queries are properly answered, $\mathcal{A}$ chooses two messages $m_0, m_1$ from $\mathcal{M}$, two public keys $pk_i$, $pk_j$ from $\mathcal{PK}$ as the sender and receiver's public key respectively and two subsets $\mathcal{SPK}$,$\mathcal{RPK}$ from $\mathcal{PK}$ such that $pk_i \in \mathcal{SPK}, pk_j \in \mathcal{RPK}, |\mathcal{SPK}|, |\mathcal{RPK}| \geq 2$, then sends them to $\mathcal{B}$. Upon receiving those information, $\mathcal{B}$ would first check whether the receiver's public key is $pk_j = g^a$, that is, the one $\mathcal{B}$ does not know its corresponding private key. If not, $\mathcal{B}$ aborts the game and outputs a random bit, otherwise $\mathcal{B}$ randomly chooses $c \in \{0, 1\}$ and encrypts $m_c$ using the Enc algorithm. Namely, $\mathcal{B}$ asks $H_1$ query about the message tuple $(m_c, g^{S_i}, g^a, g^b, g^{S_i \cdot a}, Z, g^{S_i \cdot b})$ to get $r_2^*$ , asks $H_2$ query about the message $Z \cdot g^{b \cdot r_2^*}$ to get $h_2^*$, asks $H_3$ query for the purpose of generating the proof tuples, and sets the ciphertext as $(g^b, g^{S_i \cdot a}, Z, g^{S_i \cdot b}, g^{r_2^*}, m_c \oplus h_2^*, \mathcal{SPK}, \{c\}, \{z\}, \mathcal{RPK}, \widehat{\{c\}}, \widehat{\{z\}}, \{pf\})$, where $h_2^* = H_2(Z \cdot g^{b \cdot r_2^*}), r_2^* = H_1(m_c, g^{S_i}, g^a, g^b, g^{S_i \cdot a}, Z, g^{S_i \cdot b})$. $\mathcal{B}$ sends the generated ciphertext to $\mathcal{A}$ as the challenge ciphertext. Then $\mathcal{B}$ adds the message tuple $((m_c, g^{S_i}, g^a, g^b, g^{S_i \cdot a}, Z, g^{S_i \cdot b}), r_2^*, Z \cdot g^{b \cdot r_2^*}, h_2^*)$ as a new row to the $H^{table}$ table.

- Decryption queries phase 2: In this phase, $\mathcal{A}$ can still ask decryption queries with the only constraint that $\mathcal{A}$ cannot use the challenge ciphertext as one of his queried messages. $\mathcal{B}$ answers the decryption queries using the same procedures stated in the previous decryption queries phase 1.

- Guess phase: After the decryption queries phase 2 is finished, $\mathcal{A}$ would make a guess $c' \in \{0, 1\}$ about $c$, and sends his guess to $\mathcal{B}$. $\mathcal{B}$ outputs 1 if and only if $c = c'$, otherwise, it outputs 0.

**Analysis.** Let's consider the probability that our algorithm $\mathcal{B}$ would output 1, that is, $\mathcal{A}$ succeeds in the previous game. We analyze this probability under the following two different cases:

1. When $Z \neq g^{ab}$, easily, the probability $\Pr[c = c']$ in this case should be $\frac{1}{2}$ for the reason that the challenger ciphertext is a random element from the view of $\mathcal{A}$, $\mathcal{A}$ cannot get any useful information from it.

2. When $Z = g^{ab}$ and also the intended receiver's identity is $ID_j$, we find that the challenge ciphertext produced by $\mathcal{B}$ is valid. According to our previous analysis, our simulator would reject valid given ciphertext(s) with probability at most $\frac{q_D}{2^k}$ during simulating the decryption process, which means that with probability at least $1 - \frac{q_D}{2^k}$, $\mathcal{B}$ would do a perfect simulation during the previous

game played by $\mathcal{A}$ and $\mathcal{B}$. Also, with probability $\frac{1}{n}$, $\mathcal{A}$ would choose $pk_j$ as the receiver's public key. As $\mathcal{A}$ can break our scheme with non-negligible probability $\epsilon$, the challenge ciphertext is valid when the given tuple is a DDH tuple and the receiver's public key is $pk_j$. In this case, the probability $\Pr[c = c']$ should be at least $\frac{1}{2} + \frac{1}{n}(1 - \frac{q_D}{2^k})\epsilon$.

As $\epsilon$ is non-negligible, $\mathcal{B}$ can solve the DDH problem with advantage at least $\frac{1}{n}(1 - \frac{q_D}{2^k})\epsilon$. Here we finish our proof. □

**Theorem 4.2** *Our proposed scheme holds sender privacy under the previously defined model assuming the DDH problem is hard in* $\mathbb{G}$ *where hash functions* $H_1, H_2, H_3$ *are modeled as random oracles. Concretely, if there exists such an adversary* $\mathcal{A}$ *which can break our scheme with non-negligible probability* $\epsilon$*, supposing* $\mathcal{A}$ *makes at most* $q_{H_1}$,$q_{H_2}$,$q_{H_3}$ *queries to the* $H_1, H_2, H_3$ *hash oracles respectively, and* $q_{se}$ *sender extraction queries, then we can construct another algorithm that solves the DDH problem in* $\mathbb{G}$ *with probability at least* $\frac{1}{n}(1 - \frac{q_{se}}{2^k})\epsilon$.

**Proof.** We show how to construct $\mathcal{B}$ which solves the DDH problem by interacting with $\mathcal{A}$ under our predefined model. For the sake of simplifying the description of this proof, we omit the procedures identical to those in the previous message confidentiality security proof.

- Setup phase: This phase is the same as that in the previous security proof.
- $H_1$,$H_2$,$H_3$-queries phase: When answering those three hash queries, $\mathcal{B}$ does the same as himself in the previous security proof.
- Sender extraction queries phase 1: This phase is almost the same as the decryption queries phase 1 described in the previous security proof except that $\mathcal{B}$ returns the identity of the user who generates the given ciphertext. As what we have discussed before, we still need to calculate the probability that $\mathcal{B}$ would make a perfect decryption simulation in this phase. From the previous analyze in the proof of the message confidentiality, we know that this probability is at least $1 - \frac{q_{ID}}{2^k}$.
- Challenge phase: After the sender extraction queries are properly answered, $\mathcal{A}$ chooses one message $m$ from $\mathcal{M}$, $pk_j$ from $\mathcal{PK}$ as the receiver's public key and two subsets $\mathcal{SPK}, \mathcal{RPK}$ from $\mathcal{PK}$ such that $pk_j \in \mathcal{RPK}, |\mathcal{SPK}|, |\mathcal{RPK}| \geq 2$. $\mathcal{A}$ sends them to $\mathcal{B}$. Upon receiving those information, $\mathcal{B}$ would first check whether the intended receiver is the user with public key $pk_j = g^a$, if not, $\mathcal{B}$ aborts the game and returns a random bit. Otherwise, $\mathcal{B}$ randomly chooses a index $inx$ from the indexes of the chosen subset $\mathcal{SPK}$, and encrypts $m$ using

public key $pk_j$ and the secret of the sender who has the chosen index $inx$. $\mathcal{B}$ asks $H_1$ query about the message tuple $(m, g^{S_{inx}}, g^a, g^b, g^{S_{inx}\cdot a}, Z, g^{S_{inx}\cdot b})$ to get $r_2^*$, asks $H_2$ query about the message $Z\cdot g^{b\cdot r_2^*}$ to get $h_2^*$, asks $H_3$ query in order to generate all the required proofs. Eventually, $\mathcal{B}$ sends the generated ciphertext $(g^b, g^{S_{inx}\cdot a}, Z, g^{S_{inx}\cdot b}, g^{r_2^*}, m \oplus h_2^*, \mathcal{SPK}, \{c\}, \{z\}, \mathcal{RPK}, \widehat{\{c\}}, \widehat{\{z\}}, \{pf\})$ to $\mathcal{A}$ as the challenge ciphertext. Then $\mathcal{B}$ adds the message tuple $((m_c, g^{S_{inx}}, g^a, g^b, g^{S_{inx}\cdot a}, Z, g^{S_{inx}\cdot b}), r_2^*, Z\cdot g^{b\cdot r_2^*}, h_2^*)$ as a new row to the $H^{table}$ table.

- Sender extraction queries phase 2: In this phase, $\mathcal{A}$ can still ask sender extraction queries with the only constraint that $\mathcal{A}$ cannot use the challenge ciphertext as one of his queried messages. $\mathcal{B}$ interacts with $\mathcal{A}$ in the same manner as we described in the decryption queries phase 2 of the previous security proof.

- Guess phase: After finishing the sender extraction phase 2, $\mathcal{A}$ would make a random index guess $inx'$ from the indexes of the chosen subset about $inx$, and sends his guess to $\mathcal{B}$. Algorithm $\mathcal{B}$ outputs 1 if and only if $inx = inx'$, otherwise, it outputs 0.

**Analysis.** Let's consider the probability that our algorithm $\mathcal{B}$ would output 1, that is, $\mathcal{A}$ succeeds in the previous game. We analyze this probability under the following two different cases:

1. When $Z \neq g^{ab}$, the probability $\Pr[inx = inx']$ in this case should be $\frac{1}{|\mathcal{SPK}|}$ for the reason that the challenger ciphertext is random from the view of $\mathcal{A}$, $\mathcal{A}$ cannot get any useful information from it, the best choice for him is to make a random guess.

2. When $Z = g^{ab}$ and also the intended receiver's identity is $ID_j$, we can find that the challenge ciphertext produced by $\mathcal{B}$ is valid. According to our previous analysis, our simulator would reject valid given ciphertext(s) with probability at most $\frac{q_{se}}{2^k}$ during simulating the sender extraction process, which means that with probability at least $1 - \frac{q_{se}}{2^k}$, $\mathcal{B}$ would do a perfect simulation during the previous game played by $\mathcal{A}$ and $\mathcal{B}$. Also, with probability $\frac{1}{n}$, $\mathcal{A}$ would choose $pk_j$ as the receiver's public key. As $\mathcal{A}$ can break our scheme with non-negligible probability $\epsilon$, the challenge ciphertext is valid when the given tuple is a DDH tuple and the receiver's public key is $pk_j$, in this condition, the probability $\Pr[inx = inx']$ should be at least $\frac{1}{|\mathcal{SPK}|} + \frac{1}{n}(1 - \frac{q_{se}}{2^k})\epsilon$

As $\epsilon$ is non-negligible, $\mathcal{B}$ can determine whether the given tuple is a valid DDH tuple with probability at least $\frac{1}{n}(1 - \frac{q_{se}}{2^k})\epsilon$. Here we finish our proof. □

**Theorem 4.3** *Our scheme holds receiver privacy under the predefined security model assuming the DDH problem is hard in* $\mathbb{G}$ *when hash functions* $H_1, H_2, H_3$ *are*

*modeled as random oracles. That is, if there is an adversary $\mathcal{A}$ which can break our scheme with non-negligible probability $\epsilon$, assuming $\mathcal{A}$ asks $q_{H_1}, q_{H_2}, q_{H_3}$ queries to $H_1, H_2, H_3$ respectively and $q_{re}$ receiver extraction queries during the game, then we can construct another algorithm $\mathcal{B}$ which breaks the DDH problem with probability $\frac{1}{n}(1 - \frac{q_{re}}{2^k})\epsilon$, where $n$ is a constant.*

**Proof.** The proof of theorem 3 is nearly identical to that of theorem 2 except that we require that the sender of the challenge ciphertext should be the one with public key $g^a$. □

## 4.5 Summary

In this chapter, we consider the user conditional privacy preservation problem, which is stated clearly at the beginning of our thesis, in a more complex scenario. When facing with the scenario where the authority can block the communication channels among several users by publishing a blacklist as public parameter, we also need to address the issue of proving the legitimation of the communication channel between it and its communicator. To solve the aforementioned two problems, we propose a group-based source-destination verifiable encryption scheme with blacklist checking. In order to discuss the security of our scheme, we further define three security models to capture the message confidentiality, sender privacy preservation and receiver privacy preservation properties accordingly, and then give three formal proofs under the predefined models with the help of the random oracle.

The technique used to protect users' privacy in our work may suffer from the so called cross-comparison attack and the joint conspiracy attack mentioned in [KBK+11] and [AMM99] respectively. Thus, our next work should focus on looking for another advanced privacy preservation method to make our scheme more secure under some specific attacks.

# Chapter 5

# Publicly Verifiable Secure Communication with User and Data Privacy

Security surveillance system plays an important role in the society. However, how to securely send the sensitive information from the surveillance node to the server is a critical issue which should be well addressed. In this chapter, to develop a secure communication scheme applied between the surveillance camera and the server, we propose the important and desirable security and privacy features that should be achieved by such systems, and present a secure scheme that can achieve the security goals. Our scheme ensures that encrypted datagrams not sent from the surveillance cameras can be filtrated by a public message filter while data and sender privacy is still well preserved for encrypted data sent from legitimated cameras. Furthermore, the server in our scheme is the only entity who can reveal the real sender given a ciphertext produced by it and give a proof to convince others the origination of that ciphertext without leaking its content. Such property enables the server to build a searchable database using the camera's identifier as index and also the message auditor to check the ciphertext and its origination stored in the database without any dispute. We provide the formal security models to define these security requirements and give formal security proofs in the random oracle model.

## 5.1 Introduction

**Background.** Security surveillance system is pervasively applied in many domains. For example, the transportation cameras are installed in intersections or along the highways to monitor whether a deriver is over-speeding or violating some other traffic regulations. Another example is the home security surveillance system

where cameras are installed to detect whether your home is invaded by someone who does not have the authorization. In a security surveillance system, a surveillance node records sensitive information of its monitored area at a specific frequency or when misbehavior is detected in that area, then it sends the information to a central server. The server processes those information using automated programs, stores them in a searchable database in case of any dispute and triggers further actions under certain condition. For example, the server in the transportation surveillance system would send a speeding ticket to a driver, and in the home security surveillance system, the server would send an alert to the householder when the camera in that house detects an intrusion behavior. Due to the sensitivity of information captured by security surveillance cameras, the problem of how to securely send the sensitive information from the surveillance node to the server is critical and needs to be treated and addressed seriously. In this chapter, we are focusing on designing a secure communication scheme applied between the surveillance node and the server in the security surveillance system.

**Our System Model.** Before discussing the aforementioned problem in detail, we first give a brief description of our security surveillance system model. Specifically, our surveillance system model consists of a surveillance server, a message auditor, a message filter and a number of surveillance nodes located in different areas.

As illustrated in Fig.5.1., the surveillance nodes in different locations are responsible for capturing sensitive information and sending them to the surveillance server via the public network. To reduce the server's cost on analyzing meaningless information, a message filter is enforced to ensure only information sent from the surveillance nodes can bypass it and be received by the server. The surveillance server takes the role of analyzing the information and triggering further actions depending on the analyzed result. After the information is processed, it needs to be stored in a searchable database in case of any dispute happened in the future. A message auditor is also necessary to ascertain that the original data sent by a camera is intact when it is retrieved latter.

**System Security Requirements.** Basing on the surveillance model described above, we summarize the security requirements in this system. Those requirements include:

- **Message confidentiality.** Obviously, when a surveillance node captures a misbehavior, it needs to send the sensitive information to the server without leaking its content.

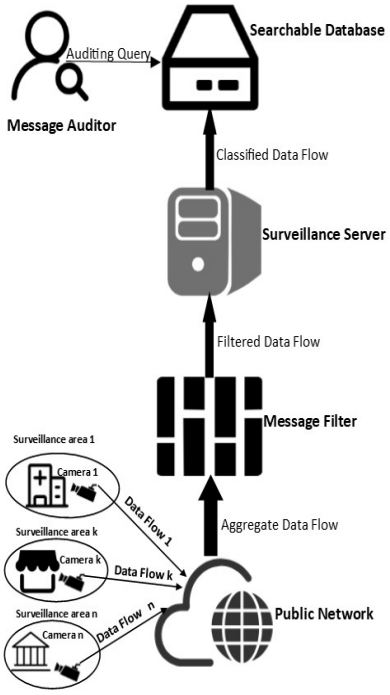


**Figure 5.1:** The security surveillance system model

- **Message public verifiability.** The message filter should be able verify whether one encrypted message is sent from a legitimated surveillance node without requiring any extra information, which means that the encrypted message should be publicly verifiable before it is processed by the surveillance server.

- **Node privacy.** The exact location of the surveillance node is also a crucial information to that node and then can be viewed as the unique identifier of it. We require that this information should be included in the transmitted encrypted message and cannot be discovered by any other users except the server. The server can reveal the identity of one certain node and build the searchable database using the node's location as index. Since the encrypted message are transmitted over the public network, there may exist misbehaved nodes which have the capability to intercept the encrypted message sent from specific surveillance nodes when the origination of that encrypted message is known, this is another reason why camera privacy is necessary.

- **Node authentication.** Since the message filter is empowered with the capability to filtrate messages not sent from surveillance nodes, each node should be able to authenticate its legitimization to the message filter.

- **Node non-repudiation.** The property node non-repudiation states that if a surveillance node have sent a message to the server, it cannot deny this behavior latter. It appears that defining this requirement in our model is not that straight-forward, so we divide it into three sub-requirements, namely, the outsider impersonation resistance, the insider impersonation resistance and receiver cheating resistance. Intuitively, if all the three requirements are satisfied in our scheme, then the non-repudiation requirement is also satisfied.

- **Data Privacy.** Given an encrypted message and its corresponding proof produced by the server, the message auditor and other users can verify the node non-repudiation property of that encrypted message, while the verification would not leak anything about the content of the sensitive information.

**Potential solution and its insufficiencies.** It seems that the signcryption scheme is promising to provide some of those aforementioned security requirements.

The primitive signcryption, first proposed in the seminal work [Zhe97], is a cryptographic tool which combines the functionalities of both the public key encryption scheme and signature scheme in a logical single step. When trying to provide desirable properties such as message confidentiality, integrity, non-repudiation and authentication simultaneously, the signcryption scheme enjoys extra benefits with respect to computational costs and communication overheads comparing to the traditional sign-then-encrypt approach. The security of signcryption scheme had never been discussed in detail until a formal security model in the multi-user setting was given by Baek et al. in [BSZ02], and since then, many efficient signcryption schemes which are proven secure have been proposed in [MM03, LYW+07, HRS14, EZ15]. However, we find that existing signcryption schemes are insufficient to address all the security goals above.

Specifically, the verification of the ciphertext in existing signcryption schemes such as [BD98, GLZ99, Ma06, SVR10] still requires extra information, which is given after the ciphertext is processed by its receiver. For example in [SVR10], the public keys of the sender and intended receiver of the given ciphertext are the required extra information. The ciphertext in signcryption schemes such as [LYW+07, HRS14] can be publicly verified to ascertain whether it is generated by one specific user, so the user's privacy cannot be properly preserved. In existing signcryption schemes such as [LQ04, LHZM10], the assurance of non-repudiation needs the involvement of the ciphertext and its corresponding plaintext, which causes the compromise of message privacy.

**Related work.** The problem how to publicly verify the ciphertext of a signcryption have been discussed decades ago. In [BD98], Bao et. al. proposed the first

signcryption scheme with public verifiability. However, in their scheme the ciphertext cannot be verified until its corresponding plaintext is given, which incurs the lost of the data privacy, this problem also exists in [Ma06]. Gamage et. al. presented a publicly verifiable signcryption used in firewalls in [GLZ99], while their scheme only consider single-user setting and the verification of the ciphertext still needs the signcrypter's public key, which incurs the lost of the user privacy, the scheme in [SVR10] and [Aim11] also has the same drawback as that in [GLZ99].

The primitive ring signature [RST01], group signature [CvH91] and ring signcryption [HSMZ05a] all provide the user with the capability to authenticate its legitimation among a group of users including itself, that is, those primitives all guarantee the user privacy during the authentication. However, each of them has insufficiency in revealing the anonymity of the user who produces a given signature or ciphertext. Specifically, in ring signatures schemes such as [FS07, ALSY06, ALSY13], one signer's anonymity can be revoked by everyone unless it issues signatures twice or more, such anonymity revocation would not work when the user is not always well-behaved. In the ring signature proposed in [LLM$^+$07], only a set of trusted authorities is allowed to revoke the anonymity of the real signer, in group signature schemes, the group manager is the only authority which can revoke the anonymity of the signer of a given signature, the revocation of the anonymity of the signer in those schemes need the participation of the trusted authority, and it is costly to maintain its availability. The problem how to revoke the anonymity of the signcrypter of a ciphertext in ring signcryption schemes is seldom considered so far, the only two work [ZYZZ08, LST08] present a ring signcryption scheme in which the actual signcrypter can prove to others that a ciphertext is generated by itself when such proof is needed, that is, only the signcrypter itself of a ciphertext can revoke its own anonymity.

**Our Contribution.** Motivated by the insufficiencies of existing signcryption schemes and the user and data privacy requirements in the security surveillance system, we present our publicly verifiable secure communication scheme with user and data privacy. This work contributes to the development of secure communication in the security surveillance system in the following two aspects.

1. The integrity and authentication of the ciphertext in our secure communication scheme are publicly verifiable and the verification does not need any extra information except the public parameters. This property benefits our scheme in enabling the message filter to filtrate information sent not from the surveillance nodes without requiring any secret.

2. Since the surveillance node in our secure communication scheme is authenti-

cated as a member of a group of legitimated nodes rather than a specific one, the node's privacy is still preserved while its legitimation is authenticated. Besides, the surveillance server in our scheme can reveal the identity of the node given a ciphertext produced by it and also give a proof to convince others about the origin of that ciphertext, and this proof would not leak any information about the plaintext, namely, the data privacy is also preserved. With this property, only the surveillance server can build the searchable database using the nodes' location as index, which is desirable in the surveillance system. Also, the message auditor and other users can retrieve the ciphertexts stored in the searchable database conveniently and verify the non-repudiation property of a ciphertext latter without requiring any secret or learning the underlying plaintext.

## 5.2 Scheme Definition and Security Models

### 5.2.1 Defining the Scheme

Our scheme is applicable among a group of users and a trusted group authority denoted by TGA. It can be defined with the following polynomial-time algorithms:

- Setup: The Setup algorithm is executed by the group authority TGA. On input the security parameter, the algorithm outputs the group public parameter PM. PM should include a user certificate repository $\mathcal{CR}$ which is initially empty.
- Registration: This Registration algorithm is executed between one group user $U_i \in \{\mathsf{U}\}$ and the TGA. When $U_i$ wants to join a group managed by the TGA, the algorithm works as following;
  - $U_i$ first produces its own signcrypiton key and unsigncryption key pair $(sik_i, unsk_i)$, then $U_i$ commits to the key pair $(sik_i, unsk_i)$ and produces the commitment $(pk_i, comsk_i)$. $U_i$ further generates a zero knowledge proof $zkpk_i$ showing it knowledge of $(sik_i, unsk_i)$ with respect to the commitment $comsk_i$.
  - $U_i$ sends $rm_i = (pk_i, comsk_i, zkpk_i)$ to TGA.
  - Upon receiving the message tuple $rm_i$ sent from $U_i$, TGA first verifies $zkpk_i$ to check whether $U_i$ satisfies certain group qualification $qul$ defined in PM, if yes, TGA computes $U_i$'s unique certificate $u_i$ from $rm_i$ and updates $\mathcal{CR}$ to include $pk_i, u_i$ as this user $U_i$'s personal certificate, then it sends $u_i$ back to $U_i$. Otherwise, TGA rejects such registration request and responds $U_i$ with a message saying "REGISTRATION REJECT ".

- Signcrypt: To send a message $m$ to $U_j$ on behalf of a group, $U_i$ executes this Signcrypt algorithm, which takes $(sik_i, u_i, pk_j, m)$ as inputs, and outputs a ciphertext $\sigma_i$. For simplicity, we say $U_i$ is the *signcrypter* of $\sigma_i$.

- Verify: Given a ciphertext $\sigma_i$ and the public parameter PM, this Verify algorithm outputs "1" if the *signcrypter* of $\sigma_i$ satisfies the group qualification *qul* stated in PM, otherwise, it outputs "0". This algorithm can be executed by anyone possessing PM and a ciphertext, it enables one party to check whether a given ciphertext is generated by a registered user of the legitimated group without revealing this user's privacy.

- Unsigncrypt: When $U_j$ receives a ciphertext $\sigma_i$, it executes the Unsigncrypt algorithm with inputs $(\sigma_i, unsk_j)$ to recover the plaintext $m$ when $U_j$ is the intended receiver of $\sigma_i$, otherwise, the algorithm outputs a symbol $\perp$.

- Open: Given a ciphertext $\sigma_i$, the intended receiver $U_j$ of $\sigma_i$ can also trace the real *signcrypter* of $\sigma_i$ using algorithm Open when $\mathsf{Unsigncrypt}(\sigma_i, unsk_j) \neq \perp$. Taking $(\sigma_i, unsk_j)$ as inputs, Open outputs the personal certificate $u_i'$ of the *signcrypter* of $\sigma_i$ and also a zero-knowledge proof $pkid_j$ showing that $U_j$ does not cheat when executing this algorithm. Here, this algorithm enables the intended receiver of $\sigma_i$ to not only disclose the privacy of the *signcrypter* of $\sigma_i$ but also convince other users that it acts honest during finding the actual *signcrypter* of $\sigma_i$.

- Justify: On input the tuple $(\sigma_i, u_i', pkid_j)$, this Justify algorithm would output the symbol "1" if the user with certificate $u_i'$ is the actual *signcrypter* of $\sigma_i$, otherwise, it outputs "0".

### 5.2.2 Formal Security Notions

**Definition 5.1 (Message Confidentiality)** *To define the security of our scheme with respect to message confidentiality, we refer to the standard definition of indistinguishability under adaptive chosen ciphertext attack* (IND-CCA2) *for public key encryption scheme[CS03c, BDPR98b], and present the following* m-IND-CCA2 *game played between a challenger* $\mathcal{C}$ *and an adversary* $\mathcal{A}$*.*

- Setup: *The challenger* $\mathcal{C}$ *takes the security parameter* $\lambda$ *as input and runs the* Setup, Registration *algorithms of our scheme to generate the group public parameter* PM *and* $t$ *users' keys.* $\mathcal{C}$ *sends* PM, $\{U\}$, $\mathcal{CR}$ *to the adversary* $\mathcal{A}$*, where* $\{U\}$ *is the collection of all the* $t$ *users and* $\mathcal{CR}$ *is the user certificate repository including all* $t$ *registered users' personal certificate.*

- Query phase 1: *In this phase, $\mathcal{A}$ can perform a number of queries bounded by a polynomial in an adaptive way, that is, each query can be depended on the responses to the previous queries. There are five types of queries which can be asked and each of them can be described as below.*

  - Signcryption key query: *$\mathcal{A}$ can select a user $U_i$ from $\{U\}$, $\mathcal{C}$ needs to find its corresponding signcryption key $sik_i$ and sends it back to $\mathcal{A}$.*
  - Unsigncryption key query: *$\mathcal{A}$ can select a user $U_i$ from $\{U\}$, $\mathcal{C}$ needs to find its corresponding unsigncryption key $unsk_i$ and sends it back to $\mathcal{A}$.*
  - Signcrypt query: *$\mathcal{A}$ chooses a message $m$ and two users $U_i, U_j \in \{U\}$ such that $(pk_i, u_i)$, $(pk_j, u_j) \in \mathcal{CR}$ as the message sender and receiver respectively, then $\mathcal{A}$ sends them to $\mathcal{C}$. $\mathcal{C}$ computes $\sigma_i = \mathsf{Signcrypt}(sik_i, u_i, pk_j, m)$ and sends $\sigma_i$ back to $\mathcal{A}$.*
  - Unsigncrypt query: *$\mathcal{A}$ chooses a ciphertext $\sigma_i$ and a user $U_j \in \{U\}$, then it sends $(\sigma_i, U_j)$ to $\mathcal{C}$. By first executing $\mathsf{Verify}(\sigma_i, \mathsf{PM})$ and then $\mathsf{Unisigncrypt}(\sigma_i, unsk_j)$, $\mathcal{C}$ responds $\mathcal{A}$ with the resulting plaintext of $\sigma_i$ if $\sigma_i$ is a valid ciphertext and $U_j$ is the intended receiver of $\sigma_i$, otherwise $\mathcal{C}$ returns nothing to $\mathcal{A}$.*
  - Open query: *$\mathcal{A}$ chooses a ciphertext $\sigma_i$ and a user $U_j \in \{U\}$, then it sends $(\sigma_i, U_j)$ to $\mathcal{C}$. If $\sigma_i$ is a valid ciphertext and $U_j$ is the intended receiver of $\sigma_i$, $\mathcal{C}$ responds $\mathcal{A}$ with $u_i$, which is the unique certificate of the signcrypter of $\sigma_i$, as well as a proof $pkid_j$, which convinces others that $U_j$ behaves honest when executing* Open *algorithm. otherwise $\mathcal{C}$ responds $\mathcal{A}$ with nothing.*

- Challenge: *After $\mathcal{A}$ decides to end the* Query phase 1*, it outputs two plaintexts $m_0, m_1$ satisfying $|m_0| = |m_1|$ and two users $U_i, U_j \in \{U\}$ as the message sender and receiver respectively. It requires that $U_j$ should not appear in any unsigncryption key queries in the* Query phase 1*, then it sends the chosen tuple $(m_0, m_1, U_i, U_j)$ to $\mathcal{C}$. Upon receiving the tuple, $\mathcal{C}$ picks a random bit $b$ from $\{0, 1\}$ and computes $\sigma = \mathsf{Signcrypt}(sik_i, u_i, pk_j.m_b)$ and returns this challenge ciphertext to $\mathcal{A}$.*

- Query phase 2: *In this phase, $\mathcal{A}$ can still ask a polynomially bounded number of queries adaptively again, and $\mathcal{C}$ acts the same as in the above* Query phase 1*. Here, the restriction is that it cannot make unsigncryption key query on $U_j$ and also cannot make an* Unsigncrypt query *on the tuple $(\sigma, U_j)$ where $\sigma$ is the challenge ciphertext and $U_j$ is its intended receiver.*

- Guess: *After $\mathcal{A}$ decides to end the* Query phase 2*, it has to output a guess $b'$. It wins the aforementioned game if $b' = b$.*

*Let $\Pr[b = b']$ denote the probability that $\mathcal{A}$ wins the game and $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{m-IND-CCA2}}$ the advantage of $\mathcal{A}$ when it wins the game, then*

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{m-IND-CCA2}} = 2|\Pr[b = b'] - 1|.$$

*We say our scheme is* m-IND-CCA2 *secure if $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{m-IND-CCA2}} \leq negl(\lambda)$ for any PPT adversary $\mathcal{A}$ where $negl(\lambda)$ is a negligible function in $\lambda$.*

**Definition 5.2 (Sender Anonymity)** *To define the security of our scheme with respect to sender anonymity, we present the following* an-CCA2 *game, which is very similar to the previously defined* m-IND-CCA2 *one, played between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$,*

- Setup: *The challenger $\mathcal{C}$ takes the security parameter $\lambda$ as input and runs the* Setup, Registration *algorithms of our scheme to generate the public parameter* PM *and $t$ users' keys. $\mathcal{C}$ sends* PM*, $\{U\}$, $\mathcal{CR}$ to the adversary $\mathcal{A}$, where $\{U\}$ is the collection of all the $t$ users and $\mathcal{CR}$ is a user certificate repository including all $t$ registered users' personal certificate.*

- Query phase 1: *In this phase $\mathcal{A}$ interacts the same with $\mathcal{C}$ as what they do in the* Query phase 1 *of the previous message confidentiality security model.*

- Challenge: *After $\mathcal{A}$ decides to end the* Query phase 1*, it outputs one message $m$ and one user $U_j \in \{U\}$ as the intended receiver. It requires that $U_j$ should not appear in any unsigncryption key queries in the* Query phase 1*, then it sends the chosen tuple $(m, U_j)$ to $\mathcal{C}$. Upon receiving the tuple, $\mathcal{C}$ chooses one index $b \in [t]$ and sets user $U_b \in \{U\}$ as the signcrypter, then it computes $\sigma = \mathsf{Signcrypt}(sik_b, u_b, pk_j, m)$ and returns this challenge ciphertext to $\mathcal{A}$.*

- Query phase 2: *In this phase, $\mathcal{A}$ can still ask a polynomially bounded number of queries adaptively again, and $\mathcal{C}$ acts identical to what it does in the above* Query phase 1*. Here, the only restriction is that it cannot make unsigncryption key query on $U_j$ and also cannot make an* Unsigncrypt query *on the tuple $(\sigma, U_j)$ where $\sigma$ is the challenge ciphertext and $U_j$ is its intended receiver.*

- Guess: *After $\mathcal{A}$ decides to end the* Query phase 2*, it has to output a guess $b' \in [t]$. It wins the aforementioned game if $b' = b$.*

*Let* $\Pr[b = b']$ *denote the probability that* $\mathcal{A}$ *wins the game and* $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{an-IND-CCA2}}$ *the advantage of* $\mathcal{A}$ *when it wins the game, then*

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{an-IND-CCA2}} = 2|\Pr[b = b'] - \frac{1}{t}|.$$

*We say our scheme is* an-CCA2 *secure if* $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{an-IND-CCA2}} \leq negl(\lambda)$ *for any PPT* $\mathcal{A}$ *where* $negl(\lambda)$ *is a negligible function in* $\lambda$*.*

**Definition 5.3 (Outside User Impersonation Resistance)** *To define the security of our scheme with respect to the property that one user outside the system cannot impersonate a legitimated user successfully, we adapt the standard security definition of existential unforgeability under adaptive chosen message attack* (UF-CMA)*[GMR88, PS96] for signature schemes to our scheme setting and present the following* O-UF-CMA *game played between a challenger* $\mathcal{C}$ *and an adversary* $\mathcal{A}$*,*

- Setup: *The challenger* $\mathcal{C}$ *takes the security parameter* $\lambda$ *as input and runs the* Setup, Registration *algorithms of our scheme to generate the group public parameter* PM *and* $t$ *users' keys.* $\mathcal{C}$ *sends* PM, $\{U\}$ *to the adversary* $\mathcal{A}$*, where* $\{U\}$ *is the collection of all the* $t$ *users. Here, as an outsider,* $\mathcal{A}$ *should not have* $\mathcal{CR}$*.*

- Query phase: *In this phase,* $\mathcal{A}$ *can perform a number of queries bounded by a polynomial in an adaptive way. As an outsider,* $\mathcal{A}$ *can ask all types of query mentioned in previous models except the* signcryption key query*.*

- Forgery: *After* $\mathcal{A}$ *decides to end the* Query phase*, it tries to forge a ciphertext* $\sigma$ *with* $U_j$ *as the message receiver, it requires that* $U_j \in \{U\}$*, then it sends the forgery* $(\sigma, U_j)$ *to* $\mathcal{C}$*.* $\mathcal{A}$ *wins the game if* $\mathsf{Verify}(\sigma, \mathsf{PM}) = 1$*.*

*Let* $\Pr[\mathcal{A}\ wins]$ *denote the probability that* $\mathcal{A}$ *wins the above game and* $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{O-UF-CMA}}$ *the advantage of* $\mathcal{A}$ *when it wins the game, then*

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{O-UF-CMA}} = \Pr[\mathcal{A}\ wins].$$

*We say our scheme is* O-UF-CMA *secure if* $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{O-UF-CMA}} \leq negl(\lambda)$ *for any PPT* $\mathcal{A}$ *where* $negl(\lambda)$ *is a negligible function in* $\lambda$*.*

**Definition 5.4 (Inside User Impersonation Resistance)** *To define the security of our scheme with respect to the property that a legitimated user in the system cannot impersonate another one successfully, we adapt the standard definition of existential unforgeability under adaptive chosen message attack* (UF-CMA) *for signature scheme to our scheme setting and present the following* I-UF-CMA *game played between a challenger* $\mathcal{C}$ *and an adversary* $\mathcal{A}$*,*

- Setup: *The challenger $\mathcal{C}$ takes the security parameter $\lambda$ as input and runs the* Setup, Registration *algorithms of our scheme to generate the group public parameter* PM *and $t$ users' keys. $\mathcal{C}$ sends* PM, $\{U\}$, $\mathcal{CR}$ *to the adversary $\mathcal{A}$, where $\{U\}$ is the collection of all the $t$ users and $\mathcal{CR}$ is a user certificate repository including all $t$ registered users' personal certificate. In this phase, $\mathcal{A}$ needs to choose one user $U_i \in \{U\}$ as the user it wants to challenge before the* Forgery *phase and sends it back to $\mathcal{C}$.*

- Query phase: *The types of queries can be asked in this phase is identical to that in the* Query phase 1 *of the previous message confidentiality security model.*

- Forgery: *After $\mathcal{A}$ decides to end the* Query phase*, it tries to forge a ciphertext $\sigma$ with $U_j$ as its receiver, it requires that $U_j \in \{U\}$, then it sends the forgery $(\sigma, U_i, U_j)$ to $\mathcal{C}$. $\mathcal{A}$ wins the game if* $\mathsf{Verify}(\sigma, \mathsf{PM}) = 1, \mathsf{Unsigncrypt}(\sigma, unsk_j) \neq \perp$ *and the signcryption key of $U_i$ is never queried in* Query phase:*.*

*Let* $\Pr[\mathcal{A}\ wins]$ *denote the probability that $\mathcal{A}$ wins the game and* $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{I-UF-CMA}}$ *the advantage of $\mathcal{A}$ when it wins the game, then*

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{I-UF-CMA}} = \Pr[\mathcal{A}\ wins].$$

*We say our scheme is* I-UF-CMA *secure with respect to the sender impersonation resistance property if* $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{I-UF-CMA}} \leq negl(\lambda)$ *for any PPT $\mathcal{A}$ where $negl(\lambda)$ is a negligible function in $\lambda$.*

**Definition 5.5 (Receiver Cheating Resistance)** *To define the security of our scheme relating to receiver cheating resistance property, we present the following* (R-UF-CMA) *game, which is similar to the predefined* (I-UF-CMA) *game, played between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$,*

- Setup: *The challenger $\mathcal{C}$ takes the security parameter $\lambda$ as input and runs the* Setup, Registration *algorithms of our scheme to generate the group public parameter* PM *and $t$ users' keys. $\mathcal{C}$ sends* PM, $\{U\}$, $\mathcal{CR}$ *to the adversary $\mathcal{A}$, where $\{U\}$ is the collection of all the $t$ users and $\mathcal{CR}$ is a user certificate repository including all $t$ registered users' personal certificate.*

- Query phase: *This phase shares no difference with the* Query phase 1 *of the previous message confidentiality security model.*

- Challenge: *After $\mathcal{A}$ decides to end the* Query phase*, it chooses a message $m$, one users $U_j$ as the intended receiver and then sends them to $\mathcal{C}$. Upon receiving the tuple $(m, U_j)$, $\mathcal{C}$ chooses a user $U_i$ satisfying $U_i \in \{U\}$ and $i \neq j$, then it computes $\sigma = \mathsf{Signcrypt}(sik_i, u_i, pk_j, m)$. $\mathcal{C}$ sends $(\sigma, U_j)$ back to $\mathcal{A}$.*

- Forgery: *When $\mathcal{A}$ receives $\sigma$, it is asked to forge a tuple $(\sigma, u_l, zkid_j)$, We say $\mathcal{A}$ wins the game if* $\mathsf{Justify}(\sigma, u_l, zkid_j) = 1 \wedge u_l \neq u_i$.

Notice. *In the challenge and forgery phase, we do not put any restriction on $U_i, U_j$, $U_l$, that is, the signcryption and unsigncryption key queries on those users are all allowed in our model.*
*Let* $\Pr[\mathcal{A}\ wins]$ *denote the probability that $\mathcal{A}$ wins the game and* $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{R-UF-CMA}}$ *the advantage of $\mathcal{A}$ when it wins the game, then*

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{R-UF-CMA}} = \Pr[\mathcal{A}\ wins].$$

*We say our scheme is* R-UF-CMA *secure with respect to the receiver cheating resistance property if* $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{R-UF-CMA}} \leq negl(\lambda)$ *for any PPT adversary $\mathcal{A}$ where $negl(\lambda)$ is a negligible function in $\lambda$.*

## 5.3 Our Concrete Construction

In this section we give a concrete construction of our proposed scheme. Consisting of the following algorithms (Setup, Registration, Signcrypt, Verify, Unsigncrypt, Open, Justify), our scheme can be described in detail as;

- Setup: Assuming there exists a trusted group authority TGA which takes the security parameter $\lambda = (\delta, \hat{\ell}, \ell_1, \ell_2, k)$ satisfying $\delta > 1, \ell_2 < \ell_1, \ell_2 \gg \ell_1 - \frac{\hat{\ell}+\ell_1}{4}$ as input, it requires that any user who wants to register as a legitimated group member should choose its secret in such a manner that the chosen secret should be a prime which lies in $\{2^{\ell_1}, \cdots, 2^{\ell_1}+2^{\ell_2}-1\}$, and TGA values this requirement as the group membership qualification $qul$. TGA also initiate a user certificate repository $\mathcal{CR}$ to include all registered users' personal certificates which can be used to uniquely identify them. TGA also chooses two large enough prime numbers $p, q$ such that $p \equiv q \equiv 3 (\text{mod } 4)$, and then produces a multiplicative group $\mathbb{Z}_n^*$ and one quadratic-residue $g$ of that group such that $n = pq, |g| = \ell_g$, TGA also gets a cyclic group $G =< g >$, it is required that $|n| = \ell_n > \ell_1$. TGA further chooses a random element $z \in \mathbb{Z}_n^*$ and two hash functions $\mathcal{H}_1 : \{0,1\}^* \rightarrow \{0,1\}^k, \mathcal{H}_2 : G \rightarrow \{0,1\}^\ell$. It define the message space $\mathcal{M} := \{0,1\}^\ell$ and ciphertext space $\mathcal{CT} := G^6 \times \{0,1\}^\ell \times \{0,1\}^k \times \{0,1\}^{\delta(\ell_2+k)} \times \{0,1\}^{\delta(\ell_g+\ell_1+k)}$, and then sets $\mathsf{PM} = (\lambda, qul, \mathcal{CR}, n, g, z, \mathcal{M}, \mathcal{CT}, \mathcal{H}_1, \mathcal{H}_2)$.

- Registration: When a user, denoted by $U_i$, wishes to be a group member, it interacts with the TGA to complete the following registration procedures;

  - $U_i$ first chooses $x_i \in [n]$ and two prime numbers $e_i, \hat{e}_i \in \{2^{\ell_1}, \cdots, 2^{\ell_1} + 2^{\ell_2} - 1\}$ randomly such that $e_i, \hat{e}_i \not\equiv 1 \text{ (mod 8)}$ and $e_i \not\equiv \hat{e}_i (\text{ mod 8})$. $U_i$

computes $\tilde{z}_i = z^{\hat{e}_i}$,$\tilde{e}_i = e_i\hat{e}_i, pk_i = g^{x_i}$ where $|\tilde{z}_i| = \tilde{\ell}$, and then generates the following two zero-knowledge proofs

$$
\begin{aligned}
p_1 &= \mathrm{ZPK}\{(e_i, \hat{e}_i, x_i) : z^{\tilde{e}_i} = \tilde{z}_i{}^{e_i} \wedge \tilde{z}_i = z^{\hat{e}_i} \wedge pk_i = g^{x_i} \wedge \\
&(2^{\ell_1} - 2^{\delta(\ell_2+k)+1}) < e_i < (2^{\ell_1} - 2^{\delta(\ell_2+k)+1}) \wedge \\
&(2^{\ell_1} - 2^{\delta(\ell_2+k)+1}) < \hat{e}_i < (2^{\ell_1} - 2^{\delta(\ell_2+k)+1})\}, \\
p_2 &= \mathrm{ZPK}\{(e_i, \hat{e}_i) : \tilde{e}_i = e_i\hat{e}_i \wedge e_i \in \mathrm{primes}(\lambda) \wedge \hat{e}_i \in \mathrm{primes}(\lambda)\}.
\end{aligned}
$$

Here the details about how to produce the proof values $p_1, p_2$ are given in [CM99], so we omit the description.Then, $U_i$ then sends $(\tilde{z}_i, \tilde{e}_i, pk_i, p_1, p_2)$ to TGA.

- Upon receiving the message tuple sent from $U_i$, TGA first verifies the two proof value $p_1, p_2$ by executing the corresponding verification algorithm described in [CM99] to check whether the secrets chosen by $U_i$ satisfies the requirement defined in $qul$, that is, to check whether $e_i, \hat{e}_i$ are prime numbers and also $e_i, \hat{e}_i \in \{2^{\ell_1}, \cdots, 2^{\ell_1} + 2^{\ell_2} - 1\}$.
- If the both verification algorithms output "1", TGA computes $u_i = \tilde{z}^{\frac{1}{\hat{e}}}$ and sets $u_i$ as $U_i$'s unique certificate, then TGA sends $u_i$ back to $U_i$. Otherwise, TGA ends the registration procedures and returns a symbol saying "REGISTRATION REJECT."
- After receiving $u_i$ from TGA, $U_i$ checks whether the equation $(u_i)^{e_i} = z$ holds, if yes, $U_i$ keeps it as its unique certificate and finishes this registration algorithm.
- Each time when a new user register itself as a legitimated group member successfully, this user $U_i$ stores two keys, its signcryption key $sik_i = e_i$ and unsigncryption key $unsk_i = x_i$, secretly. After that, TGA updates the certificate repository $\mathcal{CR}$ to include this new user's personal information $(pk_i, u_i)$.

- Signcrypt: To signcrypt a message $m \in \{0,1\}^{\ell}$ on the group's behalf and send the ciphertext to user $U_j$, $U_i$ does the following procedures;
  - It chooses $w_1, w_3 \in_R \{0,1\}^{\ell_g}$ and computes

$$
\begin{aligned}
&w_2 = \mathcal{H}_1(g||z||{y_j}^{e_i w_1 w_3}||m), \\
&c_1 = g^{w_2 w_3}, c_2 = g^{w_1 w_2 w_3}, c_3 = {y_j}^{w_2 w_3}, \\
&c_4 = u_i {y_j}^{w_1 w_2 w_3}, c_5 = g^{e_i w_1 w_3}, c_6 = g^{e_i w_1 w_2 w_3}, \\
&c_7 = m \oplus \mathcal{H}_2({y_j}^{e_i w_1 w_2 w_3}),
\end{aligned}
$$

where $e_i, u_i$ is $U_i$'s signcryption key and unique certificate respectively, $y_j$ is $U_j$'s public key.

– It chooses $r_1, r_3 \in_R \{0,1\}^{\delta(\ell_2+k)}, r_2, r_4 \in_R \{0,1\}^{\delta(\ell_g+\ell_1+k)}$ and computes

$$t_1 = \frac{(c_4)^{r_1}}{(c_3)^{r_2}}, t_2 = \frac{(c_2)^{r_1}}{(c_1)^{r_2}}, t_3 = (c_1)^{r_4}, t_4 = (c_2)^{r_3}.$$

– It computes $c_0 = \mathcal{H}_1(g||z||t_1||t_2||t_3||t_4||c_5||c_6||c_7)$.

– It computes $s_1 = r_1 - c_0(e_i - 2^{\ell_1}), s_2 = r_2 - c_0 e_i w_1, s_3 = r_3 - c_0(e_i - 2^{\ell_1}), s_4 = r_4 - c_0 e_i w_1$.

In fact, the tuple $(c_0, s_1, s_2, s_3, s_4)$ is the proof values of a zero-knowledge proof, that is,

$$\begin{aligned}(c_0, s_1, s_2, s_3, s_4) = \text{ZPK}\{(\alpha, \beta) : z = \frac{(c_4)^{\alpha}}{(c_3)^{\beta}} \wedge \\ 1 = \frac{(c_2)^{\alpha}}{(c_1)^{\beta}} \wedge c_6 = (c_1)^{\beta} \wedge c_6 = (c_2)^{\alpha} \wedge \\ (2^{\ell_1} - 2^{\delta(\ell_2+k)+1}) < \alpha < (2^{\ell_1} - 2^{\delta(\ell_2+k)+1})\},\end{aligned}$$

where $\alpha = e_i, \beta = e_i w_1$. The resulting ciphertext is $CT = (c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_0, s_1, s_2, s_3, s_4)$.

- Verify: $CT = (c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_0, s_1, s_2, s_3, s_4)$ can be publicly verified by anyone holding the public parameter PM through the execution of this Verify algorithm. A user with PM verifies whether $CT$ is valid with the following procedures;

  – It recomputes

$$\begin{aligned}t_1' = \frac{z^{c_0}(c_4)^{s_1 - c_0 2^{\ell_1}}}{(c_3)^{s_2}}, t_2' = \frac{(c_2)^{s_1 - c_0 2^{\ell_1}}}{(c_1)^{s_2}}, \\ t_3' = (c_1)^{s_4}(c_6)^{c_0}, t_4' = (c_2)^{s_3 - c_0 2^{\ell_1}}(c_6)^{c_0}.\end{aligned}$$

  – It computes

$$c_0' = \mathcal{H}_1(g||z||t_1'||t_2'||t_3'||t_4'||c_5||c_6||c_7)$$

  – If $c_0' = c_0$, then $CT$ is valid, otherwise, it is invalid.

- Unsigncrypt: When $CT$ is sent to $U_j$ which is its intended receiver or unsigncrypter, it does as following to recover the message $m$ using its unsigncryption key $x_j$;

- $U_j$ first executes the Verify algorithm to check whether $CT$ is valid, if not, $U_j$ rejects it, otherwise, it continues.
- $U_j$ computes $m' = \frac{c_7}{(c_6)^{x_j}}$.
- $U_j$ recomputes $w_2' = \mathcal{H}_1(g||z||(c_5)^{x_j}||m')$ and checks whether the equation $c_6 = (c_5)^{w_2'}$ holds.
- If yes, $U_j$ outputs $m'$ as the plaintext, otherwise, it rejects $CT$.

- Open: One user $U_j$ can reveal the unique certificate of the original signcrypter of $CT$ as long as $U_j$ is its intended receiver. Besides, $U_j$ also has the capability to convince anyone else that it cannot cheat when recovering the certificate. It does as following;
  - $U_j$ computes $u_i' = \frac{c_4}{(c_2)^{x_j}}$ and sets $u_i'$ as the certificate of the signcrypter of $CT$.
  - $U_j$ produces the proof values for the zero-knowledge proof ZPK$\{(x_j) : c_3 = (c_1)^{x_j} \wedge \frac{c_4}{u_i'} = (c_2)^{x_j}\}$. Namely, $U_j$ first chooses a random element $w_0 \in_R \{0,1\}^{\ell_g}$, then it computes $c' = \mathcal{H}_1((c_1)^{w_0}||(c_2)^{w_0}||CT), s' = w_0 - c'x_j$. So ZPK$\{(x_j) : c_3 = (c_1)^{x_j} \wedge \frac{c_4}{u_i'} = (c_2)^{x_j}\} = (c', s')$, $U_j$ publishes the tuple $(CT, u_i', c', s')$.

- Justify: When there exists any dispute about the signcrypter of $CT$ given the tuple $(CT, u_i', c', s')$, one user can easily check whether $U_j$ is cheating by executing this Justify algorithm. Justify$(CT, u_i', c', s') = 1$ when the following equation
$$c' = \mathcal{H}_1((c_1)^{s'}(c_3)^{c'}||(c_2)^{s'}(\frac{c_4}{u_i'})^{c'}||CT)$$
holds. Anyone can be convinced that $U_j$ is not cheating when Justify outputs "1".

## 5.4 Security Proofs

**Theorem 5.1** *Our secure communication scheme is* m-IND-CCA2 *secure with respect to the message confidentiality property under our predefined model assuming the Computational Diffie-Hellman problem in composite group*(CDHCG) *is hard in the chosen group system* $(G, g, n)$ *when the hash functions* $\mathcal{H}_1, \mathcal{H}_2$ *are modeled as random oracles. Concretely, if there is an adversary* $\mathcal{A}$ *which can break* m-IND-CCA2 *security with non-negligible probability* $\epsilon$*, supposing* $\mathcal{A}$ *makes at most* $q_{\mathcal{H}_2}$,$q_{is}, q_o$ *queries to the* $\mathcal{H}_2$ *oracle,* Unsigncryption key *oracle and* Open *oracle respectively, then we can construct another algorithm* $\mathcal{C}$ *solving the Computational Diffie-Hellman problem in composite group* (CDHCG) *with probability* $\frac{2\epsilon}{tq_{\mathcal{H}_2}}$*, where* $t$ *is an integer.*

**Proof.** Assuming there exists an adversary $\mathcal{A}$ which can break the m-IND-CCA2 security of our group signcryption scheme with non-negligible probability $\epsilon$, then we show how to construct another algorithm $\mathcal{C}$ solving the CDHCG problem from $\mathcal{A}$, $\mathcal{C}$ and $\mathcal{A}$ interacts as below.

- Setup: $\mathcal{C}$ is first given a group system $(G, g, n)$, where $n = pq$, $p, q$ are large safe primes, $G =< g >, (g|n) = 1, \ell_g = |g| \leq \ell = |n|$, and also a CDHCG problem instance $(g, g^a, g^b)$. $\mathcal{C}$ then executes the Setup algorithm and sends the public parameter PM to $\mathcal{A}$. Assuming there exists $t$ users in the user group $\{U\}$, $\mathcal{C}$ chooses one user $U_j \in \{U\}$ and sets its signcryption key as $sik_j = e_j$ and its certificate tuple as $(u_j = z^{\frac{1}{e_j}}, pk_j = g^a)$, the unsigncryption key $unsk_j$ is unknown. For the rest of users in $\{U\}/U_j$, their keys and identifiers are generated correctly following the Registration algorithm. Notice that after the $t$ users are all registered, the certificate repository $\mathcal{CR}$ should include all the $t$ users' personal certificate, the user group set $\{U\}$ should also be sent to $\mathcal{A}$.

- Query phase 1: In this phase, $\mathcal{A}$ can ask $\mathcal{C}$ a polynomially bounded umber of queries in an adaptive way, there are several types of query can be asked during this phase and each type of them can be answered by $\mathcal{C}$ using the following way.

  - $\mathcal{H}_1$ query: At any time, $\mathcal{A}$ can query the random oracle $\mathcal{O}_{\mathcal{H}_1}$ and $\mathcal{C}$ maintains a list $\mathcal{L}_{\mathcal{H}_1}$, which is empty initially, to answer such kind of queries. When $\mathcal{A}$ asks the oracle $\mathcal{O}_{\mathcal{H}_1}$ with a queried message denoted by $m_{\mathcal{H}_1} \in \{0,1\}^*$, $\mathcal{C}$ would first check whether this queried message has already been asked before;
    - If so, $\mathcal{C}$ retrieves the tuple $(m_{\mathcal{H}_1}, h_{\mathcal{H}_1})$, which entry is the queried message, in list $\mathcal{L}_{\mathcal{H}_1}$ and responds $\mathcal{A}$ with $h_{\mathcal{H}_1}$.
    - Otherwise, $\mathcal{C}$ chooses a element $h_{\mathcal{H}_1}$ randomly from $\{0,1\}^k$ and sends it back to $\mathcal{C}$ as the response, then $\mathcal{C}$ adds this new tuple $(m_{\mathcal{H}_1}, h_{\mathcal{H}_1})$ to $\mathcal{L}_{\mathcal{H}_1}$.
  - $\mathcal{H}_2$ query: $\mathcal{A}$ is also allowed to ask the hash oracle $\mathcal{O}_{\mathcal{H}_2}$ as it wants, the technique $\mathcal{A}$ used to respond such hash query is similar to that described in the $\mathcal{H}_1$ query. That is, $\mathcal{C}$ maintains a initially empty list $\mathcal{L}_{\mathcal{H}_2}$ and checks whether a query with queried message $m_{\mathcal{H}_2}$ has been asked before;
    - If so, $\mathcal{C}$ retrieves the tuple $(m_{\mathcal{H}_2}, h_{\mathcal{H}_2})$ in list $\mathcal{L}_{\mathcal{H}_1}$ and responds $\mathcal{A}$ with $h_{\mathcal{H}_2}$.
    - Otherwise, $\mathcal{C}$ chooses a element $h_{\mathcal{H}_2}$ randomly from $\{0,1\}^\ell$ and sends it back to $\mathcal{C}$ as the response, then $\mathcal{C}$ adds this new tuple $(m_{\mathcal{H}_2}, h_{\mathcal{H}_2})$ to $\mathcal{L}_{\mathcal{H}_2}$.

- Signcryption key query: $\mathcal{A}$ can get the signcryption keys of several users in $\{U\}$] by issuing such Signcryption key query. As $\mathcal{C}$ has the knowledge of all the signcryption keys of users in $\{U\}$, when $\mathcal{A}$ chooses a user $U_i \in \{U\}$ and sends it as the queried message to $\mathcal{C}$, $\mathcal{C}$ can easily find the corresponding signcryption key $e_i$ and sends $e_i$ as the response to this query.
- Unsigncryption key query: $\mathcal{A}$ can also get the unsigncryption keys of several users in $\{U\}$ by issuing this Signcryption key query. When $\mathcal{A}$ chooses a user $U_i \in \{U\}$ as the queried user, $\mathcal{C}$ answers this query with his knowledge of all the unsigncryption keys of users in $\{U\}$ excepts the chosen user $U_j$. That is, before deciding how to respond the query, $\mathcal{C}$ checks whether the queried user $U_i$ is the chosen user $U_j$;
  - If so, $\mathcal{C}$ aborts and outputs a symbol $\perp$.
  - Otherwise, $\mathcal{C}$ responds $\mathcal{A}$ with the corresponding unsigncryption key $x_i$ of $U_i$.
- Signcrypt query: $\mathcal{A}$ can ask $\mathcal{C}$ to signcrypt a message by issuing this Signcrypt query with the queried message denoted by $(m, U_i, U_l)$ where $m \in \mathcal{M}$ and $U_i, U_l \in \{U\}$ are acting as the signcryper and intended unsigncrypter respectively. To signcrypt $m$ on behalf of $U_i$ and let the resulted message be feasible to be unsigncrypted by $U_l$, $\mathcal{C}$ finds the signcryption key $e_i$, user identifier $u_i$ of $U_i$ and the public key $y_l$ of $U_l$, then it executes the Signcrypt algorithm of our scheme with inputs $(e_i, u_i, y_l.m)$ and sends the resulted ciphertext $CT = \mathsf{Signcrypt}(e_i, u_i, y_l.m)$ back to $\mathcal{A}$ as the response.
- Unsigncrypt query: $\mathcal{A}$ can ask $\mathcal{C}$ to unsigncrypt a ciphertext for it by issuing this Unsigncrypt query with the queried message denoted by $(CT, U_l)$ where $U_l$ is the unsigncrypter of $CT$. Basing on the given specific queried message $(CT, U_l)$ in each query, $\mathcal{C}$ uses the following two different ways to answer it;
  - When the given unsigncrypter of $CT$ is not the chosen user $U_j$, that is, $U_l \neq U_j$, $\mathcal{C}$ first executes the Verify algorithm of our scheme to check whether $CT$ is a valid ciphertext, $\mathcal{C}$ continues if $CT$ is valid, otherwise it rejects it and returns nothing to $\mathcal{A}$. Then $\mathcal{C}$ would executes the Unsigncryt algorithm to retrieve the encrypted message $m'$ from $CT$ because $\mathcal{C}$ knows the unsigncryption key of $U_l$ in this case. Finally, $\mathcal{C}$ sends $m'$ as the response back to $\mathcal{A}$.
  - When the given unsigncrypter of $CT$ is the chosen user $U_j$, that is, $U_l = U_j$, because the corresponding unsigncryption key of $U_j$ is unknown, $\mathcal{C}$ simulates the Unsigncrypt algorithm as follows;

· When given a ciphertext $CT = (c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_0, s_1, s_2, s_3, s_4)$, $\mathcal{C}$ also first executes the Verify algorithm to check whether $CT$ is a valid ciphertext, $\mathcal{C}$ continues to the next step if $CT$ is valid, otherwise, $\mathcal{C}$ rejects $CT$ and returns noting to $\mathcal{A}$.

· $\mathcal{C}$ searches the hash list $\mathcal{L}_{\mathcal{H}_1}$ thoroughly to check whether there exists a tuple $(m_{\mathcal{H}_1}, h_{\mathcal{H}_1})$ in $\mathcal{L}_{\mathcal{H}_1}$ such that $c_6 = (c_5)^{h_{\mathcal{H}_1}}$, if $\mathcal{C}$ cannot find such a tuple, it rejects $CT$ and returns noting to $\mathcal{A}$. Otherwise, it turns to the next step.

· For such a tuple $(m_{\mathcal{H}_1}, h_{\mathcal{H}_1})$, the entry $m_{\mathcal{H}_1}$ can be valued as $m_{\mathcal{H}_1} = g||z||{y_j}'||m'$, then $\mathcal{C}$ extracts ${y_j}'$ and $m'$ from $m_{\mathcal{H}_1}$. $\mathcal{C}$ latter searches the hash list $\mathcal{L}_{\mathcal{H}_1}$ thoroughly to find whether there exists a tuple $(m_{\mathcal{H}_2}, h_{\mathcal{H}_2})$ in $\mathcal{L}_{\mathcal{H}_2}$ such that $(y_j')^{h_{\mathcal{H}_1}}$ $= m_{\mathcal{H}_2}$, if there does not exist such a tuple, then $\mathcal{C}$ rejects $CT$ and returns noting to $\mathcal{A}$. Otherwise, it turns to the next step.

· For the qualified tuple $(m_{\mathcal{H}_2}, h_{\mathcal{H}_2})$, $\mathcal{C}$ computes a element $m'' = c_7 \oplus h_{\mathcal{H}_2}$ and checks whether the equation $m' = m''$ holds.

· $\mathcal{C}$ responds $\mathcal{A}$ with $m'$ if the above equation holds, otherwise it rejects $CT$ and returns noting to $\mathcal{A}$.

We argue that the above simulation of the unsigncryption process is identical to executing the Unsigncrypt algorithm from the view of the adversary $\mathcal{A}$. The simulation takes use of all the elements in $CT$ during finding the plaintext $m'$ and employs nearly the same technique to check the integrity of $CT$ after retrieving $m'$ from $CT$.

– Open query: $\mathcal{A}$ is allowed to issue this Open query at any time during this Query phase 1 with the queried message denoted by $(CT, U_l)$, $\mathcal{C}$ responds such query in the following manner when receiving the tuple $(CT, U_l)$;

∗ When the intended receiver $U_l$ of $CT$ is not the chosen user $U_j$, namely, $U_l \neq U_j$, as $\mathcal{C}$ knows the unsigncryption keys of all users in $\{U\}$ except the user $U_j$, it can execute the Open algorithm of our scheme given the tuple $(CT, U_l)$ and sends back the result denoted by the tuple $(u_i', c', s')$ as the response where $u_i'$ is the identifier of the signcrypter of $CT$ and $(c', s')$ is the proof value which proves the above statement about $u_i'$.

∗ When the intended receiver $U_l$ of $CT$ is also the chosen user $U_j$, that is, $U_l = U_j$, $\mathcal{C}$ aborts the game.

- Challenge: When $\mathcal{A}$ decides to end the Query phase 1, it chooses two messages $m_0, m_1$ from $\mathcal{M}$ such that $|m_0| = |m_1|$ and two users $U_i, U_l \in \{U\}$ as the sender

and intended receiver respectively, then it sends the tuple $(m_0, m_1, U_i, U_l)$ to $\mathcal{C}$. When the intended receiver in the given tuple is not the user $U_j$, $\mathcal{C}$ aborts the game. Otherwise, $\mathcal{C}$ chooses $m_b \in \{m_0, m_1\}$ and generates the challenge ciphertext $CT^*$ using the signcryption key $e_i$ of $U_i$ and public key $g^a$ of $U_j$ as follows;

- $\mathcal{C}$ chooses $w_1 \in \{0,1\}^{\ell_g}, w_2 \in \{0,1\}^k$ and computes

$$\begin{aligned} c_1^* &= (g^b)^{w_2}, c_2^* = (g^b)^{w_1 w_2}, c_3^* = (g^a)^{w_2}, \\ c_4^* &= u_i (g^a)^{w_1 w_2}, c_5^* = (g^b)^{e_i w_1}, c_6^* = (g^b)^{e_i w_1 w_2}. \end{aligned}$$

- As $\mathcal{C}$ knows $e_i, w_1$, it can produce the proof value $(c_0^*, s_1^*, s_2^*, s_3^*, s_4^*)$ easily following the Signcrypt algorithm of our scheme.
- $\mathcal{C}$ chooses $Z^* \in \{0,1\}^{\ell}$ and sets $c_7^* = Z^*$

$\mathcal{C}$ sends $CT^* = (c_1^*, c_2^*, c_3^*, c_4^*, c_5^*, c_6^*, c_7^*, c_0^*, s_1^*, s_2^*, s_3^*, s_4^*)$ back to $\mathcal{A}$ as the challenge ciphertext.

- Query phase 2: After receiving the challenge ciphertext $CT^*$, $\mathcal{A}$ can also issue queries in this Query phase 2, the types of query allowed in this phase is the same as that in the Query phase 1 and $\mathcal{C}$ answers them identical to what we have described above. Here, the extra restriction comparing to the Query phase 1 is that $U_j$ cannot appear in any unsigncryption key queries and $(CT^*, U_j)$ cannot appear in any unsigncryption queries in this phase where $CT^*$ is the challenge ciphertext and $U_j$ is the intended receiver of $CT^*$. Assuming the total number of Unsigncryption key query and that of open query issued by $\mathcal{A}$ in Query phase 1 and Query phase 2 is $q_{is}$ and $q_o$ respectively.

- Guess: After completing the Query Phase 2, $\mathcal{A}$ makes a guess $b'$ from $\{0,1\}$ and sends $b'$ to $\mathcal{C}$.

**Analysis.** As analyzed in many other papers [LQ04, LYW+10], it is easy to show that $\mathcal{A}$ will not able to realize that $CT^*$, in fact, is not a valid signcryption for the sender's signcryption key $e_i$ and the intended receiver's public key $g^a$ unless it asks for the hash value $\mathcal{H}_2(g^{abe_i w_1 w_2})$. In that case, the value $g^{abe_i w_1 w_2}$ would have been inserted in the hash list $\mathcal{L}_2$ exactly as one of its entry, furthermore, it does not matter if the simulation of $\mathcal{A}$'s view is no longer perfect now. Because $\mathcal{C}$ knows $e_i, w_1, w_2$, the solution $g^{ab}$ to the CDHCG problem instance $(g, g^a, g^b)$ in the composite group $\mathbb{Z}_n^*$ can be computed from that entry. At the end of the game, assuming there are $q_{\mathcal{H}_2}$ entries in $\mathcal{L}_2$, $\mathcal{C}$ randomly chooses one from $\mathcal{L}_2$, then with probability $\frac{1}{q_{\mathcal{H}_2}}$, $\mathcal{C}$ will find the solution correctly.

Let $\mathbf{E}$ be the event that the value $g^{abe_i w_1 w_2}$ is queried to $\mathcal{H}_2$ and $\tilde{\mathbf{E}}$ denotes the event that the value $g^{abe_i w_1 w_2}$ is not queried, we find that $\mathcal{C}$ solves the given problem instance in event $\mathbf{E}$ with probability $\frac{1}{q_{\mathcal{H}_2}}$. We denote $\Pr[b = b'] = \frac{1}{2} + \epsilon$ the probability that $\mathcal{A}$ wins the game, then the conditional probability $\Pr[b = b'|\tilde{\mathbf{E}}] = \frac{1}{2}$, we have

$$\begin{aligned}
\Pr[b = b'] &= \frac{1}{2} + \epsilon \\
&= \Pr[b = b'|\mathbf{E}] \Pr[\mathbf{E}] + \Pr[b = b'|\tilde{\mathbf{E}}] \Pr[\tilde{\mathbf{E}}] \\
&\leq \Pr[\mathbf{E}] + \Pr[b = b'|\tilde{\mathbf{E}}] \Pr[\tilde{\mathbf{E}}] \\
&= \Pr[\mathbf{E}] + \frac{1}{2}(1 - \Pr[\mathbf{E}]) \\
&= \frac{1}{2}(1 + \Pr[\mathbf{E}])
\end{aligned}$$

So, $\Pr[\mathbf{E}] \geq 2\epsilon$. Let $\Pr[\overline{abort}]$ denote that probability that $\mathcal{C}$ would not abort during the simulation, then we have $\Pr[\overline{abort}] = \frac{1}{t}$ where $|\{U\}| = t$ Let $\mathsf{Adv}_{\mathcal{C}}^{CDHCG}$ denote that advantage that $\mathcal{C}$ can solves the given CDHCG problem instance, then we have

$$\begin{aligned}
\mathsf{Adv}_{\mathcal{C}}^{\mathsf{CDHCG}} &= \Pr[\mathbf{E} \wedge \overline{abort}] \frac{1}{q_{\mathcal{H}_2}} \\
&\geq \Pr[\mathbf{E}] \Pr[\overline{abort}] \frac{1}{q_{\mathcal{H}_2}} \\
&\geq \frac{2\epsilon}{t q_{\mathcal{H}_2}}
\end{aligned}$$

Obviously, if $\epsilon$ is non-negligible, then $\mathsf{Adv}_{\mathcal{C}}^{\mathsf{CDHCG}}$ is non-negligible. That is, if $\mathcal{A}$ can break the m-IND-CCA2 game with non-negligible probability $\epsilon$, then $\mathcal{C}$ can solve the CDHCG problem with non-negligible probability $\frac{2\epsilon}{t q_{\mathcal{H}_2}}$, here, we finish our proof. □

**Theorem 5.2** *Our secure communication scheme is* an-CCA2 *secure under our predefined model assuming the Diffie-Hellman Decision* (DHD) *problem in composite group is hard in the chosen group system* $(G, g, n)$ *when the hash functions* $\mathcal{H}_1, \mathcal{H}_2$ *are modeled as random oracles. Concretely, if there is an adversary* $\mathcal{A}$ *which can break the* an-CCA2 *security with non-negligible probability* $\epsilon$*, supposing* $\mathcal{A}$ *makes at most* $q_{is}, q_o$ *queries to* Unsigncryption key *oracle and* Open *oracle respectively, then we can construct another algorithm* $\mathcal{C}$ *solving the Diffie-Hellman Decision problem*(DHD) *problem with probability at least* $\frac{1}{t}\epsilon$*, where* $t$ *is an integer.*

**Proof.** Assuming there exists an adversary $\mathcal{A}$ which can break the an-CCA2 security of our group signcryption scheme with non-negligible probability $\epsilon$, then we can show how to construct another algorithm $\mathcal{C}$ solving the DHD problem from $\mathcal{A}$, $\mathcal{C}$ and $\mathcal{A}$ interacts as below.

- Setup: $\mathcal{C}$ is first given a group system $(G, g, n)$, where $n = pq$, $p, q$ are large safe primes, $G =< g >, (g|n) = 1, \ell_g = |g| \leq \ell = |n|$, and also a CDHCG problem instance $(g, g^a, g^b)$. $\mathcal{C}$ then executes the Setup algorithm and sends the public parameter PM to $\mathcal{A}$. Assuming there exists $t$ users in the user group $\{U\}$, $\mathcal{C}$ chooses one user $U_j \in \{U\}$ and sets its signcryption key as $sik_j = e_j$ and its certificate tuple as $(u_j = z^{\frac{1}{e_j}}, pk_j = g^a)$, the unsigncryption key $unsk_j$ is unknown. For the rest of users in $\{U\}/U_j$, their keys and identifiers are generated correctly following the Registration algorithm. Notice that after the $t$ users are all registered, the certificate repository $\mathcal{CR}$ should include all the $t$ users' personal certificate, the user group set $\{U\}$ should also be sent to $\mathcal{A}$.

- Query phase 1: In this phase, $\mathcal{A}$ can ask a polynomially bounded umber of queries in an adaptive way. As the types of queries allowed in this phase and the strategy used by $\mathcal{C}$ to answer each of those queries are all the same as that mentioned in the Query phase 1 of the previous security proof, we omit the description here.

- Challenge: When $\mathcal{A}$ decides to end the Query phase 1, it chooses one message $m$ from $\mathcal{M}$ and one user $U_l \in \{U\}$ as the intended receiver, then it sends the tuple $(m, U_l)$ to $\mathcal{C}$. If the intended receiver in the given tuple is not the user $U_j$, $\mathcal{C}$ aborts the game. Otherwise, $\mathcal{C}$ chooses an index $b \in [t]$ and generates the challenge ciphertext $CT^*$ using the signcryption key $e_b$ of $U_b$ and public key $g^a$ of $U_j$ as follows;

  - $\mathcal{C}$ chooses $w_3^* \in_R \{0,1\}^{\ell_g}$ and asks the $\mathcal{H}_1$ oracle with message $g||z||S^{e_b w_3^*}||m$ to get $w_2^* = \mathcal{H}_1(g||z||S^{e_b w_3^*}||m)$. As $\mathcal{C}$ controls $\mathcal{H}_1$ oracle, it does the same as what we have described in the Query phase 1 of the previous security proof to answer this query.
  - $\mathcal{C}$ sets

$$c_1^* = g^{w_2^* w_3^*}, c_2^* = s^{w_2^* w_3^*}, c_3^* = g^{a w_2^* w_3^*},$$
$$c_4^* = u_b S^{w_2^* w_3^*}, c_5^* = s^{e_b w_3^*}, c_6^* = s^{e_b w_2^* w_3^*},$$
$$c_7^* = m \oplus \mathcal{H}_2(S^{e_b w_2^* w_3^*}).$$

    Here, the $\mathcal{H}_2$ oracle is also controlled by $\mathcal{C}$, the strategy used by $\mathcal{C}$ to answer the query with message $e_b w_2^* w_3^*$ during generating the challenge ciphertext is also identical to that described in the previous security proof.

– $\mathcal{C}$ tries to forge a zero-knowledge proof like

$$(c_0^*, s_1^*, s_2^*, s_3^*, s_4^*) = \text{ZPK}\{(\alpha, \beta) : z = \frac{(c_4^*)^\alpha}{(c_3^*)^\beta} \wedge$$
$$1 = \frac{(c_2^*)^\alpha}{(c_1^*)^\beta} \wedge c_6^* = (c_1^*)^\beta \wedge c_6^* = (c_2^*)^\alpha \wedge$$
$$(2^{\ell_1} - 2^{\delta(\ell_2+k)+1}) < \alpha < (2^{\ell_1} - 2^{\delta(\ell_2+k)+1})\}.$$

To do it, $\mathcal{C}$ first chooses $c_0^* \in_R \{0,1\}^\ell, s_1^*, s_3^* \in_R \{0,1\}^{\delta(\ell_2+k)}, s_2^*, s_4^* \in_R \{0,1\}^{\delta(\ell_g+\ell_1+k)}$, then it computes

$$t_1^* = \frac{z^{c_0^*}(c_4^*)^{s_1^* - c_0^* 2^{\ell_1}}}{(c_3^*)^{s_2^*}}, t_2^* = \frac{(c_2^*)^{s_1^* - c_0^* 2^{\ell_1}}}{(c_1^*)^{s_2^*}},$$
$$t_3^* = (c_1^*)^{s_4^*}(c_6^*)^{c_0^*}, t_4^* = (c_2^*)^{s_3^* - c_0^* 2^{\ell_1}}(c_6^*)^{c_0^*}.$$

$\mathcal{C}$ programs $\mathcal{H}_1$ in such a way that it sets

$$c_0^* = \mathcal{H}_1(g||z||t_1^*||t_2^*||t_3^*||t_4^*||c_5^*||c_6^*||c_7^*)$$

and then adds this tuple $(g||z||t_1^*||t_2^*||t_3^*||t_4^*||c_5^*||c_6^*||c_7^*, c_0^*)$ to the hash list $\mathcal{L}_{\mathcal{H}_1}$.

– After finishing the above procedures, $\mathcal{C}$ sets the challenge ciphertext $CT^* = (c_1^*, c_2^*, c_3^*, c_4^*, c_5^*, c_6^*, c_7^*, c_0^*, s_1^*, s_2^*, s_3^*, s_4^*)$. Here, in fact, the tuple $(c_0^*, s_1^*, s_2^*, s_3^*, s_4^*)$ is valid from the view of the adversary $\mathcal{A}$ because $\mathcal{C}$ controls $\mathcal{H}_1$ oracle.

- Query phase 2: After receiving the challenge ciphertext $CT^*$, $\mathcal{A}$ can also issue queries in this Query phase 2, the types of query allowed in this phase is the same as that in the Query phase 1 and $\mathcal{C}$ answers them identical to what we have described above. Here, the extra restriction comparing to the Query phase 1 is that $U_j$ cannot appear in any unsigncryption key query and $(CT^*, U_j)$ cannot appear in any unsigncryption query and open query in this phase where $CT^*$ is the challenge ciphertext and $U_j$ is the intended receiver of $CT^*$. Assuming the total number of Unsigncryption key query and that of open query issued by $\mathcal{A}$ in Query phase 1 and Query phase 2 is $q_{is}$ and $q_o$.

- Guess: After completing the Query Phase 2, $\mathcal{A}$ makes a guess $b'$ from $[t]$ and sends $b'$ to $\mathcal{C}$.

**Analysis.** Let $\Pr[b = b']$ denote the probability that $\mathcal{A}$ wins the game, we consider this probability in the following two different scenes,

- When $(g, s, g^a, S) \in \mathcal{Q}(G)$, that is, $\log_g g^a \neq \log_s S$, the challenge ciphertext

$CT^*$ can be valued as an one-time pad indeed and it would not reveal any information about the choice $b$ of $\mathcal{C}$. In that case, $\mathcal{A}$ gains no help from this challenge ciphertext and it has no choice but to make a random choice $b'$ about $b$, so the probability $\Pr[b = b'] = \frac{1}{t}$.

- When $(g, s, g^a, S) \in \mathcal{DH}(G)$, that is, $\log_g g^a = \log_s S$, the challenge ciphertext $CT^*$, in fact, is a valid ciphertext which encrypts $m$ and $u_b$. In this case, let $\Pr[\overline{abort}]$ denote the probability that the above simulation would not abort, then $\Pr[\overline{abort}] = \frac{1}{t}$, if $\mathcal{A}$ can break the an-CCA2 security of our scheme with non-negligible probability $\epsilon$, then $\Pr[b = b'] = \frac{1}{t} + (\frac{t-1}{t})^{q_{is}+q_o}\frac{1}{t}\epsilon$.

Let $\mathsf{Adv}_{\mathcal{C}}^{\mathsf{DHD}}$ denote the probability that $\mathcal{C}$ can solve the DHD problem, then we have

$$\mathsf{Adv}_{\mathcal{C}}^{\mathsf{DHD}} = \frac{1}{t}\epsilon$$

Namely, if $\mathcal{A}$ can break the an-CCA2 security of our scheme with non-negligible probability $\epsilon$, then $\mathcal{C}$ can break the DHD problem with non-negligible probability $\frac{1}{t}\epsilon$. Here, we finish our proof. □

**Theorem 5.3** *Our secure communication scheme is* O-UF-CMA *secure under our predefined model assuming the Strong RSA problem is hard in the chosen group system* $(G, g, n)$ *when the hash functions* $\mathcal{H}_1, \mathcal{H}_2$ *are modeled as random oracles. Concretely, if there is an adversary* $\mathcal{A}$ *which can break the* O-UF-CMA *security with non-negligible probability* $\epsilon$*, supposing* $\mathcal{A}$ *makes at most* $q_{\mathcal{H}_1}, q_{\mathcal{H}_2}, q_s$ *queries to the* $\mathcal{H}_1$*,* $\mathcal{H}_2$ *oracle and* Signcrypt *oracle respectively, then we can construct another algorithm* $\mathcal{C}$ *solving the Strong RSA problem with probability* $(\epsilon - \frac{1}{v})^2\frac{1}{q_{\mathcal{H}_1}+q_{\mathcal{H}_2}+q_s}$ *where* $|G| = v$.

**Proof.** Assuming there exists an adversary $\mathcal{A}$ which can break the O-UF-CMA security of our scheme with non-negligible probability $\epsilon$, then we can show how to construct another algorithm $\mathcal{C}$ solving the Strong RSA problem from $\mathcal{A}$, $\mathcal{C}$ and $\mathcal{A}$ interacts as below.

- Setup: $\mathcal{C}$ is first given a group system $(G, g, n)$ and an element $z \in G$, where $n = pq$, $p, q$ are large safe primes, $G =< g >, |G| = v, (g|n) = 1, \ell_g = |g| \leq \ell = |n|$. $\mathcal{C}$ then executes the Setup algorithm and sends the public parameter PM to $\mathcal{A}$. Assuming there exists $t$ users in the user group $\{U\}$. For each user $U_i \in \{U\}$, $\mathcal{C}$ executes the Registration algorithm of our scheme and generates $U_i$'s public key $pk_i$, user certificate $u_i$, signcryption key $sik_i$ and unsigncryption key $unsk_i$ correctly. After all $t$ users are properly registered, $\mathcal{C}$ also sends the user group set $\{U\}$ to $\mathcal{A}$.

- Query phase: In this phase, $\mathcal{A}$ can ask a polynomially bounded umber of queries in an adaptive way. However, to imitate the behavior of the adversary

$\mathcal{A}$ which is neither a registered user nor colluding with any registered user in the system, we only allow $\mathcal{A}$ to ask the $\mathcal{H}_1$ query, $\mathcal{H}_2$ query, Signcrypt query, Unisgncrypt query and Open query. As the description of the types of queries allowed in this phase and the strategy used by $\mathcal{C}$ when answering each of those queries are all the same as that aforementioned in the Query phase 1 of the previous message confidentiality security proof, we omit it here.

- Forgery: In this phase, $\mathcal{A}$ itself chooses a message $m \in \mathcal{M}$ and also one user $U_j$ as the intended message receiver, then it asks the $\mathcal{H}_1$, $\mathcal{H}_2$ oracles and tries to give a forgery $CT' = (c'_1, c'_2, c'_3, c'_4, c'_5, c'_6, c'_7, c'_0, s'_1, s'_2, s'_3, s'_4)$.

**Analysis.** In fact, the given forgery $CT'$ can be represented as a variant of the Schnorr signature[Sch89] $(h', \delta')$ of the message $m'$ where $m' = (c'_1, c'_2, c'_3, c'_4, c'_5, c'_6, c'_7)$, $h' = c'_0, \delta' = (s'_1, s'_2, s'_3, s'_4)$. The adversary $\mathcal{A}$ wins the above game when $\mathcal{A}$ can produce a valid ciphertext $(m', h', \delta')$ such that $\mathsf{Verify}(m', h', \delta') = 1$, let $\Pr[\mathcal{A}\text{ wins}]$ denote the probability of the event that $\mathcal{A}$ wins the game. According to the *Forking Lemma*[PS00], we can treat this adversary $\mathcal{A}$ as a turning machine with a random tape $R'$, if $\Pr[\mathcal{A}\text{ wins}] = \epsilon$ is non-negligible, that is, $\mathcal{A}$ can produce a valid message signature pair $(m', h', \delta')$ successfully with non-negligible probability, then $\mathcal{C}$ can control $\mathcal{A}$ and execute a replay attack with the same random tape $R'$, and would produce another valid signature $(h'', \delta'')$ of the same message $m'$ where $h' \neq h'' \wedge \delta' \neq \delta''$ with probability $\frac{(\epsilon - \frac{1}{|G|})^2}{Q}$, where $|G|$ denotes the order of $G$ and $Q$ is the total number of hash and signing queries asked during the game.

Basing on the above result, when $\mathcal{C}$ gets $CT' = (m', h', \delta')$ and $CT'' = (m', h'', \delta'')$, it can retrieve the personal certificate $u'$ embedded in $m' = (c'_1, c'_2, c'_3, c'_4, c'_5, c'_6, c'_7)$ by computing $u' = \frac{c'_4}{(c'_2)^{x_j}}$ as it knows the unsigncryption key $x_j$ of the intended receiver of $CT'$. Given the two tuples $(h', \delta') = (c'_0, s'_1, s'_2, s'_3, s'_4), (h'', \delta'') = (c''_0, s''_1, s''_2, s''_3, s''_4)$, we have

$$s'_1 = r_1 - c'_0(e' - 2^{\ell_1}), s'_2 = r_2 - c'_0 e' w_1,$$
$$s'_3 = r_3 - c'_0(e' - 2^{\ell_1}), s'_4 = r_4 - c'_0 e' w_1,$$

$$s''_1 = r_1 - c''_0(e' - 2^{\ell_1}), s''_2 = r_2 - c''_0 e' w_1,$$
$$s''_3 = r_3 - c''_0(e' - 2^{\ell_1}), s''_4 = r_4 - c''_0 e' w_1.$$

We can further compute $e' = \frac{s'_1 - s''_1}{c''_0 - c'_0} + 2^{\ell_1} = \frac{s'_3 - s''_3}{c''_0 - c'_0} + 2^{\ell_1}$,$e'w_1 = \frac{s'_2 - s''_2}{c''_0 - c'_0} = \frac{s'_4 - s''_4}{c''_0 - c'_0}$.

According to the Verify algorithm of our scheme, if $\mathsf{Verify}(CT') = \mathsf{Verify}(CT'') = 1$, then we have $\frac{(c'_4)^{e'}}{(c'_3)^{e'w_1}} = (u')^{e'} = z$. That is, if there exist an adversary $\mathcal{A}$ which can win our O-UF-CMA game with negligible probability $\epsilon$, then we can use it to

find a pair of value $(u', e')$ such that $(u')^{e'} = z$ to break the Strong RSA problem with probability $(\epsilon - \frac{1}{v})^2 \frac{1}{q_{\mathcal{H}_1}+q_{\mathcal{H}_2}+q_s}$ where $|G| = v$. Here we finish our proof. □

**Theorem 5.4** *Our secure communication scheme is* I-UF-CMA *secure under our predefined model assuming the Discrete Logarithm* (DL) *problem is hard in the composite group system* $(G, g, n)$ *when the hash functions* $\mathcal{H}_1, \mathcal{H}_2$ *are modeled as random oracles. Concretely, if there is an adversary* $\mathcal{A}$ *which can break the* I-UF-CMA *security with non-negligible probability* $\epsilon$*, supposing* $\mathcal{A}$ *makes at most* $q_{\mathcal{H}_1}, q_{\mathcal{H}_2}, q_{sk}, q_s$ *queries to the* $\mathcal{H}_1$*,* $\mathcal{H}_2$ *oracle,* Signcryption key *oracle and* Signcrypt *oracle respectively, then we can construct another algorithm* $\mathcal{C}$ *that solves the DL problem with probability at least* $(\epsilon - \frac{1}{v})^2 \frac{1}{q_{\mathcal{H}_1}+q_{\mathcal{H}_2}+q_s}$ *where* $|G| = v$.

**Proof.** Assuming there exists an adversary $\mathcal{A}$ which can break the I-UF-CMA security of our scheme with non-negligible probability $\epsilon$, then we can show how to construct another algorithm $\mathcal{C}$ solving the DL problem from $\mathcal{A}$, $\mathcal{C}$ and $\mathcal{A}$ interacts as below.

- Setup: $\mathcal{C}$ is first given a group system $(G, g, n)$ and a DL problem instance $(u', z)$ where $(u', z) \in G^2$, where $n = pq$, $p, q$ are large safe primes, $G =< g >$ , $|G| = v, (g|n) = 1, \ell_g = |g| \leq \ell = |n|$. $\mathcal{C}$ then executes the Setup algorithm and sends the public parameter PM to $\mathcal{A}$. Assuming there exists $t$ users which consist of the user group $\{U\}$. $\mathcal{C}$ randomly chooses one user $U_i \in \{U\}$ as the user which it wants $\mathcal{A}$ to impersonate. For each user $U_j \in \{U\}/U_i$, $\mathcal{C}$ executes the Registration algorithm of our scheme for that user and generates $U_j$'s public key $pk_j$, user certificate $u_j$, signcryption key $sik_j$ and unsigncryption key $unsk_j$ correctly. For the specific user $U_i$, $\mathcal{C}$ sets its certificate $u_i = u'$, $\mathcal{C}$ further random chooses $x_i \in \mathbb{Z}_v$, then sets $U_i$'s public key $pk_i = g^{x_i}$ and unsigncryption key $unsk_i = x_i$. Assuming there exists a user certificate repository $\mathcal{CR}$ which includes all the $t$ users' certificate, after all $t$ users are properly registered, $\mathcal{C}$ also sends the user group set $\{U\}$ and also $\mathcal{CR}$ to $\mathcal{A}$.

- Query phase: In this phase, $\mathcal{A}$ can ask a polynomially bounded umber of queries in an adaptive way and is allowed to ask all types of query described previously. However, if $\mathcal{A}$ make signcryption key query on user $U_i$ and make Signcrypt query with queried message $(m, U_l, U_j)$ where $U_l = U_i$, $\mathcal{C}$ would abort the game. As the description of the types of query allowed in this phase and the strategy used by $\mathcal{C}$ when answering each of those queries are all the same as that aforementioned in the Query phase 1 of the previous message confidentiality security proof, we omit it here. Assuming $\mathcal{A}$ makes at most $q_{\mathcal{H}_1}, q_{\mathcal{H}_2}, q_{sk}, q_s$ queries to the $\mathcal{H}_1$, $\mathcal{H}_2$ oracle, Signcryption key oracle and Signcrypt oracle respectively in this phase.

- Forgery: In this phase, $\mathcal{A}$ itself chooses a message $m \in \mathcal{M}$ and also one user $U_j$ as the intended message receiver, then it asks the $\mathcal{H}_1$, $\mathcal{H}_2$ oracles and tries to forge a ciphertext $CT' = (c'_1, c'_2, c'_3, c'_4, c'_5, c'_6, c'_7, c'_0, s'_1, s'_2, s'_3, s'_4)$ on behalf of $U_i$.

**Analysis.** In fact, the given forgery $CT'$ can be represented as a variant of the Schnorr signature [Sch89] $(h', \delta')$ of the message $m'$ where $m' = (c'_1, c'_2, c'_3, c'_4, c'_5, c'_6, c'_7)$, $h' = c'_0, \delta' = (s'_1, s'_2, s'_3, s'_4)$. The adversary $\mathcal{A}$ wins the above game when $\mathcal{A}$ can produce a valid ciphertext $(m', h', \delta')$ such that $\mathsf{Verify}(m', h', \delta') = 1$ and $\mathsf{Open}(\mathsf{m'}, \mathsf{h'}, \delta', \mathsf{unsk_j}) = u'$, let $\Pr[\mathcal{A}$ wins$]$ denote the probability of the event that $\mathcal{A}$ wins the game. According to the *Forking Lemma* [PS00], we can treat this adversary $\mathcal{A}$ as a turning machine with a random tape $R'$, if $\Pr[\mathcal{A}$ wins$] = \epsilon$ is non-negligible, that is, $\mathcal{A}$ produces a valid message signature pair $(m', h', \delta')$ successfully, then $\mathcal{C}$ can control $\mathcal{A}$ and execute a replay attack with the same random tape $R'$, and would produce another valid signature $(h'', \delta'')$ of the same message $m'$ where $h' \neq h'' \wedge \delta' \neq \delta''$ with probability $\frac{(\epsilon - \frac{1}{v})^2}{Q}$, where $|G| = v$ and $Q$ is the total number of hash and signing queries asked during the game.

Basing on the above result, when $\mathcal{C}$ gets $CT' = (m', h', \delta')$ and $CT'' = (m', h'', \delta'')$ such that $(h', \delta') = (c'_0, s'_1, s'_2, s'_3, s'_4), (h'', \delta'') = (c''_0, s''_1, s''_2, s''_3, s''_4)$, we have

$$\begin{aligned} s'_1 &= r_1 - c'_0(e' - 2^{\ell_1}), s'_3 = r_3 - c'_0(e' - 2^{\ell_1}), \\ s''_1 &= r_1 - c''_0(e' - 2^{\ell_1}), s''_3 = r_3 - c''_0(e' - 2^{\ell_1}). \end{aligned}$$

We can further compute $e' = \frac{s'_1 - s''_1}{c''_0 - c'_0} + 2^{\ell_1} = \frac{s'_3 - s''_3}{c''_0 - c'_0} + 2^{\ell_1}$.

According to the $\mathsf{Verify}$ algorithm of our scheme, if $\mathsf{Verify}(CT') = \mathsf{Verify}(CT'') = 1$, then we have $\frac{(c'_4)^{e'}}{(c'_3)^{e' w_1}} = (u')^{e'} = z$. Let $\Pr[\overline{abort}]$ denote the probability that our simulation would not abort and $\mathsf{Adv}^{DL}_{\mathcal{C}}$ denote the probability that $\mathcal{C}$ would break the DL problem, then we have

$$\begin{aligned} \mathsf{Adv}^{DL}_{\mathcal{C}} &= \Pr[\overline{abort}] \frac{(\epsilon - \frac{1}{v})^2}{Q} \\ &= (\frac{1}{t}) \frac{(\epsilon - \frac{1}{v})^2}{Q}, \\ &\geq (\epsilon - \frac{1}{v})^2 \frac{1}{t(q_{\mathcal{H}_1} + q_{\mathcal{H}_2} + q_s)} \end{aligned}$$

Namely, if there exist an adversary $\mathcal{A}$ which can win our $\mathsf{I\text{-}UF\text{-}CMA}$ game with negligible probability $\epsilon$, then we can use it to break the DL problem with probability $(\epsilon - \frac{1}{v})^2 \frac{1}{t(q_{\mathcal{H}_1} + q_{\mathcal{H}_2} + q_s)}$, where $t$ is an integer. Here we finish our proof. □

**Theorem 5.5** *Our secure communication scheme is* $\mathsf{R\text{-}UF\text{-}CMA}$ *secure under our predefined model assuming the Discrete Logarithm* $(\mathsf{DL})$ *problem is hard in the com-*

*posite group system* $(G, g, n)$ *when the hash functions* $\mathcal{H}_1, \mathcal{H}_2$ *are modeled as random oracles. Concretely, if there is an adversary* $\mathcal{A}$ *which can break the* R-UF-CMA *security with non-negligible probability* $\epsilon$*, supposing* $\mathcal{A}$ *makes at most* $q_{\mathcal{H}_1}, q_o$ *times of queries to the* $\mathcal{H}_1$ *and* Open *oracle respectively, then we can construct another algorithm* $\mathcal{C}$ *that solves the DL problem with probability* $(\epsilon - \frac{1}{v})^2 \frac{1}{q_{\mathcal{H}_1}+q_o}$ *where* $|G| = v$*.*

**Proof.** Assuming there exists an adversary $\mathcal{A}$ which can break the I-UF-CMA security of our scheme with non-negligible probability $\epsilon$, then we can show how to construct another algorithm $\mathcal{C}$ solving the DL problem from $\mathcal{A}$, $\mathcal{C}$ and $\mathcal{A}$ interacts as below.

- Setup: $\mathcal{C}$ is first given a group system $(G, g, n)$ and an element $z \in G$, where $n = pq$, $p, q$ are large safe primes, $G =< g >, |G| = v, (g|n) = 1, \ell_g = |g| \leq \ell = |n|$. $\mathcal{C}$ then executes the Setup algorithm and sends the public parameter PM to $\mathcal{A}$. Assuming there exists $t$ users in the user group $\{U\}$. For each user $U_i \in \{U\}$, $\mathcal{C}$ executes the Registration algorithm of our scheme and generates $U_i$'s public key $pk_i$, user certificate $u_i$, signcryption key $sik_i$ and unsigncryption key $unsk_i$ correctly. Assuming $\mathcal{C}$ has a user certificate repository $\mathcal{CR}$ which includes all users' personal certificate. After all $t$ users are properly registered, $\mathcal{C}$ also sends the user group set $\{U\}$ and also the certificate repository $\mathcal{CR}$ to $\mathcal{A}$.

- Query phase: In this phase, $\mathcal{A}$ can ask a polynomially bounded umber of queries in an adaptive way. As the types of queries allowed in this phase and the strategy used by $\mathcal{C}$ when answering each of those queries are all the same as that aforementioned in the Query phase 1 of the previous security proof, we omit the description here. Assuming $\mathcal{A}$ makes at most $q_{\mathcal{H}_1}, q_o$ times of queries to the $\mathcal{H}_1$ and Open oracle respectively

- Challenge: In this phase, $\mathcal{A}$ chooses one message $m \in \mathcal{M}$ and one user $U_j \in \{U\}$ as the intended receiver of the chosen message, then $\mathcal{A}$ sends the tuple $(m, U_j)$ to $\mathcal{C}$. After receiving it, $\mathcal{C}$ chooses one user $U_i$ such that $U_i \neq U_j$ as the signcrypter of $m$ and generates the challenge ciphertext as $CT' = \mathsf{Signcrypt}(sik_i, u_i, pk_j, m) = (c'_1, c'_2, c'_3, c'_4, c'_5, c'_6, c'_7, c'_0, s'_1, s'_2, s'_3, s'_4)$, then $\mathcal{C}$ sends this tuple $(CT', U_j)$ back to $\mathcal{A}$.

- Forgery: Upon receiving $(CT', U_j)$, $\mathcal{A}$ tries to give a tuple $(CT', u', c', s')$ such that $u' \in \mathcal{CR}$ to convince others that the signcrypter of $CT'$ is the user with personal certificate $u'$.

**Analysis.** $\mathcal{A}$ wins the above game when $\mathsf{Justify}(CT', u', c', s') = 1 \wedge u' \neq u_i$. Let $\Pr[\mathcal{A}\text{ wins}]$ denote the probability of the event that $\mathcal{A}$ wins the above game. In

fact, the forged tuple $(CT', u', c', s')$ can be valued as a variant of the Schnorr signature $(m', h', \delta')$ where $m' = (CT', u'), h' = c', \delta' = s'$. According to the *Forking Lemma*[PS00], we can treat this adversary $\mathcal{A}$ as a turning machine with a random tape $R'$, if $\Pr[\mathcal{A} \text{ wins}] = \epsilon$ is non-negligible, that is, $\mathcal{A}$ produces a valid message signature pair $(m', h', \delta')$ successfully, then $\mathcal{C}$ can control $\mathcal{A}$ and execute a replay attack with the same random tape $R'$, and would produce another valid signature $(h'', \delta'')$ of the same message $m'$ where $h' \neq h'' \wedge \delta' \neq \delta''$ with probability $\frac{(\epsilon - \frac{1}{v})^2}{Q}$, where $|G| = v$ and $Q$ is the total number of hash and signing queries asked during the game.

When $\mathcal{C}$ gets two message signature pairs $(m', h', \delta')$ and $(m', h', \delta')$ such that $\mathsf{Justify}(m', h', \delta') = \mathsf{Justify}(m', h'', \delta'') = 1$, it can find the following relationship between the four values in our scheme;

$$s' = w_0 - c'x', s'' = w_0 - c''x',$$

where $w_0, x'$ are unknown, we derive the value $x' = \frac{s' - s''}{c'' - c'}$ from them.

According to our scheme, if $\mathsf{Justify}(CT', u', c', s') = 1$, then we have $\frac{c'_4}{u'} = (c'_2)^{x'}$. Namely, if $\mathcal{A}$ can win the above game with non-negligible probability $\epsilon$, then we can construct another algorithm form $\mathcal{A}$ to break the DL problem with probability $(\epsilon - \frac{1}{v})^2 \frac{1}{q_{\mathcal{H}_1} + q_o}$. Here we finish the proof. □

## 5.5 Summary

In this chapter, to securely send the sensitive information from the surveillance node to the server in the security surveillance system, we present a publicly verifiable secure communication scheme with user and data privacy. With our scheme, the message filter can filtrate information sent not from the surveillance nodes without requiring any secret and compromising the nodes' privacy. In our scheme, the anonymity of one node can only be revoked by the server, which enables only the server to build the searchable database using the node's location as index. Besides, given a ciphertext, the surveillance server can also give a proof to convince anyone the origination of it without leaking the data privacy. Such property enables the message auditor and others to check the origination of a ciphertext latter without knowing the underlying plaintext. We give formal security models and proofs to argue that our scheme satisfies all the required security requirements.

# Part II

# Access Control Encryption

# Chapter 6

# ACE with compact ciphertext size and decentralized sanitizers

We present an access control encryption (ACE) scheme which enjoys advantages over previous works in several aspects. Our scheme ensures not only compact ciphertext size but also small size of keys installed in each user in the ACE system. Besides, our scheme is believed to be the first implementation of ACE with decentralized sanitizers. Comparing to ACE constructions with only one sanitizer, our scheme is more secure and reliable since it does not suffer from the so called single point failure. To discuss the security of our scheme in detail, we present two models catering to the no-read rule and no-write rule security requirements respectively. Additionally, our extended no-write rule model allows the corruption of some sanitizers in the ACE system and thus is more stronger than the one proposed in schemes with only one sanitizer. We prove the security of our scheme under the two models.

## 6.1 Introduction

**Background.** The recently proposed primitive Access Control Encryption(ACE) [DHO16] gives the first cryptographic realization of the classical Bell-Lapudala access control model [BL96]. By giving different roles to different users, it enables fine-grained access control in terms of which messages are allowed to be received and sent respectively by one specific user, which properties are also defined as the read rule and write rule of the ACE. In [DHO16], Damgard et. al. assume that there must exist one special party named sanitizer in ACE to fully control the communication channel, which means all outgoing messages must pass through the sanitizer, otherwise, the desirable access control strategies, no-read-up and the no-write down exactly, can never be enforced successfully. In fact, such assumption is also held in subsequent papers related to the ACE [FGKO17][TZMT17]. Furthermore, the ACE

enjoys extra benefit with the existence of the sanitizer, that is, the sanitizer can prohibit communications between even corrupted senders and corrupted receivers, and it is this property that had seldom be considered by previous cryptographic primitives.

Since the role played by the sanitizer is of vital importance in ACE, the security concerns on it should be considered carefully and thoroughly. According to the previous works, the minimum requirement of the sanitizer is that it should be semi-trusted. Explicitly, the sanitizer is trusted to follow the protocol specification honestly but may try to learn additional information by utilizing other attacks such as collusion. However, we argue that the sanitizer should also be extremely reliable, namely, the so called single point failure should never happen. Literally, if the sanitizer cannot work properly in some instances, such as out-of-power or off-line, none of the security properties, the no-read-rule and no-write-rule indeed, of the ACE scheme would be guaranteed further. In this chapter, we show our interest on the problem how to increase the reliability of the sanitizer in ACE and therefore increase the robustness of the whole ACE scheme.

**Existing ACEs.** In [DHO16], Damgard et. al. aims to construct the ACE scheme in such a way that the functionality of the sanitizer is simplified as much as possible and the knowledge it learns is minimal. Namely, the sanitizer must process every incoming ciphertexts in an easy way, and the processing depends neither on the original message of the ciphertext nor on the access policies in the system. Even the resulted ACE scheme satisfies the requirements mentioned above properly, the sanitizer in the scheme still learns too much. Explicitly, in Damgard's work, the sanitizer utilizes keyed-sanitation algorithm to process received ciphertexts, which requires the sanitizer to be installed with a set of sanitizing keys which are indeed the additive inverse of all uses' encryption keys in the ACE. This fact implies that the sanitizer can create new access polices and then manipulate the communication in the system when collusion attack is allowed. More precisely, the sanitizer can collude with one user and allocate several encryption keys, which are not owned by this user before, to it, the sanitizer therefore empowers this user with the capability to send messages to users who it cannot communicate with previously. In this case, the sanitizer actually creates new access policies. To solve such problem, Fuchsbauer et. al. [FGKO17] gives new construction of the ACE in which the sanitizer does not need sanitizing keys to do sanitation.

The newly constructed ACE scheme in [FGKO17] improved Damgard's scheme in two aspects. To begin with, the complexity, in terms of key and ciphertext size, of the proposed scheme in [FGKO17] is polylogarithmic in terms of the numbers of identities $n$ in the system under standard cryptographic assumptions, while that of the scheme in [DHO16] is exponential in $n$ under the same kind of assumptions.

Most importantly, the constructions in [DHO16] need the sanitizer to store some secret information, such requirement could cause several security issues mentioned above and then incur the proposed ACE scheme insecure. In contrast, the scheme proposed in [FGKO17] does not need secret keys to do sanitation and therefore does not have such problem. As a result, it significantly reduces the probability of breaking the security of the scheme. Also, benefiting from the non-keyed sanitation, the sanitizing algorithm of the sanitizer in [FGKO17] is oblivious to the keys of the possible receivers of a incoming ciphertext. However, the sanitizer in [DHO16] has to sanitize each component of the incoming ciphertext using a receiver-dependent procedure. Literally, the scheme in [FGKO17] also has advantage over that in [DHO16] with respect to the efficiency of the sanitizing algorithm.

**Motivations.** Since the proposing of the primal work in [DHO16], one successive work is primarily on improving the efficiency of the construction from standard cryptographic assumptions [FGKO17]. There also exists a work giving new constructions from non-standard ones such as cryptographic obfuscation or learning with error(LWE) assumption [TZMT17]. It has been declared in [FGKO17] that it is still unclear whether there exists ACE scheme with compact size ciphertext under standard assumptions. Considering there is no compact size ciphertext ACE scheme under non-standard assumptions found in previous works, in this chapter, we are trying to give such a ACE construction. Besides, as we have discussed before, the ACE scheme with only one sanitizer empowers the sanitizer with the capability of producing new access policies. Furthermore, such scheme also encounters single-point-failure problem. Either of those problems would incur the insecurity of the ACE system. Since such problems in ACE systems has seldom been considered or addressed before, we treat it as one of our concern in this chapter.

**Contributions.** In this chapter, our contribution focuses on the following two aspects; We give the first ACE scheme construction with compact size ciphertext. Our construction borrows idea from the primitive anonymous broadcast encryption [LPQ12], the resulted scheme keeps not only the ciphertext size compact but also the key size of each users in the ACE compact. Literally, the key size of our ACE scheme is polylogarithmic in terms of the number of layers in the ACE, while that in [DHO16] and [FGKO17] is exponential and polylogarithmic respectively in terms of the number of users. As the number of layers should be at least smaller than that of users in the ACE, the key size of our construction should be more compact than that in previous construction mentioned before.

Our more significant contribution of this chapter is giving a decentralized implementation of the sanitizer in the ACE to restrain its capability and also increase its reliability. In this chapter, we use secret sharing technique to allow the sanitizing

key to be shared among $n$ sanitizers, and only exact $t$ of them can collaboratively transform a ciphertext into a valid sanitized ciphertext. Which can then be correctly decrypted by its receivers. In our construction, each of the $n$ sanitizers is installed with an unique sanitizing key and would execute the same sanitizing algorithm on the ciphertexts, which are either not sanitized or partially sanitized. One ciphertext can only be viewed as a partially sanitized ciphertext and cannot be decrypted by its intended receivers until it is processed by $t$ sanitizers. Unlike previous scheme with only one sanitizer, our construction distributes the sanitizing functionality of the origin ACE among $n$ sanitizers. It is impossible for one of sanitizers in our construction to produce a new access policy, so our construction imposes restriction on the capability of the sanitizer. Besides, aone message sender in our ACE construction can choose the $t$ sanitizers itself to collaboratively produce a valid sanitized ciphertext. Even some of the $n$ sanitizers cannot provide service or off-line, the whole ACE system can never encounter the single-point-failure and still work as normal. Our construction improves the reliability of the sanitizer and even the robustness of the whole ACE system.

### 6.1.1 Chapter Organization

The rest of this chapter is organized as follows. In Section 6.2, we first formalize useful notations and primitives, then we give a definition of our ACE scheme. We also present two security models to cover the no-read rule and no-write rule property in this part. In Section 6.3, in order to make the description of our scheme more understandable, we first present a new notion of "sanitizing pipeline", then we present a concrete construction of our ACE with compact size ciphertext and decentralized sanitizers. We give security proofs in Section 6.4 and conclude this chapter in Section 6.5.

## 6.2 Primitives and Definitions

*Notations.* Here, for the benefit of consistency, we give the notations used throughout the whole chapter. There are always three types of users involved in the ACE scheme, we denote them the message sender *Se*, the message sanitizer *San* and the message receiver *Re* separately. For a specific user, he can play the role of both *Se* and *Re*. We use $ke, kd$ to represent this user's encryption and decryption key respectively. Assuming there are $l$ layers in the ACE system, when a user in layer $\alpha \in [l]$ can send messages to a receiver in layer $\beta \in [l]$, we use the notation $\alpha \times \beta \to 1$ to denote such access policy, otherwise $\alpha \times \beta \to 0$. We use the access policy set $P : [l] \times [l] \to \{0, 1\}$ to cover the collection of all the access polices defined in the

ACE system. When there exist $n$ message sanitizers in our ACE definition, we assume each of them hold a unique secret sanitizing key $ks$. In order to keep the consistency of the description of our ACE system, we use [u+1,u+n] to denote the list $\{u+1, u+2, \cdots, u+n\}$ and to represent identities of the $n$ sanitizers. For simplicity, we use the notation $[u+n]$ to represent all identities of users involved in the ACE definition.

### 6.2.1 Primitives

**Broadcast Encryption.** The cryptographic primitive Broadcast Encryption(BE) was first introduced by Fiat et. al. in [FN93], it ensures that a message sender can choose a group of users and send encrypted messages to them, also, only users in the chosen group can decrypt such ciphertext using their private keys. According to [BGW05], a broadcast encryption system, without lose of generality, consists of the following four algorithms:

- $\mathsf{Setup}(\lambda, n, \ell)$. Takes as input the security parameter $\lambda$, the number of receivers $n$ and the maximal size $\ell \leq n$ of a broadcast recipient group, it outputs the master public and secret key $mpk, msk$.

- $\mathsf{KeyGen}(i, msk)$. On inputs an index $i \in \{1, \cdots, n\}$ and the master secret key $msk$, this algorithm outputs a decryption key $d_i$ for that user.

- $\mathsf{Enc}(S, mpk)$. On input a subset $S \subset \{1, \cdot, n\}$ and the master public key $mpk$, when $|S| \leq \ell$, this algorithm outputs a pair $(Hdr, K)$ where $Hdr$ is called the herder and $K \in \mathcal{K}$ is a message encryption key. Where $\mathcal{K}$ is the key space of one symmetric encryption scheme $\mathcal{E}_{sym}$. Let $SymEnc$ and $SymDec$ be the encryption and decryption algorithm of $\mathcal{E}_{sym}$ respectively. Let $M$ be the message to be broadcast to the users in set $S$ and $C_M \leftarrow SymEnc(K, M)$ be the encryption of $M$ under the symmetric key $K$. The broadcast message to users in $S$ consists of $(S, Hdr, C_M)$.

- $\mathsf{Dec}(S, i, d_i, Hdr, mpk)$. Takes as input a subset $S \subset \{0, \cdots, n\}$, a user with identity $i \in \{1, \cdots, n\}$ and the decryption key $d_i$, if $i \in S$, then the algorithm outputs a symmetric key $K \in \mathcal{K}$. $K$ then can be used to decrypt $C_M$ to obtain $M$.

Since its invention in [FN93], many BE systems have been proposed and enjoy various flavors; some of BE schemes [BH08, KSAS15] may follow the traditional public key encryption construction while most of them [BGW05, Del07, DPP07, GW09, BWZ14, PPSS13] apply the hybrid encryption methodology, with such technique, the whole ciphertext of the BE system can be represented as a tuple with two

parts $(\mathsf{Hdr}, \mathsf{C}_{\mathsf{sk,m}})$ where $\mathsf{Hdr}$ is an encryption of a symmetric key $sk$ and $\mathsf{C}_{\mathsf{sk,m}}$ is a symmetric encryption of the broadcast contents using the $sk$. The chosen receiver group in the BE schemes could be static [FN93, ZWM13] or dynamic [DPP07], if the receiver group can be changed by the broadcaster freely, we say it is dynamic, otherwise static. In some BE schemes, the receivers' keys can even be revoked [SCG+16, LMG+17] to achieve forward-secrecy. One fundamental security property of the BE is collision resistance, a broadcast encryption is said to be $(t, n)-$collusion secure if for any subset $\mathcal{R}$, where $\mathcal{R} \subset \mathcal{U}, |\mathcal{R}| = r, r \leq t, |\mathcal{U}| = n$, users in $\mathcal{R}$ can by no means infer information about the broadcast messages, the broadcast encryption with $(n, n)-$collusion secure is said to be fully collusion resistance, BE schemes of such kind can be found in [BGW05, DPP07].

Considering providing key-privacy, or receiver anonymity [BBDP01], in BE schemes, Libert et. al. [LPQ12] proposed the anonymous broadcast encryption, the two constructions given in that paper are generic and take one IND-CCA secure PKE as their basis. To hide the real intended receivers of a broadcast content, the schemes are constructed in such a manner that the resulted ciphertext size is linear with the number of users in the BE system. This paper also declared that it should be impossible to construct anonymous broadcast encryption with compact size ciphertext. Latter, [FP12] gives the first anonymous broadcast encryption with sublinear ciphertexts, however, the proposed schemes is based the anonymous IBE and such requirement is stronger than that in [LPQ12], besides, the security of the schemes is analyzed under the so-called subset cover framework, which is seldom used before. It seems feasible to give anonymous broadcast encryption when the intended receiver is static, and the construction in [ZWM13] gives evidence to this statement.

**Secreting Sharing.** The secret sharing scheme has been invented decades ago and usually acts as a very important ingredient in many cryptographic protocols such as secure multi-party computation, threshold cryptography, access control, attribute-based encrypiton and generalized oblivious transfer [Tas11].

Generally, one secret sharing scheme involves a dealer who holds the whole secret $s$, a set of $n$ parties, each of whom holds a share of the secret of the dealer, and a collection $\mathcal{AS}$ of subsets of users from the $n$ parties called the access structure. The secret sharing scheme can be defined using the following algorithms;

- $\mathsf{Setup}(\lambda, n)$. Given the security parameter $\lambda$ and the whole $n$ parties involved in the scheme, the dealer produces a secret $s$ it wants to share as well as an access structure $\mathcal{AS}$. The dealer keeps $s$ and sends $\mathcal{AS}$ to all the $n$ parties. Note that if $\mathcal{AS}$ is the collection of all possible subsets of at least $t$ users from the $n$ parties, then such scheme is called the specific $t$ out of $n$ threshold secret

sharing.

- $\mathsf{ShDis}(\mathcal{AS}, s, i)$. Given $\mathcal{AS}, s$ and one user's identity $i$, the dealer distributes one secret share $v_i$ to that user.

- $\mathsf{SeCon}(\mathcal{AS}, \{v_i\})$. For one user with identity $i$, if he has received a set of secret shares $\{v_i\}$ from other users and those users can form at least one subset in $\mathcal{AS}$, then this user can reconstruct the whole secret $s$ from $\{v_i\}$

The scheme ensures that any subset in the access structure $\mathcal{AS}$ can reconstruct the secret from shares of users in that subset collaboratively and any subset not in $\mathcal{AS}$ can reveal nothing even partial information about the secret. Despite the fact that there exist many different types of secret sharing schemes at present such as the threshold kind in [Sha79], the undirected $s-t$ connectivity kind in [BI92] and the monotone formulae construction kind in [BL88], we give a secret sharing scheme which is similar to the classical shamir one but does not inherit the threshold characteristic. Explicitly, our scheme demands that each subset of the access structure contains only $t$ users and only the exact $t$ users in one specific subset of the access structure can reconstruct the secret collaboratively.

### 6.2.2 Defining Our ACE

An ACE scheme with decentralized sanitizers is defined by the following polynomial time algorithms:

- $\mathsf{Setup}(P, \lambda)$. On input the security parameter $\lambda$ and a access policy set $P : [u] \times [u] \to \{0, 1\}$, the $\mathsf{Setup}$ algorithm outputs a master secret key $msk$ and the public parameter $pp$, which include the description of the message space $\mathcal{M}$, the ciphertext space $\mathcal{C}$ and the sanitized ciphertext space $\mathcal{C}'$.

- $\mathsf{KeyGen}(msk, i, t)$. On input $msk$, an identity $i \in [u + n]$ and a user type $t \in \{Se, Re, San\}$, the key generation algorithm $\mathsf{KeyGen}$ produces the following different types of keys accordingly:

  - $ke_i = \mathsf{KeyGen}(msk, i, Se)$ when the user with identity $i$ is a message sender, $t = Se$, and $ke_i$ is called the encryption key for that user.
  - $kd_i = \mathsf{KeyGen}(msk, i, Re)$ when the user with identity $i$ acts as a message receiver, $t = Re$, and $kd_i$ is called the decryption key for that user.
  - $ks_i = \mathsf{KeyGen}(msk, i, San)$ when the user with identity $i$ plays the role of a message sanitizer, $t = San$, and $ks_i$ is called the sanitizing key for that user.

- $\mathsf{Enc}(ke_i, m)$. The encryption algorithm $\mathsf{Enc}$, on input an encryption key $ke_i$ and a message $m \in \mathcal{M}$, outputs a ciphertext $c \in \mathcal{C}$.

- $\mathsf{Sanit}(c, SP_l)$. For one incoming ciphertext $c \in \mathcal{C}$, a sanitizer in one chosen sanitizing pipeline $SP_l$ would process it using this sanitation algorithm $\mathsf{Sanit}$ with its own sanitizing key, and then relay the result to another sanitizer in the same path, and the next sanitizer would do the same as its predecessor. Our ACE scheme with decentralized sanitizers requires that $c$ should be processed by all $t$ sanitizers in the sanitizing pipeline $SP_l$ collaboratively before becoming a valid sanitized ciphertext $c' \in \mathcal{C}'$.

- $\mathsf{Dec}(c', kd_j)$. On input a sanitized ciphertext $c' \in \mathcal{C}'$ and a decryption key $kd_j$, the decryption algorithm $\mathsf{Dec}$ recovers the message $m' \in \mathcal{M} \cup \{\perp\}$.

### 6.2.3 Security Notions for Our ACE

Our ACE scheme must satisfy requirements formalized below:

**Definition 6.1 (Correctness)** *For all $m \in \mathcal{M}, i, j \in [u]$ such that $P(i, j) = 1$:*

$$\Pr[\mathsf{Dec}(kd_j, \underbrace{\mathsf{Sanit}(ks_t, \cdots, \mathsf{Sanit}(ks_l, \mathsf{Enc}(ke_i, m))))}_{t}) \neq m] \leq negl(\lambda)$$

*with $(pp, msk) \leftarrow \mathsf{Setup}(1^\lambda, P), ke_i \leftarrow \mathsf{KeyGen}(msk, i, Se)$,$kd_j \leftarrow \mathsf{KeyGen}(msk, j, Re)$ and $ks_l \leftarrow \mathsf{KeyGen}(msk, l, San)$, where $l \in [u+1, u+n]$. The above notation denotes that the encrypted message should be processed by exact $t$ different sanitizers in the same sanitizing pipeline before becoming a valid sanitized ciphertext and then being decrypted to a valid plaintext, otherwise, the probability of a correct decryption should be negligible. The probability is taken over the random coins of all involved algorithms.*

**Definition 6.2 (No-Read Rule)** *To define the No-Read Rule in our ACE scheme, we consider the game, which is played between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$, described in Table 6.1.*

*Where $P : [u] \times [u] \rightarrow \{0, 1\}$ is the given access policy set and $t \in \{Se, Re, San\}$, $|m_0| = |m_1|$, $i \in [u]$ and for all queries to $\mathcal{O}_G$ with $q = (j, Re)$, the equation $P(i, j) = 0$ always hold. we say that the adversary $\mathcal{A}$ wins the No-Read game if its output $b' = b$.*

*Let $\Pr[\mathcal{A}$ wins the No-Read game$]$ denote the probability the $\mathcal{A}$ wins the predefined game and $\mathsf{Adv}_{\mathcal{A}}^{No\text{-}Read}(ACE)$ its advantage to win the game, then an ACE scheme is said to satisfy the No-Read Rule if for all probabilistic polynomial time*

**Table 6.1:** The No-Read Rule

| Game Definition | Oracle Definition |
|---|---|
| 1. $(pp, msk) \leftarrow \mathsf{Setup}(1^\lambda, P)$;<br>2. $(m_0, m_1, i) \leftarrow \mathcal{A}^{\mathcal{O}_G(\cdot),\mathcal{O}_E(\cdot)}(pp)$;<br>3. $b \leftarrow \{0,1\}$<br>4. $c \leftarrow \mathsf{Enc}(\mathsf{KeyGen}(msk, i, Se), m_b)$<br>5. $c' \leftarrow \underbrace{\mathsf{Sanit}^{\mathcal{O}_\mathsf{G}()}(\cdots, \mathsf{Sanit}^{\mathcal{O}_\mathsf{G}()}(ks_1, c))}_{t}$<br>6. $b' \leftarrow \mathcal{A}^{\mathcal{O}_G(\cdot),\mathcal{O}_E(\cdot)}(c')$ | $\mathcal{O}_G(i,t)$ :<br>1. $k_i \leftarrow \mathsf{KeyGen}(msk, i, t)$<br><br>$\mathcal{O}_E(i,m)$ :<br>1. $ke_i \leftarrow \mathsf{KeyGen}(msk, i, Se)$;<br><br>2. $c \leftarrow \mathsf{Enc}(ke_i, m)$<br>3. $c' \leftarrow \underbrace{\mathsf{Sanit}^{\mathcal{O}_\mathsf{G}()}(\cdots, \mathsf{Sanit}^{\mathcal{O}_\mathsf{G}()}(ks_1, c))}_{t}$ |

*(PPT) algorithm* $\mathcal{A}$

$$\mathsf{Adv}_{\mathcal{A}}^{\textit{No-Read}}(ACE) = 2|\Pr[\mathcal{A} \textit{ wins the game}] - \frac{1}{2}| \leq negl(\lambda).$$

*Remark.* The No-Read Rule model in [DHO16] also covers the sender anonymity, or key-privacy, property when the second, fourth step of our game definition is changed to

$$(m_0, m_1, i_0, i_1) \leftarrow \mathcal{A}^{\mathcal{O}_G(\cdot),\mathcal{O}_E(\cdot)}(pp), c \leftarrow \mathsf{Enc}(\mathsf{KeyGen}(msk, i_b, Se), m_b)$$

accordingly and the requirement $P(i,j) = 0$ is changed to

$$m_0 = m_1, P(i_0, j) = P(i_1, j).$$

It is easy to find that our model can be extended to guarantee the sender anonymity with the above minimal modification, and the corresponding security proof would not be changed a lot indeed. Here, for simplicity , we first concentrate on the basic No-Read property.

**Definition 6.3 (No-Write Rule)** *To define the No-Write Rule in our ACE scheme, we consider the game, which is played between a challenger* $\mathcal{C}$ *and a stateful adversary* $\mathcal{A}$*, described in Table 6.2.*

*Let* $Q_S$ *(resp.* $Q$*) be the set of queries issued by* $\mathcal{A}$ *to* $\mathcal{O}_S$ *(resp. both* $\mathcal{O}_S$ *and* $\mathcal{O}_R$*). Let* $I_S$ *be all the identities* $i \in [u]$ *such that* $(i, Se) \in Q_S$ *and let* $J$ *be the set of all identities* $j \in [u]$ *such that* $(j, Re) \in Q$*. When* $(l, San) \notin Q, i' \in I_S \cup \{0\}$ *and* $\forall i \in I_S, j \in J, P(i,j) = 0$*, if the adversary's final output* $b' = b$*, we say that* $\mathcal{A}$ *wins the No-Write game defined above.*

*Let* $\Pr[A$ *wins the No-Write game*$]$ *denote the probability when the event* $b' = b$ *happens and* $\mathsf{Adv}_{\mathcal{A}}^{\textit{No-Write}}(ACE)$ *denote* $\mathcal{A}$*'s advantage when* $\mathcal{A}$ *wins this game, then*

**Table 6.2:** The No-Write Rule

| Game Definition | Oracle Definition |
|---|---|
| 1. $(pp, msk) \leftarrow \mathsf{Setup}(1^\lambda, P)$;<br>2. $(i', c, m) \leftarrow \mathcal{A}^{\mathcal{O}_S(\cdot),\mathcal{O}_E(\cdot)}(pp)$;<br>3. $ke_{i'} \leftarrow \mathsf{KeyGen}(msk, i', Se)$;<br>4. $\underbrace{ks_l \leftarrow \mathsf{KeyGen}(msk, l, San), \cdots, ks_t}_{t}$;<br>5. $b \leftarrow \{0, 1\}$;<br>*if* $b = 0, c' \leftarrow \underbrace{\mathsf{Sanit}(\cdots, \mathsf{Sanit}(ks_1, \mathsf{Enc}(ke_{i'}, m)))}_{t}$;<br>*if* $b = 1, c' \leftarrow \underbrace{\mathsf{Sanit}(\cdots, \mathsf{Sanit}(ks_1, c))}_{t}$<br>6. $b' \leftarrow \mathcal{A}^{\mathcal{O}_S(\cdot),\mathcal{O}_R(\cdot)}(c')$ | $\mathcal{O}_S(j, Se)$ :<br>1. $k \leftarrow \mathsf{KeyGen}(msk, j, Se)$<br><br>$\mathcal{O}_R(j, Re)$ :<br>1.$k \leftarrow \mathsf{KeyGen}(msk, j, Re)$<br><br>$\mathcal{O}_E(i, r)$ :<br><br>1. $ke_i \leftarrow \mathsf{KeyGen}(msk, i, Se)$;<br>2. $c \leftarrow \mathsf{Enc}(ke_i, r)$<br>3. $c' \leftarrow \underbrace{\mathsf{Sanit}(\cdots, \mathsf{Sanit}(ks_1, c))}_{t}$ |

*we say an ACE scheme satisfies the No-Write Rule if for all PPT* $\mathcal{A}$

$$\mathsf{Adv}_{\mathcal{A}}^{No\text{-}Write}(ACE) = 2|\Pr[A \text{ wins the game}] - \frac{1}{2}| \leq negl(\lambda)$$

*Remark.* The No-Write security model here also ensures that any set of senders, even corrupted, cannot transfer any information to any set of receivers unless at least one of the senders is allowed to communicate with at least one of the receivers by the access policy $P$. In fact, the above model does not consider whether the sanitizers could be corrupted and thus is valued as the Simplified No-Write Rule. However, in our ACE scheme with decentralized sanitizers, such corruption is permitted, moreover, our scheme even allows the $\mathcal{A}$ to corrupt more than $t$ sanitizers as long as the corrupted sanitizers cannot form a valid "sanitizing pipeline". We give another security model, the extended no-write rule model, to formalize such property.

Before defining a security model allowing the corruption of the decentralized sanitizers in our ACE, we would like to present a simple analysis about the security goal of the adversary captured by such model. According to what we have discussed previously, the sanitizer in [DHO16], in fact, can produce new access policies using the sanitizing key held by itself, which actually violate the No-Write Rule security requirement of the ACE scheme and should be forbidden, while our ACE with decentralize sanitizers gives a promising way to prevent such behavior. In order to produce a valid sanitized ciphertext, our scheme enforces $t$ sanitizers in one specific "sanitizing pipeline" to execute the sanitizing algorithm collaboratively rather than only one single sanitizer. The result is that, if one new access policy is produced, there should be at least $t$ nodes in one specific pipeline colluded together. So, when

**Table 6.3:** The Extended No-Write Rule

| Extended No-Write Rule | |
|---|---|
| Game Definition | Oracle Definition |
| 1. $(pp, msk) \leftarrow \mathsf{Setup}(1^\lambda, P)$;<br>2. $(m_0, m_1, j) \leftarrow \mathcal{A}^{\mathcal{O}_S(\cdot),\mathcal{O}_E(\cdot)}(pp)$;<br>3. $kd_j \leftarrow KeyGen(msk, j, Re)$<br>4. $ke_i^{'} \leftarrow \mathcal{A}^{\mathcal{O}_{San}(\cdot),\mathcal{O}_S(\cdot)}(pp)$<br>5. $b \leftarrow \{0,1\}$<br>6. $c \leftarrow \mathsf{Enc}(ke_i{}', m_b)$<br>7. $c' \leftarrow \underbrace{\mathsf{Sanit}(\cdots, \mathsf{Sanit}(ks_1, c))}_{t}$<br>8. $b' \leftarrow \mathcal{A}^{\mathcal{O}_{San}(\cdot),\mathcal{O}_E(\cdot)}(c')$ | $\mathcal{O}_{\mathcal{S}}(i, Se)$ :<br>1. $ke_i \leftarrow \mathsf{KeyGen}(msk, i, Se)$<br><br>$\mathcal{O}_{\mathcal{R}}(j, Re)$ :<br>1. $kd_j \leftarrow \mathsf{KeyGen}(msk, j, Re)$<br>$\mathcal{O}_{San}(l, San)$ :<br>1. $ks_l \leftarrow \mathsf{KeyGen}(msk, l, San)$<br><br>$\mathcal{O}_E(i, msg)$ :<br>1. $ke_i \leftarrow \mathsf{KeyGen}(msk, i, Se)$;<br>2. $c \leftarrow \mathsf{Enc}(ke_i, msg)$ |

defining security model with respect to the sanitizer in our scheme, the goal of the adversary is to produce a new access policy without corrupting all $t$ users in one specific "sanitizing pipeline" .

**Definition 6.4 (Extended No-Write Rule)** *To define a model capturing the security of the sanitizers, we consider the game, which is played between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$, described in Table 6.3.*

*Let $Q_S, Q_R$ and $Q_{San}$ be the set of queries issued by $\mathcal{A}$ to $\mathcal{O}_S$, $\mathcal{O}_R$ and $\mathcal{O}_{San}$ respectively. Let $I_S$ be all the identities $i \in [u]$ such that $(i, Se) \in \mathcal{Q}_S$, $J_R$ be the set of all identities $j \in [u]$ such that $(j, Re) \in Q_R$ and $L_{San}$ be all identities $l \in [u+1, u+n]$ such that $(l, San) \in Q_{San}$ respectively. We have*

- $\forall i \in I_S, j \in J, P(i, j) = 0$,
- *There exists no "sanitizing pipeline" whose users are all included in $L_{San}$.*

*If the adversary's final output $b' = b$, we say that $\mathcal{A}$ wins the Extended No-Write Rule game defined above. Let $\Pr[A \text{ wins the game}]$ denote the probability that $b' = b$ and $\mathsf{Adv}_{\mathcal{A}}^{Ex\text{-}No\text{-}Write}(ACE)$ denote $\mathcal{A}$'s advantage when $\mathcal{A}$ wins this game, then we say an ACE scheme satisfies the Extended No-Write Rule if for all PPT $\mathcal{A}$*

$$\mathsf{Adv}_{\mathcal{A}}^{Ex\text{-}No\text{-}Write}(ACE) = 2|\Pr[A \text{ wins the game}] - \frac{1}{2}| \leq negl(\lambda)$$

In fact, we find the above two security models, the simplified no-write rule and the extended no-write rule, are considering the same security issue with only minimal

differences. Namely, the former model defines one user $i$'s no-write property in such a manner that $i$ cannot send messages to another user $j$ when the access policy $P(i,j) = 0$ even $i, j$ are all corrupted or $i$ gets help from users who also cannot send messages to $j$, while in the extended no-write rule model, user $i$'s no-write property is defined similarly but with the exception that $i$ can also gets help from at most $t-1$ sanitizers in one sanitizing pipeline rather than just from other users. Intuitively, the extended no-write rule model defined here should have already covered the simplified no-write rule model and is thus stronger than it.

## 6.3 The ACE with Decentralized Sanitizers

In this section, we first illustrate how to construct a sanitizing cluster and how a new "sanitizing pipeline" with $t$ sanitizers is formed when there are $n$ sanitizers existed in the cluster, we also show you that the whole number of sanitizing pipelines and sanitizers in the sanitizing cluster can be increased in an on-demand manner. After that, we give a description of our ACE scheme with compact ciphertext size and decentralized sanitizers in detail.

### 6.3.1 The Sanitizing Cluster and Sanitizing Pipelines

We assume all sanitizers in our ACE system constitute a sanitizing clusters. Our ACE with decentralized sanitizers requires that only $t$ sanitizers can collaboratively fulfill the sanitizing algorithm properly and converts one incoming ciphertext into a valid sanitized ciphertext which can then be decrypted by the intended receiver. To save the computational cost of the sanitizers in the sanitizing cluster when they do sanitization, we introduce the notion sanitizing pipeline. A sanitizing pipeline can be valued as a path predefined by the system authority containing a collection of exact $t$ sanitizers chosen by it from the sanitizer cluster. one ciphertext can never be transformed into a valid sanitized ciphertext until it is processed by every nodes in the pipeline chosen in advance by the message sender. The system authority can actually produce as many sanitizing pipelines as it wants, and the collection of all the pipelines is represented as $\{SP\}$ which should be known by all the nodes in the ACE system. Given a polynomial $F(x)$ with degree $t-1$ such that $F(0) = y$, which is the secret to be shared. When one user with identity $j$ wants to join the sanitizing cluster as a sanitizer, the system authority chooses $x_j \in \mathbb{Z}_p$ and computes $y_j = F(x_j)$, then $y_j$ is allocated to this user as one of its secret, then the system authority would also produce a new sanitizing pipeline $sp_l$ and add this user as one member of this pipeline. Furthermore, as the authority knows all the $t$ sanitizers in $sp_l$, another secret value $f_j = g^{-\prod_{i \neq j \wedge i \in sp_l} \frac{x_i}{x_i - x_j}}$ is computed by the authority in

advance and then distributed to that sanitizer $j$. When one user with identifier $j$ gets its own secret share $(y_j, f_j)$ and the sanitizing pipeline identifier $sp_l$, it can work as a valid sanitizer member in the sanitizer cluster.

### 6.3.2 Our ACE Scheme With Compact Ciphertext Size and Decentralized Sanitizers

Our ACE scheme with compact ciphertext size and decentralized sanitizers (AC-CDS) is defined by the following algorithms;

- $\mathsf{Setup}(\lambda)$: This ACE system setup algorithm is executed by the system authority. Given the security parameter $\lambda$, a bilinear map group system $\mathcal{BM} = (p, g, \mathbb{G}, \mathbb{G}_1, e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_1)$ is generated such that $|p| = \lambda$, $g, h \in \mathbb{G}$ are two randomly selected generators of $\mathbb{G}$ and a secret value $\gamma \in \mathbb{Z}_p$ is chosen, sets $w = g^{\gamma}$. The authority also chooses a cryptographic hash function $\mathcal{H} : \{0,1\}^{\lambda} \rightarrow \mathbb{Z}_p$ which will be viewed as the random oracle in the security analysis. The authority also initializes the sanitizing clusters and sanitizing pipelines using the initialization algorithm defined above, after that, assuming there are $n$ sanitizers and $|\{SP\}|$ sanitizing pipelines in the ACE system, notice that each element in $\{SP\}$ contains a list of sanitizers' identities and represents a unique sanitizing pipeline. Assuming there are $u$ users which can play the role of the message sender and the message receiver, and each of them lays in one specific layer, supposing there are $\mu$ layers at most in the ACE system, let $S_{L_\beta}$ denote the collection of identities of users laying in the $\beta$-th layer where $1 \leq \beta \leq \mu$, only the authority knows $\mathcal{AC} = (S_{L_1}, S_{L_2}, \cdots, S_{L_\alpha}, \cdot, S_{L_\mu})$, that is, only the authority has the knowledge of which user lays in which layer for all the $u$ users. The authority also knows the whole sanitizing key $y \in \mathbb{Z}_p^*$. The authority defines the key space $\mathcal{KM} = \mathbb{G}_1$,the ciphertext space $\mathcal{C} = \mathbb{G}^6$, the sanitized ciphertext space $\mathcal{C}' = \mathbb{G}^6$ respectively. The public parameter $pp = (\mathcal{BM}, w, \mathcal{H}, \{SP\}, \mathcal{KM}, \mathcal{C}, \mathcal{C}')$, the master secret key $msk = (\mathcal{AC}, \gamma, y)$.

- $\mathsf{KeyGen}(msk, pp, i, L_\beta, ty)$: When given $pp, msk$, one specific users' identity $i$, the layer $L_\beta$ this user lays in and its user type $ty \in \{Se, Re, San\}$, the key generation algorithm is executed by the authority as follows;

  - When $ty = Se$, that is, the authority needs to generate an encryption key $ke_i$ for the user with identity $i$. The authority chooses $x_i \xleftarrow{R} \mathbb{Z}_p^*$ for this user with identity $i$ and sets $ke_i$ as;

$$ke_i = (h^{\prod_{i \in S_{L_\beta} \cup \cdots \cup S_{L_\mu}} (\gamma + H(i))}, h^{x_i \prod_{i \in S_{L_\beta} \cup \cdots \cup S_{L_\mu}} (\gamma + H(i))}, g^{-x_i \gamma}, e(g,h)^{x_i}, g^y)$$

– When $ty = Re$, that is, the authority needs to generate a decryption key $kd_i$ for that user. When user with identity $i$ lays in layer $L_\beta$, he can receiver messages sent from layers below its own, that is, his decryption key should be able to decrypt messages sent from layers from $L_1$ to $L_\beta$. Here the authority construct the decryption key of this user in such a manner that $kd_i$ contains $\beta$ components and each component is response for decrypting ciphertexts from one specific layer. $kd_i$ can be represented as;

$$\begin{aligned} kd_i = (kd_{i0} &= g^{\frac{1}{\gamma+H(i)}}, kd_{i1} = h^{\frac{\prod_{l\neq i \wedge l\in S_{L_1}\cup\cdots\cup S_{L_\mu}}{}^{(\gamma+H(l))-1}}{\gamma}}, \\ kd_{i2} &= h^{\frac{\prod_{l\neq i \wedge l\in S_{L_2}\cup\cdots\cup S_{L_\mu}}{}^{(\gamma+H(l))-1}}{\gamma}}, \cdots\cdots \\ kd_{i\beta} &= h^{\frac{\prod_{l\neq i \wedge l\in S_{L_\beta}\cup\cdots\cup S_{L_\mu}}{}^{(\gamma+H(l))-1}}{\gamma}}). \end{aligned}$$

– When $ty = San$, that is, the authority needs to generate a sanitizing key $ks_i$ for that user. To do this, the authority chooses a $m-1$ degree function $F(x)$ such that $F(0) = y$. For each sanitizer $j$ in the specific sanitizing pipeline, denoted by $sp_l$, the authority allocate a $x_j \xleftarrow{R} \mathbb{Z}_p^*$ to it and computes $y_j = F(x_j)$, the sanitizing key $ks_i$ of the sanitizer with identity $i$ should be;

$$ks_i = g^{-y_i \prod_{j\neq i \wedge j\in SP_l} \frac{x_j}{x_j - x_i}}$$

- $\mathsf{Enc}(m, ke_i, pp)$: Our ACE scheme borrows idea from the hybrid encryption scheme, that is, the asymmetric encryption scheme actually encrypts a symmetric encryption key, the real ciphertext is an encryption of the origin message using a symmetric key encryption scheme with the symmetric key encrypted by the previous asymmetric encryption scheme. Here, we only focus on the asymmetric part of our whole ACE and just use $SE_{sk}(m)$ to represent the symmetric encryption part. When given a message $m \in \mathcal{M}$, one message sender with identity $i$ in layer $L_\beta$ encrypts it as follows;

$$\begin{aligned} k_0 &, k_1, r_s \xleftarrow{R} \mathbb{Z}_p^* \\ C_1 &= g^{-x_i\gamma k_1}, C_2 = g^{-x_i\gamma k_1 r_s} g^{-k_0\gamma} g^y, \\ C_3 &= h^{k_1 x_i \prod_{i\in S_{L_\beta}\cup\cdots\cup S_{L_\mu}}{}^{(\gamma+H(i))}}, \\ C_4 &= h^{k_1 x_i r_s \prod_{i\in S_{L_\beta}\cup\cdots\cup S_{L_\mu}}{}^{(\gamma+H(i))}} h^{k_0 \prod_{i\in S_{L_\beta}\cup\cdots\cup S_{L_\mu}}{}^{(\gamma+H(i))}} \\ C_5 &= e(g,h)^{x_i k_1}, C_6 = e(g,h)^{x_i k_1 r_s} \end{aligned}$$

The symmetric key should be $sk = e(g,h)^{k_0}$, the real ciphertext should be

$C_7 = SE_{sk}(m)$. So , the whole ciphertext of our ACE is the tuple $\mathcal{CT} = (L_\beta, C_1, C_2, C_3, C_4,$
$C_5, C_6, C_7)$. The message sender then chooses one sanitizing pipeline $SP_l$ from all pipelines which are hard-wired with this sender.

- $\mathsf{Sanit}(\mathcal{CT}^v, pp, ks_{lv+1})$: Given a ciphertext $\mathcal{CT}^v = (L_\beta, C_1^v, C_2^v, C_3^v, C_4^v, C_5^v, C_6^v,$ $Ci^v)$, no matter whether it is received from the message sender or from a sanitizer's predecessor, this sanitizer does as follows;

$$\begin{aligned}
r_{v+1} &\overset{R}{\leftarrow} \mathbb{Z}_p^*, C_1^{v+1} = g^{-x_i\gamma k_1}, C_2^{v+1} = g^{-x_i\gamma k_1 r_s} g^{-k_0\gamma} g^y ks_{lv+1} (C_1^v)^{r_{v+1}},\\
C_3^{v+1} &= h^{k_1 x_i \prod_{i\in S_{L_\beta}\cup\cdots\cup S_{L_\mu}} (\gamma+H(i))},\\
C_4^{v+1} &= h^{k_1 x_i r_s \prod_{i\in S_{L_\beta}\cup\cdots\cup S_{L_\mu}} (\gamma+H(i))} h^{k_0 \prod_{i\in S_{L_\beta}\cup\cdots\cup S_{L_\mu}} (\gamma+H(i))} (C_4^v)^{r_{v+1}}\\
C_5^{v+1} &= e(g,h)^{x_i k_1}, C_6^{v+1} = e(g,h)^{x_i k_1 r_s} (C_5^v)^{r_{v+1}}, C_7^{v+1} = C_7^v
\end{aligned}$$

After this sanitizer proceeds the incoming ciphertext as above properly, it would relay the partially sanitized ciphertext to the next sanitizer laying in the same sanitizing pipeline as itself if it is not the final sanitizer in this pipeline, otherwise, this sanitizer would relay the sanitized ciphertext to the intended receiver.
Notice that all sanitizers in $SP_l$ will do the same as what we described above. When one ciphertext tuple $\mathcal{CT} = (L_\beta, C_1, C_2, C_3, C_4, C_5, C_6)$ goes through the sanitizing pipeline $SP_l$ and is processed by each of the $t$ sanitizers in $SP_l$, the finally sanitized ciphertext should be represent as:

$$\begin{aligned}
C_1^t &= C_1 = g^{-x_i\gamma k_1}, C_2^t = g^{-x_i\gamma k_1 r_s} g^{-k_0\gamma} g^y (C_1)^{r_1+r_2+\cdots+r_t} ks_{l1} ks_{l2} \cdots ks_{lt},\\
C_3^t &= C_3 = h^{k_1 x_i \prod_{i\in S_{L_\beta}\cup\cdots\cup S_{L_\mu}} (\gamma+H(i))},\\
C_4^t &= h^{k_1 x_i r_s \prod_{i\in S_{L_\beta}\cup\cdots\cup S_{L_\mu}} (\gamma+H(i))} h^{k_0 \prod_{i\in S_{L_\beta}\cup\cdots\cup S_{L_\mu}} (\gamma+H(i))} (C_3)^{r_1+r_2+\cdots+r_t}\\
C_5^t &= C_5 = e(g,h)^{x_i k_1}, C_6^t = e(g,h)^{x_i k_1 r_s} (C_5)^{r_1+\cdots+r_t}, C_7^t = C_7
\end{aligned}$$

As we can see,

$$\begin{aligned}
& ks_{l1} ks_{l2} \cdots ks_{lt}\\
= \; & g^{-y_{l1} \prod_{j\neq l1\wedge j\in SP_l} \frac{x_j}{x_j - x_{l1}}} g^{-y_{l2} \prod_{j\neq l2\wedge j\in SP_l} \frac{x_j}{x_j - x_{l2}}} \cdots g^{-y_{lt} \prod_{j\neq lt\wedge j\in SP_l} \frac{x_j}{x_j - x_{lt}}}\\
= \; & g^{-F(0)} = g^{-y}
\end{aligned}$$

When the last sanitizer in $SP_l$ has executed its sanitizing algorithm on one incoming partially sanitized ciphertext, he can just send $CT' = (L_\beta, C_1', C_2', C_3',$

$C'_4$) to the intended receivers, where

$$\begin{aligned}
C'_1 &= C_2^t = g^{-x_i\gamma k_1 r_s} g^{-k_0\gamma} g^y (C_1)^{r_1+r_2+\cdots+r_t} ks_{l1} ks_{l2} \cdots ks_{lt} \\
&= g^{-(x_i k_1(r_s+r_1+\cdots+r_t)+k_0)\gamma} \\
C'_2 &= C_4^t = h^{k_1 x_i r_s \prod_{i\in S_{L_\beta}\cup\cdots\cup S_{L_\mu}} (\gamma+H(i))} h^{k_0 \prod_{i\in S_{L_\beta}\cup\cdots\cup S_{L_\mu}} (\gamma+H(i))} (C_3)^{r_1+\cdots+r_t} \\
&= h^{(k_1 x_i(r_s+r_1+\cdots+r_t)+k_0)\prod_{i\in S_{L_\beta}\cup\cdots\cup S_{L_\mu}} (\gamma+H(i))} \\
C'_3 &= C_6^t = e(g,h)^{x_i k_1(r_s+r_1+r_2+\cdots+r_t)} \\
C'_4 &= SE_{sk}(m) \text{ where } sk = e(g,h)^{k_0}
\end{aligned}$$

and $L_\beta$ denotes the layer this message sender lays in.

- $\mathsf{Dec}(kd_j, \mathcal{CT}', pp)$: When given a properly sanitized ciphertext $\mathcal{CT}'$ and one user's decryption key $kd_j$, this user would first judge whether he is able to recover the origin message of the received ciphertext by checking whether the layer the receiver lays in is higher than that of the message sender. If the receiver can decrypt the ciphertext, it does as follows;

$$\begin{aligned}
sk' &= \frac{e(C'_1, kd_{j\beta}) e(C'_2, kd_{j0})}{C'_3}, \text{ sets } K = (x_i k_1(r_s + r_1 + \cdots + r_t) + k_0) \\
&= \frac{e(g^{-K\gamma}, h^{\frac{\prod_{l\neq i\wedge l\in S_{L_\beta}\cup\cdots\cup S_{L_\mu}} (\gamma+H(l))-1}{\gamma}}) e(h^{K\prod_{i\in S_{L_\beta}\cup\cdots\cup S_{L_\mu}} (\gamma+H(i))}, g^{\frac{1}{\gamma+H(i)}})}{e(g,h)^{K-k_0}} \\
&= \frac{e(g,h)^{K(1-\prod_{l\neq i\wedge l\in S_{L_\beta}\cup\cdots\cup S_{L_\mu}} (\gamma+H(l)))} e(g,h)^{K\prod_{l\neq i\wedge l\in S_{L_\beta}\cup\cdots\cup S_{L_\mu}} (\gamma+H(l))}}{e(g,h)^{K-k_0}} \\
&= e(g,h)^{k_0} \\
m' &= DE_{sk'}(C'_4)
\end{aligned}$$

## 6.4 Security Proofs

**Theorem 6.1** *Our ACE scheme holds the No-Read Rule property assuming the $(f,g,F)-GDDHE$ problem is hard in the group system $\mathcal{BM} = (p, g_0, h_0, \mathbb{G}_1, \mathbb{G}_T, e(\cdot,\cdot))$ when the hash function $H$ is modeled as random oracle. Concretely, if there is an adversary $\mathcal{A}$ which can break our scheme with non-negligible probability $\epsilon$, supposing $\mathcal{A}$ makes at most $q_H, q_{ke}, q_{kd}$ queries to the $H$ hash oracle,encryption key query oracle and decryption key query oracle respectively, then we can construct another algorithm $\mathcal{B}$ that solves the $(f,g,F)$-GDDHE problem in the given group system with advantage at least $\frac{1}{2}\cdot(\frac{q_H-1}{q_H})^{q_{kd}}\cdot\frac{1}{q_H}\cdot\epsilon$, where $q_H, q_{kd}$ are defined above.*

**Proof.** Assuming the adversary $\mathcal{A}$ can break our ACE scheme with non-negligible probability $\epsilon$, we show how to construct another algorithm $\mathcal{B}$ solving the $(f,g,F)$-

GDDHE problem from $\mathcal{A}$, $\mathcal{B}$ and $\mathcal{A}$ interacts as below.

- Setup. The algorithm $\mathcal{B}$ is first given a $(f, g, F) - GDDHE$ problem instance

$$\mathcal{BM} = (p, \mathbb{G}_1, \mathbb{G}_T, e(\cdot,\cdot)), (g_0, g_0^{\gamma}, g_0^{\gamma^2}, \cdots, g_0^{\gamma^{t-1}}, g_0^{\gamma \cdot f(\gamma)}, \quad g_0^{k \cdot \gamma \cdot f(\gamma)}) \in \mathbb{G}_1,$$
$$(h_0, h_0^{\gamma}, h_0^{\gamma^2}, \cdots, h_0^{\gamma^{2n}}, \quad h_0^{k \cdot g(\gamma)}) \in \mathbb{G}_1 \quad \text{and } T \in \mathbb{G}_T$$

  where $p$ is a reasonably large prime number and also the order of $\mathbb{G}_1$ and $\mathbb{G}_T$.

  Given the bilinear map group system $\mathcal{BM} = (p, g_0, h_0, \mathbb{G}_1, \mathbb{G}_T, e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T)$ from the problem instance, where $|p| = \lambda$, $g_0, h_0 \in \mathbb{G}_1$ are generators of $\mathbb{G}_1$, $\mathcal{B}$ then executes the setup algorithm of our ACE scheme. $\mathcal{B}$ also initializes the sanitizing clusters and sanitizing pipelines using the initialization algorithm defined in our ACE scheme, assuming there are $s$ sanitizers and $|\{SP\}|$ sanitizing pipelines in the ACE system after the execution, notice that each element in $\{SP\}$ contains a list of sanitizers' identities and represents a unique sanitizing pipeline. Assuming there are $u$ users in the user set $\{ID_u\} = \{ID_1, ID_2, \cdots, ID_u\}$, where $u < n$, each of them lays in one specific layer, supposing there are $\mu$ layers at most in the ACE system, let $S_{L_\beta}$ denote the collection of identities of users laying in the $\beta$-th layer where $1 \leq \beta \leq \mu$, only $\mathcal{B}$ knows $\mathcal{AC} = (S_{L_1}, S_{L_2}, \cdots, S_{L_\beta}, \cdot, S_{L_\mu})$, that is, only $\mathcal{B}$ has the knowledge of which user lays in which layer for all the $u$ users. $\mathcal{B}$ also chooses one element $y \in \mathbb{Z}_p^*$ as the system sanitizing key. $\mathcal{B}$ defines the key space $\mathcal{KM} = \mathbb{G}_1$,the ciphertext space $\mathcal{C} = \mathbb{G}^6$, the sanitized ciphertext space $\mathcal{C}' = \mathbb{G}^6$ respectively. $\mathcal{B}$ also chooses a cryptographic hash function $H : \{0,1\}^{\lambda} \rightarrow \mathbb{Z}_p$ which will be viewed as the random oracle in the security analysis.

  To enable $\mathcal{B}$ to answer the three different types of queries, it randomly chooses $\{I_1, I_2, \cdots, I_{q_H}\}$ from $\mathbb{Z}_p$, where $q_H \leq t$ is the number of queries asked to the $H$ oracle. It also picks a random index $i^* \in \{1, 2, \cdots, q_H\}$. $\mathcal{B}$ defines one polynomial function $f(x)$ as

$$f(x) = \prod_{i=1, i \neq i*}^{q_H} (x + I_i) = F_{q_H - 1} x^{q_H - 1} + \cdots + F_2 x^2 + F_1 x^1 + F(0)$$

  $\mathcal{B}$ then sets $g = g_0^{f(\gamma)}$.

  Assuming the user with identity $ID_{i*}$ lays in the layer $L_\beta$. $\mathcal{B}$ defines a function $g(x)$ as

$$g(x) = \prod_{ID \in S_{L_\beta} \cup S_{L_{\beta+1}}, \cdots, S_{L_\mu}} (x + H(ID))$$

$\mathcal{B}$ then sets

$$h = h_0^{g(\gamma)}, w = g_0^{\gamma \cdot f(\gamma)} = g^{\gamma}$$

$\mathcal{B}$ can also compute the tuple $(h, h^{\gamma}, \cdots, h^{\gamma^n})$ from the given instance and the predefined function. $\mathcal{B}$ then sets $h^{\gamma} = h_1$, then the above instance turns to

$$(h_1^{\frac{1}{\gamma}}, h_1, h_1^{\gamma}, \cdots, h_1^{\gamma^{n-1}}).$$

$\mathcal{B}$ gives the public parameter $pp = (\mathcal{BM}, \{ID_u\}, \{SP\}, \mathcal{KM}, \mathcal{C}, \mathcal{C}')$, $\mathcal{B}$ itself knows $y$.

- Hash query phase. At any time, $\mathcal{A}$ can query the random oracle $H$, the total number of distinct query to $H$ is $q_H$. In order to answer such query, $\mathcal{B}$ maintains a list $\mathcal{L}_H$ and responds the query as follows.
  For one query with input $ID$ to the oracle $H$, $\mathcal{B}$ first checks whether there already has been one tuple $(ID, I)$ in the $\mathcal{L}_H$.
  - If yes, $\mathcal{B}$ responds with $H(ID) = I$ to the adversary.
  - Otherwise, let $ID$ be the $i$-th distinct query to H, $\mathcal{B}$ responds with $H(ID) = I_i$ to the adversary. $\mathcal{B}$ then add this new tuple $(ID, I_i)$ to the $\mathcal{L}_H$.

- Encryption key query phase. $\mathcal{A}$ can also ask encryption key queries to $\mathcal{B}$ at any time. When $\mathcal{A}$ wants to get the encryption key of user with identity $ID$, $\mathcal{B}$ answers such key query as follows.
  - As $\mathcal{B}$ knows the access policy $\mathcal{AC}$, it would first check which layer the user with identity $ID$ lays in.
  - Supposing the user lays in the layer $L_\delta$, $\mathcal{B}$ would then find all users laying in or above this layer, that is, all users' identities in the set $S_{L_\delta} \cup S_{L_\delta+1}, \cdots, S_{L_\mu}$.
  - $\mathcal{B}$ issues multiple hash queries to $H$ for each identity in the aforementioned set. $\mathcal{B}$ computes the coefficients of $F_{L_\delta}(x) = \prod_{ID \in S_{L_\delta} \cup S_{L_\delta+1}, \cdots, S_{L_\mu}} (x + H(ID))$
  - $\mathcal{B}$ chooses a random value $x_i$, then it computes the following values:

$$\begin{aligned}
ke_{ID}^0 &= h_1^{F_{L_\delta}(\gamma)} = h_0^{\gamma \cdot F_{L_\beta}(\gamma) \cdot F_{L_\delta}(\gamma)}, \\
ke_{ID}^1 &= h_1^{x_i \cdot F_{L_\delta}(\gamma)} = h_0^{x_i \cdot \gamma \cdot F_{L_\beta}(\gamma) \cdot F_{L_\delta}(\gamma)}, \\
ke_{ID}^2 &= g^{-x_i \gamma} = g_0^{-x_i \cdot \gamma \cdot f(\gamma)}, \\
ke_{ID}^4 &= e(g, h_1)^{x_i} = e(g_0, h_0)^{\gamma \cdot f(\gamma) \cdot F_{L_\beta}(\gamma) \cdot x_i}, \quad ke_{ID}^5 = g^y = g_0^{f(\gamma) \cdot y}
\end{aligned}$$

Obviously, all the above values are computable by using the given challenge problem instance, the defined function $f(x)$,$F_{L_\beta}(x)$ and $F_{L_\delta}(x)$ .

– $\mathcal{B}$ gives $ke_{ID} = (ke^0_{ID}, ke^1_{ID}, ke^2_{ID}, ke^3_{ID}, ke^4_{ID}, ke^5_{ID})$ as the respond to $\mathcal{A}$.

- Decryption key query phase. $\mathcal{A}$ can query the decryption keys of users in the ACE system at any time. When $\mathcal{A}$ sends the decryption key query with identity $ID$ as the queried message to $\mathcal{A}$, $\mathcal{B}$ answers it using the following steps.

  – $\mathcal{B}$ first checks whether $H(ID) = I_{i^*}$, if yes, $\mathcal{B}$ aborts the simulation. Otherwise, $\mathcal{B}$ turns to the next step.

  – $\mathcal{B}$ then identifies the layer this user lays in, say $L_\delta$. If $\beta < \delta$, $\mathcal{B}$ aborts the simulation too. That is, $\mathcal{A}$ cannot answer decryption key queries for users who can decrypt the challenge ciphertext.

  – Otherwise, let the symbol $Q_{L_\delta}$ denote $ID_i \neq ID$ and $ID_i \in S_{L_\delta} \cup S_{L_\delta+1}, \cdots, S_{L_\mu}$, it defines a function $F_{\frac{L_\delta}{x}}(x)$ as

$$F_{\frac{L_\delta}{x}}(x) = \frac{\prod Q_{L_\delta}(x + H(i)) - \prod Q_{L_\delta}(H(i))}{x}$$

  – $\mathcal{B}$ computes the following values;

$$\begin{aligned}
kd^0_{ID} &= g^{\frac{1}{\alpha+H(ID)}} = g_0^{\frac{f(\gamma)}{\gamma+H(ID)}} = g_0^{\prod_{ID_i \notin \{ID, ID_{i^*}\}}(\gamma+H(ID_i))} \\
kd^1_{ID} &= h_1^{\frac{\prod_{ID_i \neq ID \text{ and } ID_i \in S_{L_1} \cup S_{L_2}, \cdots, S_{L_\mu}}(\gamma+H(ID_i))-1}{\gamma}} \\
&= h_1^{F_{\frac{L_1}{\gamma}}(\gamma)} h_1^{\frac{\prod Q_{L_1} H(ID_i)-1}{\gamma}} \\
&= h_1^{F_{\frac{L_1}{\gamma}}(\gamma)} h^{\prod Q_{L_1} H(ID_i)-1} \\
&= h_0^{\gamma \cdot f(\gamma) \cdot F_{\frac{L_1}{\gamma}}(\gamma)} h_0^{f(\gamma) \cdot \prod Q_{L_1} H(ID_i)-1} \\
kd^2_{ID} &= h_1^{\frac{\prod_{ID_i \neq ID \text{ and } ID_i \in S_{L_2} \cup S_{L_3}, \cdots, S_{L_\mu}}(\gamma+H(ID_i))-1}{\gamma}} \\
&= h_1^{F_{\frac{L_2}{\gamma}}(\gamma)} h_1^{\frac{\prod Q_{L_2} H(ID_i)-1}{\gamma}}
\end{aligned}$$

$$
\begin{aligned}
kd_{ID}^2 &= h_1^{\frac{\prod ID_i \neq ID \text{ and } ID_i \in S_{L_2} \cup S_{L_3}, \cdots, S_{L_\mu} (\gamma + H(ID_i)) - 1}{\gamma}} \\
&= h_1^{\frac{F_{L_2}}{\gamma}(\gamma)} h_1^{\frac{\prod Q_{L_2} H(ID_i) - 1}{\gamma}} \\
&= h_1^{\frac{F_{L_2}}{\gamma}(\gamma)} h^{\prod Q_{L_2} H(ID_i) - 1} \\
&= h_0^{\gamma \cdot f(\gamma) \cdot \frac{F_{L_2}}{\gamma}(\gamma)} h_0^{f(\gamma) \cdot \prod Q_{L_2} H(ID_i) - 1} \\
\cdots & \\
\cdots & \\
kd_{ID}^\delta &= h_1^{\frac{\prod ID_i \neq ID \text{ and } ID_i \in S_{L_\delta} \cup S_{L_{\delta+1}}, \cdots, S_{L_\mu} (\gamma + H(ID_i)) - 1}{\gamma}} \\
&= h_1^{\frac{F_{L_\delta}}{\gamma}(\gamma)} h_1^{\frac{\prod Q_{L_\delta} H(ID_i) - 1}{\gamma}} \\
&= h_1^{\frac{F_{L_\delta}}{\gamma}(\gamma)} h^{\prod Q_{L_\delta} H(ID_i) - 1} \\
&= h_0^{\gamma \cdot f(\gamma) \cdot \frac{F_{L_\delta}}{\gamma}(\gamma)} h_0^{f(\gamma) \cdot \prod Q_{L_\delta} H(ID_i) - 1}
\end{aligned}
$$

As we can see, all the above values can be computed from the given problem instance and the defined functions.

– $\mathcal{B}$ sends $kd_{ID} = (kd_{ID}^0, kd_{ID}^1, \cdots, kd_{ID}^\beta)$ as the respond to $\mathcal{A}$.

- Challenge phase. In this phase, $\mathcal{B}$ randomly chooses two messages $m_0, m_1$ from the message space $\mathcal{M}$ and the identity $ID^*$ he wishes to challenge, then it sends $(m_0, m_1, ID^*)$ to $\mathcal{B}$. Upon receiving this tuple, $\mathcal{B}$ does as following.

  – $\mathcal{B}$ would first check whether $H(ID^*) = I_{i^*}$, if not, $\mathcal{B}$ aborts the simulation. Otherwise, $\mathcal{B}$ continues.

  – Let $S_{L_{\beta\mu}}$ denote the set $S_{L_\beta} \cup S_{L_\beta+1}, \cdots, S_{L_\mu}$, $\mathcal{B}$ chooses a random bit $b \in \{0, 1\}$ and encrypts $m_b$ using the encryption key of the user with identity $ID^*$, the generated sanitized ciphertext $C'$ is;

$$
\begin{aligned}
C_1' &= g_0^{-k \cdot \gamma \cdot f(\gamma)}, C_2' = h_1^{k \cdot g(\gamma)}, \\
C_3' &= \frac{e(g_0^{\gamma^2 \cdot f(\gamma)}, h_0^{kg(\gamma)})}{T^{\prod ID \in S_{L_{\beta\mu}} (H(ID))} \cdot e(g_0^{k \cdot \gamma \cdot f(\gamma)}, h_0^{q(\gamma)})}. \\
q(\gamma) &= \frac{\prod ID \in S_{L_{\beta\mu}} (\gamma + H(ID)) - \prod ID \in S_{L_{\beta\mu}} (H(ID))}{\gamma} \\
sk &= e(g_0^{f(\gamma)}, h_0^{k \cdot g(\gamma)}), \qquad C_4' = SE_{sk}(m_b)
\end{aligned}
$$

  – $\mathcal{B}$ then sends $(C_1', C_2', C_3', C_4')$ to $\mathcal{A}$ as the challenger ciphertext.

- Guess. Finally, the adversary $\mathcal{A}$ outputs a guess $b' \in \{0, 1\}$ about the encrypted message $m_b$, if $b = b'$, then $\mathcal{A}$ wins the game.

**Analysis.** Here, we finished the simulation part of our proof. Let $\Pr[b = b']$ be the probability $\mathcal{A}$ wins the above game, let $Adv_{\mathcal{B}}{}^{(f,g,F)-gddhe}$ denote the advantage of $\mathcal{B}$ in solving the given $(f, g, F)$-GDDHE problem instance and $\epsilon$ denote the probability $\mathcal{A}$ breaks our ACE scheme, to complete the proof, we give the following probability analysis;

- When T is a random element in $\mathbb{G}_T$, it is easy to verify that the challenge ciphertext $CT' = (C'_1, C'_2, C'_3C'_4)$ is like a one-time-pad, so $\mathcal{A}$ cannot get any useful information from $CT'$. In this case $\Pr[b = b'] = \frac{1}{2}$

- When $T = e(g_0, h_0)^{k \cdot f(\gamma)}$, when the symmetric key used to encrypt $m_b$ is $sk = e(g_0^{f(\gamma)}, h_0^{k \cdot g(\gamma)}) = e(g, h)^k$, we have

$$
\begin{aligned}
&C'_1 = g_0^{-k \cdot \gamma \cdot f(\gamma)} = g^{-k}, C'_2 = h_1^{k \cdot g(\gamma)} = h_1^{k \prod_{ID \in S_{L_\beta} \cup S_{L_{\beta+1}, \cdots, S_{L_\mu}}} (\gamma + H(ID))}, \\
&C'_3 = \frac{e(g_0^{\gamma^2 \cdot f(\gamma)}, h_0^{kg(\gamma)})}{T^{\prod_{ID \in S_{L_\beta} \cup S_{L_{\beta+1}, \cdots, S_{L_\mu}}} (H(ID))} \cdot e(g_0^{k \cdot \gamma \cdot f(\gamma)}, h_0^{q(\gamma)})} \\
&= \frac{e(g, h_1)^k}{T^{\prod_{ID \in S_{L_\beta} \cup S_{L_{\beta+1}, \cdots, S_{L_\mu}}} (H(ID))} \cdot e(g_0, h_0)^{k \cdot f(\gamma) \cdot \gamma \cdot q(\gamma)}} \\
&= \frac{e(g, h_1)^k}{e(g_0, h_0)^{k \cdot f(\gamma) \cdot \prod_{ID \in S_{L_\beta} \cup S_{L_{\beta+1}, \cdots, S_{L_\mu}}} (H(ID))} \cdot e(g_0, h_0)^{k \cdot f(\gamma) \cdot \gamma \cdot q(\gamma)}} \\
&= \frac{e(g, h_1)^k}{e(g_0, h_0)^{k \cdot f(\gamma) \cdot g(\gamma)}} = \frac{e(g, h_1)^k}{e(g, h)^k}, C'_4 = SE_{sk}(m_b)
\end{aligned}
$$

  So the given challenge ciphertext $CT' = (C'_1, C'_2, C'_3, C'_4)$ is valid and can be viewed as an encryption of $m_b$ from the point of $\mathcal{A}$. In this case, we first analyze the probability of the event our simulation would not abort. Here, let $\Pr[NA]$ be the probability of such event, then $\Pr[NA] = (\frac{q_H - 1}{q_H})^{q_{kd}}$, where $q_H, q_{kd}$ is the number of queries to the $H$ oracle and the decryption key query oracle respectively. As $\mathcal{A}$ can choose the target identity with probability $\frac{1}{q_H}$ and can break our ACE with probability $\epsilon$, then $\Pr[b = b'] = \frac{1}{2} + (\frac{q_H - 1}{q_H})^{q_{kd}} \cdot \frac{1}{q_H} \cdot \epsilon$.

Form what we have analyzed, we have

$$
\begin{aligned}
\mathsf{Adv}_{\mathcal{B}}{}^{(f,g,F)-gddhe} &= \Pr[b = b' | T = e(g_0, h_0)^{k \cdot f(\gamma)}] - \Pr[b = b' | T \neq e(g_0, h_0)^{k \cdot f(\gamma)}] \\
&= \frac{1}{2} \cdot (\frac{q_H - 1}{q_H})^{q_{kd}} \cdot \frac{1}{q_H} \cdot \epsilon
\end{aligned}
$$

Here, we finish our proof. □

**Theorem 6.2** *Our ACE scheme holds the Extended No-Write Rule property assuming the $(f, g, F)-GDDHE$ problem is hard in the group system $\mathcal{BM} = (p, g_0, h_0, \mathbb{G}_1, \mathbb{G}_T, e(\cdot, \cdot))$ when the hash function $H$ is modeled as random oracle. Concretely, if*

*there is an adversary $\mathcal{A}$ which can break our scheme with non-negligible probability $\epsilon$, supposing $\mathcal{A}$ makes at most $q_H, q_{ke}$ queries to the $H$ hash oracle,encryption key query oracle respectively, then we can construct another algorithm $\mathcal{S}$ that solves the $(f, g, F)$-GDDHE problem in the given group system with advantage at least $\frac{1}{2} \cdot \frac{1}{q_H} \cdot \epsilon$, where $q_H, q_{kd}$ are defined above.*

**Proof.** Assuming the adversary $\mathcal{A}$ can break our ACE scheme with non-negligible probability $\epsilon$, we show how to construct another algorithm $\mathcal{S}$ solving the $(f, g, F)$-GDDHE problem from $\mathcal{A}$, $\mathcal{S}$ and $\mathcal{A}$ interacts as below.

- Init. In this phase, $\mathcal{A}$ first outputs the user $j$ with identity $ID_j$ as the user it wants to send messages to.

- Setup. In this phase, $\mathcal{S}$ does the same as $\mathcal{B}$ in the setup phase of the previous security proof. Eventually, $\mathcal{S}$ gives the public parameter $pp = (\mathcal{BM}, \{ID_u\}, \{SP\}, \mathcal{KM}, \mathcal{C}, \mathcal{C}')$ to $\mathcal{A}$ and keeps $y$ as the secret.

- Hash query phase. What $\mathcal{S}$ does in this phase is also identical to $\mathcal{B}$ in the hash query phase of the previous security proof, so we omit the description.

- Decryption key query phase. In this phase, $\mathcal{A}$ can only ask the decryption key of the user $j$ he wants to sends messages to. Literally, when $\mathcal{A}$ queries the decryption key of a user whose identity is not $ID_j$, $\mathcal{S}$ aborts the simulation. Otherwise, $\mathcal{S}$ answers the decryption key query the same as $\mathcal{B}$ does in the corresponding phase of the previous security proof.

- Encryption key query phase. In this phase, $\mathcal{A}$ can query as many as users' encryption keys with the only retraction that $P(i, j) = 0$ holds for each $i$ of those users. To answer such encryption key queries, $\mathcal{S}$ does identical to $\mathcal{B}$ in the corresponding phase of the last security proof.

- Sanitizing key query phase. In this phase, $\mathcal{A}$ can query the sanitizing key of multiple sanitizers. The restriction here is that some of the sanitizers queried in this phase cannot form a "sanitizing pipeline ". Because $\mathcal{S}$ knows the system sanitizing key $y$, he can answer such query using the sanitizing key generation algorithm described in the concrete construction of the ACE scheme.

- Challenge phase. Let $S_{L_{\beta\mu}}$ denote the set $S_{L_\beta} \cup S_{L_\beta+1}, \cdots, S_{L_\mu}$. In this phase, $\mathcal{A}$ chooses two message $m_0, m_1 \in \mathcal{M}$, then it sends them to $\mathcal{S}$. $\mathcal{S}$ chooses a random bit $b \in \{0, 1\}$, then it generates the sanitized ciphertext $C' =$

$(C'_1, C'_2, C'_3, C'_4)$ as;

$$
\begin{aligned}
&C'_1 = g_0^{-k\cdot\gamma\cdot f(\gamma)}, C'_2 = h_1^{k\cdot g(\gamma)},\\
&C'_3 = \frac{e(g_0^{\gamma^2\cdot f(\gamma)}, h_0^{kg(\gamma)})}{T^{\prod_{ID\in S_{L_{\beta\mu}}}(H(ID))}\cdot e(g_0^{k\cdot\gamma\cdot f(\gamma)}, h_0^{q(\gamma)})}.\\
&q(\gamma) = \frac{\prod_{ID\in S_{L_{\beta\mu}}}(\gamma + H(ID)) - \prod_{ID\in S_{L_{\beta\mu}}}(H(ID))}{\gamma}\\
&sk = e(g_0^{f(\gamma)}, h_0^{k\cdot g(\gamma)}), \qquad C'_4 = SE_{sk}(m_b)
\end{aligned}
$$

$\mathcal{S}$ then sends $(C'_1, C'_2, C'_3, C'_4)$ to $\mathcal{A}$ as the challenger ciphertext.

- Guess phase. Finally, the adversary $\mathcal{A}$ outputs a guess $b' \in \{0, 1\}$ about the encrypted message $m_b$, if $b = b'$, then $\mathcal{A}$ wins the game.

**Analysis.** Let $\Pr[b = b']$ be the probability $\mathcal{A}$ wins the above game, let $Adv_{\mathcal{C}}^{(f,g,F)-gddhe}$ denote the advantage of $\mathcal{S}$ in solving the given $(f, g, F)$-GDDHE problem instance and $\epsilon$ denote the probability $\mathcal{A}$ breaks our ACE scheme, to complete the proof, we give the following probability analysis;

- When T is a random element in $\mathbb{G}_T$, it is easy to verify that the challenge ciphertext $CT' = (C'_1, C'_2, C'_3C'_4)$ is like a one-time-pad, so $\mathcal{A}$ cannot get any useful information from $CT'$. In this case $\Pr[b = b'] = \frac{1}{2}$

- When $T = e(g_0, h_0)^{k\cdot f(\gamma)}$, when the symmetric key used to encrypt $m_b$ is $sk = e(g_0^{f(\gamma)}, h_0^{k\cdot g(\gamma)}) = e(g, h)^k$, we have

$$
\begin{aligned}
C'_1 &= g_0^{-k\cdot\gamma\cdot f(\gamma)} = g^{-k}, C'_2 = h_1^{k\cdot g(\gamma)} = h_1^{k\prod_{ID\in S_{L_\beta}\cup S_{L_{\beta+1}},\cdots,S_{L_\mu}}(\gamma+H(ID))},\\
C'_3 &= \frac{e(g_0^{\gamma^2\cdot f(\gamma)}, h_0^{kg(\gamma)})}{T^{\prod_{ID\in S_{L_\beta}\cup S_{L_{\beta+1}},\cdots,S_{L_\mu}}(H(ID))}\cdot e(g_0^{k\cdot\gamma\cdot f(\gamma)}, h_0^{q(\gamma)})}\\
&= \frac{e(g, h_1)^k}{T^{\prod_{ID\in S_{L_\beta}\cup S_{L_{\beta+1}},\cdots,S_{L_\mu}}(H(ID))}\cdot e(g_0, h_0)^{k\cdot f(\gamma)\cdot\gamma\cdot q(\gamma)}}\\
&= \frac{e(g, h_1)^k}{e(g_0, h_0)^{k\cdot f(\gamma)\cdot\prod_{ID\in S_{L_\beta}\cup S_{L_{\beta+1}},\cdots,S_{L_\mu}}(H(ID))}\cdot e(g_0, h_0)^{k\cdot f(\gamma)\cdot\gamma\cdot q(\gamma)}}\\
&= \frac{e(g, h_1)^k}{e(g_0, h_0)^{k\cdot f(\gamma)\cdot g(\gamma)}}\\
&= \frac{e(g, h_1)^k}{e(g, h)^k}\\
C'_4 &= SE_{sk}(m_b)
\end{aligned}
$$

So the given challenge ciphertext $C' = (C'_1, C'_2, C'_3, C'_4)$ is valid and can be viewed as an encryption of $m_b$ from the point of $\mathcal{A}$. In this case, we first

analyze the probability of the event our simulation would not abort. Here, let $\Pr[NA]$ be the probability of such event, then $\Pr[NA] = \frac{1}{q_H}$, where $q_H$ is the number of queries to the $H$ oracle. As $\mathcal{A}$ can break our ACE with probability $\epsilon$, then in this case $\Pr[b = b'] = \frac{1}{2} + \frac{1}{q_H} \cdot \epsilon$.

Form what we have analyzed, we have

$$\begin{aligned} \mathsf{Adv}_{\mathcal{S}}^{(f,g,F)-gddhe} &= \Pr[b = b' | T = e(g_0, h_0)^{k \cdot f(\gamma)}] - \Pr[b = b' | T \neq e(g_0, h_0)^{k \cdot f(\gamma)}] \\ &= \frac{1}{2} \cdot \frac{1}{q_H} \cdot \epsilon \end{aligned}$$

Here, we finish our proof. □

## 6.5 A Further Discussion

We propose a pure cryptographic solution in this chapter to resist cyberattacks on centralized ACE systems. In practice, giant IT companies, such as IBM and Google, would prefer to use the traditional fault tolerance solutions with replicated servers to protect their centralized systems from being compromised.

While it seems that traditional replica-based solutions are more attractive, we argue that our solution has advantages over them because of the following reasons: First, since our software solution does not involve high expenses to buy and maintain expensive servers comparing to conventional replica-based solutions, it is more cost effective. Which makes our solution more suitable for startups and small size organizations. Second, our software solution can be deployed over either ASIC (application specified integrated circuit) or FPGA (field programmable gate array)or personal computers, while a replica-based solution may have to be deployed on servers with X86 architecture. That is, the deployment of our solution enjoys high flexibility than conventional hardware-based solutions. Last, our solution doesn't require extra system level management procedures to guarantee the security of it. Namely, the security of our software solution is based on the intractability of well studied computational assumptions and thus can be quantitive analysis, while that of the replica-based solutions lie in the trustiness level of the system manager, which could be a person or a department. Which means that, when applying the replica-based solutions, there must exist extra and strict management procedures to ensure that the system manager always works honest and is highly trusted.

We should also admit that our solution has constraints comparing to the replica-based solutions. Firstly, since our solution, in fact, shifts certain amount of computational task from the server to the end nodes, the users in our system need to spend more computational power and time on generating required ciphertexts. So,

our solution imposes more burden on end users comparing to replica-based solutions. Secondly, as a software-based solution, our scheme can hardly be more efficient than the traditional hardware-based solutions with replicated servers. And when comparing the system throughput provided by the two types of solutions, our solution also cannot earn any advantage over the traditional fault tolerance solutions with replicated servers.

## 6.6 Summary

In this chapter, we present an access control encryption(ACE) with compact size ciphertext and decentralized sanitizers. Our construction is also believed to be the first one considering using multiple sanitizers rather than one. Our extended no-write rule model and the given corresponding proof show that our ACE is more secure and reliable because of the utilization of decentralized sanitizers. The security of our scheme is proven under non-standard assumptions with the help of the random oracle. Our next work focuses on presenting ACE which can be proven secure without random oracle and under standard assumptions.

# Chapter 7

# Thesis Conclusion

In this chapter, we summarize the work presented in this thesis and list some research directions for future work.

## 7.1 Conclusion

### 7.1.1 Communication Schemes with User and Data Privacy

In our first work, we formalize the notion user conditional privacy preservation and then propose a privacy preserving source-verifiable encryption scheme which maintains message confidentiality and sender conditional privacy. We also give a short discussion of constructing a server-aided variant of our scheme.

In the second work, we consider a more complex scenario where the user and data privacy should be preserved. In that scenario, a user is required to be able to prove the legitimation of the communication channel between it and its communicator without leaking their privacy. We find such scenario is realistic when there exists a authority in the system which maintains a publicly published blacklist to block communication channels between specific message senders and receivers. We present a group-based source-destination verifiable encryption scheme with blacklist checking which can address the issues in the scenario properly. Our construction utilizes the zero-knowledge proof of membership and also zero-knowledge of inequality technique.

Our two aforementioned works all give answers to the problem how to preserve the user conditional privacy. However, those two solutions have the same insufficiency. Namely, since the receiver in the proposed schemes is the only parity which can revoke the anonymity of the sender of a given ciphertext, and no one else in the system has the capability to verify whether the receiver behaves honestly during identifying the actual sender, it can manipulate the origination of one ciphertext successfully. To tackle this problem, we develop a secure communication scheme

applied between the surveillance camera and the server in the camera surveillance system. With our scheme, the server can give a proof to convince others the origination of a ciphertext without leaking its content. Such property enables the server to build a searchable database using the camera's identifier as index and also the message auditor to check the ciphertext and its origination stored in the database without any dispute.

### 7.1.2 Access Control Encryption

In our fourth work, we give the first ACE scheme construction with decentralized sanitizers. The resulted scheme keeps not only the ciphertext size but also the key size of each users compact. Our more significant contribution to the ACE is giving a decentralized implementation of the sanitizer. Unlike previous ACE schemes with only one sanitizer, our construction distributes the sanitizing functionality of the origin ACE among $n$ sanitizers. It is impossible for one of sanitizers in our construction to produce a new access policy, so our construction imposes restriction on the capability of the sanitizer. Besides, as one message sender in our ACE scheme can choose the $t$ sanitizers itself to collaboratively produce a valid sanitized ciphertext, even some of the $n$ sanitizers cannot provide service or are off-line, the whole ACE system can still work as normal. So our construction improves the reliability of the sanitizer and even the robustness of the whole ACE system.

## 7.2 Future Work

We put forward the following research directions as our future work.

1. The realization of the user conditional privacy preservation property in our first three works cannot avoid the usage of zero-knowledge proof, which makes the resulted schemes rather inefficient. We are still working on finding more efficient cryptographic tools to construct encryption schemes which preserve the user conditional privacy.

2. The access control encryption is a cryptographic realization of the classical Bell-LaPadula model. However, this primitive only realize a partial of the functionalities of that model. Besides, there still exists another useful access control model, the Chinese wall model concretely, which has never been well studied and realized using cryptographic tools. We cannot, at present, ascertain whether the whole functionalities of the Bell-LaPadula model and Chinese wall model can be implemented using cryptographic tools. So we leave it as one of our future work.

# Bibliography


[ABC+08] Michel Abdalla, Mihir Bellare, Dario Catalano, Eike Kiltz, Tadayoshi Kohno, Tanja Lange, John Malone-Lee, Gregory Neven, Pascal Paillier, and Haixia Shi. Searchable encryption revisited: Consistency properties, relation to anonymous ibe, and extensions. *J. Cryptology*, 21(3):350–391, 2008.

[ACJT00a] Giuseppe Ateniese, Jan Camenisch, Marc Joye, and Gene Tsudik. A practical and provably secure coalition-resistant group signature scheme. In *Advances in Cryptology - CRYPTO 2000, 20th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2000, Proceedings*, pages 255–270, 2000.

[ACJT00b] Giuseppe Ateniese, Jan Camenisch, Marc Joye, and Gene Tsudik. A practical and provably secure coalition-resistant group signature scheme. In *CRYPTO 2000*, pages 255–270. Springer, 2000.

[Aim11] Laila El Aimani. Generic constructions for verifiable signcryption. In *Information Security and Cryptology - ICISC 2011 - 14th International Conference, Seoul, Korea, November 30 - December 2, 2011. Revised Selected Papers*, pages 204–218, 2011.

[ALSY06] Man Ho Au, Joseph K. Liu, Willy Susilo, and Tsz Hon Yuen. Constant-size id-based linkable and revocable-iff-linked ring signature. In *Progress in Cryptology - INDOCRYPT 2006, 7th International Conference on Cryptology in India, Kolkata, India, December 11-13, 2006, Proceedings*, pages 364–378, 2006.

[ALSY13] Man Ho Au, Joseph K. Liu, Willy Susilo, and Tsz Hon Yuen. Secure id-based linkable and revocable-iff-linked ring signature with constant-size construction. *Theor. Comput. Sci.*, 469:1–14, 2013.

[AMM99] Jun Anzai, Natsume Matsuzaki, and Tsutomu Matsumoto. A quick group key distribution scheme with "entity revocation". In *Advances in Cryptology - ASIACRYPT '99, International Conference on the Theory*

*and Applications of Cryptology and Information Security, Singapore, November 14-18, 1999, Proceedings*, pages 333–347, 1999.

[Ate04] Giuseppe Ateniese. Verifiable encryption of digital signatures and applications. *ACM TISSEC*, 7(1):1–20, 2004.

[Bao00] Feng Bao. An efficient verifiable encryption scheme for encryption of discrete logarithms. In *Smart Card Research and Applications*, pages 213–220. Springer, 2000.

[BBDP01] Mihir Bellare, Alexandra Boldyreva, Anand Desai, and David Pointcheval. Key-privacy in public-key encryption. In *Advances in Cryptology - ASIACRYPT 2001, 7th International Conference on the Theory and Application of Cryptology and Information Security, Gold Coast, Australia, December 9-13, 2001, Proceedings*, pages 566–582, 2001.

[BBR99] Eli Biham, Dan Boneh, and Omer Reingold. Breaking generalized diffie–hellman modulo a composite is no easier than factoring. *Information Processing Letters*, 70(2):83 – 87, 1999.

[BBW06] Adam Barth, Dan Boneh, and Brent Waters. Privacy in encrypted content distribution using private broadcast encryption. In *International Conference on Financial Cryptography and Data Security*, pages 52–64. Springer, 2006.

[BCK96] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. Keying hash functions for message authentication. In *Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 1996, Proceedings*, pages 1–15, 1996.

[BCOP04] Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. Public key encryption with keyword search. In *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*, pages 506–522, 2004.

[BD98] Feng Bao and Robert H. Deng. A signcryption scheme with signature directly verifiable by public key. In *Public Key Cryptography, First International Workshop on Practice and Theory in Public Key Cryptography, PKC '98, Pacifico Yokohama, Japan, February 5-6, 1998, Proceedings*, pages 55–59, 1998.

[BDPR98a] Mihir Bellare, Anand Desai, David Pointcheval, and Phillip Rogaway. Relations among notions of security for public-key encryption schemes. In *CRYPTO '98*, pages 26–45. Springer, 1998.

[BDPR98b] Mihir Bellare, Anand Desai, David Pointcheval, and Phillip Rogaway. Relations among notions of security for public-key encryption schemes. In *Advances in Cryptology - CRYPTO '98, 18th Annual International Cryptology Conference, Santa Barbara, California, USA, August 23-27, 1998, Proceedings*, pages 26–45, 1998.

[BF01] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the weil pairing. In *Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings*, pages 213–229, 2001.

[BFPV11] Olivier Blazy, Georg Fuchsbauer, David Pointcheval, and Damien Vergnaud. Signatures on randomizable ciphertexts. In *PKC 2011*, pages 403–422. Springer, 2011.

[BGW05] Dan Boneh, Craig Gentry, and Brent Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. In *Advances in Cryptology - CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings*, pages 258–275, 2005.

[BH08] Dan Boneh and Michael Hamburg. Generalized identity based and broadcast encryption schemes. In *Advances in Cryptology - ASIACRYPT 2008, 14th International Conference on the Theory and Application of Cryptology and Information Security, Melbourne, Australia, December 7-11, 2008. Proceedings*, pages 455–470, 2008.

[BI92] Michael Bertilsson and Ingemar Ingemarsson. A construction of practical secret sharing schemes using linear block codes. In *Advances in Cryptology - AUSCRYPT '92, Workshop on the Theory and Application of Cryptographic Techniques, Gold Coast, Queensland, Australia, December 13-16, 1992, Proceedings*, pages 67–79, 1992.

[BL88] Josh Cohen Benaloh and Jerry Leichter. Generalized secret sharing and monotone functions. In *Advances in Cryptology - CRYPTO '88, 8th Annual International Cryptology Conference, Santa Barbara, California, USA, August 21-25, 1988, Proceedings*, pages 27–35, 1988.

[BL96] David Elliott Bell and Leonard J. LaPadula. Secure computer systems: A mathematical model, volume II. *Journal of Computer Security*, 4(2/3):229–263, 1996.

[Bon98] Dan Boneh. The decision diffie-hellman problem. In *Algorithmic Number Theory, Third International Symposium, ANTS-III, Portland, Oregon, USA, June 21-25, 1998, Proceedings*, pages 48–63, 1998.

[Bou00] Fabrice Boudot. Efficient proofs that a committed number lies in an interval. In *Advances in Cryptology - EUROCRYPT 2000, International Conference on the Theory and Application of Cryptographic Techniques, Bruges, Belgium, May 14-18, 2000, Proceeding*, pages 431–444, 2000.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *CCS '93, Proceedings of the 1st ACM Conference on Computer and Communications Security, Fairfax, Virginia, USA, November 3-5, 1993.*, pages 62–73, 1993.

[BSZ02] Joonsang Baek, Ron Steinfeld, and Yuliang Zheng. Formal proofs for the security of signcryption. In *Public Key Cryptography, 5th International Workshop on Practice and Theory in Public Key Cryptosystems, PKC 2002, Paris, France, February 12-14, 2002, Proceedings*, pages 80–98, 2002.

[BWZ14] Dan Boneh, Brent Waters, and Mark Zhandry. Low overhead broadcast encryption from multilinear maps. In *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I*, pages 206–223, 2014.

[CD00] Jan Camenisch and Ivan Damgård. Verifiable encryption, group encryption, and their applications to separable group signatures and signature sharing schemes. In *ASIACRYPT 2000*, pages 331–345. Springer, 2000.

[CDS94] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *Advances in Cryptology - CRYPTO '94, 14th Annual International Cryptology Conference, Santa Barbara, California, USA, August 21-25, 1994, Proceedings*, pages 174–187, 1994.

[CFT98] Agnes Hui Chan, Yair Frankel, and Yiannis Tsiounis. Easy come - easy go divisible cash. In *Advances in Cryptology - EUROCRYPT '98, International Conference on the Theory and Application of Cryptographic*

*Techniques, Espoo, Finland, May 31 - June 4, 1998, Proceeding*, pages 561–575, 1998.

[CGH04] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited. *J. ACM*, 51(4):557–594, 2004.

[CM99] Jan Camenisch and Markus Michels. Proving in zero-knowledge that a number is the product of two safe primes. In *Advances in Cryptology - EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2-6, 1999, Proceeding*, pages 107–122, 1999.

[CS98a] Ronald Cramer and Victor Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In *Advances in Cryptology — CRYPTO '98: 18th Annual International Cryptology Conference Santa Barbara, California, USA August 23–27, 1998 Proceedings*, pages 13–25, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg.

[CS98b] Ronald Cramer and Victor Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In *Advances in Cryptology - CRYPTO '98, 18th Annual International Cryptology Conference, Santa Barbara, California, USA, August 23-27, 1998, Proceedings*, pages 13–25, 1998.

[CS03a] Jan Camenisch and Victor Shoup. Practical verifiable encryption and decryption of discrete logarithms. In *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings*, pages 126–144, 2003.

[CS03b] Jan Camenisch and Victor Shoup. Practical verifiable encryption and decryption of discrete logarithms. In *CRYPTO 2003*, pages 126–144. Springer, 2003.

[CS03c] Ronald Cramer and Victor Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM J. Comput.*, 33(1):167–226, 2003.

[CvH91] David Chaum and Eugène van Heyst. Group signatures. In *Advances in Cryptology - EUROCRYPT '91, Workshop on the Theory and Application of of Cryptographic Techniques, Brighton, UK, April 8-11, 1991, Proceedings*, pages 257–265, 1991.

[CW79] Larry Carter and Mark N. Wegman. Universal classes of hash functions. *J. Comput. Syst. Sci.*, 18(2):143–154, 1979.

[CYHL14] Tat Wing Chim, Siu.Ming Yiu, Lucas CK Hui, and Victor OK Li. Vspn: Vanet-based secure and privacy-preserving navigation. *Computers, IEEE Transactions on*, 63(2):510–524, 2014.

[DC06] Shanshan Duan and Zhenfu Cao. Efficient and provably secure multi-receiver identity-based signcryption. In *Information Security and Privacy, 11th Australasian Conference, ACISP 2006, Melbourne, Australia, July 3-5, 2006, Proceedings*, pages 195–206, 2006.

[DDN91] Danny Dolev, Cynthia Dwork, and Moni Naor. Non-malleable cryptography (extended abstract). In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing, May 5-8, 1991, New Orleans, Louisiana, USA*, pages 542–552, 1991.

[Del07] Cécile Delerablée. Identity-based broadcast encryption with constant size ciphertexts and private keys. In *Advances in Cryptology - ASIACRYPT 2007, 13th International Conference on the Theory and Application of Cryptology and Information Security, Kuching, Malaysia, December 2-6, 2007, Proceedings*, pages 200–215, 2007.

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

[DHO16] Ivan Damgård, Helene Haagh, and Claudio Orlandi. Access control encryption: Enforcing information flow with cryptography. In *Theory of Cryptography - 14th International Conference, TCC 2016-B, Beijing, China, October 31 - November 3, 2016, Proceedings, Part II*, pages 547–576, 2016.

[DKNS04] Yevgeniy Dodis, Aggelos Kiayias, Antonio Nicolosi, and Victor Shoup. Anonymous identification in ad hoc groups. In *EUROCRYPT 2004*, pages 609–626. Springer, 2004.

[DPP07] Cécile Delerablée, Pascal Paillier, and David Pointcheval. Fully collusion secure dynamic broadcast encryption with constant-size ciphertexts or decryption keys. In *Pairing-Based Cryptography - Pairing 2007, First International Conference, Tokyo, Japan, July 2-4, 2007, Proceedings*, pages 39–59, 2007.

[ELO13] Hasoo Eun, Hoonjung Lee, and Heekuck Oh. Conditional privacy preserving security protocol for NFC applications. *IEEE Trans. Consumer Electronics*, 59(1):153–160, 2013.

[EZ15] Graham Enos and Yuliang Zheng. An id-based signcryption scheme with compartmented secret sharing for unsigncryption. *Inf. Process. Lett.*, 115(2):128–133, 2015.

[FFS88] Uriel Feige, Amos Fiat, and Adi Shamir. Zero-knowledge proofs of identity. *J. Cryptology*, 1(2):77–94, 1988.

[FGKO17] Georg Fuchsbauer, Romain Gay, Lucas Kowalczyk, and Claudio Orlandi. Access control encryption for equality, comparison, and more. In *Public-Key Cryptography - PKC 2017 - 20th IACR International Conference on Practice and Theory in Public-Key Cryptography, Amsterdam, The Netherlands, March 28-31, 2017, Proceedings, Part II*, pages 88–118, 2017.

[FN93] Amos Fiat and Moni Naor. Broadcast encryption. In *Advances in Cryptology - CRYPTO '93, 13th Annual International Cryptology Conference, Santa Barbara, California, USA, August 22-26, 1993, Proceedings*, pages 480–491, 1993.

[FO97] Eiichiro Fujisaki and Tatsuaki Okamoto. Statistical zero knowledge protocols to prove modular polynomial relations. In *Advances in Cryptology - CRYPTO '97, 17th Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 1997, Proceedings*, pages 16–30, 1997.

[FOPS01] Eiichiro Fujisaki, Tatsuaki Okamoto, David Pointcheval, and Jacques Stern. RSA-OAEP is secure under the RSA assumption. In *Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings*, pages 260–274, 2001.

[FP12] Nelly Fazio and Irippuge Milinda Perera. Outsider-anonymous broadcast encryption with sublinear ciphertexts. In *Public Key Cryptography - PKC 2012 - 15th International Conference on Practice and Theory in Public Key Cryptography, Darmstadt, Germany, May 21-23, 2012. Proceedings*, pages 225–242, 2012.

[FS86] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Advances in Cryptology*

*- CRYPTO '86, Santa Barbara, California, USA, 1986, Proceedings*, pages 186–194, 1986.

[FS07] Eiichiro Fujisaki and Koutarou Suzuki. Traceable ring signature. In *Public Key Cryptography - PKC 2007, 10th International Conference on Practice and Theory in Public-Key Cryptography, Beijing, China, April 16-20, 2007, Proceedings*, pages 181–200, 2007.

[Fuc10] Georg Fuchsbauer. Commuting signatures and verifiable encryption and an application to non-interactively delegatable credentials. *IACR Cryptology ePrint Archive*, 2010:233, 2010.

[Gam85] Taher El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Trans. Information Theory*, 31(4):469–472, 1985.

[GDM02] Nicolás González-Deleito and Olivier Markowitch. An optimistic multi-party fair exchange protocol with reduced trust requirements. In *ICISC 2001*, pages 258–267. Springer, 2002.

[GK15] Jens Groth and Markulf Kohlweiss. One-out-of-many proofs: Or how to leak a secret and spend a coin. *EUROCRYPT 2015*, pages 253–280, 2015.

[GLZ99] Chandana Gamage, Jussipekka Leiwo, and Yuliang Zheng. Encrypted message authentication by firewalls. In *Public Key Cryptography, Second International Workshop on Practice and Theory in Public Key Cryptography, PKC '99, Kamakura, Japan, March 1-3, 1999, Proceedings*, pages 69–81, 1999.

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *J. Comput. Syst. Sci.*, 28(2):270–299, 1984.

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

[GMW86] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity and a methodology of cryptographic protocol design (extended abstract). In *27th Annual Symposium on Foundations of Computer Science, Toronto, Canada, 27-29 October 1986*, pages 174–187, 1986.

[Gol03] Oded Goldreich. *Foundations of cryptography: volume 1, basic tools*. Cambridge university press, 2003.

[GSY99] Eli Gafni, Jessica Staddon, and Yiqun Lisa Yin. Efficient methods for integrating traceability and broadcast encryption. In *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, pages 372–387, 1999.

[GW09] Craig Gentry and Brent Waters. Adaptive security in broadcast encryption systems (with short ciphertexts). In *Advances in Cryptology - EUROCRYPT 2009, 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26-30, 2009. Proceedings*, pages 171–188, 2009.

[HBCC13] Daojing He, Jiajun Bu, Sammy Chan, and Chun Chen. Handauth: Efficient handover authentication with conditional privacy for wireless networks. *IEEE Trans. Computers*, 62(3):616–622, 2013.

[HCW13] Lein Harn, Chin-Chen Chang, and Hsiao-Ling Wu. An anonymous multi-receiver encryption based on RSA. *I. J. Network Security*, 15(4):307–312, 2013.

[HM12] Jan Hajny and Lukas Malina. Practical revocable anonymous credentials. In *Communications and Multimedia Security*, pages 211–213. Springer, 2012.

[HP10] Gertjan P Halkes and Johan A Pouwelse. Verifiable encryption for p2p block exchange. In *Peer-to-Peer Computing (P2P), IEEE Tenth International Conference on*, pages 1–4. IEEE, 2010.

[HRS14] Javier Herranz, Alexandre Ruiz, and Germán Sáez. Signcryption schemes with threshold unsigncryption, and applications. *Des. Codes Cryptography*, 70(3):323–345, 2014.

[HSMZ05a] Xinyi Huang, Willy Susilo, Yi Mu, and Futai Zhang. Identity-based ring signcryption schemes: cryptographic primitives for preserving privacy and authenticity in the ubiquitous world. In *Advanced Information Networking and Applications, 2005. AINA 2005. 19th International Conference on*, volume 2, pages 649–654, 2005.

[HSMZ05b] Xinyi Huang, Willy Susilo, Yi Mu, and Futai Zhang. Identity-based ring signcryption schemes: Cryptographic primitives for preserving privacy

and authenticity in the ubiquitous world. In *19th International Conference on Advanced Information Networking and Applications (AINA 2005), 28-30 March 2005, Taipei, Taiwan*, pages 649–654, 2005.

[JM97] Camenisch Jan and Stadler Markus. Proof systems for general statements about discrete logarithms. *Technical report, Dept. of Computer Science, ETH Zürich*, 260, 1997.

[KBK+11] Emile J. C. Kelkboom, Jeroen Breebaart, Tom A. M. Kevenaar, Ileana Buhan, and Raymond N. J. Veldhuis. Preventing the decodability attack based cross-matching in a fuzzy commitment scheme. *IEEE Trans. Information Forensics and Security*, 6(1):107–121, 2011.

[KPW97] Seungjoo Kim, Sangjoon Park, and Dongho Won. Group signatures for hierarchical multigroups. In *Information Security, First International Workshop*, pages 273–281. Springer, 1997.

[KSAS15] Jongkil Kim, Willy Susilo, Man Ho Au, and Jennifer Seberry. Adaptively secure identity-based broadcast encryption with a constant-sized ciphertext. *IEEE Trans. Information Forensics and Security*, 10(3):679–693, 2015.

[LASZ14] Joseph K Liu, Man Ho Au, Willy Susilo, and Jianying Zhou. Linkable ring signature with unconditional anonymity. *IEEE Transactions on Knowledge and Data Engineering*, 26(1):157–165, 2014.

[LHZM10] Zhenhua Liu, Yupu Hu, Xiangsong Zhang, and Hua Ma. Certificateless signcryption scheme in the standard model. *Inf. Sci.*, 180(3):452–464, 2010.

[LK14] Yehuda Lindell and Jonathan Katz. *Introduction to modern cryptography*. Chapman and Hall/CRC, 2014.

[LLM+07] Dennis Y. W. Liu, Joseph K. Liu, Yi Mu, Willy Susilo, and Duncan S. Wong. Revocable ring signature. *J. Comput. Sci. Technol.*, 22(6):785–794, 2007.

[LLZ+08] Rongxing Lu, Xiaodong Lin, Haojin Zhu, Pin-Han Ho, and Xuemin Shen. ECPP: efficient conditional privacy preservation protocol for secure vehicular communications. In *INFOCOM 2008. 27th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies, 13-18 April 2008, Phoenix, AZ, USA*, pages 1229–1237, 2008.

[LMG+17] Jianchang Lai, Yi Mu, Fuchun Guo, Willy Susilo, and Rongmao Chen. Fully privacy-preserving and revocable id-based broadcast encryption for data access control in smart city. *Personal and Ubiquitous Computing*, 21(5):855–868, 2017.

[LPQ12] Benoît Libert, Kenneth G. Paterson, and Elizabeth A. Quaglia. Anonymous broadcast encryption: Adaptive security and efficient constructions in the standard model. In *Public Key Cryptography - PKC 2012 - 15th International Conference on Practice and Theory in Public Key Cryptography, Darmstadt, Germany, May 21-23, 2012. Proceedings*, pages 206–224, 2012.

[LQ04] Benoît Libert and Jean-Jacques Quisquater. Efficient signcryption with key privacy from gap diffie-hellman groups. In *Public Key Cryptography - PKC 2004, 7th International Workshop on Theory and Practice in Public Key Cryptography, Singapore, March 1-4, 2004*, pages 187–200, 2004.

[LST08] Fagen Li, Masaaki Shirase, and Tsuyoshi Takagi. Analysis and improvement of authenticatable ring signcryption scheme. *IACR Cryptology ePrint Archive*, 2008:373, 2008.

[LYW+07] Chung Ki Li, Guomin Yang, Duncan S. Wong, Xiaotie Deng, and Sherman S. M. Chow. An efficient signcryption scheme with key privacy. In *Public Key Infrastructure, 4th European PKI Workshop: Theory and Practice, EuroPKI 2007, Palma de Mallorca, Spain, June 28-30, 2007, Proceedings*, pages 78–93, 2007.

[LYW+10] Chung Ki Li, Guomin Yang, Duncan S. Wong, Xiaotie Deng, and Sherman S. M. Chow. An efficient signcryption scheme with key privacy and its extension to ring signcryption. *Journal of Computer Security*, 18(3):451–473, 2010.

[Ma06] Changshe Ma. Efficient short signcryption scheme with public verifiability. In *Information Security and Cryptology, Second SKLOIS Conference, Inscrypt 2006, Beijing, China, November 29 - December 1, 2006, Proceedings*, pages 118–129, 2006.

[Mao97] Wenbo Mao. Publicly verifiable partial key escrow. *Information and Communications Security*, pages 409–413, 1997.

[Mao03] Wenbo Mao. *Modern cryptography: theory and practice*. Pearson Education India, 2003.

[Mau94] Ueli M. Maurer. Towards the equivalence of breaking the diffie-hellman protocol and computing discrete algorithms. In *Advances in Cryptology - CRYPTO '94, 14th Annual International Cryptology Conference, Santa Barbara, California, USA, August 21-25, 1994, Proceedings*, pages 271–281, 1994.

[MM03] John Malone-Lee and Wenbo Mao. Two birds one stone: Signcryption using RSA. In *Topics in Cryptology - CT-RSA 2003, The Cryptographers' Track at the RSA Conference 2003, San Francisco, CA, USA, April 13-17, 2003, Proceedings*, pages 211–225, 2003.

[MSK02] Shigeo Mitsunari, Ryuichi Sakai, and Masao Kasahara. A new traitor tracing. *IEICE transactions on fundamentals of electronics, communications and computer sciences*, 85(2):481–484, 2002.

[Odl84] Andrew M. Odlyzko. Discrete logarithms in finite fields and their cryptographic significance. In *Advances in Cryptology: Proceedings of EUROCRYPT 84, A Workshop on the Theory and Application of of Cryptographic Techniques, Paris, France, April 9-11, 1984, Proceedings*, pages 224–314, 1984.

[PCS03] Jung-Min Park, Edwin K. P. Chong, and Howard Jay Siegel. Constructing fair-exchange protocols for e-commerce via distributed computation of RSA signatures. In *Proceedings of the Twenty-Second ACM Symposium on Principles of Distributed Computing*, pages 172–181, 2003.

[PPSS13] Duong Hieu Phan, David Pointcheval, Siamak Fayyaz Shahandashti, and Mario Strefler. Adaptive CCA broadcast encryption with constant-size secret keys and ciphertexts. *Int. J. Inf. Sec.*, 12(4):251–265, 2013.

[PS96] David Pointcheval and Jacques Stern. Security proofs for signature schemes. In *Advances in Cryptology - EUROCRYPT '96, International Conference on the Theory and Application of Cryptographic Techniques, Saragossa, Spain, May 12-16, 1996, Proceeding*, pages 387–398, 1996.

[PS00] David Pointcheval and Jacques Stern. Security arguments for digital signatures and blind signatures. *J. Cryptology*, 13(3):361–396, 2000.

[RH07] Maxim Raya and Jean-Pierre Hubaux. Securing vehicular ad hoc networks. *Journal of Computer Security*, 15(1):39–68, 2007.

[RLL09] Jian Ren, Yun Li, and Tongtong Li. Providing source privacy in mobile ad hoc networks. In *Mobile Adhoc and Sensor Systems, 2009. MASS'09. IEEE 6th International Conference on*, pages 332–341. IEEE, 2009.

[RS91] Charles Rackoff and Daniel R. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In *Advances in Cryptology - CRYPTO '91, 11th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1991, Proceedings*, pages 433–444, 1991.

[RSA83] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems (reprint). *Commun. ACM*, 26(1):96–99, 1983.

[RST01] Ronald L. Rivest, Adi Shamir, and Yael Tauman. How to leak a secret. In *Advances in Cryptology - ASIACRYPT 2001, 7th International Conference on the Theory and Application of Cryptology and Information Security, Gold Coast, Australia, December 9-13, 2001, Proceedings*, pages 552–565, 2001.

[SCG+16] Willy Susilo, Rongmao Chen, Fuchun Guo, Guomin Yang, Yi Mu, and Yang-Wai Chow. Recipient revocable identity-based broadcast encryption: How to revoke some recipients in IBBE without knowledge of the plaintext. In *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security, AsiaCCS 2016, Xi'an, China, May 30 - June 3, 2016*, pages 201–210, 2016.

[Sch89] Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*, pages 239–252, 1989.

[Sha79] Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979.

[Sha84] Adi Shamir. Identity-based cryptosystems and signature schemes. In *Advances in Cryptology, Proceedings of CRYPTO '84, Santa Barbara, California, USA, August 19-22, 1984, Proceedings*, pages 47–53, 1984.

[Sta96] Markus Stadler. Publicly verifiable secret sharing. In *EUROCRYPT '96*, pages 190–199. Springer, 1996.

[SVR10] S. Sharmila Deva Selvi, S. Sree Vivek, and C. Pandu Rangan. Identity based public verifiable signcryption scheme. In *Provable Security - 4th International Conference, ProvSec 2010, Malacca, Malaysia, October 13-15, 2010. Proceedings*, pages 244–260, 2010.

[Tas11] Tamir Tassa. Generalized oblivious transfer by secret sharing. *Des. Codes Cryptography*, 58(1):11–21, 2011.

[TV09] Stephen R. Tate and Roopa Vishwanathan. Improving cut-and-choose in verifiable encryption and fair exchange protocols using trusted computing technology. In *Data and Applications Security XXIII, 23rd Annual IFIP WG 11.3 Working Conference*, pages 252–267. Springer, 2009.

[TZMT17] Gaosheng Tan, Rui Zhang, Hui Ma, and Yang Tao. Access control encryption based on LWE. In *Proceedings of the 4th ACM International Workshop on ASIA Public-Key Cryptography, APKC@AsiaCCS 2017, Abu Dhabi, United Arab Emirates, April 2, 2017*, pages 43–50, 2017.

[YM18] Zhongyuan Yao and Yi Mu. Ace with compact ciphertext size and decentralized sanitizers. *International Journal of Foundations of Computer Science*, 2018.

[YM19] Zhongyuan Yao and Yi Mu. Publicly verifiable secure communication with user and data privacy in public surveillance system. *Personal and Ubiquitous Computing*, 2019.

[YMY17] Zhongyuan Yao, Yi Mu, and Guomin Yang. Group-based source-destination verifiable encryption with blacklist checking. In *Information Security Practice and Experience - 13th International Conference, ISPEC 2017, Melbourne, VIC, Australia, December 13-15, 2017, Proceedings*, pages 186–203, 2017.

[ZCZ16] Y. Zhang, Q. Chen, and S. Zhong. Privacy-preserving data aggregation in mobile phone sensing. *IEEE Transactions on Information Forensics and Security*, 11(5):980–992, 2016.

[Zha08] Justin Zhan. Privacy-preserving collaborative data mining. *Computational Intelligence Magazine, IEEE*, 3(2):31–41, 2008.

[Zhe97] Yuliang Zheng. Digital signcryption or how to achieve cost(signature & encryption) << cost(signature) + cost(encryption). In *Advances in Cryptology - CRYPTO '97, 17th Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 1997, Proceedings*, pages 165–179, 1997.

[ZM15] Jianhong Zhang and Jian Mao. An improved anonymous multi-receiver identity-based encryption scheme. *Int. J. Communication Systems*, 28(4):645–658, 2015.

[ZWM13] Leyou Zhang, Qing Wu, and Yi Mu. Anonymous identity-based broadcast encryption with adaptive security. In *Cyberspace Safety and Security - 5th International Symposium, CSS 2013, Zhangjiajie, China, November 13-15, 2013, Proceedings*, pages 258–271, 2013.

[ZYZZ08] Mingwu Zhang, Bo Yang, Shenglin Zhu, and Wenzheng Zhang. Efficient secret authenticatable anonymous signcryption scheme with identity privacy. In *Intelligence and Security Informatics, IEEE ISI 2008 International Workshops: PAISI, PACCF, and SOCO 2008, Taipei, Taiwan, June 17, 2008. Proceedings*, pages 126–137, 2008.