University of Wollongong

Research Online

University of Wollongong Thesis Collection 1954-2016

University of Wollongong Thesis Collections

2015

Preservation of privacy in public-key cryptography

Hui Cui
University of Wollongong

Follow this and additional works at: https://ro.uow.edu.au/theses

University of Wollongong

Copyright Warning

You may print or download ONE copy of this document for the purpose of your own research or study. The University does not authorise you to copy, communicate or otherwise make available electronically to any other person any copyright material contained on this site.

You are reminded of the following: This work is copyright. Apart from any use permitted under the Copyright Act 1968, no part of this work may be reproduced by any process, nor may any other exclusive right be exercised, without the permission of the author. Copyright owners are entitled to take legal action against persons who infringe their copyright. A reproduction of material that is protected by copyright may be a copyright infringement. A court may impose penalties and award damages in relation to offences and infringements relating to copyright material. Higher penalties may apply, and higher damages may be awarded, for offences and infringements involving the conversion of material into digital or electronic form.

Unless otherwise indicated, the views expressed in this thesis are those of the author and do not necessarily represent the views of the University of Wollongong.

Recommended Citation

Cui, Hui, Preservation of privacy in public-key cryptography, Doctor of Philosophy thesis, School of Computer Science and Software Engineering, University of Wollongong, 2015. https://ro.uow.edu.au/theses/4480

Research Online is the open access institutional repository for the University of Wollongong. For further information contact the UOW Library: research-pubs@uow.edu.au

UNIVERSITY OF WOLLONGONG

COPYRIGHT WARNING

You may print or download ONE copy of this document for the purpose of your own research or study. The University does not authorise you to copy, communicate or otherwise make available electronically to any other person any copyright material contained on this site. You are reminded of the following:

Copyright owners are entitled to take legal action against persons who infringe their copyright. A reproduction of material that is protected by copyright may be a copyright infringement. A court may impose penalties and award damages in relation to offences and infringements relating to copyright material. Higher penalties may apply, and higher damages may be awarded, for offences and infringements involving the conversion of material into digital or electronic form.

Preservation of Privacy in Public-Key Cryptography

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

Doctor of Philosophy

from

THE UNIVERSITY OF WOLLONGONG

by

Hui Cui
School of Computer Science and Software Engineering
October 2015

© Copyright 2015

by

Hui Cui

All Rights Reserved

Dedicated to

My family

Declaration

This is to certify that the work reported in this thesis was done by the author, unless specified otherwise, and that no part of it has been submitted in a thesis to any other university or similar institution.

Hui Cui
October 19, 2015

Abstract

An important part of cryptography is about constructing and analyzing protocols that overcome the attacks of adversaries related to various features in information security such as confidentiality, data integrity and authentication. In this thesis, we focus on confidentiality and data integrity, especially confidentiality, under several practical security scenarios. Confidentiality refers to the privacy of encrypted data, but with increasing concern for the protection of personal information, this notion has been extended to user privacy, which is referred to as anonymity in this thesis. Data integrity refers to maintaining and assuring the accuracy and consistency of data over its entire life-cycle, which is crucial to the design, implementation and usage of any system which stores, processes, or retrieves data.

The work in this thesis can be divided into two aspects:

Firstly, we consider confidentiality and anonymity under a scenario where an untrusted third party is involved in a cryptographic system to check all the messages received. According to different settings of the application, we construct three encryption schemes with a gateway involving in verification. In more detail, our first construction to achieve confidentiality and anonymity for an encryption scheme with a gateway is simply designed, and it has a limited and strict environment requirement for the application; our second construction strengthens the security one step further but still has drawbacks in terms of anonymity; our third construction has perfect security properties which fully hides the identity of the targeted user in the system.

Then we study confidentiality, anonymity and integrity under a special attack called related-key attack, where an adversary can induce modifications in a hardware stored key and subsequently observe the output of the cryptographic primitive under this modified key. Related-key attacks are viewed merely as a way to study the security of blockciphers, but they emerge as real threats in practice and are of interest for primitives beyond blockciphers. In this thesis, the security of cryptographic

primitives such as signcryption, public-key encryption and proof of retrievability under related-key attacks is explored. We start from signcryption schemes secure against related-key attacks, and propose three signcryption schemes that are resistant to linear related-key attacks following the line of security enhancement. Then we focus on related-key attack secure public-key encryption schemes from a point of view different from the existing work related to this problem, and we propose three public-key encryption schemes with private keys composed of multiple elements that are secure against related-key attacks. Lastly, we pay attention to data integrity in the setting of related-key attacks, and show how data integrity can be broken in a proof of retrievability system by a related-key attack adversary and how to prevent such attacks.

Acknowledgement

I would like to express my sincere appreciation to my supervisor Professor Yi Mu, for his kindness and patience. Without him I would not have had a chance of gaining a scholarship to study in the University of Wollongong. During my research, he provided many suggestions to help me deal with the difficulties in the study, and gave me encouragement when I was frustrated in my research. This thesis could not have been done without his guidance.

I am deeply grateful to my co-supervisor Doctor Man Ho (Allen) Au, for his stringency, perseverance and preciseness. He gave his support for my learning experience in various ways. He not only taught me the academic knowledge, but also the methodologies of doing research. It is from him that I learned the importance of rigorous attitude and enthusiasm when doing research. It would not have been possible for me to finish this thesis without his supervision.

I want to express my heartfelt gratitude to my friends: Kel Magrath, Bob Colvin, Jacqueline Adriaanse, Sally Purdy, Andrea Kornhoff, Phuong Nguyen, Nurul Syazana, Kiara Stapleton and Thanh Nguyen, for the opportunity to be a committee member of the Illawarra Committee for International Students (ICIS) and all the good times with; Colin Hadfield, Helen Hadfield, Rodney Woollard and Christine Woollard, for lifts to the church activities and the help in my daily life; Xin Wang, Xiyan Li and Jie Liu, for being supportive flatmates; Xiaoming Zheng, Yueyue Jiang, Ying Guo and Guoseng Huang, for the friendship in my first months in Australia; Wei Gao, Yafu Ji, Rongmao Chen, Fuchun Guo, Jinguang Han, Nan Li, Ibrahim Elashry, Shams Quzi, Guomin Yang, Willy Susilo and Yong Yu, for the advices on my studies.

I wish to take this opportunity to express my thanks to Professor Tianjie Cao, who was my supervisor when I was doing a master degree at the China University of Mining and Technology. It was he who introduced me to the field of cryptography and inspired me to further study.

I would like to thank my family for their unconditional support, both financially and emotionally throughout my study. In particular, the understanding shown by my parents, grandparents and husband is greatly appreciated.

Publications

I published the following papers related to this thesis.

1. Hui Cui, Yi Mu, Man Ho Au: Anonymous Signcryption Against Linear Related-Key Attacks. ProvSec 2013: 165-183.
2. Hui Cui, Yi Mu, Man Ho Au: Public-Key Encryption Resilient to Linear Related-Key Attacks. SecureComm 2013: 182-196.
3. Hui Cui, Yi Mu, Man Ho Au: Verifiable and Anonymous Encryption in Asymmetric Bilinear Maps. TrustCom/ISPA/IUCC 2013: 704-711.
4. Hui Cui, Yi Mu, Man Ho Au: Public-Key Encryption Resilient Against Linear Related-Key Attacks Revisited. TrustCom 2014: 268-275.
5. Hui Cui, Yi Mu and Man Ho Au: Signcryption Secure Against Linear Related-key Attacks. The Computer Journal 57(10): 1472-1483 (2014).
6. Hui Cui, Yi Mu, Man Ho Au: Anonymous Broadcast Encryption with an Untrusted Gateway. International Journal of Security and Networks 9(1): 20-29 (2014).
7. Hui Cui, Yi Mu and Man Ho Au: Proof of Retrievability with Public Verifiability Resilient Against Related-Key Attacks. IET Information Security 9(1): 43-49 (2015).

Publications listed below are beyond the scope of this thesis.

1. Hui Cui, Yi Mu, Man Ho Au: Complete Robustness in Identity-Based Encryption. ProvSec 2014: 342-349.
2. Hui Cui, Yi Mu, Fuchun Guo: Server-Aided Identity-Based Anonymous Broadcast Encryption. International Journal of Security and Networks 8(1): 29-39 (2013).

3. Hui Cui, Tianjie Cao: A Secure Anonymous Identity-Based Key Agreement Protocol for Distributed Computer Networks. Journal of Networks 6(9): 1337-1343 (2011).

4. Tianjie Cao, Shi Huang, Hui Cui, Yipeng Wu, Qihan Luo: Controlled Secret Leakage. Journal of Computers 4(1): 61-68 (2009).

5. Hui Cui, Tianjie Cao: A Novel Anonymous and Authenticated Key Exchange Protocol. Journal of Networks 4(10): 985-992 (2009).

6. Tianjie Cao, Hui Cui: Digital Signature Techniques. China Cryptography Development Report 2008: 32-74 (2008).

7. Tianjie Cao, Shi Huang, Hui Cui, Yipeng Wu, Qihan Luo: Controlled Secret Leakage. CIS 2007: 868-872.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Cryptography is the practice and study of techniques for secure communication in the presence of third parties, called adversaries, and is originally synonymous with encryption — a process of converting a message into an incomprehensible format. Traditionally, the security requirement of an encryption scheme is to provide privacy (confidentiality) of the encrypted data. Nevertheless, privacy of data is not the only goal in cryptography with the encryption schemes. So far, the field has been expanded to various aspects of information security such as integrity, authentication, and non-repudiation. Integrity protects a message from unauthorized modification. Authentication assures the originality of a message from the anticipated sender. Non-repudiation prevents a sender from denying that it has sent a message.

1.1 Background

The fundamental security of an encryption scheme requires it to provide privacy for the encrypted data. To meet various kinds of data privacy requirements, popular formalizations such as indistinguishability (semantic security) [GM84] or non-malleability [DDN91], under either chosen plaintext or various kinds of chosen ciphertext attacks are directed. However, privacy of data is not the unique goal in cryptography with encryption schemes. In recent years, the privacy of users has been of equal concern, which leads the research community to pursue anonymity properties in designing cryptographic primitives. With regard to encryption, key privacy [BBDP01], also known as receiver anonymity, was first introduced in public-key encryption to prevent a ciphertext leaking any information about the public key under which it was created, and then it was extended to the identity-based setting under the notion of anonymity [ABC+05].

Traditionally, data privacy and key privacy are discussed under a two-party protocol involving a sender and a receiver. With the development of modern technology, third parties are widely used in many cryptographic primitives, and have become a crucial part of preserving information security in the digital world. A natural question which arises is how to fulfil security protection if third parties are involved in cryptographic systems. In some of the mechanisms such as private key generators in identity-based environments [BF01], they are fully trusted. However, this assumption, in practice, is too strong, as it is hard to make sure that a third party is fully honest. An alternative way to get around this dilemma is to trust a third party to some degree rather than to fully trust it. This weaker but practical assumption has been popularly studied in many cryptographic protocols like server-aided systems [MKI88], certificate-based mechanisms [Gen03], and so on. In this thesis, we discuss privacy preserving under a special scenario which involves a third party, called gateway, to check the identity of the user in the encryption system.

On the other hand, the adversaries in the security notions under the classical security models are assumed to have no access to the private keys. However, this is impractical in real systems, as in many situations the adversary might get some partial information about the keys through methods which are not anticipated by the designer of the system and, correspondingly, not taken into account when arguing for its security. Besides, it has been proved that the adversary can make physical side-channels leak partial information about internal states of program executions through recent timing [Koc96], cold-boot [HSH+08] and virtual-machine attacks [RTSS09], or use fault injection techniques to tamper with and induce modifications to the internal state of the device [BS97]. Such attacks, referred to as key-leakage attacks, come in a large variety. This observation brings a question of how to protect privacy and achieve security in cryptography under complex conditions. To handle this problem, various security notions including leakage resistant cryptography [DP08], semantic security for wiretap channels [BTV12], cryptography secure against related-key attacks and tampering [BCM11] are proposed. In this thesis, we study the security of several cryptographic primitives under one intensively investigated attack of this kind called related-key attack (RKA) [BC10], where an adversary induces modifications in a hardware-stored key, and subsequently observes the output of the cryptographic primitive under this modified key.

In terms of the related-key attack, we can see that in its security model, modifications to the key used are denoted by a function chosen by the adversary [BC10].

In other words, there exists a function denoting the relations between a related key and the original key. This reminds us of the fact that some users like to update their keys regularly according to a predefined function. As a result of this preference, a question is raised: will the security of cryptographic systems still be intact if the relations of users' keys are detected by the adversary? To address this problem, we take outsourced storage as an instance, where clients store their important data to remote third parties without leaving a copy in their local computers. Since these clients do not own data themselves, it is crucial for them to make sure that their data are not lost or corrupted. In order to reduce the waste of communication bandwidth caused by downloading large amounts of data just to check data integrity, various efficient checking protocols, called proof of retrievability, have been proposed to allow data integrity to be checked without completely downloading the data. In this thesis, we shed light on the fragility of these systems, and explore the techniques that are able to protect them from related-key attacks.

1.2 Scope of This Study

In this thesis, we focus on solving the aforementioned problems in the cryptographic world. Specifically, we deal with them from the following aspects.

1. Achieving security in encryption systems involving third parties. As far as we know, third parties involved cryptographic systems have been discussed in the context of signature [MKI88], public-key encryption [Gen03], authenticated key exchange protocols [ACFP05], and so on. To address a practical scenario where a gateway is present in checking the identity of the receiver when a sender communicates with receivers in an anonymous encryption system, we put forward the notion of verifiable and anonymous encryption. A gateway-based verifiable anonymous encryption is a three-party protocol to enable an outside sender to securely and anonymously transmit a ciphertext to an inside receiver who belongs to a large group of users when taking into account the presence of a gateway which is not fully trusted. In addition to the semantic security of the plaintext, our goal is also to provide anonymity with respect to a malicious gateway and curious receivers in the system. With this in mind, in this thesis, we strengthen the security model step by step, and design the concrete gateway-based anonymous encryption schemes to meet the

corresponding security requirements.

2. Signcryption secure against related-key attacks. To combine the functions of digital signature and encryption in a single step with a cost lower than that required by signature-then-encryption approach, signcryption [Zhe97] is introduced, of which security notions such as semantic security, unforgeability, and anonymity have been formalized under classical security model in [BSZ07, LQ04]. In this thesis, we take these security definitions one step further in the context of related-key attacks. Specifically, under related-key attacks, the adversary in the security games is allowed to obtain the outcome of the signcryption scheme under the modified private keys of both sides — the sender and the receiver. After formally defining the corresponding security notions of signcryption secure against related-key attacks, we propose several concrete RKA secure signcryption schemes with regard to different security settings.

3. Related-key attack secure public-key encryption schemes. How to build public-key encryption schemes secure against related-key attacks has been well studied in [Wee12, BPT12]. Wee [Wee12] presented the first public-key encryption schemes resistant to linear related-key attacks in the standard model from adaptive trapdoor relations. Bellare, Paterson and Thomson [BPT12] constructed the RKA secure schemes for public-key encryption in the standard model, based on the results of a framework to enable the construction of identity-based encryption schemes that are secure under related-key attacks. The methods to achieve RKA security in public-key encryption in [Wee12] and [BPT12] are efficient and easy to perform, but we found that their schemes of public-key encryption for full RKA security are executed under the situation of the private key being a single element. Due to this observation, in this dissertation, we focus on exploring other techniques different from those in [Wee12, BPT12] to make the public-key encryption schemes with multi-element private keys be resistant to related-key attacks.

4. New attacks to publicly verifiable proof of retrievability. Storing data to a third party has become a trend, and there are increasing numbers of clients who store their important data to a remote server. As users in this case no longer physically possess the storage of their data, and the remote storage provider may not be trusted, traditional cryptographic primitives for the purpose of

data privacy protection cannot be directly adopted. It is desirable to design a protocol by which users can verify the integrity of the data stored at the remote servers without downloading all the data. A scheme that accomplishes this is called a proof of retrievability (PoR) protocol, which was firstly introduced in [JJ07]. In this study, inspired by the phenomenon that users may set a function to update their keys at certain intervals, we strengthen the security of proof of retrievability schemes by divulging the information of the function to the adversary in the security model.

1.3 Structure of This Thesis

In Chapter 1, the background of the study and an introduction that describes the research scope of this dissertation are briefly presented.

In Chapter 2, we review the basic cryptographic notions, mathematical tools and complexity assumptions that are associated with this research.

Chapter 3 is mainly about the construction of secure protocols under a scenario called gateway-based verifiable and anonymous encryption. Firstly, we describe the framework and security model of gateway-based verifiable and anonymous encryption in a broadcast setting, and provide an efficient construction according to the formal framework of gateway-based verifiable and anonymous broadcast encryption, as well as its security proof based on the complexity assumptions. Secondly, we define the algorithms and security requirements of a verifiable and anonymous encryption system with an untrusted gateway, detail one verifiable and anonymous encryption scheme in asymmetric bilinear maps, and prove its confidentiality and anonymity under the random oracle model. Thirdly, after pointing out a shortage of the security reduction of the above verifiable and anonymous encryption system, we propose another concrete construction of verifiable and anonymous encryption on the basis of zero-knowledge proof of knowledge.

The emphasis of Chapter 4 is to build signcryption schemes that are secure against related-key attacks. In the first place, we define our security model of signcryption under related-key attacks. In the second place, we propose a specific construction of RKA secure signcryption, and prove its security in the random oracle model, and then we further consider ciphertext anonymity in our signcryption scheme in the setting of related-key attacks. Finally, we strengthen the security model of signcryption by removing the assumption that both the sender and the

receiver are honest, and then we put forward an RKA secure signcryption scheme under this setting, as well as its security proof.

The focus of Chapter 5 is to design of related-key attack secure public-key encryption schemes. First of all, after the analysis of a linear attack on the Cramer-Shoup cryptosystem [CS01], we propose an efficient public-key encryption scheme which is resistant to related-key attacks, and prove its RKA security without random oracles. Next, after pointing out a related-key attack on an existing scheme, we present a public-key encryption scheme with RKA security from bilinear maps, and analyze its RKA security under chosen ciphertext attacks. Lastly, we provide a public-key encryption scheme resistant to related-key attacks without pairings, and show its security proof in the setting of chosen ciphertext attacks.

The goal of Chapter 6 is to achieve data integrity in proof of retrievability systems under related-key attacks. To begin with, we review a publicly verifiable proof of retrievability system and describe how an adversary breaks its data integrity to deceive users under related-key attacks. Later in this chapter, we propose a proof of retrievability scheme with public verification which is secure against related-key attacks, and prove its security in the random oracle model.

The contributions of this thesis are concluded in Chapter 7.

Chapter 2

Preliminaries

In this chapter, we review some basic cryptographic tools related to this research.

2.1 Notations

Below we explain the meanings of the symbols that will be used throughout the thesis.

Denote $\mathbb{N}$ by the set of natural numbers $\{1, 2, \ldots\}$ and Z by the set of integers $\{\ldots, -2, -1, 0, 1, 2, \ldots\}$. We use Z_p to denote the set $\{0, \ldots, p-1\}$ and Z_p^* $= \{k | 1 \leq k \leq p$ and $\gcd(k, p) = 1\}$ the set of positive integers smaller than p and relatively prime to p. The security parameter is denoted by $\lambda \in \mathbb{N}$, which will sometimes be written in its unary representation, 1^λ.

If S is a set, then $|S|$ is its cardinality. If S is a non-empty set and $a \in S$, then a is an element belonging to S. If $s_1, s_2 \in \{0,1\}^*$ are strings, $s_1 || s_2 \in \{0,1\}^*$ is the concatenation of binary strings s_1 and s_2.

2.2 Bilinear Pairings and Some Complexity Assumptions

In this section, we recall the definitions of bilinear pairings and bilinear pairing groups, and then recall some computational assumptions related to this study.

2.2.1 Bilinear Pairings

Let G and $\hat{G}$ be two multiplicative cyclic groups of prime order $p \geq 2^\lambda$. Let g be a generator of G and $\hat{g}$ be a generator of $\hat{G}$. We define $\hat{e} : G \times \hat{G} \rightarrow G_T$ to be a bilinear map if it has the following properties [BF01, Jou00, JN03]:

1. Bilinear: for all $g \in G$, $\hat{g} \in \hat{G}$ and a, $b \in Z_p^*$, we have $\hat{e}(g^a, \hat{g}^b) = \hat{e}(g, \hat{g})^{ab}$.
2. Non-degenerate: $\hat{e}(g, \hat{g}) \neq 1$.

We say that $(G, \hat{G})$ is a bilinear group if the group action in $(G, \hat{G})$ can be computed efficiently and there exists a group G_T and an efficiently computable bilinear map $\hat{e} : G \times \hat{G} \to G_T$ as above. Typically, G and $\hat{G}$ are subgroups of the group of points on an elliptic curve over a finite field, G_T will be a subgroup of the multiplicative group of a related finite field, and the map $\hat{e}$ will be derived from either the Weil or Tate pairing on the elliptic curve.

Galbraith, Paterson and Smart classify pairing into three types in [GPS08].

- Type 1: $G = \hat{G}$, or there exists efficiently computable isomorphism between the two groups.
- Type 2: $G \neq \hat{G}$, and here is no efficiently computable homomorphism from G to $\hat{G}$ but there exists an efficiently computable homomorphism $\psi : \hat{G} \to G$.
- Type 3: $G \neq \hat{G}$, and there are no efficiently computable homomorphisms between G and $\hat{G}$,

Generally speaking, type 1 is known as symmetric pairing, while type 2 and type 3 are known as asymmetric pairing. In this thesis, bilinear pairings of Type 1 and Type 3 will be used.

2.2.2 Complexity Assumptions

Suppose that Groupgen is a probabilistic polynomial-time algorithm that inputs a security parameter 1^λ, and outputs a triplet (G, p, g) where G is a group of order p that is generated from g, and p is a prime number.

Computational DL. The computational discrete log (DL) problem is that for any probabilistic polynomial-time algorithm, it is difficult to compute a given (g, g^a), where $g \in G$, $a \in Z_p^*$ are chosen independently and uniformly at random.

Computational DL Oracle $\mathcal{O}_g^{\mathsf{DL}}$. This oracle is defined over a cyclic group G, On input a value $Y \in G$, this oracle outputs $a \in Z_p^*$ such that $Y = g^a$.

Computational DH. The computational Diffie-Hellman (DH) problem is that for

any probabilistic polynomial-time algorithm, it is difficult to compute g^{ab} given (g, g^a, g^b), where $g \in G$, a, $b \in Z_p^*$ are chosen independently and uniformly at random.

Decisional DH. The decisional Diffie-Hellman (DH) problem is that for any probabilistic polynomial-time algorithm, it is difficult to distinguish (g, g^a, g^b, g^{ab}) from (g, g^a, g^b, Z), where g, $Z \in G$, a, $b \in Z_p^*$ are chosen independently and uniformly at random.

A variant of this is that the ensembles $\{G,\ g,\ f,\ g^r,\ f^r\}$ and $\{G,\ g,\ f,\ g^{r_1},\ f^{r_2}\}$ are computationally indistinguishable, where $(G,\ p,\ g) \leftarrow \text{Groupgen}(1^\lambda)$, and the elements $g,\ f \in G$, $r,\ r_1,\ r_2 \in Z_p$ are chosen independently and uniformly at random.

A Basic Scheme Based on DDH. Since the introduction of the DDH assumption [Bon98], there have been several interesting applications. Note that the DDH assumption readily gives a chosen-plaintext attack (CPA) secure public-key encryption scheme. Let the public key consist of random elements $g,\ f,\ g^{x_1},\ f^{x_2} \in G$, and the secret key consist of random elements $x_1,\ x_2 \in Z_p$. The encryption of a message $M \in G$ is given by $(C_1,\ C_2,\ C_3) = (g^r,\ f^r,\ (g^{x_1} f^{x_2})^r \cdot M)$, where $r \in Z_p$ is a random element. The message M can be recovered with the secret key x_1, x_2 by computing $M = C_3 \cdot (C_1)^{-x_1} \cdot (C_2)^{-x_2}$.

Computational BDH. The computational bilinear Diffie-Hellman (BDH) problem is that for any probabilistic polynomial-time algorithm, it is difficult to compute $\hat{e}(g,g)^{abc}$ given (g, g^a, g^b, g^c), where $g \in G$, a, b, $c \in Z_p^*$ are chosen independently and uniformly at random.

In the asymmetric setting, it will be that it is difficult to compute $\hat{e}(g,\hat{g})^{abc}$ given (g, g^a, g^b, $\hat{g}$, $\hat{g}^a$, $\hat{g}^c$), where $g \in G$, $\hat{g} \in \hat{G}$, a, b, $c \in Z_p^*$ are chosen independently and uniformly at random.

Decisional BDH. The decisional bilinear Diffie-Hellman (BDH) problem is that for any probabilistic polynomial-time algorithm, it is difficult to distinguish (g, g^a, g^b, g^c, $\hat{e}(g,g)^{abc}$) from (g, g^a, g^b, g^c, Z), where $g \in G$, $Z \in G_T$, a, b, $c \in Z_p^*$ are chosen independently and uniformly at random.

In the asymmetric setting, it will be that it is difficult to distinguish (g, g^a, g^b, $\hat{g}$, $\hat{g}^a$, $\hat{g}^c$, $\hat{e}(g,\hat{g})^{abc}$) from (g, g^a, g^b, $\hat{g}$, $\hat{g}^a$, $\hat{g}^c$, Z), where $g \in G$, $\hat{g} \in \hat{G}$, $Z \in G_T$, a, b,

$c \in Z_p^*$ are chosen independently and uniformly at random.

2.3 Cryptographic Tools

In this section, we give a brief introduction on some cryptographic tools needed in this research.

2.3.1 Hash Functions

A hash function, $H : \{0,1\}^* \rightarrow \{0,1\}^\lambda$, is an algorithm that can map data of arbitrary length to data of a fixed length λ. In this thesis, we require that hash functions have to be collision resistant. A family of collision resistant hash functions is a set of hash functions with the following properties [Dam87]:

1. There is a probabilistic polynomial time algorithm, which on input a value of security parameter selects uniformly and randomly a member of the family with the given value attached.

2. All functions in the family are computable in polynomial time.

3. The problem of finding $x \neq y$ such that $h(x) = h(y)$ for a given h in the family is computationally impossible to solve.

The random oracle model, introduced by Bellare and Rogaway [BR93] as a paradigm for designing efficient protocols, assumes that all parties including the adversary have access to a public, truly random hash function H. In practice, this ideal hash function H is instantiated as a concrete cryptographic hash function. Though from a theoretical perspective, a security proof in the random oracle model is only a heuristic indication of the security of the system when instantiated with a specific hash function, this model is extremely useful for designing simple, efficient and highly practical solutions for many problems. On the contrary, in the standard model (without random oracles), there are no idealized oracle accesses, and the security is proven using only the standard complexity assumptions. Hence, from the point of security, a proof in the standard model is preferable to a proof in the random oracle model.

2.3.2 Strong One-Time Signature

The notion of digital signature was first proposed by Diffie and Hellman [DH76]. Generally, a signature scheme consists of the following four algorithms: parameter generation algorithm Setup, key generation algorithm Keygen, signing algorithm Sign and verifying algorithm Verify.

1. Setup(1^λ) $\rightarrow$ *params*: Taking a security parameter λ as input, this algorithm outputs the public parameters *params*.

2. Keygen(*params*) $\rightarrow$ (VK, SK): Taking the public parameters *params* as input, this algorithm outputs a verifying (public) key VK and a signing (private) key SK.

3. Sign(*params*, m, SK) $\rightarrow \sigma$: Taking the public parameters *params*, a message m and the signing key SK as input, this algorithm outputs a signature σ.

4. Verify(*params*, VK, m, σ) $\rightarrow$ *true*/*false*: Taking the public parameters *params*, the verifying key VK, a pair (m, σ) of message and signature as input, this algorithm outputs *true* for a valid signature or *false* for an invalid signature.

We require that a signature scheme $\mathcal{SIG}$ is correct if for any $\lambda \in \mathbb{N}$, *params* $\leftarrow$ Setup(1^λ), (VK, SK) $\leftarrow$ Keygen(*params*), and $\sigma \leftarrow$ Sign(*params*, m, SK), we have that Verify(*params*, VK, m, σ) = *true*.

Existential Unforgeability [GMR88]. A signature scheme $\mathcal{SIG}$ = (PG, KG, Enc, Dec) is existentially unforgeable under adaptive chosen message attacks (EUF-CMA secure) if the advantage function referring to the security game defining in Figure 2.1

$$\mathbf{Adv}^{\text{EUF-CMA}}_{\mathcal{SIG},\Phi}(\mathcal{A}) = \Pr[\mathcal{SIG}^{\mathcal{A}} \Rightarrow \mathit{true}]$$

is negligible in the security parameter λ for any adversary algorithm $\mathcal{A}$.

Strong One-Time Signature. A signature scheme (Setup, Signkeygen, Sign, Verify) is a strong one-time signature (OTS) scheme, if for a stateful adversary $\mathcal{A}$ it

proc Initialize	proc Sign(m)
$M \leftarrow \emptyset$	$\sigma \leftarrow$ Sign($params$, m, SK)
$params \leftarrow$ Setup(1^λ)	$M \leftarrow M \cup \{m\}$
(VK, SK) $\leftarrow$ Keygen($params$)	Return σ
Return VK	proc Finalize(m, σ)
	Return Verify($params$, VK, m, σ) $= true$ $\wedge$ $(m \notin M)$

Figure 2.1: Game defining EUF-CMA security for $\mathcal{SIG}$ = (SKg, SSig, SVer).

holds that

$$\Pr\left[\begin{array}{l|l} & params \leftarrow \text{Setup}(1^\lambda). \\ \text{Verify}(params, \text{VK}, & (\text{VK}, \text{SK}) \leftarrow \text{Keygen}(params). \\ m', \sigma') = true \wedge & m \leftarrow \mathcal{A}(\text{VK}). \\ (m', \sigma') \neq (m, \sigma) & \sigma \leftarrow \text{Sign}(params, m, \text{SK}). \\ & (m', \sigma') \leftarrow \mathcal{A}(\sigma). \end{array}\right]$$

is a negligible function in the security parameter λ.

A Strong One-Time Signature Scheme from DDH. We present a strong one-time signature scheme in [Gro06], which is secure under the difficulty of computing discrete log and H is collision resistant.

- Key generation. Choose random $g \in G$, a, b, $c \in Z_p^*$, a collision resistant hash function $H : G \to Z_p^*$, and set $u_1 = g^a$, $u_2 = g^b$, $u_3 = g^c$. The verification key is $VK = (u_1, u_2, u_3)$, and the signing key is $SK = (a, b, c)$.

- Sign. To sign a message $M \in G$,

  1. choose a random element $e \in Z_p^*$, and compute

  $$w = c + e \cdot a + (H(M) + e) \cdot b.$$

  2. output the signature $\sigma = (e, w)$.

- Verify. To verify a signature $\sigma = (e, w)$, check the equation

  $$g^w = u_3 {u_1}^e {u_2}^{H(M)+e}.$$

  If the equation holds, output 1; otherwise, output 0.

2.3.3 Public-Key Encryption

A public key encryption scheme is composed of the following four algorithms [CS01]: parameter generation algorithm PG, key generation algorithm KG, encryption algorithm Enc and decryption algorithm Dec.

- $\mathrm{PG}(1^\lambda) \to pars$: Taking a security parameter λ as input, this algorithm outputs the public parameters $pars$.
- $\mathrm{KG}(pars) \to (ek, dk)$: Taking the public parameters $pars$ as input, this algorithm outputs an encryption (public) key ek and a decryption (private) key dk.
- $\mathrm{Enc}(pars, ek, m) \to C$: Taking the public parameters $pars$, an encryption (public) key ek and a plaintext m as input, this algorithm outputs a ciphertext C.
- $\mathrm{Dec}(pars, ek, dk, C) \to m/\perp$: Taking the public parameters $pars$, an encryption (public) key ek, a decryption (private) key dk and a ciphertext C as input, this algorithm outputs m for a valid ciphertext or $\perp$ for an invalid ciphertext.

We require that a public key encryption scheme $\mathcal{PKE}$ is correct if for any $\lambda \in \mathbb{N}$, $pars \leftarrow \mathrm{PG}(1^\lambda)$, $(ek, dk) \leftarrow \mathrm{KG}(pars)$, and $C \leftarrow \mathrm{Enc}(pars, ek, m)$, we have that $\mathrm{Dec}(pars, ek, dk, C) = m$.

IND-CCA Security [NY90]. A public-key encryption scheme $\mathcal{PKE} = (\mathrm{PG}, \mathrm{KG}, \mathrm{Enc}, \mathrm{Dec})$ is indistinguishable under chosen-ciphertext attacks (IND-CCA security) if for a stateful adversary algorithm $\mathcal{A}$, the advantage in the following game is negligible in the security parameter λ.

1. $pars \leftarrow \mathrm{PG}(1^\lambda)$.
2. $(ek, dk) \leftarrow \mathrm{KG}(pars)$.
3. $(m_0, m_1) \leftarrow \mathcal{A}^{\mathrm{Dec}(dk,\cdot)}(pars, ek)$ such that $|m_0| = |m_1|$.
4. $C^* \leftarrow \mathrm{Enc}(pars, ek, m_d)$ where $d \in \{0, 1\}$.
5. $d' \leftarrow \mathcal{A}^{\mathrm{Dec}(dk,\cdot)}(C^*)$.
6. Output d'.

Here $\mathrm{Dec}(dk, \cdot)$ is an oracle that on an input C, it returns $\mathrm{Dec}(dk, C)$. We restrict algorithm $\mathcal{A}$ to make queries C such that $C = C^*$.

We define the advantage of algorithm $\mathcal{A}$ in the above game as

$$\mathbf{Adv}^{\text{IND-CCA}}_{\mathcal{A},\mathcal{PKE}}(\lambda) \stackrel{\text{def}}{=} |\Pr[d = d'] - 1/2|.$$

The weaker notion of IND-CPA security (i.e., secure against chosen plaintext attacks) is obtained in the above security game when depriving algorithm $\mathcal{A}$ of the the access to the decryption oracle.

Key Privacy. The notion of key privacy (or anonymity) under chosen plaintext and chosen ciphertext attacks for a public-key encryption scheme is introduced by Bellare, Boldyreva, Desai and Pointcheval [BBDP01]. A public-key encryption scheme $\mathcal{PKE} = (\mathrm{PG}, \mathrm{KG}, \mathrm{Enc}, \mathrm{Dec})$ is anonymous under chosen ciphertext attacks (ANON-CCA security) if for a stateful adversary algorithm $\mathcal{A}$, the advantage in the following game is negligible in the security parameter λ.

1. $pars \leftarrow \mathrm{PG}(1^\lambda)$.
2. $(ek_0, dk_0) \leftarrow \mathrm{KG}(pars)$; $(ek_1, dk_1) \leftarrow \mathrm{KG}(pars)$.
3. $m \leftarrow \mathcal{A}^{\mathrm{Dec}(dk_0,\cdot),\ \mathrm{Dec}(dk_1,\cdot)}(pars, ek_0, ek_1)$.
4. $C^* \leftarrow \mathrm{Enc}(pars, ek_d, m)$ where $d \in \{0, 1\}$.
5. $d' \leftarrow \mathcal{A}^{\mathrm{Dec}(dk,\cdot)}(C^*)$.
6. Output d'.

Here for $i \in \{0, 1\}$, $\mathrm{Dec}(dk_i, \cdot)$ is an oracle that on an input C, it returns $\mathrm{Dec}(dk_i, C)$. We restrict algorithm $\mathcal{A}$ to make queries C such that $C = C^*$.

We define the advantage of algorithm $\mathcal{A}$ in the above game as

$$\mathbf{Adv}^{\text{ANON-CCA}}_{\mathcal{A},\mathcal{PKE}}(\lambda) \stackrel{\text{def}}{=} |\Pr[d = d'] - 1/2|.$$

Likewise, the weaker notion of ANON-CPA security (i.e., secure against chosen plaintext attacks) is obtained in the above security game when depriving algorithm $\mathcal{A}$ of access to the decryption oracle.

2.3.4 Zero-Knowledge Proof-of-Knowledge

The notion of zero-knowledge proof was put forward by Goldwasser, Micali and Rackoff in [GMR89]. In a zero-knowledge proof protocol, a prover convinces a verifier that a statement is true, while the verifier learns nothing except the validity of the assertion. A proof of knowledge [BG92] is a protocol where the verifier is convinced that the prover knows a certain quantity w satisfying some kinds of relation R with respect to a commonly known string x. If a proof-of-knowledge protocol can be done in such a way that the verifier learns nothing other that the validity of the statement, this protocol is called a zero-knowledge proof of knowledge (ZKPoK) protocol [GMR89]. Hitherto, various efficient ZKPoK protocols about knowledge of discrete logarithms and their relations have been proposed [CS97, Bou00, CL01, CS03], of which some are used in the anonymous systems to prove their possession of certificates for authentication without revelation of certificates.

Zero-Knowledge of Proof-of-Knowledge of Discrete Logarithm. Let G be a finite cyclic group with prime order p. Let g be a generator of G. Let $H : \{0,1\}^* \rightarrow \{0,1\}^\lambda$ be a collision resistant hash function. Denote PK$\{(x) : Y = g^x\}$ by a ZKPoK protocol that allows prover P to prove the knowledge of $x \in Z_p$ such that $Y = g^x$ for some $Y \in G$. Following the description in [Sch91], we outline this ZKPoK protocol as follows.

1. Commitment. P randomly chooses $\rho \in Z_p$, computes $T = g^\rho$, and sends T to V.

2. Challenge. V randomly chooses $c \in \{0,1\}^\lambda$ and sends c to P.

3. Response. P computes $z = \rho - cx \bmod p$, and sends z to V.

4. Verify. V outputs accept if $T = Y^c g^z$.

Any ZKPoK protocol can be turned into non-interactive form, which is called signature of knowledge [CS97], by setting the challenge to the hash value of the commitment together with the message to be signed [FS86].

A Signature of Knowledge of Discrete Logarithm. We turn PK$\{(x) : Y = g^x\}$ to a signature of knowledge SPK$\{(x) : Y = g^x\}(m)$ as follows. A pair $(c, z) \in \{0,1\}^\lambda \times Z_p$ satisfying $c = H(g||Y||Y^c g^z||m)$ is a signature of knowledge of the discrete logarithm of Y to base g for message $m \in \{0,1\}^*$ [CS97].

1. Commitment. P randomly chooses $\rho \in Z_p$, and computes $T = g^\rho$.
2. Challenge. P computes $c = H(g||Y||T||m)$.
3. Response. P computes $z = \rho - cx \bmod p$, and outputs (c, z) as SPK.
4. Verify. Anyone can verify SPK by testing if

$$c = H(g||Y||Y^c g^z||m).$$

Informally, a non-interactive simulation-sound zero-knowledge proof system for a language L with a witness relation R_L is a tuple of probabilistic polynomial-time algorithms (Gen, P, V, S_1, S_2) with the following properties.

1. Perfect completeness. For $(x, w) \in R_L$ it holds that

$$\Pr\left[\mathrm{V}(1^\lambda, x, \pi, \sigma) = 1 \,\middle|\, \begin{array}{c} \sigma \leftarrow \mathrm{Gen}(1^\lambda) \\ \pi \leftarrow \mathrm{P}(1^\lambda, x, w, \sigma) \end{array}\right] = 1,$$

   where the probability is taken over by the internal randomness of the public parameter generation algorithm Gen, the prover algorithm P and the verifier algorithm V.

2. Zero knowledge. For every probabilistic polynomial-time algorithm $\mathcal{A}$ such that the quantity

$$\left|\Pr\left[b \,\middle|\, \begin{array}{c} \sigma \leftarrow \mathrm{Gen}(1^\lambda) \\ b \leftarrow \mathcal{A}^{\mathrm{P}(1^\lambda,\sigma)}(1^\lambda, \sigma) \end{array}\right] - \Pr\left[b \,\middle|\, \begin{array}{c} (\sigma, \tau) \leftarrow \mathrm{S}_1(1^\lambda) \\ b \leftarrow \mathcal{A}^{\mathrm{S}_2'(1^\lambda,\cdot,\cdot,\tau)}(1^\lambda, \sigma) \end{array}\right]\right|$$

   is negligible in the security parameter λ, where $\mathrm{S}_2'(1^\lambda, x, w, \tau) = \mathrm{S}_2(1^\lambda, x, \tau)$.

3. Simulation soundness. For every probabilistic polynomial-time algorithm $\mathcal{A}$ such that the quantity

$$\Pr\left[x \notin L, \pi \notin Q \text{ and } \mathrm{V}(1^\lambda, x, \pi, \sigma) = 1 \,\middle|\, \begin{array}{c} (\sigma, \tau) \leftarrow \mathrm{S}_1(1^\lambda) \\ (x, \pi) \leftarrow \mathcal{A}^{\mathrm{S}_2(1^\lambda,\cdot,\tau)}(1^\lambda, \sigma) \end{array}\right]$$

   is negligible in the security parameter λ, where Q is the set of S_2's answers to algorithm $\mathcal{A}$'s oracle queries.

Chapter 3

Achieving Security in Verifiable and Anonymous Encryption Systems Involving Third Parties

The distribution and availability of digital information in modern life and work lead to new opportunities for providing support to individuals. This ubiquity of information also creates new challenges for the protection of both the provided information and the privacy of its users, which requires the communication mechanisms to allow some specification of the access policies and protect the privacy of the users at the same time. The emphasis of this chapter is designing secure protocols under a scenario where for any incoming messages, the receiver's identity will be checked by a third party.

3.1 Introduction

In the cryptographic world, third parties such as private key generators (PKGs) in identity-based encryption play an important role in protecting information. From the point of security, a necessary assumption in a third-party involved system is that the third party should be fully trusted. However, this hypothesis is impractical in the real world because it is hard to make a third party fully trusted by all its users. As a result, some cryptographic protocols use a third party with trust at a certain level rather than full trust.

3.1.1 Scenario

Considering the scenario as shown in Figure 3.1. The gateway of organization A rejects any inbound traffic unless a message is really for at least one member in organization A. In this scenario, the gateway might not be regarded as honest in

terms of whom the intended receivers would be. It could be the case that an agent who works for organization A sends an inbound encrypted message to organization A, where only the agent knows the receivers. On the other hand, a sender outside organization A should be able to make sure that the message for the target receiver(s) inside organization A will not be rejected by the gateway while maintaining the message confidentiality and the receiver anonymity. To ensure maximum security, we should also consider that the gateway could potentially collude with the corrupted receivers who will leak all the personal information related to their identities in order to compromise other users.

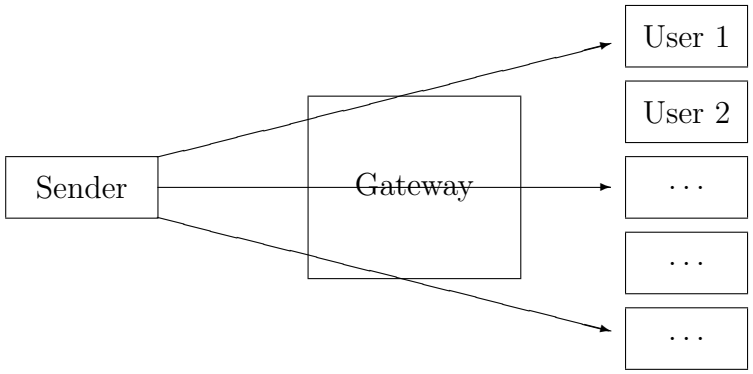

Figure 3.1: Scenario.

Briefly, this scenario can be seen as an untrusted gateway being involved for verification in an anonymous encryption system. To address this issue, we put forward the notion of verifiable and anonymous encryption as our access control mechanism, which is a three-party protocol enabling an outside sender to transmit a ciphertext to the inside user(s)[1] under the verification of gateway who determines whether to broadcast this ciphertext or not. Unlike general encryption schemes, because of the involvement of an untrusted gateway, in this case collusion attack may be possible between the gateway and the corrupted users: the gateway may collude with the corrupted users to obtain the message content, and the gateway may collude with the corrupted users to guess the identity of the privileged user(s). All in all, a verifiable and anonymous encryption system should maintain confidentiality and anonymity while preventing collusion attack between the gateway and the corrupted users, such that an outside sender can securely transmit information to at least one inside user through a gateway.

[1]We assume that the insider user(s) belong(s) to a group composed of a large number of users.

3.1.2 Our Results

In order to design an encryption system for the aforementioned scenario, we propose three different frameworks with which one can design a protocol covering three parties (the sender, the gateway and the receivers) to protect both the message content and the privacy of receivers. Within these frameworks, when a sender, say Alice, wants to transmit a message to an internal user, say Bob, she encrypts her message with an algorithm to generate a ciphertext which is composed of two parts: encryption of the message and verification of the receiver's identity. The gateway checks whether Bob is an inside user with the verification part while Bob decrypts the ciphertext with the encryption part.

In our first construction, we consider a broadcast environment in which the message will be sent to multiple receivers, and name it as gateway-based verifiable and anonymous broadcast encryption. We apply a technique called public key encryption and identity search (PKEIS) for the realization, where each member in the organization sends a "trapdoor" related to its private key and identity token to the gateway through a public channel. The gateway can then use this piece of information to check whether the receivers belong to the organization by blindly verifying the identity token embedded in the trapdoor. To address the issue of receiver privacy, we make use of anonymous broadcast encryption in the encryption process, so a privileged receiver can only be sure that it is one of receivers of the message and learns nothing about other receivers of the message.

The key challenge in building a gateway-based verifiable and anonymous broadcast encryption scheme is preventing collusion attacks. We define the following security notions to capture the dishonest behaviours of the sender, the gateway and the receivers, respectively.

1. The confidentiality (or semantic security) of the message. Confidentiality here means that the gateway cannot obtain any information about the plaintext from the ciphertext, even if it colludes with the non-privileged receivers.

2. The anonymity of the receivers. Anonymity here means that the gateway has no idea who will be the privileged receivers from the ciphertext, even if it colludes with the corrupted receivers, which also implies that the gateway cannot distinguish the relations between the ciphertexts and the trapdoors with the information it obtains from the corrupted receivers.

With regard to the concrete construction of gateway-based verifiable and anonymous broadcast encryption, we ask a third party called authority center (AC) to generate a public and private key pair for the gateway and assign every potential receiver a secret identity token, whilst the receivers generate their own public and private key pairs. Every potential receiver will then send a "trapdoor", which is embedded with its private key and identity token, to the gateway through a public channel. Whenever a sender wants to send a message through this gateway, it will yield a ciphertext composed of two parts: encryption of the message and verification of receivers' identity tokens. The gateway could check whether the identity tokens (not the data items or keywords) used in the broadcast encryption are from the data stored by the gateway with the verification part while the privileged receivers decrypt the ciphertext to gain the plaintext with the encryption part. Concerning the security, our gateway-based verifiable and anonymous broadcast encryption scheme protects the confidentiality of message content and the privacy of privileged receivers according to a formal model which we specify under collusion attacks, which means that the gateway learns nothing from the ciphertext even if it colludes with the corrupted receivers.

In the following part, we remove the broadcast condition and simply consider an anonymous encryption scheme involving an untrusted gateway for verification, and show some interesting methods to solve the problem with stronger security than the first construction as secret identity tokens are no longer required.

In our second construction, we make use of a trusted third party as key generation center (KGC) [ARP03] to generate partial private keys for the users (i.e., the KGC does not have access to the private keys of the users), and then allow users to generate their own (full) private keys (from partial private keys). Also, we require every user in the system to generate a trapdoor and send it to the gateway through a public channel, which assures the unlinkability between the users and the trapdoors. The gateway checks the validity of the trapdoors with the public keys of the users, and maintains them in a trapdoor list (without identity information). Once a ciphertext comes in, the gateway checks it using the trapdoor list stored on its side. Seemingly, our problem has been addressed well. However, in the above construction, with respect to the security reduction, on the one hand, the adversaries are not allowed to issue a public key query on the user identity of the challenge phase. Nevertheless, this limitation is restricted so that it can only be applied in special circumstances. On the other hand, its security is reduced in the random oracle model, which may

be insecure in some cases [Nac05].

To overcome these drawbacks, we resort to a cryptographic primitive called zero-knowledge proof of knowledge [GMR89] to hide the message and the identity information of the privileged user. In this way, the adversary could be given both identities and public keys of all the users while retaining the confidentiality of the message content and the anonymity of the privileged user. In our third construction, we use a trusted third party named certificate authority (CA) to issue certificates, which are actually the signatures of the users' public keys and identity information generated by CA with its master key. Now the verification part of the ciphertext is replaced by a signature based on a non-interactive zero-knowledge proof of knowledge scheme, which in our case consists of the plaintext and the certificate of the privileged user, and the gateway checks the validity of this zero-knowledge proof of knowledge. The problem here is how to generate certificates for the users in an efficient way under the setting of bilinear groups. Fortunately, we found that Abe, Groth, Haralambiev and Ohkubo [AGHO11] have presented an efficient structure-preserving signature in asymmetric bilinear maps, for which the proof is extractable and therefore yields an efficient non-interactive zero-knowledge proof of knowledge system.

3.1.3 Related Work

Verifiable Encryption. A related but not closely related work is verifiable encryption. Verifiable encryption has the property that the validity of ciphertext can be verified without knowledge of a private key, and it has been used for fair exchange [ASW98, Bao98], key escrow schemes [PS00], signature sharing schemes [FR95] and publicly verifiable secret sharing [Sta96]. The concept of verifiable encryption was first introduced by Stadler [Sta96] with the cut-and-choose methodology in the context of publicly verifiable secret sharing schemes in 1996. Then, Asokan, Shoup and Waidner [ASW98] proposed a more general form of verifiable encryption with perfect separability for the purpose of fair exchange of signatures in 1998. Bao [Bao98] gave a verifiable encryption scheme without using the cut-and-choose methodology, but it failed to provide semantic security [GM84]. In 2001 Camenisch and Lysyanskaya [CL01] proposed an anonymous verifiable encryption scheme which did not use the cut-and-choose methodology, but the prover needed to know the private key of the receiver. In 2003 Camenisch and Shoup [CS03] introduced a verifiable encryption system that provides chosen ciphertext security and avoids inefficient cut-and-choose

proofs, but it requires the use the Paillier encryption function [Pai99].

Broadcast Encryption. Broadcast encryption (BE) is designed to address the problem of broadcasting a message to an arbitrary subset S from a universe of receivers U who are listening on a broadcast channel. Since it was introduced by Fiat and Navor [FN94] in 1993, various BE schemes have been proposed [BGW05, DPP07, Del07, DF, FN94, GW09] from different aspects such as strength of security notions, public and private key storage requirements, ciphertext length and computational costs. Concerning making ciphertexts to be as short as possible, schemes in [BGW05] and [GW09] are close to optimal. With regard to the privacy of receivers, Barth, Boneh and Waters [BBW06] first considered privacy of users in broadcast encryption under the context of encryption systems, Then Libert, Paterson and Quaglia [LPQ11] gave a generalized and unified security definition for anonymous broadcast encryption.

Private Information Retrieval. Private information retrieval (PIR) protocols introduced by Chor, Goldreich, Kushilevitz and Sudan [CGKS95] allow users to retrieve some data items or search some data items from a public database without revealing to the database administrator which items they retrieve or search, but the data should be public. Boneh, Crescenzo, Ostrovsky and Persiano [BCOP04] proposed public key encryption with keyword search (PEKS) to enable one, say Alice, to provide a key to the gateway such that the gateway can test whether the word "urgent" (or other words) is a keyword in the email without learning anything else about the email, but it needs a secure channel between Alice and the email server. In order to solve this problem, Baek, Safavi-Naini and Susilo [BSNS05, BSNS08] provided a secure channel free public key encryption with keyword search (SCF-PEKS) scheme.

3.1.4 Organization

The remainder of this chapter is organized as follows. In Section 3.2, we describe the framework and security model of gateway-based verifiable and anonymous broadcast encryption, review the related complexity assumptions, provide an efficient instantiation according to the formal framework of gateway-based verifiable and anonymous

broadcast encryption, as well as its security proof based on the BDH complexity assumptions. In Section 3.3, we define the algorithms and security requirements of a verifiable and anonymous encryption system with an untrusted gateway, introduce the generic bilinear group model and the complexity assumptions that our proof of security depends on, detail one verifiable and anonymous encryption scheme in asymmetric bilinear maps, and prove its confidentiality and anonymity in the random oracle model. In Section 3.4, after pointing out a shortage of the security reduction in the verifiable and anonymous encryption system in Section 3.3, we present another construction on the basis of zero-knowledge proof of knowledge, as well as its security proof. Finally, we conclude this chapter in Section 3.5.

3.2 Gateway-Based Verifiable and Anonymous Broadcast Encryption

In a gateway-based verifiable and anonymous broadcast encryption system, whenever a sender wants to broadcast a message, the ciphertext needs to be firstly checked by the gateway; otherwise, the message will not be transferred to the privileged receivers.

3.2.1 Definition and Security Model

We firstly present a formal framework of our primitive gateway-based verifiable and anonymous broadcast encryption, and then we describe the adversary model and games between an adversary and a challenger.

In gateway-based verifiable and anonymous broadcast encryption, these three parties are involved: sender, receivers and gateway. Every receiver generates a trapdoor according to its identity token, and sends it to the gateway. The sender generates and sends an encrypted messages which we call "PKEIS ciphertext". The gateway receives a PKEIS ciphertext, and performs verification according to the trapdoors received from the receivers. If the ciphertext passes the verification, the gateway broadcasts the ciphertext; otherwise, it rejects the ciphertext.

In our framework, every receiver R_i generates its own public and private key pair (pk_{R_i}, sk_{R_i}). An authority centre (AC) provides an identity token ID_i for every potential receiver R_i and a public and private key pair (pk_G, sk_G) for the gateway W. Every receiver R_i generates a trapdoor T_i on identity token ID_i under

private key sk_{R_i}, and then sends this trapdoor to gateway W. Gateway W maintains a trapdoor list L_T; whenever it receives a trapdoor T_i from a receiver R_i, it adds this trapdoor to the trapdoor list L_T. With the private key sk_G and the trapdoor list L_T, gateway W can verify the ciphertext to determine whether to broadcast it or not.

Let $\mathcal{L}$ be an index set associated with the number of the privileged recipients. A gateway-based verifiable and anonymous broadcast encryption scheme is specified by the following seven randomized algorithms: Setup, Make-GKey, Make-RKey, Encrypt, Verify and Decrypt.

- Setup(λ): Taking a security parameter λ as input, this algorithm outputs a common parameters $params$.

- Make-GKey($params$): Taking the public parameters $params$ as input, this algorithm outputs the public and private key pair (pk_G, sk_G) for gateway W.

- Make-RKey($params$): Taking the public parameters $params$ as input, this algorithm outputs the public and private key pair (pk_{R_i}, sk_{R_i}) for receiver R_i.

- Trapdoor($params$, sk_{R_i}, ID_i): Taking the public parameters $params$, the secret key sk_{R_i} and the identity token ID_i of the receiver R_i as input, this algorithm outputs the trapdoor T_i for ID_i.

  Receiver R_i sends its trapdoor T_i to gateway W through a public broadcast channel. Once gateway W receives a trapdoor T_i, it will add T_i to the trapdoor list L_T which is initially empty.

- Encrypt($params$, M, pk_G, $\{pk_{R_i}, ID_i\}_{i\in\mathcal{L}}$): Taking the public parameters $params$, the message M, the public key pk_G of gateway W, the public keys $\{pk_{R_i}\}_{i\in\mathcal{L}}$ and the corresponding identity tokens $\{ID_i\}_{i\in\mathcal{L}}$ of receivers $\{R_i\}_{i\in\mathcal{L}}$ as input, this algorithm outputs the ciphertext C.

- Verify($params$, sk_G, L_T, C): Taking the public parameters $params$, the private key sk_G of gateway W, the trapdoor list L_T for all the receivers and the ciphertext C as input, this algorithm outputs C in case of success or $\perp$ in case of failure.

- Decrypt($params$, sk_{R_i}, T_i, C): Taking the public parameters $params$, the private key sk_{R_i} and the trapdoor T_i of the receiver R_i and the ciphertext C

as input, this algorithm outputs the message M for a member of privileged receivers or $\perp$ for a member of non-privileged receivers.

Let $\mathcal{R} = \{R_1, \ldots, R_n\}$ for $n \in \mathbb{N}$ be the recipient set. We require that our system is correct, meaning that for all $R_i \in \mathcal{R}$, if $params \leftarrow \text{Setup}(\lambda)$, $(pk_G, sk_G) \leftarrow \text{Make-GKey}(params)$, $(pk_{R_i}, sk_{R_i}) \leftarrow \text{Make-RKey}(params)$, $T_i \leftarrow \text{Trapdoor}(params, sk_{R_i}, ID_i)$, $C \leftarrow \text{Encrypt}(params, M, pk_G, \{pk_{R_i}, ID_i\}_{i \in \mathcal{L}})$, and $C \leftarrow \text{Verify}(params, sk_G, L_T, C)$, then $M = \text{Decrypt}(params, sk_{R_i}, T_i, C)$.

The goal of a gateway-based verifiable and anonymous broadcast encryption scheme is to send a message to multiple receivers under the verification of the gateway while keeping the confidentiality of the message content and the anonymity of the receivers. In a gateway-based verifiable and anonymous broadcast encryption protocol, which involves three parties: a sender, a gateway and the potential receivers, collusion resistance has to be dealt with: (1) a collusion of the corrupted receivers and the gateway may help the adversary recover the message from the ciphertext; (2) a collusion of the corrupted receivers and the gateway may help the adversary identify the identity tokens of the privileged receivers, or help the adversary discern the trapdoors involved in the ciphertext. In our security model, collusion will be modeled by the Corrupt queries, which will provide the secret data of the gateway and the corrupted receivers to the adversary. Then, from all the information, the adversary will be allowed to do anything it wants to.

As a consequence, we provide the adversary with two Corrupt oracles in the security model assuming that the adversary has known the public keys of receivers in the system: a Gateway_Corrupt oracle that outputs the trapdoor list stored by the gateway, and a Receiver_Corrupt oracle that outputs the related information (including public key, private key, identity token, trapdoor) of corrupted receiver R_i. We denote the set of all the public keys of potential receivers by PK, and the corruption list of corrupted receivers by L_{CR}.

- Gateway_Corrupt$(PK) \rightarrow L_T$, takes the public key set PK as input, this algorithm outputs all the trapdoors in the trapdoor list L_T.
- Receiver_Corrupt$(pk_{R_i}) \rightarrow (sk_{R_i}, ID_i, T_i)$, takes the public key pk_{R_i} as input, this algorithm outputs the corresponding private key, identity token and trapdoor in a 3-tuple (sk_{R_i}, ID_i, T_i).

Confidentiality. We define a chosen plaintext attack security for gateway-based verifiable and anonymous broadcast encryption to ensure the confidentiality of the

message content. More precisely, confidentiality is defined using a game between an adversary algorithm $\mathcal{A}$ and a challenger algorithm $\mathcal{B}$ such that algorithm $\mathcal{A}$ cannot distinguish a ciphertext intended for one message from a ciphertext intended for another message.

1. **Initialization.** Algorithm $\mathcal{B}$ runs the Setup algorithm to obtain the public parameters $params$. Then, algorithm $\mathcal{B}$ generates the public and private key pair (pk_G, sk_G) for gateway W. Algorithm $\mathcal{B}$ gives the public parameters $params$, the public and private key pair (pk_G, sk_G) to algorithm $\mathcal{A}$.

   For every receiver $R_i \in \mathcal{R}$, algorithm $\mathcal{B}$ generates a public and private key pair (pk_{R_i}, sk_{R_i}). Algorithm $\mathcal{B}$ gives their public keys pk_{R_i} to algorithm $\mathcal{A}$ while keeping their private keys sk_{R_i} secret.

2. **Query Phase 1.** Algorithm $\mathcal{A}$ adaptively query pk_{R_i} to the Receiver_Corrupt oracle. Algorithm $\mathcal{B}$ forwards the corresponding private key, identity token and trapdoor in a 3-tuple (sk_{R_i}, ID_i, T_i) to algorithm $\mathcal{A}$, and adds (pk_{R_i}, sk_{R_i}, ID_i, T_i) to list L_{CR}.

3. **Query Phase 2.** Algorithm $\mathcal{A}$ issues $PK = \{pk_{R_1}, \ldots, pk_{R_n}\}$ to the Gateway_Corrupt oracle. Algorithm $\mathcal{B}$ forwards the trapdoor list $L_T = \{T_1, \ldots, T_n\}$ to algorithm $\mathcal{A}$.

   Note that algorithm $\mathcal{A}$ knows nothing about the relations between the public key pk_{R_i} and the trapdoor T_i except those in list L_{CR}.

4. **Challenge.** When algorithm $\mathcal{A}$ decides that Phase 2 is over, it outputs two messages M_0^*, M_1^*, a public key set $PK^* \subset PK$ on which it wishes to be challenged. The only constraint is that none of $pk_{R_i} \in PK^*$ appears in Phase 1. To generate the challenge ciphertext, algorithm $\mathcal{B}$ retrieves the identity token set S^* corresponding to PK^*, chooses a random bit $\gamma \in \{0, 1\}$, and runs the encryption algorithm on M_γ^* to obtain the ciphertext C^*. It sends C^* as the ciphertext to algorithm $\mathcal{A}$.

5. **Query Phase 3.** Algorithm $\mathcal{A}$ continues to adaptively query pk_{R_i} to the Receiver_Corrupt oracle of which $pk_{R_i} \notin PK^*$, as in Phase 1.

6. **Guess.** Algorithm $\mathcal{A}$ outputs its guess $\gamma' \in \{0, 1\}$ for γ, and it wins the game if $\gamma = \gamma'$.

We refer to algorithm $\mathcal{A}$ as an IND-CPA adversary. We define the advantage of algorithm $\mathcal{A}$ in attacking a gateway-based verifiable and anonymous broadcast encryption scheme $\varepsilon =$ (Setup, Make-GKey, Make-RKey, Trapdoor, Encrypt, Verify, Decrypt) as

$$\mathbf{Adv}_{\varepsilon,\mathcal{A}} = |\Pr[\gamma = \gamma'] - 1/2|$$

The probability is over the random bits used by the challenger and the adversary.

Definition 3.1 *We say that a gateway-based verifiable and anonymous broadcast encryption scheme ε is (t, q_T, ϵ)-IND-CPA secure if for any IND-CPA adversary algorithm $\mathcal{A}$ that runs in time t, makes at most q_T Receiver_Corrupt queries, we have that $Adv_{\varepsilon,\mathcal{A}} < \epsilon$.*

There is another stronger version of security, the chosen ciphertext security, where the adversary is not only allowed to issue adaptive Receiver_Corrupt queries, but also allowed to issue decryption queries.

Definition 3.2 *We say that a gateway-based verifiable and anonymous broadcast encryption scheme ε is (t, q_T, ϵ)-IND-CPA secure if ε is (t, q_T, 0, ϵ)-IND-CCA secure.*

Anonymity. We define the following game to ensure that the adversary cannot distinguish a ciphertext intended for one recipient set from a ciphertext intended for another recipient set. More precisely, receiver anonymity is defined using a game between an adversary algorithm $\mathcal{A}$ and a challenger algorithm $\mathcal{B}$.

1. **Initialization.** Algorithm $\mathcal{B}$ runs Setup to obtain the public parameters $params$. Then algorithm $\mathcal{B}$ generates the public and private key pair (pk_G, sk_G) for gateway W. Algorithm $\mathcal{B}$ gives the public parameters $params$, the public and private key pair (pk_G, sk_G) to algorithm $\mathcal{A}$.

   For every receiver $R_i \in \mathcal{R}$, algorithm $\mathcal{B}$ generates an identity token ID_i, a public and private key pair (ID_i, pk_{R_i}, sk_{R_i}) as well as the trapdoor T_i. Algorithm $\mathcal{B}$ gives their public keys (pk_{R_i}) to algorithm $\mathcal{A}$.

2. **Query Phase 1.** Algorithm $\mathcal{A}$ adaptively query pk_{R_i} to the Receiver_Corrupt oracle. Algorithm $\mathcal{B}$ forwards the corresponding private key, identity token and trapdoor in a 3-tuple (sk_{R_i}, ID_i, T_i) to algorithm $\mathcal{A}$, and adds (pk_{R_i}, sk_{R_i}, ID_i, T_i) to list L_{CR}.

3. **Query Phase 2.** Algorithm $\mathcal{A}$ issues $PK = \{pk_{R_1}, \ldots, pk_{R_n}\}$ to the Gateway_Corrupt oracle. Algorithm $\mathcal{B}$ forwards the trapdoor list $L_T = \{T_1, \ldots, T_n\}$ to algorithm $\mathcal{A}$.

   Note that algorithm $\mathcal{A}$ knows nothing about the relations between the public key pk_{R_i} and the trapdoor T_i except those in list L_{CR}.

4. **Challenge.** When algorithm $\mathcal{A}$ decides that Phase 2 is over, it outputs a messages M^*, and two public key sets PK_0^*, $PK_1^* \subset PK$ on which it wishes to be challenged where PK_0^*, PK_1^* are of equal size l. The only constraint is that none of $pk_{R_i} \in PK_0^* \cup PK_1^*$ appears in Phase 1. To generate the challenge ciphertext, algorithm $\mathcal{B}$ retrieves the identity token sets S_0^*, S_1^* respectively corresponding to PK_0^*, PK_1^*, chooses a random bit $\gamma \in \{0, 1\}$, and runs the encryption algorithm on PK_γ^*, S_γ^* to obtain the ciphertext C^*. It sends C^* as the ciphertext to algorithm $\mathcal{A}$.

5. **Query Phase 3.** Algorithm $\mathcal{A}$ continues to adaptively query pk_{R_i} to the Receiver_Corrupt oracle, as in Phase 1.

6. **Guess.** Algorithm $\mathcal{A}$ outputs its guess $\gamma' \in \{0, 1\}$ for γ, and it wins the game if $\gamma = \gamma'$.

We refer to such an algorithm $\mathcal{A}$ as an ANON-IND-CPA adversary. We define the advantage of algorithm $\mathcal{A}$ in attacking a gateway-based verifiable and anonymous broadcast encryption scheme ε = (Setup, Make-GKey, Make-RKey, Trapdoor, Encrypt, Verify, Decrypt) as

$$\mathbf{Adv}_{\varepsilon,\mathcal{A}} = \left|\Pr[\gamma = \gamma'] - 1/2\right|.$$

The probability is over the random bits used by the challenger and the adversary.

Definition 3.3 *We say that a gateway-based verifiable and anonymous broadcast encryption scheme ε is (t, q_T, ϵ)-ANON-IND-CPA secure if for any ANON-IND-CPA adversary algorithm $\mathcal{A}$ that runs in time t, makes q_T Receiver_Corrupte query, we have that $Adv_{\varepsilon,\mathcal{A}} < \epsilon$.*

There is another stronger version of security, the chosen ciphertext security, where the adversary is not only allowed to issue adaptive Receiver_Corrupt queries, but also allowed to issue decryption queries.

Definition 3.4 *We say that a gateway-based verifiable and anonymous broadcast encryption scheme ε is (t, q_T, ϵ)-ANON-IND-CPA secure if ε is $(t, q_T, 0, \epsilon)$-ANON-IND-CCA secure.*

3.2.2 Complexity Assumptions

We define a variant version of computational BDH assumption and decisional BDH assumption, respectively. If n is a positive integer ($\in \mathbb{N}$), we use $[n]$ to denote the set $\{1, \ldots, n\}$.

Computational X-BDH. We say that an algorithm $\mathcal{A}$ has advantage $\mathrm{Adv}_{\mathcal{A}}^{\mathrm{XBDH}} = \epsilon$ in solving the computational X-BDH problem in $(G, \hat{G})$ if

$$\Pr[\mathcal{A}(g, g^a, \{g^{b_i}\}_{i\in[n]}, \hat{g}, \hat{g}^a, \hat{g}^c, \mathcal{O}_g^{\mathsf{DL}}) = \{\hat{e}(g,\hat{g})^{ab_kc}\}_{k\in\mathcal{L}}] \geq \epsilon,$$

where $\mathcal{L} \subset [n]$ and all inputs to $\mathcal{O}_g^{\mathsf{DL}}$ belongs to the set $\{g^{b_i}\}_{i\in[n]\setminus\mathcal{L}}$. The probability is over the random choice of generators g of G and $\hat{g}$ of $\hat{G}$, the random choice of exponents $a, \{b_i\}_{i\in[n]}, c$ in Z_p^* and the random bits used by algorithm $\mathcal{A}$.

Decisional X-BDH. We say that an algorithm $\mathcal{A}$ outputs a bit $\gamma \in \{0,1\}$ has advantage $\mathrm{Adv}_{\mathcal{A}}^{\text{D-XBDH}} = \epsilon$ in solving the Decisional X-BDH problem in $(G, \hat{G})$ if

$$\begin{aligned}&(\mathsf{st}, \mathcal{L}) \leftarrow \mathcal{A}(g, g^a, \{g^{b_i}\}_{i\in[n]}, \hat{g}, \hat{g}^a, \hat{g}^c, \mathcal{O}_g^{\mathsf{DL}}),\\ &\left|\Pr[\mathcal{A}(\mathsf{st}, \mathcal{O}_g^{\mathsf{DL}}, \{\hat{e}(g,\hat{g})^{ab_kc}\}_{k\in\mathcal{L}}) = 0] - \Pr[\mathcal{A}(\mathsf{st}, \mathcal{O}_g^{\mathsf{DL}}, \{Z_k\}_{k\in\mathcal{L}}) = 0]\right| \geq \epsilon,\end{aligned}$$

where $\mathcal{L} \subset [n]$, and all inputs to $\mathcal{O}_g^{\mathsf{DL}}$ belongs to the set $\{g^{b_i}\}_{i\in[n]\setminus\mathcal{L}}$. The probability is over the random choice of generators g of G and $\hat{g}$ of $\hat{G}$, the random choice of exponents $a, \{b_i\}_{i\in[n]}, c$ in Z_p^*, the random choice of $\{Z_k\}_{i=1}^{l} \in G_T$ and the random bits used by algorithm $\mathcal{A}$.

In the decisional X-BDH assumption, we refer to the distribution of $(g,\ g^a,\ \{g^{b_i}\}_{i\in[n]},\ \hat{g},\ \hat{g}^a,\ \hat{g}^c,\ \{\hat{e}(g,\hat{g})^{ab_kc}\}_{k\in\mathcal{L}})$ over $G^{n+2} \times \hat{G}^3 \times G_T^{|\mathcal{L}|}$ as $\mathcal{P}_{XBDH}$, and the distribution on the right as $\mathcal{R}_{XBDH}$.

Definition 3.5 *We say that the (t, ϵ)-Decisional BDH assumption holds in $(G, \hat{G})$ if no t-time algorithm has advantage at least ϵ in solving the Decisional BDH problem in $(G, \hat{G})$.*

Similarly, we say that the (t, ϵ)-Decisional X-BDH assumption holds in $(G, \hat{G})$ if no t-time algorithm has advantage at least ϵ in solving the Decisional X-BDH problem in $(G, \hat{G})$.

3.2.3 Proposed Scheme

Suppose that there are a gateway W and a recipient set $\mathcal{R} = \{R_1, \ldots, R_n\}$ in the system. The AC assigns an identity token ID_i for receiver R_i where $i = 1, \ldots, n$, and a public and private key pair (pk_G, sk_G) for gateway W. Each receiver R_i generates its public and private key pair (pk_{R_i}, sk_{R_i}), and it publishes the public key pk_{R_i} while keeping the private key sk_{R_i} secret. After that, receiver R_i computes its trapdoor T_i corresponding to identity ID_i with private key sk_{R_i}, and sends T_i to gateway W. Every time gateway W receives a trapdoor T_i, it will add T_i to the trapdoor list L_T. When gateway W receives an outside ciphertext, it checks whether this message is for some inside receivers ($R_i \in \mathcal{R}$) with its private key sk_G and the trapdoor list L_T. If so, gateway W broadcasts this ciphertext; otherwise, it rejects and outputs $\perp$.

Our gateway-based verifiable and anonymous broadcast encryption scheme consists of the following seven algorithms.

1. Setup(λ): This algorithm takes a security parameter λ as input. It chooses two groups $G, \hat{G}$ of prime order $p \geq 2^\lambda$. It constructs a bilinear pairing $\hat{e} : G \times \hat{G} \rightarrow G_T$. It defines a hash function $H_1 : \{0,1\}^* \rightarrow \hat{G}$. It outputs $params = (G, \hat{G}, G_T, q, \hat{e}, g, \hat{g}, H_1)$ as the public parameters where $g \in G$.

2. Make-GKey($params$): This algorithm takes the public parameters $params$ as input. It chooses $x \in Z_p^*$ uniformly at random and computes $X = g^x$. It outputs the public and private key pair (pk_G, sk_G) = (X, x) for gateway W.

3. Make-RKey($params$): This algorithm takes the public parameters $params$ as input. For $i = 1, \ldots, n$, it chooses $y_i \in Z_p^*$ uniformly at random and computes $Y_i = g^{y_i}$. It outputs the public and private key pair (pk_{R_i}, sk_{R_i}) = (Y_i, y_i) for receiver R_i.

4. Trapdoor($params$, sk_{R_i}, ID_i): This algorithm takes the public parameters $params$, and the secret key sk_{R_i} and the identity token ID_i of receiver R_i as input. It computes $T_i = H_1(ID_i)^{y_i}$ where $i = 1, \ldots, n$. It outputs T_i as the trapdoor for receiver R_i.

5. Encrypt($params$, M, pk_G, $\{pk_{R_i}, ID_i\}_{i \in \mathcal{L}}$): This algorithm takes the public parameters $params$, the message $M \in G_T$, the public key pk_G of gateway W,

and the set of recipients' public keys and identity tokens $\{Y_i, ID_i\}_{i\in\mathcal{L}}$ as input. For all $i \in \mathcal{L}$, it chooses $r \in Z_p^*$, and computes

$$C_{1,i} = M \cdot \hat{e}(Y_i, \hat{g})^r, \quad C_2 = g^r, \quad C_{3,i} = \hat{e}(X, \hat{g})^r \cdot \hat{e}(Y_i, H_1(ID_i))^r.$$

It outputs $C = (\{C_{1,i}\}_{i\in\mathcal{L}}, C_2, \{C_{3,i}\}_{i\in\mathcal{L}})$ as the ciphertext.

6. Verify($params$, sk_G, L_T, C): This algorithm takes the public parameters $params$, the secret key sk_G of gateway W, the trapdoor list L_T and a ciphertext C as input. It parses the ciphertext as $C_{1,i}$, C_2, $C_{3,i}$ for all $i \in \mathcal{L}$. Next, it checks whether there exists a trapdoor $T \in L_T$ such that $\hat{e}(C_2, \hat{g}^x \cdot T) = C_{3,i}$ for all $i \in \mathcal{L}$. If so, it outputs the ciphertext C.

7. Decrypt($params$, sk_{R_i}, T_i, C): This algorithm takes the public parameters $params$, the secret key sk_{R_i} and the trapdoor T_i of receiver R_i and the ciphertext C as input. If receiver R_i is a privileged receiver, it outputs $M = C_{1,i} \cdot \hat{e}(C_2, \hat{g}^{-y_i})$. Otherwise, it outputs a failure symbol $\perp$.

Efficiency. Our scheme achieves $O(1)$-size public keys and $O(n)$-size ciphertexts and constant-size private keys. Note that the ciphertext is linear in the size of S not in the maximum number of decryption keys that can be distributed. Besides, because $\hat{e}(Y_i, \hat{g})$, $\hat{e}(X, \hat{g})$ and $\hat{e}(Y_i, H_1(ID_i))$ can be pre-computed, pairing computations are greatly reduced. In our scheme, encryption needs no pairing computation and $2 \cdot t + 2$ exponentiation computation while decryption needs one pairing computation and one exponentiation computation.

3.2.4 Security Analysis

We present the security reduction of our gateway-based verifiable and anonymous broadcast encryption scheme by showing that it is secure under the games previously defined.

In our scheme, public keys of receiver R_i and gateway W will be made known to all the involved parties, while identity token ID_i of receiver R_i generated by the AC is known to receiver R_i and the outside sender. Private key sk_{R_i} of receiver R_i generated by receiver R_i is only known to receiver R_i, private key sk_G of gateway W generated by the AC is known to gateway W, and trapdoor T_i of receiver R_i generated by receiver R_i is known to gateway W and receiver R_i. Note that only

receiver R_i knows the correlations between its public key pk_{R_i}, private key sk_{R_i} and trapdoor T_i.

Theorem 3.1 *The above scheme is confidential assuming that the* (t, ϵ)*-Decisional* X*-BDH assumption holds in* $(G, \hat{G})$.

Proof. Suppose there exists a (t, ϵ)-algorithm $\mathcal{A}$ against the confidentiality of our gateway-based verifiable and anonymous broadcast scheme. Algorithm $\mathcal{A}$ may get help from gateway W and q_T corrupted receivers. The former will give its public and private key pair (pk_G, sk_G) and answer the Gateway_Corrupt query on the set of all the public keys in the system PK, and the latter will answer the Receiver_Corrupt query on pk_{R_i}.

Thus, we construct an algorithm $\mathcal{B}$ that solves the (t, ϵ)-Decisional X-BDH problem. Specifically, algorithm $\mathcal{B}$ is given $(g, g^a, \{g^{b_i}\}_{i\in[n]}, \hat{g}, \hat{g}^a, \hat{g}^c, \mathcal{O}_{DL})$ and the definition of G, $\hat{G}$, G_T together with the bilinear map $\hat{e}$ as the problem instance. Recall that at some point algorithm $\mathcal{B}$ has to output a set $\mathcal{L}^* \subset [n]$ and receives a set $\{Z_k\}_{k\in\mathcal{L}^*}$. We defer the description of this step to later. Based on the problem instance, algorithm $\mathcal{B}$ creates the system parameters.

- **Initialization.** The system parameters are generated based on the problem instance. To generate the system parameters, algorithm $\mathcal{B}$ sets $\hat{g}' = \hat{g}^c$, and then
  - outputs $params = (G, \hat{G}, G_T, q, \hat{e}, g, \hat{g}', H_1)$ as the public parameters, where H_1 is a random oracle controlled by algorithm $\mathcal{B}$.
  - generates n identity tokens $ID_1, \ldots, ID_n$, and sets a corruption list L_{CR} which is initially empty.
  - chooses $x \in Z_p^*$ uniformly at random, computes $X = g^x$, and outputs $(pk_G, sk_G) = (X, x)$ as the public and private key pair of gateway W.
  - outputs $PK = \{g^{b_1}, \ldots, g^{b_n}\} = \{Y_1, \ldots, Y_n\}$ as the public keys of n potential receivers $R_1, \ldots, R_n$.
- **Phase 1.** Algorithm $\mathcal{A}$ queries pk_{R_i} to the Receiver_Corrupt oracle. To answer it, algorithm $\mathcal{B}$
  - issues a query to oracle $\mathcal{O}_g^{\mathsf{DL}}$ on input Y_i, and obtains the value b_i such that $Y_i = g^{b_i}$.

- computes the trapdoor $T_i = H_1(ID_i)^{b_i}$.
- outputs the corresponding private key, identity token and trapdoor in a 3-tuple $(sk_{R_i}, ID_i, T_i) = (b_i, ID_i, H_1(ID_i)^{b_i})$ as the answer.
- updates the corruption list L_{CR} with $(pk_{R_i}, sk_{R_i}, ID_i, T_i)$.

- **Phase 2.** Algorithm $\mathcal{A}$ queries PK to the Gateway_Corrupt oracle. To answer it, algorithm $\mathcal{B}$
  - chooses $r_i \in Z_p^*$ uniformly at random, and computes $T_i = \hat{g}'^{r_i}$.
  - outputs all the trapdoors $\{T_1, \ldots, T_n\}$ to algorithm $\mathcal{A}$.

  Note that algorithm $\mathcal{B}$ has implicitly assumed $H_1(ID_i) = \hat{g}'^{r_i/b_i}$. If algorithm $\mathcal{A}$ queries ID_i without issuing a Receiver_Corrupt query on Y_i, algorithm $\mathcal{B}$ aborts. This happens with negligible probability since the value ID_i is hidden from algorithm $\mathcal{A}$ if the corrupt query on Y_i is not issued. Otherwise, algorithm $\mathcal{B}$ returns $\hat{g}'^{r_i/b_i}$ as the hash value of ID_i. Thus, simulation is perfect in the random oracle model.

- **Challenge.** Algorithm $\mathcal{A}$ outputs two messages $M_0^*, M_1^* \in G_T$ and a public key set $PK^* \subset PK$ where PK^* is of size l, with the restriction that $pk_{R_i} \in PK^*$ does not exist in list L_{CR}. At this stage algorithm $\mathcal{B}$ submits $\mathcal{L}^*$[2], and receives $\{Z_k\}_{k\in\mathcal{L}^*}$. The task of algorithm $\mathcal{B}$ is to distinguish if $Z_k = \hat{e}(g,\hat{g})^{ab_kc}$ for all $k \in \mathcal{L}^*$. To create the challenge ciphertext with this problem instance, algorithm $\mathcal{B}$
  - chooses an identity token set $S^* \subset \{ID_1, \ldots, ID_n\}$ of size l.
  - selects a random bit $\gamma \in \{0,1\}$, sets $C_2^* = g^a$, and computes, for all $i \in \mathcal{L}^*$,
  $$C_{1,i}^* = M_\gamma^* \cdot Z_i, \quad C_{3,i}^* = \hat{e}(g^a, \hat{g}^c)^x \cdot \hat{e}(g^a, \hat{g}^c)^{r_i}.$$
  - responds with the challenge ciphertext $C^* = (\{C_{1,i}^*\}_{i\in\mathcal{L}^*}, C_2^*, \{C_{3,i}^*\}_{i\in\mathcal{L}^*})$.

  For all $k \in \mathcal{L}^*$, if $Z_k = \hat{e}(g,\hat{g})^{ab_kc}$, we have
  $$\begin{aligned} C_{1,k}^* &= M_\gamma^* \cdot Z_k = M_\gamma^* \cdot \hat{e}(g^{b_k}, \hat{g}^c)^a = M_\gamma^* \cdot \hat{e}(Y_i, \hat{g}')^a, \\ C_{3,k}^* &= \hat{e}(g^a, \hat{g}^c)^x \cdot \hat{e}(g^a, \hat{g}^c)^{r_k} = \hat{e}(g^x, \hat{g}')^a \cdot \hat{e}(g^{b_k}, (\hat{g}^c)^{r_k/b_k})^a \\ &= \hat{e}(X, \hat{g}')^a \cdot \hat{e}(Y_k, \hat{g}'^{r_k})^a = \hat{e}(X, \hat{g}')^a \cdot \hat{e}(Y_k, H_1(ID_k))^a, \end{aligned}$$

[2]For notational convenience, let $\mathcal{L}^*$ be an index set such that $\{Y_i\}_{i\in\mathcal{L}^*} = PK^*$.

where $ID_k \in S^*$, $Y_k \in PK^*$.

Hence, when $Z_k = \hat{e}(g, \hat{g})^{ab_k c}$, meaning that the input of algorithm $\mathcal{B}$ is sampled from $\mathcal{P}_{XBDH}$, then C^* is a valid encryption of M^*_γ under the public key set PK^*_γ chosen by algorithm $\mathcal{A}$. On the other hand, when Z_k is uniform and independent in G_T, meaning that the input of algorithm $\mathcal{B}$ is sampled from $\mathcal{R}_{XBDH}$, then C^* is independent of γ in the view of algorithm $\mathcal{A}$.

- **Phase 3.** Algorithm $\mathcal{A}$ continues to adaptively query the Receiver_Corrupt oracle on pk_{R_i} with the restriction that $pk_{R_i} \notin PK^*$. Algorithm $\mathcal{B}$ responds as in Phase 1.

- **Guess.** Finally, algorithm $\mathcal{A}$ outputs a guess $\gamma' \in \{0, 1\}$. If $\gamma = \gamma'$, algorithm $\mathcal{B}$ outputs 1 meaning that it wins the game. Otherwise, it outputs 0.

If $Z_k = \hat{e}(g, \hat{g})^{ab_k c}$, the simulation is perfect and algorithm $\mathcal{A}$ must satisfy $|\Pr[\gamma = \gamma'] | = 1/2 + \epsilon$. On the other hand, if $Z_k \in G_T$, the challenge ciphertext C^* contains no information on γ and thus $\Pr[\gamma = \gamma'] = 1/2$. The overall probability that algorithm $\mathcal{B}$ solves the X-DBDH problem correctly is

$$\begin{aligned}\Pr[\mathcal{B}(g, g^a, g^c, \hat{g}, \hat{g}^a, \hat{g}^b, Z)] &= 1/2 \cdot (1/2 + \epsilon) + 1/2 \cdot 1/2 \\ &= 1/2 + \epsilon/2.\end{aligned}$$

This completes the proof of Theorem 1.

Theorem 3.2 *The above scheme is anonymous assuming that the* (t, ϵ)*-Decisional* X*-BDH assumption holds in* $(G, \hat{G})$.

Proof. Suppose there exists a (t, ϵ)-algorithm $\mathcal{A}$ against the confidentiality of our gateway-based verifiable and anonymous broadcast scheme. Algorithm $\mathcal{A}$ may get help from gateway W and q_T corrupted receivers. The former will give its public and private key pair (pk_G, sk_G) and answer the Gateway_Corrupt query on the set of all the public keys in the system PK, and the latter will answer the Receiver_Corrupt query on pk_{R_i}.

Thus, we construct an algorithm $\mathcal{B}$ that solves the (t, ϵ)-Decisional X-BDH problem. Specifically, algorithm $\mathcal{B}$ is given $(g, g^a, \{g^{b_i}\}_{i \in [n]}, \hat{g}, \hat{g}^a, \hat{g}^c, \mathcal{O}_{DL})$ and the definition of G, $\hat{G}$, G_T together with the bilinear map $\hat{e}$ as the problem instance. Recall that at some point algorithm $\mathcal{B}$ has to output a set $\mathcal{L}^* \subset [n]$ and receives a set $\{Z_k\}_{k \in \mathcal{L}^*}$. We defer the description of this step to later. Based on the problem instance, $\mathcal{B}$ creates the system parameters as follows.

- **Initialization.** The same as in Theorem 3.1.
- **Phase 1.** The same as in Theorem 3.1.
- **Phase 2.** The same as in Theorem 3.1.
- **Challenge.** Algorithm $\mathcal{A}$ outputs a message $M^* \in G_T$ and two public key sets PK_0^*, $PK_1^* \subset PK$ with the restriction that $pk_{R_i} \in PK_0^* \cup PK_1^*$ does not exist in list L_{CR}, where PK_0^*, PK_1^* are of the same size l. At this stage algorithm $\mathcal{B}$ submits $\mathcal{L}^{*3}$, and receives $\{Z_k\}_{k\in\mathcal{L}^*}$. The task of algorithm $\mathcal{B}$ is to distinguish if $Z_k = \hat{e}(g,\hat{g})^{ab_kc}$ for all $k \in \mathcal{L}^*$. To create the challenge ciphertext, algorithm $\mathcal{B}$
  - chooses two identity token sets $S_0^*, S_1^* \subset \{ID_1, \ldots, ID_n\}$ of equal size l.
  - selects a random bit $\gamma \in \{0,1\}$, sets $C_2^* = g_1 = g^a$, and computes, for all $i \in \mathcal{L}^*$,

$$C_{1,i}^* = M^* \cdot Z_i, \quad C_{3,i}^* = \hat{e}(g^a, \hat{g}^c)^x \cdot \hat{e}(g^a, \hat{g}^c)^{r_i}.$$

  - responds with the challenge ciphertext $C^* = (\{C_{1,i}^*\}_{i\in\mathcal{L}^*}, C_2^*, \{C_{3,i}^*\}_{i\in\mathcal{L}^*})$.

  For all $k \in \mathcal{L}^*$, if $Z_k = \hat{e}(g,\hat{g})^{ab_kc}$, we have

$$\begin{aligned} C_{1,k}^* &= M^* \cdot Z_k = M^* \cdot \hat{e}(g^{b_k}, \hat{g}^c)^a = M^* \cdot \hat{e}(Y_i, \hat{g}')^a, \\ C_{3,k}^* &= \hat{e}(g^a, \hat{g}^c)^x \cdot \hat{e}(g^a, \hat{g}^c)^{r_k} = \hat{e}(g^x, \hat{g}')^a \cdot \hat{e}(g^{b_k}, (\hat{g}^c)^{r_k/b_k})^a \\ &= \hat{e}(X, \hat{g}')^a \cdot \hat{e}(Y_k, \hat{g}'^{r_k})^a = \hat{e}(X, \hat{g}')^a \cdot \hat{e}(Y_k, H_1(ID_k))^a. \end{aligned}$$

  where $ID_k \in S_\gamma^*$, $Y_k \in PK_\gamma^*$.

  Hence, when $Z_k = \hat{e}(g,\hat{g})^{ab_kc}$, meaning that the input of algorithm $\mathcal{B}$ is sampled from $\mathcal{P}_{XBDH}$, then C^* is a valid encryption of M^* under the public key set PK_γ^* chosen by algorithm $\mathcal{A}$. On the other hand, when Z_k is uniform and independent in G_T, meaning that the input of algorithm $\mathcal{B}$ is sampled from $\mathcal{R}_{XBDH}$, then C^* is independent of γ in the view of algorithm $\mathcal{A}$.
- **Phase 3.** Algorithm $\mathcal{A}$ continues to adaptively query the Receiver_Corrupt oracle on pk_{R_i} with the restriction that $pk_{R_i} \notin PK_0^* \cup PK_1^*$. Algorithm $\mathcal{B}$ responds as in Phase 1.

[3]For notational convenience, let $\mathcal{L}^*$ be an index set such that $\{Y_i\}_{i\in\mathcal{L}^*} = PK_0^* \cup PK_1^*$.

- **Guess.** Finally, algorithm $\mathcal{A}$ outputs a guess $\gamma' \in \{0, 1\}$. If $\gamma = \gamma'$, algorithm $\mathcal{B}$ outputs 1 meaning that it wins the game. Otherwise, it outputs 0.

If $Z_k = \hat{e}(g, \hat{g})^{ab_k c}$, the simulation is perfect and algorithm $\mathcal{A}$ must satisfy $|\Pr[\gamma = \gamma'] | = 1/2+\epsilon$. On the other hand, if $Z_k \in G_T$, the challenge ciphertext C^* contains no information on γ and thus $\Pr[\gamma = \gamma'] = 1/2$. The overall probability that algorithm $\mathcal{B}$ solves the X-DBDH problem correctly is

$$\begin{aligned}\Pr[\mathcal{B}(g, g^a, g^c, \hat{g}, \hat{g}^a, \hat{g}^b, Z)] &= 1/2 \cdot (1/2 + \epsilon) + 1/2 \cdot 1/2 \\ &= 1/2 + \epsilon/2.\end{aligned}$$

This completes the proof of Theorem 2.

CCA Secure Scheme. The result of Boneh and Katz [BK05] can be applied to the above CPA secure scheme, and thus we can obtain a new scheme that is provably CCA secure in the random oracle model.

3.3 Verifiable and Anonymous Encryption under an Untrusted Gateway

In this section, we simply consider that there is one receiver in a verifiable and anonymous system, and achieve it under a stronger security model without the use of secret identity tokens.

3.3.1 Definition and Security Model

We firstly present a formal framework of verifiable and anonymous encryption under an untrusted gateway, and then we describe how an adversary will be allowed to interfere in the protocol with games between an adversary algorithm and a challenger algorithm.

Let $S = \{ID_1, \ldots, ID_n\}$ be the user set in the verifiable and anonymous encryption system, where ID_i ($i \in \{1, \ldots, n\}$) represents the identity information (name, student ID, etc.) of user ID_i. Our framework is specified by the following algorithms.

- Setup(λ) $\rightarrow$ ($params$, msk, (pk_G, sk_G)): Taking a security parameter λ as input, this algorithm outputs the common system parameters $params$, the

master key msk, the public key and private key pair (pk_G, sk_G) for gateway W.

Generally speaking, this algorithm is run by the KGC.

- Make-Partial-USKey($params$, msk, ID_i) $\rightarrow$ psk_i: Taking the common system parameters $params$, the master key msk and the identity information ID_i as input, this algorithm outputs a partial private key psk_i for user ID_i.

  Generally speaking, this algorithm is run by the KGC and its output is sent to user ID_i through a confidential and authentic channel.

- Set-USValue($params$, ID_i) $\rightarrow$ x_i: Taking the common system parameters $params$ and the identity information ID_i as input, this algorithm outputs a secret value x_i for user ID_i.

- Make-USKey($params$, psk_i, x_i) $\rightarrow$ sk_i: Taking the common system parameters $params$, the partial private key psk_i and the secret value x_i as input, this algorithm outputs a (complete) private key sk_i for user ID_i.

- Make-UPKey($params$, x_i) $\rightarrow$ pk_i: Taking the common system parameters $params$ and the secret value x_i as input, this algorithm outputs a public key pk_i for user ID_i.

  Both Make-USKey and Make-UPKey are run by user ID_i itself, after running Set-USValue, and they share the same secret value x_i. Separating them means that there is no need for a temporal requirement on the generation of public and private keys in the scheme. Usually, user ID_i is the only one in possession of x_i and sk_i, and x_i will be chosen at random from a suitable and large set.

- Make-Trapdoor($params$, x_i) $\rightarrow$ T_i: Taking the common parameters $params$ and the secret value x_i as input, this algorithm outputs a trapdoor T_i for user ID_i.

  Each user sends its own trapdoor through a secure channel to gateway W, which then stores all the received trapdoors in a trapdoor list L_T.

- Encrypt($params$, ID_i, pk_i, pk_G, M) $\rightarrow$ C: Taking the common system parameters $params$, the identity identity $ID_i \in S$ with the corresponding public key pk_i, the public key pk_G of gateway W and a message M in the message space as input, this algorithm outputs a ciphertext C.

- Verify($params$, L_T, sk_G, C) $\rightarrow C$: Taking the common system parameters $params$, the trapdoor list L_T, the private key sk_G of gateway W and the ciphertext C as input, this algorithm outputs a ciphertext C in case of success or $\perp$ in case of failure.

- Decrypt($params$, sk_i, C) $\rightarrow M$: Taking the common system parameters $params$, the private key sk_i of user ID_i and the ciphertext C as input, this algorithm outputs a message M for a privileged receiver or $\perp$ for a non-privileged receiver.

We say that a verifiable and anonymous system under an untrusted gateway is correct, meaning that for all $ID_i \in S$, if ($params$, msk, (pk_G, sk_G)) $\leftarrow$ Setup(λ), $psk_i \leftarrow$ Make-Partial-USKey($params$, msk, ID_i), $x_i \leftarrow$ Set-USValue($params$, ID_i), $sk_i \leftarrow$ Make-USKey($params$, psk_i, x_i), $pk_i \leftarrow$ Make-UPKey($params$, x_i), $T_i \leftarrow$ Make-Trapdoor($params$, x_i), $C \leftarrow$ Encrypt($params$, ID_i, pk_i, pk_G, M), $C \leftarrow$ Verify($params$, L_T, sk_G, C), then $M =$ Decrypt($params$, sk_i, C).

In our model, since KGC is a trusted third party, we do not allow the adversaries to have access to the master key, but we allow them to request partial private keys, or private keys, or both, for identities of their choice. The following is a list of the queries that an adversary algorithm $\mathcal{A}$ against verifiable and anonymous encryption may carry out. We define a challenger algorithm $\mathcal{B}$ to respond to these queries.

1. Make-Partial-USKey for user ID_i: Algorithm $\mathcal{B}$ responds by running algorithm Make-Partial-USKey to generate the partial private key sk_i for user ID_i.

2. Make-USKey for user ID_i: Algorithm $\mathcal{B}$ responds by running algorithm Make-USKey to generate the private key sk_i for user ID_i (first running Set-USValue for user ID_i if necessary).

3. Make-UPKey for user ID_i: We assume that public keys are available to algorithm $\mathcal{A}$. When receiving a public key request for user ID_i, algorithm $\mathcal{B}$ responds by running algorithm Set-UPKey to generate the public key pk_i for user ID_i (first running Set-USValue for user ID_i if necessary).

4. Make-Trapdoor for user ID_i: Algorithm $\mathcal{B}$ responds by running algorithm Make-Trapdoor to generate the trapdoor T_i for user ID_i.

Confidentiality and Anonymity. It is formalized by the indistinguishability game, and the adversary has to guess which plaintext and which identity has been encrypted in the challenge ciphertext. Note that we provide the adversary with the public and private key pair of gateway W, the trapdoor list and the private information of some non-privileged users, which models a collusion between gateway W and the corrupted users.

We define a security model for verifiable and anonymous encryption to ensure the confidentiality of the plaintext and the anonymity of the privileged user, which we call ANON-IND-CPA security. More precisely, confidentiality and anonymity are defined using a game between an adversary algorithm $\mathcal{A}$ and a challenger algorithm $\mathcal{B}$ such that algorithm $\mathcal{A}$ cannot distinguish a ciphertext decrypted to one plaintext under one identity from a ciphertext decrypted to another plaintext under another identity.

1. **Initialization.** Algorithm $\mathcal{B}$ runs the Setup algorithm to obtain the public parameters $params$ and the master key msk. Then, algorithm $\mathcal{B}$ generates the public and private key pair (pk_G, sk_G) for gateway W. Also, algorithm $\mathcal{B}$ generates the trapdoor list $L_T = \{T_1, \ldots, T_n\}$, which stores the trapdoors of all the users in the system.

   Algorithm $\mathcal{B}$ gives the public parameters $params$, the public and private key pair (pk_G, sk_G), and the trapdoor list L_T to algorithm $\mathcal{A}$ while keeping the master key msk to itself.

2. **Query Phase 1.** Algorithm $\mathcal{A}$ issues a sequence of queries, each query being either a Make-Partial-USKey query, a Make-USKey query, a Make-UPKey query or a Make-Trapdoor query on $ID_i \in S$. These queries may be issued adaptively.

3. **Challenge.** When algorithm $\mathcal{A}$ decides that Phase 1 is over, it outputs two messages M_0^*, M_1^* of the same size, two users ID_0^*, $ID_1^* \in S$ on which it wishes to be challenged. The only constraint is that ID_0^*, ID_1^* do not appear in Phase 1. To generate the challenge ciphertext, algorithm $\mathcal{B}$ chooses d, $e \in \{0, 1\}$, and runs the encryption algorithm on M_d^*, ID_e^* to obtain the ciphertext C^*. It sends C^* as the ciphertext to algorithm $\mathcal{A}$.

4. **Query Phase 2.** Algorithm $\mathcal{A}$ continues to issue Make-Partial-USKey queries,

Make-USKey queries, Make-UPKey queries and Make-Trapdoor queries on $ID_i \in S \setminus \{ID_0^*, ID_1^*\}$, as in Phase 1.

5. **Guess.** Algorithm $\mathcal{A}$ outputs its guess d', $e' \in \{0,1\}$ for d, e, and it wins the game if $d = d'$ and $e = e'$.

We refer to such an adversary algorithm $\mathcal{A}$ as an ANON-IND-CPA adversary. We define the advantage of the adversary algorithm $\mathcal{A}$ in attacking a verifiable and anonymous encryption scheme $\prod$ = (Setup, Make-Partial-USKey, Set-USValue, Make-USKey, Make-UPKey, Make-Trapdoor, Encrypt, Verify, Decrypt) as

$$\mathbf{Adv}_{\prod,\mathcal{A}} = |\Pr[d = d' \wedge e = e'] - 1/4|.$$

The probability is over the random bits used by the challenger and the adversary.

There is another stronger version of security, the ANON-IND-CCA security, where the adversary is not only allowed to issue the above queries adaptively, but also allowed to issue decryption queries.

3.3.2 Proposed Scheme

Based on the techniques in [ARP03], we give a secure verifiable and anonymous encryption scheme. Let $S = \{ID_1, \ldots, ID_n\}$ be the recipient set in the system. Let $\hat{e} : G \times \hat{G} \to G_T$ be a bilinear map over bilinear groups G, $\hat{G}$ of prime order p with generators $g \in G$, $\hat{g} \in \hat{G}$ respectively. Our verifiable and anonymous encryption scheme in asymmetric bilinear maps consists of the following six algorithms.

- Setup(λ): This algorithm takes a security parameter λ as input. It runs as follows to generate the system parameters.

  1. Selects s, $\beta \in Z_p^*$ uniformly at random, and computes $g_1 = g^s$, $g_2 = \hat{g}^s$, $g_3 = \hat{g}^\beta$.
  2. Defines a hash function $H_1 : \{0,1\}^* \to \hat{G}$.
  3. Chooses $x \in Z_p^*$ uniformly at random, computes $X = g^x$, and sets (pk_G, sk_G) = (X, x) as the public key and private key pair for gateway W.

  Thus, the common system parameters are $params$ = (g, g_1, $\hat{g}$, g_2, g_3, H_1). The master key is $msk = s \in Z_p^*$.

- Make-Partial-USKey($params$, msk, ID_i): This algorithm takes the common system parameters $params$, the master key msk and the identity information ID_i as input. It outputs $psk_i = H_1(ID_i)^s$ as the partial private key associated for user ID_i.

- Set-Secret-USValue($params$, ID_i): This algorithm takes the common system parameters $params$ and an identity ID_i as input. It selects $x_i \in Z_p^*$ uniformly at random, and outputs x_i as the secret value for user ID_i.

- Make-USKey($params$, psk_i, x_i): This algorithm takes the common system parameters $params$, the partial private key psk_i and the secret value x_i as input. It outputs a private key $sk_i = psk_i^{x_i} = H_1(ID_i)^{sx_i}$ for user ID_i.

- Make-UPKey($params$, x_i): This algorithm takes the common system parameters $params$ and the secret value x_i as input. It outputs a public key $pk_i = (X_i, Y_i)$ for user ID_i, where $X_i = g^{x_i}$ and $Y_i = g_1^{x_i} = g^{sx_i}$.

  Note that the validity of the public key pk_i can be checked by the equation $\hat{e}(X_i, g_2) = \hat{e}(Y_i, \hat{g})$.

- Make-Trapdoor($params$, x_i): This algorithm takes the common system parameters $params$ and the secret value x_i as input.

  It outputs a trapdoor $T_i = (\hat{X}_i, \hat{Y}_i)$ for user ID_i, where $\hat{X}_i = \hat{g}^{x_i}$ and $\hat{Y}_i = g_2^{x_i} = \hat{g}^{sx_i}$.

  Note that gateway W can check the validity of T_i by the equations $\hat{e}(X_i, \hat{g}) = \hat{e}(g, \hat{X}_i)$ and $\hat{e}(Y_i, \hat{g}) = \hat{e}(g, \hat{Y}_i)$, but it is not aware of user ID_i and the corresponding trapdoor T_i. We assume that gateway W stores all T_i in a trapdoor list L_T.

- Encrypt($params$, ID_i, pk_i, pk_G, M): This algorithm takes the common parameters $params$, the identity information $ID_i \in S$ with the corresponding public key $pk_i = (X_i, Y_i)$, the public key pk_G of gateway W and a message $M \in G_T$ as input. It runs as follows to generate the ciphertext.

  1. Chooses $r \in Z_p^*$ uniformly at random, computes

     $$C_1 = g^r, \quad C_2 = M \cdot \hat{e}(Y_i, H_1(ID_i))^r, \quad C_3 = \hat{e}(X, g_3)^r \cdot \hat{e}(Y_i, \hat{g})^r.$$

  2. Outputs the ciphertext $C = (C_1, C_2, C_3)$.

- Verify($params$, L_T, sk_G, C): This algorithm takes the common parameters $params$, the trapdoor list L_T, the private key sk_G of gateway W and the ciphertext C as input. It parses the ciphertext C as (C_1, C_2, C_3), and checks whether

$$\hat{e}(g_1, \hat{X}_i) = \hat{e}(g, \hat{Y}_i), \quad C_3 = \hat{e}(C_1, g_3)^x \cdot \hat{e}(C_1, \hat{Y}_i).$$

  If both of the two equations hold, it outputs the ciphertext C. Otherwise, it outputs a failure symbol $\perp$.

- Decrypt($params$, sk_i, C): This algorithm takes the common system parameters $params$, the private key sk_i of user ID_i and the ciphertext C as input. If user ID_i is a privileged receiver, it outputs

$$M = C_2/\hat{e}(C_1, sk_i) = C_2/\hat{e}(C_1, H_1(ID_i)^{sx_i}).$$

  Otherwise, it outputs a failure symbol $\perp$.

Our construction also achieves traceability, i.e., the PKG can reveal the identities of the privileged users if necessary.

- Query-Trace($params$, C): This algorithm takes the common system parameters $params$ and the ciphertext C as input. It outputs a trapdoor T_i via the Verify algorithm.

- Trace($params$, pk_i, psk_i, T_i): This algorithm takes the common parameters $params$, the public key pk_i, the partial private key psk_i and the trapdoor T_i as input. It checks whether

$$\hat{e}(g, \hat{Y}_i) = \hat{e}(Y_i, \hat{g}), \quad \hat{e}(X_i, H_1(ID_i)^s) = \hat{e}(Y_i, H_1(ID_i)).$$

  If such ID_i can be found, it outputs ID_i. Otherwise, it outputs a failure symbol $\perp$.

3.3.3 Security Analysis

We present the security reduction of our verifiable and anonymous encryption scheme by showing that it is confidential and anonymous under the Decisional BDH assumption in the random oracle model.

Theorem 3.3 *The above scheme is confidential and anonymous in the random oracle model assuming that the* (t, ϵ)*-Decisional BDH assumption holds in* $(G, \hat{G})$.

Proof. Suppose there exists a (t, ϵ)-algorithm $\mathcal{A}$ that breaks the confidentiality and anonymity of our verifiable and anonymous scheme. We can construct an algorithm $\mathcal{B}$ that solves the (t, ϵ)-Decisional BDH problem, which is given as input a random tuple $(g, g^a, g^c, \hat{g}, \hat{g}^a, \hat{g}^b, Z)$, and outputs 1 ($Z$ is $\hat{e}(g, \hat{g})^{abc}$) or 0 ($Z$ is a random element in G_T).

- **Initialization.** To generate the system parameters, algorithm $\mathcal{B}$
  - sets $g_1 = g^a$, $g_2 = \hat{g}^a$, $g_3 = \hat{g}^b$, and outputs $params = (g, g_1, \hat{g}, g_2, g_3, H_1)$ as the public parameters, where H_1 is a random oracle controlled by algorithm $\mathcal{B}$.
  - chooses $x \in Z_p^*$ uniformly at random, computes $X = g^x$, and outputs $(pk_G, sk_G) = (X, x)$ as the public and private key pair of gateway W.
  - for each user $ID_i \in S$, chooses $x_i \in Z_p^*$, and outputs $T_i = (\hat{X}_i, \hat{Y}_i) = (\hat{g}^{x_i}, g_2^{x_i})$ as a trapdoor.

  Algorithm $\mathcal{B}$ sends $params$, (pk_G, sk_G) and the trapdoor list $L_T = \{T_1, \ldots, T_n\}$ to algorithm $\mathcal{A}$.

- H_1**-queries.** At any time algorithm $\mathcal{A}$ can query the random oracle on ID_i. To respond to these queries, algorithm $\mathcal{B}$ maintains a list L_H of tuples $(ID_i, H_1(ID_i), r_i, x_i, psk_i, sk_i, pk_i, T_i)$ which is initially empty. When algorithm $\mathcal{A}$ issues a hash query on identity ID_i, if (ID_i, r_i) already exists in list L_H, algorithm $\mathcal{B}$ responds with $H_1(ID_i)$; otherwise, algorithm $\mathcal{B}$ chooses $r_i \in Z_p^*$ uniformly at random, outputs $H_1(ID_i) = g_3^{r_i}$, and completes list L_H with $(ID_i, g_3^{r_i}, r_i, \cdot, \cdot, \cdot, \cdot, \cdot)$.

- **Phase 1.** Algorithm $\mathcal{A}$ queries $ID_i \in S$ to a series of oracles, each of which is either Make-Partial-USKey, Make-USKey, Make-UPKey, or Make-Trapdoor.
  - Make-Partial-USKey: Algorithm $\mathcal{B}$ runs the following steps to answer.
    1. If (ID_i, r_i, psk_i) already exists in list L_H, algorithm $\mathcal{B}$ responds with $psk_i = g_1^{r_i}$.

2. Otherwise, algorithm $\mathcal{B}$ chooses $r_i \in Z_p^*$ uniformly at random, outputs $psk_i = g_1^{r_i}$, and completes list L_H with (ID_i, $\cdot$, r_i, $\cdot$, psk_i, $\cdot$, $\cdot$, $\cdot$).

To see psk_i is a correct partial private key with respect to ID_i. Let $r_i' = r_i/b$, thus we have that

$$psk_i = g_1^{r_i} = g_3^{ar_i'} = (g_3^{r_i'})^a = H_1(ID_i)^a.$$

– Make-USKey: Algorithm $\mathcal{B}$ performs the following steps to respond.

1. If (ID_i, r_i, x_i, sk_i) already exists in list L_H, algorithm $\mathcal{B}$ responds with $sk_i = g_1^{r_i x_i}$.
2. Otherwise, algorithm $\mathcal{B}$ chooses r_i, $x_i \in Z_p^*$, outputs $sk_i = g_1^{r_i x_i}$, and completes list L_H with (ID_i, $\cdot$, r_i, x_i, $\cdot$, sk_i, $\cdot$, $\cdot$).

To see sk_i is a correct private key with respect to ID_i. Let $r_i' = r_i/b$, thus we have

$$sk_i = g_1^{r_i x_i} = (g_3^{ar_i'})^{x_i} = H_1(ID_i)^{ax_i}.$$

– Make-UPKey: Algorithm $\mathcal{B}$ executes the following procedure to answer.

1. If (ID_i, x_i, pk_i) already exists in list L_H, algorithm $\mathcal{B}$ responds with $pk_i = (X_i, Y_i) = (g^{x_i}, g_1^{x_i})$.
2. Otherwise, algorithm $\mathcal{B}$ chooses $x_i \in Z_p^*$ uniformly at random, outputs $pk_i = (X_i, Y_i) = (g^{x_i}, g_1^{x_i})$, and completes list L_H with (ID_i, $\cdot$, r_i, x_i, $\cdot$, $\cdot$, pk_i, $\cdot$).

– Make-Trapdoor: Algorithm $\mathcal{B}$ runs as follows to respond.

1. If (ID_i, x_i, T_i) already exists in list L_H, algorithm $\mathcal{B}$ responds with $T_i = (\hat{X}_i, \hat{Y}_i) = (\hat{g}^{x_i}, g_2^{x_i})$.
2. Otherwise, algorithm $\mathcal{B}$ chooses $x_i \in Z_p^*$ uniformly at random, outputs $T_i = (\hat{X}_i, \hat{Y}_i) = (\hat{g}^{x_i}, g_2^{x_i})$, and completes list L_H with (ID_i, $\cdot$, $\cdot$, x_i, $\cdot$, $\cdot$, $\cdot$, T_i).

- **Challenge.** Algorithm $\mathcal{A}$ outputs two messages M_0^*, $M_1^* \in G_T$, two users ID_0^*, $ID_1^* \in S$, where ID_0^*, ID_1^* do not appear in Phase 2. Algorithm $\mathcal{B}$ executes as follows to answer.

– Responds to the H_1-queries to obtain r_i^*, for $i \in \{0, 1\}$. Let $(ID_i^*, g_3^{r_i^*}, r_i^*, \cdot, \cdot, \cdot, \cdot, \cdot)$ be the corresponding tuple on list L_H.

– Selects $d, e \in \{0, 1\}$, sets $C_1^* = g^c$, and computes

$$C_2^* = M_d^* \cdot Z, \quad C_3^* = \hat{e}(g^c, g_3)^x \cdot \hat{e}(g^c, g_2)^{x_e^*},$$

where $x_e^* \in \{x_0^*, x_1^*\}$.

– Outputs the challenge ciphertext $C^* = (C_1^*, C_2^*, C_3^*)$.

To see this, let $x_e^* = 1/r_e^*$, we have that

$$\begin{aligned} C_2^* &= M_d^* \cdot Z = M_d^* \cdot \hat{e}(g^{a \cdot x_e^*}, \hat{g}^{b \cdot r_e^*})^c \\ &= M_d^* \cdot \hat{e}(Y_e^*, H_1(ID_e^*))^c, \\ C_3^* &= \hat{e}(g^c, g_3)^x \cdot \hat{e}(g^c, g_2)^{x_e^*} = \hat{e}(X, g_3)^c \cdot \hat{e}(g^{a \cdot x_e^*}, \hat{g})^c \\ &= \hat{e}(X, g_3)^c \cdot \hat{e}(Y_e^*, \hat{g})^c. \end{aligned}$$

Hence, when Z equals $\hat{e}(g, \hat{g})^{abc}$, then C^* is a valid encryption of M_d^* of ID_e^* chosen by algorithm $\mathcal{A}$. On the other hand, when Z is uniform and independent in G_T, then C^* is independent of γ in the view of algorithm $\mathcal{A}$.

- **Phase 2.** Algorithm $\mathcal{A}$ continues to adaptively query $ID_i \in S \setminus \{ID_0^*, ID_1^*\}$ to oracles Make-Partial-USKey, Make-USKey, Make-UPKey or Make-Trapdoor. Algorithm $\mathcal{B}$ responds as in Phase 1.

- **Guess.** Finally, algorithm $\mathcal{A}$ outputs d', $e' \in \{0, 1\}$. If $d = d'$ and $e = e'$, algorithm $\mathcal{B}$ outputs 1, meaning that algorithm $\mathcal{A}$ wins the game. Otherwise, algorithm $\mathcal{B}$ outputs 0.

We can see that if $Z = \hat{e}(g, \hat{g})^{abc}$, the simulation is exactly the same as the real attack, and algorithm $\mathcal{A}$ will output $d' = d$ and $e' = e$ with probability $1/4 + \epsilon$. Otherwise, if Z is uniformly random, then the advantage of algorithm $\mathcal{A}$ is nil and it outputs $d' = d$ and $e' = e$ with probability $1/4$. Thus, we have that the probability of algorithm $\mathcal{B}$ in solving the decisional BDH problem is

$$\begin{aligned} \Pr[\mathcal{B}(g, g^a, g^c, \hat{g}, \hat{g}^a, \hat{g}^b, Z)] &= 1/2 \cdot (1/4 + \epsilon) + 1/2 \cdot 1/4 \\ &= 1/4 + \epsilon/2. \end{aligned}$$

This completes the proof of Theorem 1.

CCA Secure Scheme. The result of Boneh and Katz [BK05] can be applied to the above CPA secure verifiable and anonymous encryption scheme, and thus we can obtain a verifiable and anonymous encryption scheme that is provably CCA secure in the random oracle model.

3.4 Verifiable and Anonymous Encryption under an Untrusted Gateway with Stronger Security

In the security proof of the scheme in the previous section, the adversaries are not allowed to issue the public key query on user ID_i^* where ID_i^* is the identity in the challenge phase; otherwise, malicious users in the system can easily obtain the identity of the privileged user when colluding with the untrusted gateway W. To some extent, this constraint is not very reasonable, as in most cases the users are required to publish both their identities and public keys in a public bulletin. Therefore, this system cannot be widely used in the practical environment. In this section, based on a zero-knowledge proof of knowledge scheme, we propose a new scheme with stronger security in which the adversaries may query the public keys and the certificates of all the users in the system.

3.4.1 Definition

Let $S = \{ID_1, \ldots, ID_n\}$ be the user set in the verifiable and anonymous encryption system, where ID_i ($i \in \{1, \ldots, n\}$) represents the identity information (name, student ID, etc.) of user ID_i. The new definition is specified by six randomized algorithms: Setup, Make-UKey, Certificate, Encrypt, Verify and Decrypt.

- Setup(λ) $\rightarrow$ ($params$, msk, mpk): Taking a security parameter λ as input, this algorithm outputs the common system parameters $params$, the master secret key msk and the master public key mpk. Generally, this algorithm is run by the CA.

- Make-UKey($params$, ID_i) $\rightarrow$ (sk_i, pk_i): Taking the common system parameters $params$ and an identity ID_i as input, this algorithm outputs a private

and public key pair (sk_i, pk_i) for user ID_i. Definitely, this algorithm is run by user ID_i itself.

- Certificate($params$, ID_i, pk_i, msk) $\rightarrow Cert_i$: Taking the common parameters $params$, an identity ID_i, a public key pk_i and the master secret key msk as input, this algorithm outputs a certificate $Cert_i$ for user ID_i. Generally, this algorithm is run by the CA.

- Encrypt($params$, pk_i, $Cert_i$, M) $\rightarrow C$: Taking the common parameters $params$, a public key pk_i, a certificate $Cert_i$ and a message M in the message space as input, this algorithm outputs a ciphertext C via zero-knowledge proof of knowledge.

- Verify($params$, mpk, C) $\rightarrow C$: Taking the common system parameters $params$, and the ciphertext C as input, this algorithm outputs C in case of success or $\perp$ in case of failure. Obviously, this algorithm is run by gateway W.

- Decrypt($params$, sk_i, C) $\rightarrow M$: Taking the common parameters $params$, the private key sk_i of user ID_i and the ciphertext C as input, this algorithm outputs the message M in the case of being a privileged receiver or $\perp$ in the case of being a non-privileged receiver.

Note that to simplify the construction, here the key generation algorithm for gateway W is removed from the system, and we require users to send the trapdoors to gateway W via a secure channel rather than a public channel. As mentioned in [BSNS05, BSNS08], arming gateway W with a key pair is an efficient way to remove the secure channel; but this is not the emphasis of this section, so we remove it to make the scheme simpler.

We say that a verifiable and anonymous system under an untrusted gateway is correct, meaning that for all $ID_i \in S$, if ($params$, msk, mpk) $\leftarrow$ Setup(λ), (sk_i, pk_i) $\leftarrow$ Make-UKey($params$, ID_i), $Cert_i \leftarrow$ Certificate($params$, ID_i, pk_i, msk), $C \leftarrow$ Encrypt($params$, pk_i, $Cert_i$, M), $C \leftarrow$ Verify($params$, mpk, C), then $M =$ Decrypt($params$, sk_i, C).

Confidentiality and Anonymity. The definition of this security mostly follows that in Section 3.3 by replacing the queries in Phase 1 and 2 by Make-UKey and Certificate queries without the disallowance of querying the challenge identity ID^*. As the use of zero-knowledge proof of knowledge in the encryption phase to

yield the ciphertext is required, the preservation of privacy actually turns to the security implied by the zero-knowledge proof of knowledge system.

3.4.2 Construction

In this construction, we use a trusted certificate authority (CA) to issue certificates, which are actually signatures of the users' public keys generated by a CA with its master key, for users. Note that if a user accidentally reveals its secret key or an attacker actively compromises it, the user itself may request revocation of its certificate. Alternatively, the user's organization may request revocation if the user leaves the company or changes position and is no longer entitled to use the key.

In our verifiable and anonymous encryption system under the setting of asymmetric bilinear maps, the public keys and the identity information of the users in the system might be made available to any sender. To encrypt a message M to user ID_i, the sender makes use of its public key pk_i to compute a ciphertext C_i, and generates a signature of ($Cert_i$, C_i) based on a proof of knowledge scheme in [GMR89], where $Cert_i$ is the certificate of user ID_i.

Let $S = \{ID_1, \ldots, ID_n\}$ be the recipient set in the system. Let $\hat{e} : G \times T \to G_T$ be a bilinear map over bilinear groups G, T of prime order p with generators $g \in G$, $h \in T$ respectively. Let $H : \{0,1\}^* \to Z_p^*$ be a collision resistant hash function. Our verifiable and anonymous encryption scheme consists of the following six algorithms.

- Setup(λ): This algorithm takes a security parameter λ as input. It chooses random u, v, w, $z \in Z_p^*$, and computes

$$U = g^u, \quad V = h^v, \quad W = h^w, \quad Z = h^z.$$

  The common system parameters are $params = (g, h)$, the master public key is $mpk = (U, V, W, Z)$, and the master secret key is $msk = (u, v, w, z)$.

- Make-UKey($params$, ID_i): This algorithm takes the common system parameters $params$ and the identity information $ID_i \in T$ as input. It chooses a random $x_i \in Z_p^*$, and computes $Y_i = g^{x_i}$. It outputs a public and private key pair (pk_i, sk_i) = (Y_i, x_i) for user ID_i.

- Certificate($params$, ID_i, pk_i, msk): This algorithm takes the common parameters $params$, the identity information ID_i, the public key $pk_i = Y_i$ and the

master secret key $msk = (u, v, w, z)$ as input. It chooses a random $s \in Z_p^*$, and computes

$$R_i = g^s, \quad S_i = g^{z-sv} {Y_i}^{-w}, \quad T_i = (h \cdot {ID_i}^{-u})^{\frac{1}{s}}.$$

It outputs $Cert_i = (R_i, S_i, T_i)$ as a certificate for user ID_i.

Note that here we use the optimal structure-preserving signature proposed in [AGHO11], which is very efficient in asymmetric bilinear groups.

- Encrypt($params$, pk_i, $Cert_i$, M): This algorithm takes the common parameters $params$, the public key $pk_i = Y_i$, the certificate $Cert_i$ and a message $M \in \mathcal{M}$ as input. It runs as follows to generate the ciphertext.

  1. Chooses a random $r \in Z_p^*$, and computes

  $$C_1 = g^r, \quad C_2 = M \cdot {Y_i}^r.$$

  2. Generates a signature for $(Cert_i, C_1, C_2)$ based on the zero-knowledge proof of knowledge, which in fact is based on the non-interactive version of the Schnorr's proof system presented in [Sch90, Sch91].

     – Chooses random $\beta_0, \beta_1, \beta_Y, \beta_M, \beta_R, \beta_S, \beta_T, \beta_D \in Z_p^*$, $g_1, g_2 \in G$ and $h_1, h_2 \in H$, and computes

     $$\begin{aligned} A_0 &= g_1^{\beta_0} g_2^{\beta_Y}, \quad A_M = M \cdot g_1^{\beta_M}, \\ A_Y &= Y_i \cdot g_1^{\beta_Y}, \quad B_0 = h_1^{\beta_1} h_2^{\beta_T}, \\ B_R &= R_i \cdot g^{\beta_R}, \quad B_S = S_i \cdot g^{\beta_S}, \\ B_T &= T_i \cdot {h_1}^{\beta_T}, \quad B_D = ID_i \cdot g^{\beta_D}. \end{aligned}$$

     – Chooses random $K_0, K_1, K_2, K_3, K_4, K_5, K_Y, K_M, K_R, K_S, K_T, K_D, K_r \in Z_p^*$, and computes

     $$\begin{aligned} E_r &= g^{K_r}, \quad E_0 = g_1^{K_0} g_2^{K_Y}, \\ E_1 &= A_0^{-K_r} g_1^{K_2} g_2^{K_3}, \quad E_2 = A_Y^{-K_r} g_1^{K_3} g_1^{K_M}, \\ F_0 &= h_1^{K_1} h_2^{K_T}, \quad F_1 = B_1^{-K_R} h_1^{K_4} h_2^{K_5}, \\ F_2 &= \hat{e}(g_1, W)^{K_Y} \hat{e}(g, V)^{K_R} \hat{e}(g, h)^{K_S}, \\ F_3 &= \hat{e}(U, h)^{K_D} \hat{e}(B_R, h_1)^{K_T} \hat{e}(g, B_T)^{K_R} \hat{e}(g, h_1)^{K_T}. \end{aligned}$$

– Computes

$$c = H(C_1, C_2, E_r, E_0, E_1, E_2, F_0, F_1, F_2, F_3),$$
$$d_r = K_r - cr, \quad d_0 = K_0 - c\beta_0, \quad d_1 = K_1 - c\beta_1,$$
$$d_2 = K_2 - c\beta_0 r, \quad d_3 = K_3 - c\beta_Y r,$$
$$d_4 = K_4 - c\beta_1\beta_R, \quad d_5 = K_5 - c\beta_T\beta_R,$$
$$d_Y = K_Y - c\beta_Y, \quad d_M = K_M - c\beta_M, \quad d_R = K_R - c\beta_R,$$
$$d_S = K_S - c\beta_S, \quad d_T = K_T - c\beta_T, \quad d_D = K_D - c\beta_D.$$

Thus, we obtain $C_3 = (c,\ d_r,\ d_0, ..., d_5,\ d_Y,\ d_M,\ d_R,\ d_S,\ d_T,\ d_D,\ A_0,\ A_M,\ A_Y,\ B_0,\ B_R,\ B_S,\ B_T,\ B_D)$, a signature of knowledge for $\mathbb{M} = (Cert_i, C_1, C_2)$ as

$$\text{SPK}\left\{\begin{array}{l|l} (r, \beta_M, \beta_0, & A_0 = g_1^{\beta_0} g_2^{\beta_Y} \wedge 1 = A_0^{-r} g_1^{\alpha_0} g_2^{\alpha_Y} \wedge \\ \alpha_0, \alpha_1, \beta_Y, & C_1 = g^r \wedge \frac{A_M}{C_2} = A_Y^{-r} g_1^{\alpha_Y} \cdot g_1^{\beta_M} \wedge \\ \beta_D, \beta_T, & B_0 = h_1^{\beta_1} h_2^{\beta_T} \wedge 1 = B_0^{-\beta_R} h_1^{\alpha_1} h_2^{\alpha_T} \wedge \\ \alpha_T, \alpha_Y) & \frac{\hat{e}(B_R,V)\hat{e}(B_S,h)\hat{e}(A_Y,W)}{\hat{e}(g,Z)} = \hat{e}(g,W)^{\beta_Y} \\ & \hat{e}(g,V)^{\beta_R}\hat{e}(g,h)^{\beta_S} \wedge \frac{\hat{e}(U,B_D)\hat{e}(B_R,B_T)}{\hat{e}(g,h)} = \\ & \hat{e}(U,h)^{\beta_D}\hat{e}(B_R,h_1)^{\beta_T}\hat{e}(g,B_T)^{\beta_R}\hat{e}(g,h_1)^{\alpha_T} \end{array}\right\}(\mathbb{M}),$$

where $\alpha_0 = \beta_0 \cdot r$, $\alpha_Y = \beta_Y \cdot r$, $\alpha_1 = \beta_1 \cdot \beta_R$, $\alpha_T = \beta_T \cdot \beta_R$.

3. Outputs the ciphertext $C = (C_1, C_2, C_3)$.

- Verify($params$, mpk, C): This algorithm takes the common parameters $params$, the master public key mpk and the ciphertext C as input. It parses the ciphertext C as (C_1, C_2, C_3), and checks the validity of C_3.

  – Computes

$$B_1 = \frac{\hat{e}(B_R,V)\hat{e}(B_S,h)\hat{e}(A_Y,W)}{\hat{e}(g,Z)}, \quad B_2 = \frac{\hat{e}(U,B_D)\hat{e}(B_R,B_T)}{\hat{e}(g,h)},$$
$$E_r = C_1^c g^{d_r}, \quad E_0 = A_0^c g_1^{d_0} g_2^{d_Y},$$
$$E_1 = A_0^{-d_r} g_1^{d_2} g_2^{d_3}, \quad E_2 = (\frac{A_M}{C_2})^c A_Y^{-d_r} g_1^{d_3} g_1^{d_M},$$
$$F_0 = B_0^c h_1^{d_1} h_2^{d_T}, \quad F_1 = B_0^{-d_R} h_1^{d_4} h_2^{d_5},$$
$$F_2 = B_1^c \hat{e}(g_1,W)^{d_Y}\hat{e}(g,V)^{d_R}\hat{e}(g,h)^{d_S},$$
$$F_3 = B_2^c \hat{e}(U,h)^{d_D}\hat{e}(B_R,h_1)^{d_T}\hat{e}(g,B_T)^{d_R}\hat{e}(g,h_1)^{d_5}.$$

– If $c = H(C_1, C_2, E_r, E_0, E_1, E_2, F_0, F_1, F_2, F_3)$, it outputs the ciphertext C. Otherwise, it outputs $\perp$.

- Decrypt($params$, sk_i, C): This algorithm takes the common system parameters $params$, the private key sk_i of user ID_i and the ciphertext C as input. If user ID_i is a privileged receiver, it outputs $M = C_2 \cdot C_1^{-x_i}$. Otherwise, it outputs a failure symbol $\perp$.

Efficiency. In order to improve the efficiency of the zero-knowledge proof of knowledge system, the optimal structure-preserving signature proposed in [AGHO11] is used to create the certificate for the user, which is very efficient in asymmetric bilinear groups. Thus, our verifiable and anonymous encryption system, considering the security it has achieved, can be regarded as an efficient one.

3.4.3 Security Proof

Theorem 3.4 *The above scheme is ANON-IND-CCA secure assuming that the decisional DH assumption holds in G and* SPK *is a secure zero-knowledge proof of knowledge system.*

Proof. Given a ciphertext $C^* = (C_1^*, C_2^*, C_3^*)$, under the decisional DH assumption, it is clear that algorithm $\mathcal{A}$ cannot obtain the plaintext M and the public key Y_i used in C^* from the encryption part (C_1^*, C_2^*). On the other hand, because of the security properties of the zero-knowledge proof of knowledge scheme, algorithm $\mathcal{A}$ has negligible probability to learn the identity of the privileged user from the verification part C_3^*. Also, C_3^* can be regarded as an one-time signature scheme built from a proof of knowledge system [ES02], which makes the scheme secure against chosen ciphertext attacks. Consequently, the scheme achieves the ANON-IND-CCA security. Note that as C_3^* is based on a zero-knowledge proof of knowledge scheme, algorithm $\mathcal{A}$ acquires nothing even it colludes with CA.

It is obvious that the security of above scheme depends on the security of the zero-knowledge proof of knowledge system, which is detailed below.

- **Completeness.** It is clear from the verification phase.

- **Soundness.** Assume that there are $(c, d_r, d_0, ..., d_5, d_Y, d_M, d_R, d_T, d_S, d_T)$ and $(c', d'_r, d'_0, ..., d'_5, d'_Y, d'_M, d'_R, d'_T, d'_S, d'_T)$ associated to the same $(E_r,$

$E_0, ..., E_2,\ F_0, ..., F_3,\ B_0, ..., B_2$), then we have

$$\begin{aligned}
E_r &= C_1^c g^{d_r} = C_1^{c'} g^{d'_r},\\
E_0 &= A_0^c g_1^{d_0} g_2^{d_Y} = A_0^{c'} g_1^{d'_0} g_2^{d'_Y},\\
E_1 &= A_0^{-d_r} g_1^{d_2} g_2^{d_3} = A_0^{-d'_r} g_1^{d'_2} g_2^{d'_3},\\
E_2 &= (\frac{A_M}{C_2})^c A_Y^{-d_r} g_1^{d_3} g_1^{d_M} = (\frac{A_M}{C_2})^{c'} A_Y^{-d'_r} g_1^{d'_3} g_1^{d'_M},\\
F_0 &= B_0^c h_1^{d_1} h_2^{d_T} = B_0^{c'} h_1^{d'_1} h_2^{d'_T},\\
F_1 &= B_1^{-d_R} h_1^{d_4} h_2^{d_5} = B_1^{-d'_R} h_1^{d'_4} h_2^{d'_5},\\
F_2 &= B_1^c \hat{e}(g_1, W)^{d_Y} \hat{e}(g, V)^{d_R} \hat{e}(g, h)^{d_S}\\
&= B_1^{c'} \hat{e}(g_1, W)^{d'_Y} \hat{e}(g, V)^{d'_R} \hat{e}(g, h)^{d'_S},\\
F_3 &= B_2^c \hat{e}(U, h)^{d_D} \hat{e}(B_R, h_1)^{d_T} \hat{e}(g, B_T)^{d_R} \hat{e}(g, h_1)^{d_5}\\
&= B_2^{c'} \hat{e}(U, h)^{d'_D} \hat{e}(B_R, h_1)^{d'_T} \hat{e}(g, B_T)^{d'_R} \hat{e}(g, h_1)^{d'_5},
\end{aligned}$$

where

$$B_1 = \frac{\hat{e}(B_R, V)\hat{e}(B_S, h)\hat{e}(A_Y, W)}{\hat{e}(g, Z)}, \quad B_2 = \frac{\hat{e}(U, B_D)\hat{e}(B_R, B_T)}{\hat{e}(g, h)}.$$

Therefore, we obtain

$$\begin{aligned}
C_1 &= g^{\frac{d'_r - d_r}{c - c'}} = g^r, \quad A_0 = g_1^{\frac{d'_0 - d_0}{c - c'}} g_2^{\frac{d'_Y - d_Y}{c - c'}} = g_1^{\beta_0} g_2^{\beta_Y},\\
(\frac{A_M}{C_2}) &= A_Y^{\frac{d_r - d'_r}{c - c'}} g_1^{\frac{d'_3 - d_3}{c - c'}} g_1^{\frac{d'_M - d_M}{c - c'}} = A_Y^{-r} g_1^{\alpha_Y} \cdot g_1^{\beta_M},\\
B_0 &= h_1^{\frac{d'_1 - d_1}{c - c'}} h_2^{\frac{d'_T - d_T}{c - c'}} = h_1^{\beta_1} h_2^{\beta_T},\\
B_1 &= \hat{e}(g_1, W)^{\frac{d'_Y - d_Y}{c - c'}} \hat{e}(g, V)^{\frac{d'_R - d_R}{c - c'}} \hat{e}(g, h)^{\frac{d'_S - d_S}{c - c'}}\\
&= \hat{e}(g, W)^{\beta_Y} \hat{e}(g, V)^{\beta_R} \hat{e}(g, h)^{\beta_S},\\
B_2 &= \hat{e}(U, h)^{\beta'_D} \hat{e}(B_R, h_1)^{\beta'_T} \hat{e}(g, B_T)^{\beta'_R} \hat{e}(g, h_1)^{\alpha'_T}\\
&= \hat{e}(U, h)^{\beta_D} \hat{e}(B_R, h_1)^{\beta_T} \hat{e}(g, B_T)^{\beta_R} \hat{e}(g, h_1)^{\alpha_T},
\end{aligned}$$

where $\beta'_D = \frac{d'_D - d_D}{c - c'}$, $\beta'_T = \frac{d'_T - d_T}{c - c'}$, $\beta'_R = \frac{d'_R - d_R}{c - c'}$ and $\alpha'_T = \frac{d'_5 - d_5}{c - c'}$.

- **Zero-Knowledge.** To simulate a zero-knowledge proof of knowledge protocol without knowing the exponents, a simulator S can proceed as follows.
  - It chooses random numbers c, d_r, d_0, d_1, d_2, d_3, d_4, d_5, d_Y, d_M, d_R, d_T,

d_S, $d_T \in Z_p^*$, and computes

$$B_1 = \frac{\hat{e}(B_R, V)\hat{e}(B_S, h)\hat{e}(A_Y, W)}{\hat{e}(g, Z)},$$

$$B_2 = \frac{\hat{e}(U, B_D)\hat{e}(B_R, B_T)}{\hat{e}(g, h)},$$

$$E_r = C_1^c g^{d_r}, \quad E_0 = A_0^c g_1^{d_0} g_2^{d_Y},$$

$$E_1 = A_0^{-d_r} g_1^{d_2} g_2^{d_3}, \quad E_2 = (\frac{A_M}{C_2})^c A_Y^{-d_r} g_1^{d_3} g_1^{d_M},$$

$$F_0 = B_0^c h_1^{d_1} h_2^{d_T}, \quad F_1 = B_0^{-d_R} h_1^{d_4} h_2^{d_5},$$

$$F_2 = B_1^c \hat{e}(g_1, W)^{d_Y} \hat{e}(g, V)^{d_R} \hat{e}(g, h)^{d_S},$$

$$F_3 = B_2^c \hat{e}(U, h)^{d_D} \hat{e}(B_R, h_1)^{d_T} \hat{e}(g, B_T)^{d_R} \hat{e}(g, h_1)^{d_5}.$$

- It sets $c = H(C_1, C_2, E_r, E_0, E_1, E_2, F_0, F_1, F_2, F_3)$.
- It outputs (c, d_r, d_0, d_1, d_2, d_3, d_4, d_5, d_Y, d_M, d_R, d_T, d_S, d_T).

We can see that S can simulate the zero-knowledge proof of knowledge system perfectly.

3.5 Summary

In this chapter, we put emphasis on designing a secure three-party protocol enabling an outside sender to transmit a ciphertext to at least one inside user under the verification of an untrusted gateway. After analyzing its security requirements, we depict three different schemes to achieve it.

In the first construction, we consider this scenario in a broadcast environment, and call it gateway-based verifiable and anonymous broadcast encryption, which is a combination of verifiable encryption and anonymous broadcast encryption with an untrusted gateway verifying that the privileged receivers of an inbound ciphertext belong to the organization without leaking the information of both the plaintext and the privileged receivers. To solve such a problem, we introduce a new notion: public key encryption and identity search (PKEIS), which is derived from private information retrieval (PIR) [CGKS95] and public key encryption with keyword search (PEKS) [BCOP04]. In PKEIS, the gateway stores all the trapdoors generated by receivers, and when it receives an inbound ciphertext, it decides to broadcast it or not by verifying this ciphertext with the stored trapdoors and its private key. Performing this method in a gateway-based verifiable and anonymous broadcast encryption

system, the gateway can complete verification without learning anything about the plaintext and the personal information of the receivers even if it colludes with the corrupted receivers. We also provide a concrete implementation of this new notion from bilinear pairings, and prove its security in the random oracle model. Although our construction is based on bilinear pairings, it is very efficient because encryption needs no pairing computation while decryption only needs one.

Since the first scheme requires the identity tokens to be secret, which is not common in applications, in the second and third constructions, we simplify it to the single receiver case to make it more feasible and securer, and call it verifiable and anonymous encryption under an untrusted gateway. In the second verifiable and anonymous encryption scheme, we follow the method used in the first one without the requirement of secret identity tokens: the gateway stores the trapdoors generated by the users in the system in a trapdoor list, and when it receives an outside ciphertext, it decides to send it or not by checking this ciphertext with the trapdoor list. Seemingly, it solves our problem; however, in this construction, the adversaries are not allowed to issue a public key query on ID_i^* where ID_i^* is the identity in the challenge phase, which is too strict to be widely applied in the real world, where the users are usually required to make their identity information and the corresponding public keys public. In the third verifiable and anonymous encryption scheme, we consider making use of zero-knowledge proof of knowledge to generate a signature for verification so that the scheme could be secure even when the adversaries are given both the identity information and the public keys of all the users in the system. It is clear that zero-knowledge proof of knowledge is very useful to achieve verifiable and anonymous encryption securely. The problem here is how to improve the efficiency of the scheme, as most of the zero-knowledge proof of knowledge systems are costly in computation. Fortunately, we can easily solve it by making the structure of the certificate in our system be optimal structure-preserving in asymmetric bilinear maps.

Some results of this chapter have been published in "The 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (IEEE TrustCom-13)" and "International Journal of Security and Networks".

Chapter 4

Signcryption Secure Against Related-Key Attacks

In cryptographic world, signcryption simultaneously performs the functions of both digital signature and encryption in a way that is more efficient than signing and encrypting separately. Signcryption has several applications including secure and authentic email, electronic commerce, mobile commerce, and so on. The focus of this chapter is to enhance the traditional security models of signcryption under related-key attacks.

4.1 Introduction

Encryption and digital signature are used to achieve confidentiality and authenticity of a message, respectively. In some scenarios such as secure email, both primitives are needed. Resorting to the signature-then-encryption approach can solve this problem, but the computational cost and the communication overhead are high. In 1997, Zheng [Zhe97] proposed a cryptographic primitive called Signcryption to combine the functions of digital signature and encryption in a single step with a cost lower than that required by the signature-then-encryption approach. Later Baek, Steinfeld and Zheng [BSZ07] formalized and defined security notions for signcryption via semantic security against adaptive chosen ciphertext attack and existential unforgeability against adaptive chosen message attack.

Modern notions of security like semantic security [GM82] or CCA security [NY90] in encryption are formulated in a very strong way such that the adversary can fully control almost all aspects of the system, meaning that the adversary is able to encrypt messages and decrypt ciphertexts at will, but in these definitions the adversary is assumed to have no access to the private keys, which seems impractical in real systems. In many situations, the adversary might get some partial information

about private keys through methods which are not anticipated by the designer of the system and, correspondingly, not taken into account when arguing its security. Such attacks, referred to as key-leakage attacks, come in a large variety. An important example is side-channel attack [MR04] that exploits information leakage from the implementation of an algorithm, where an adversary observes some "physical output" of a computation (such as radiation, power, temperature, or running time), in addition to the "logical output" of the computation. To achieve these security requirements, it is required to capture security under the scenarios where some information of the keys is leaked to the adversary. If an adversary is allowed to tamper with the private key stored in a cryptographic hardware device and observes the result of the cryptographic primitive under this modified private key, there is a related-key attack (RKA) [GLM+04, BC10, BCM11]. The key here could be a signing key of a certificate authority or a decryption key of an encryption scheme. In related-key attacks, the adversary attempts to break an cryptographic system by invoking it with several private keys satisfying some known relations.

4.1.1 Our Results

Although related-key attack security has been achieved in various cryptographic primitives [BCM11], so far there is no such work in signcryption. Inspired by this challenge, in this chapter we focus on the design of signcryption schemes that are secure against linear related-key attacks [Wee12]. At the outset, we need to formalize the security model for signcryption under related-key attacks, in addition to the original security requirements [BSZ07]. Motivated by the fact that user privacy has recently become another pursuit in the cryptographic world, we also consider anonymity under related-key attacks in signcryption.

Suppose that the signcryption system is composed of algorithms, public parameters, as well as the private and public key pairs of the sender and the receiver, respectively, of which the private and public keys are subject to related-key attacks, and the public parameters are system-wide, i.e., they are set beforehand and independent of users. In a protocol run, all these parameters can be tampered when distributed via a public channel.

In a signcryption system, the designcryption needs the private key of the receiver while the signcryption needs the private key of the sender, hence we consider

related-key attacks on private keys of both sides: a chosen ciphertext and related-key attack and a chosen message and related-key attack. The designcryption oracle is forbidden when the signcryption[1] is equal to the challenge signcryption and the derived receiver's key matches the original one. Likewise, the signcryption oracle will not work if the given message is equal to the output message and the derived sender's key matches the original one. In addition, we add anonymity to this security model, which requires the signcryption to be anonymous to others except the real receiver.

To begin with, we assume that both the sender and the receiver are honest so that they will not be involved in the attacks, and we call the corresponding security notions as outsider chosen ciphertext security under related-key attacks (OCC-RKA security), outsider chosen message security under related-key attacks (OCM-RKA security) and outsider anonymity under related-key attacks (OANON-RKA security). The difficulty here is how to designcrypt a signcryption C with the receiver's private key $\phi(sk_R)$, and how to signcrypt a plaintext m with the sender's private key $\phi(sk_S)$, where ϕ denotes a related-key deriving function [BK03]. A well known method is key homomorphism [BCM11, Wee12], which, under chosen ciphertext attacks and chosen message attacks, can reduce a signcryption scheme secure against related-key attacks to a general signcryption scheme. This is because key homomorphism enables the designcryption of a signcryption C with the related private key $\phi(sk_R)$ to equal the designcryption of another signcryption C' with the original private key sk_R, and the signcryption of a plaintext m with the related private key $\phi(sk_S)$ to equal the signcryption of another plaintext m' with the original private key sk_S, respectively.

Unfortunately, key homomorphism fails when the signcryption C' equals the challenge signcryption in the chosen ciphertext security game, and the message m' equals the output message in chosen message security game. We solve this problem with the following techniques.

- We make use of a strong one-time signature scheme to elicit a *tag* to construct the signcryption. Additionally, we restrict that the signcryption C and the derivation signcrypiton C' share the same *tag*. If C and the challenge signcryption share the same *tag*, then

[1]To distinguish from the ciphertext generated by an encryption scheme, we refer to the result outputted by a signcryption scheme as signcryption.

1. C has to equal the challenge signcryption because of the one-time signature scheme.
2. C' and the challenge signcryption also share the same *tag*, and thus C' can be decrypted by issuing the decryption oracle in the chosen ciphertext attack security game.

Note that we employ a one-time signature scheme in our construction because one-time signature is a very simple method to construct a chosen ciphertext attack secure scheme according to the result in [CHK04] by Cannetti, Haleve, and Katz.

- In order to obtain the full OCC-RKA security of signcryption, what we need to handle is making the signcryption C' equal the challenge signcryption but $\phi(sk_R) \neq sk_R$. Due to the onewayness property of adaptive trapdoor relations [Wee10] implied in the signcryption, we can simply formulate that the challenge signcryption is an invalid signcryption for any receiver's private key $sk'_R \neq sk_R$, which means that a valid signcryption with the public parameters uniquely decides a consistent private key.

- With regard to the full OCM-RKA security of signcryption, we need to tackle the problem in which the signcryption equals the output signcryption where $\phi(sk_S) \neq sk_S$. We adopt a collision resistant hash function in the signcryption, which disables an adversary to output a valid signcryption for any sender's private key $sk'_S \neq sk_S$ such that a valid signcryption with public parameters can only be constructed by a unique corresponding private key.

Next, we try to obtain outsider anonymity in a related-key attack secure signcryption scheme, which asks the ciphertext to be anonymous to others except the real receiver given the honest participants. With a minor change to our normal signcryption scheme, we easily make a signcryption scheme with outsider anonymity in the context of related-key attacks, of which the OANON-RKA security can be proved in the random oracle model.

After that, we consider the security one step further and allow the adversary to approach the private keys involved in the signcryption system, and rename the security notions as chosen ciphertext security under related-key attacks (CC-RKA), chosen message security under related-key attacks (CM-RKA) and anonymity under related-key attacks (ANON-RKA). To achieve such security in a signcryption

scheme, we mainly modify the signing algorithm in the original outsider anonymous signcryption scheme, resulting an anonymous signcryption system which is secure against related-key attacks even when the private key information in the system is leaked to the adversary.

4.1.2 Related Work

Signcryption. Zheng [Zhe97] first proposed a cryptographic primitive "Signcryption" to combine the functions of digital signature and encryption in a single step with a cost lower than that required by the signature-then-encryption approach in 1997. Then Baek, Steinfeld and Zheng [BSZ07] formalized and defined security notions for signcryption. In 2002, Malone-Lee [ML02] proposed the first identity-based signcryption scheme and claimed that their scheme achieves both privacy and unforgeability. Libert and Quisquater [LQ03] pointed out that the scheme in [ML02] is not semantically secure in privacy, because the signature of the message is not hidden in the signcrypted message. Libert and Quisquater [LQ03] also proposed a signcryption scheme with ciphertext anonymity [LQ04] based on gap Diffie-Hellman assumption, but Yang, Wong and Deng [YWD05] found that it was not secure. Chow, Yiu, Hui and Chow [CYHC03] designed an identity-based signcryption scheme with public verifiability and forward security. Concurrently, Boyen [Boy03] extended the security model in [ML02] via adding three new security notions: ciphertext unlinkability, ciphertext authentication and ciphertext anonymity. In addition, there are also some schemes concentrating on efficiency [BLMQ05, LYW+07, LYW+10]. Barreto, Libert, McCullagh and Quisquater[BLMQ05] constructed an identity-based signcryption scheme to improve the efficiency. Chung et al. [LYW+07] described a key privacy preserving signcryption scheme with high efficiency and simple design, and then they extended it to a ring signcryption scheme based on the technique due to Boneh, Gentry, Lynn and Shacham[BGLS03].

Related-Key Attack Secure Cryptography. In 2004, Micali and Reyzin [MR04] put forward a comprehensive framework for modeling security against side-channel attacks, which relies on the assumption that there is no leakage of information in the absence of computation. Later, Halderman et al. [HSH+08] described a set of attacks violating the assumption of the framework of Micali and Reyzin. Specially, their "cold boot" attacks showed that a significant fraction of the bits of a cryptographic

key can be recovered if the key is ever stored in memory, of which the framework was modeled by Akavia, Goldwasser and Vaikuntanathan [AGV09]. Actually, physical techniques like fault injection can be used to falsify, inducing the internal state of the devices being modified, if given physical access to the hardware devices [Bih93]. Bellare and Kohno [BK03] investigated related-key attacks from a theoretical point of view and presented an approach to formally handle the notion of related-key attacks. Lucks [Luc04] presented some constructions for block ciphers and pseudorandom function generators, solving the open problem in related-secret security whether or not related-key secure blockciphers exist. Bellare and Cash [BC10] provided the first constructions to create related-secret pseudorandom bits. Applebaum, Harnik and Ishai [AHI11] gave RKA secure symmetric encryption schemes, which can be used in garbled circuits in secure computation. Bellare, Cash and Miller [BCM11] proposed approaches to build high-level primitives secure against related-key attacks such as signatures, CCA secure public-key encryption, identity-based encryption, based on RKA secure pseudorandom functions. Other work about cryptographic systems with RKA security includes signatures [GOR11, BPT12], CCA secure public-key encryption [Wee12, BPT12], and identity-based encryption [BPT12].

4.1.3 Organization

The remainder of this chapter is organized as follows. In Section 4.2, we briefly present the concepts associated with this work and define our security model of signcryption under related-key attacks. In Section 4.3, we propose a specific construction of RKA secure signcryption from BDH, and prove its security in the random oracle model, and then we further consider ciphertext anonymity in our signcryption scheme in the setting of related-key attacks. In Section 4.4, we firstly strengthen the security model of signcryption in Section 4.2 by removing the assumption that both the sender and the receiver are honest, and then we put forward a specific construction of RKA secure signcryption from BDH, and prove its security in the random oracle model. Finally, we conclude this chapter in Section 4.5.

4.2 Background

We firstly describe the framework of signcryption, and then after reviewing some concepts related to the RKA security, we give the security definitions of signcryption

under related-key attacks.

4.2.1 Signcryption

A signcryption scheme is composed of the following five algorithms [LYW+10]: Setup, Keygen, Signcrypt, Designcrypt and Verify.

- Setup(1^λ) $\rightarrow$ ($params$, $\mathcal{M}$, $\mathcal{C}$): Taking a security parameter λ as input, this algorithm outputs public parameters $params$, the message space $\mathcal{M}$ and the signcryption space $\mathcal{C}$.

- Keygen(1^λ, $params$) $\rightarrow$ (sk_R, pk_R), (sk_S, pk_S): Taking a security parameter λ and public parameters $params$ as input, this algorithm outputs two private and public key pairs (sk_R, pk_R), (sk_S, pk_S).

- Signcrypt(1^λ, $params$, m, sk_S, pk_R) $\rightarrow$ C: Taking a security parameter λ, public parameters $params$, a plaintext $m \in \mathcal{M}$, a private key sk_S and a public key pk_R as input, this algorithm outputs a signcryption $C \in \mathcal{C}$.

- Designcrypt(1^λ, $params$, C, sk_R, pk_S) $\rightarrow$ (m, σ, sk_R, pk_S)/$\perp$: Taking a security parameter λ, public parameters $params$, a signcryption C, a private key sk_R and a public key pk_S as input, this algorithm outputs either a triple (m, σ, sk_R, pk_S) where $m \in \mathcal{M}$ is the plaintext, σ is a signnature, sk_R is a private key, and pk_S is a public key, or $\perp$ in case of failure.

- Verify(1^λ, $params$, m, σ, sk_R, pk_S) $\rightarrow$ *true*/*false*: Taking a security parameter λ, public parameters $params$, a plaintext m, a signature σ, a private key sk_R and a public key pk_S as input, this algorithm outputs *true* for a valid signcryption or *false* for an invalid signcryption.

We require that a signcryption system is correct, meaning that if ($params$, $\mathcal{M}$, $\mathcal{C}$) $\leftarrow$ Setup(1^λ), (sk_R, pk_R), (sk_S, pk_S) $\leftarrow$ Keygen(1^λ, $params$), $C \leftarrow$ Signcrypt(1^λ, $params$, m, sk_S, pk_R), then (m, σ, sk_R, pk_S) $\leftarrow$ Designcrypt(1^λ, $params$, C, sk_R, pk_S), $true \leftarrow$ Verify(1^λ, $params$, m, σ, sk_R, pk_S).

4.2.2 RKA Security for Outsider Secure Signcryption

Our definition of related-key deriving functions follows the notion given in [BK03]. Briefly, a class Φ of related-key deriving functions ϕ: $sk_u \rightarrow sk_u$ is a finite set of

functions with the same domain and range which can map a key to a related key. Additionally, the class of functions should allow an efficient membership test, and its functions should be efficiently computable. Note that in this chapter, we consider only the class Φ^+ as linear shifts.

The family Φ^+. Any function $\phi : Z_p^* \to Z_p^*$ in this class is indexed by $\triangle \in Z_p^*$, where $\phi_\triangle(sk_u) := sk_u + \triangle$.

Let RKA.Signcrypt(sk_S, pk_R, $\cdot$, $\cdot$) be an oracle that on an input (m, ϕ), it returns Signcrypt(m, $\phi(sk_S)$, pk_R), and RKA.Designcrypt(sk_R, pk_S, $\cdot$, $\cdot$) be an oracle that on an input (C, ϕ), it returns Designcrypt(C, $\phi(sk_R)$, pk_S). We constrain algorithm $\mathcal{A}$ to only make queries (m, ϕ) such that $\phi \in \Phi$ and (m, $\phi(sk_S)$) $\neq$ (m^*, sk_S), and (C, ϕ) such that $\phi \in \Phi$ and (C, $\phi(sk_R)$) $\neq$ (C^*, sk_R). In addition, we assume that both the sender and the receiver in our signcrypiton system are honest. In other words, the adversary will not have access to the private keys of the sender and the receiver in the security games.

OCC-RKA Security. A signcryption scheme is Φ-OCC-RKA secure if for a stateful adversary $\mathcal{A}$ it holds that

$$\left| \Pr\left[d = d' \;\middle|\; \begin{array}{l} params \leftarrow \text{Setup}(1^\lambda). \\ (sk_R, pk_R), (sk_S, pk_S) \leftarrow \text{Keygen}(1^\lambda, params). \\ (m_0, m_1) \leftarrow \mathcal{A}^{\text{RKA.Signcrypt}(sk_S, pk_R, \cdot, \cdot)}(pk_S, pk_R). \\ (m_0, m_1) \leftarrow \mathcal{A}^{\text{RKA.Designcrypt}(sk_R, pk_S, \cdot, \cdot)}(pk_S, pk_R). \\ d \in \{0, 1\}. \\ C^* \leftarrow \text{Signcrypt}(1^\lambda, params, m_d, sk_S, pk_R). \\ d' \leftarrow \mathcal{A}^{\text{RKA.Designcrypt}(sk_R, pk_S, \cdot)}(C^*). \end{array} \right] - 1/2 \right|$$

is a negligible function in the security parameter λ.

OCM-RKA Security. A signcryption scheme is Φ-OCM-RKA secure if for a stateful adversary $\mathcal{A}$ it holds that

$$\Pr\left[\begin{array}{l} \text{Designcrypt}(1^\lambda, \\ params, C^*, sk_R) \\ \rightarrow (m^*, \sigma^*, sk_R, pk_S) \\ \wedge\ \text{Verify}(m^*, \sigma^*, \\ sk_R, pk_S) \neq \perp \end{array} \;\middle|\; \begin{array}{l} params \leftarrow \text{Setup}(1^\lambda). \\ (sk_R, pk_R), (sk_S, pk_S) \leftarrow \text{Keygen}(1^\lambda, params). \\ m^* \leftarrow \mathcal{A}^{\text{RKA.Signcrypt}(sk_S, pk_R, \cdot, \cdot)}(pk_S, pk_R). \\ m^* \leftarrow \mathcal{A}^{\text{RKA.Designcrypt}(sk_R, pk_S, \cdot, \cdot)}(pk_S, pk_R). \\ C^* \leftarrow \mathcal{A}(pk_R, pk_S, m^*) \end{array} \right]$$

is a negligible function in the security parameter λ.

Following the security model of ciphertext anonymity defined by Libert and Quisquater in [LQ04] (note that the signcryption scheme given in [LQ04] is demonstrated neither semantically secure nor anonymous under chosen plaintext attacks [YWD05]), we introduce a weaker anonymity under related-key attacks, OANON-RKA, assuming that the communication parties (the sender and the receiver) are honest.

OANON-RKA Security. Denoted by RKA.Signcrypt(sk_S, pk_R, $\cdot$, $\cdot$) an oracle given an input (m, ϕ) returns the result of Signcrypt(m, $\phi(sk_S)$, pk_R), and denoted by RKA.Designcrypt(sk_R, pk_S, $\cdot$, $\cdot$) an oracle given an input (C, ϕ) returns the result of Designcrypt(C, $\phi(sk_R)$, pk_S). A signcryption scheme is Φ-OANON-RKA secure if for a stateful adversary $\mathcal{A}$ it holds that

$$\left| \Pr \left[\begin{array}{l|l} k = d & params \leftarrow \text{Setup}(1^\lambda). \\ k' = d' & (sk_{R,0}, pk_{R,0}), (sk_{S,0}, pk_{S,0}) \leftarrow \text{Keygen}(1^\lambda, params). \\ & (sk_{R,1}, pk_{R,1}), (sk_{S,1}, pk_{S,1}) \leftarrow \text{Keygen}(1^\lambda, params). \\ & (pk_S, pk_R) \leftarrow \mathcal{A}(1^\lambda, params, pk_{S,0}, pk_{S,1}, pk_{R,0}, pk_{R,1}) \\ & (pk_{S,0}, pk_{S,1}) \leftarrow \mathcal{A}^{\text{RKA.Signcrypt}(sk_S, pk_R, \cdot, \cdot)}(pk_S, pk_R). \\ & (pk_{R,0}, pk_{R,1}) \leftarrow \mathcal{A}^{\text{RKA.Designcrypt}(sk_R, pk_S, \cdot, \cdot)}(pk_S, pk_R). \\ & d, d' \in \{0, 1\}. \\ & C^* \leftarrow \text{Signcrypt}(1^\lambda, params, \\ & m^*, sk_{S,d'}, pk_{R,d}). \\ & k' \leftarrow \mathcal{A}^{\text{RKA.Signcrypt}(sk_{R,d}, pk_{S,d'}, \cdot)}(C^*). \\ & k \leftarrow \mathcal{A}^{\text{RKA.Designcrypt}(sk_{R,d}, pk_{S,d'}, \cdot)}(C^*). \end{array} \right] - 1/4 \right|$$

is a negligible function in the security parameter λ, where $pk_S \in \{pk_{S,0}, pk_{S,1}\}$, $pk_R \in \{pk_{R,0}, pk_{R,1}\}$. Additionally, we restrict that algorithm $\mathcal{A}$ to only make queries (C, ϕ) such that $\phi \in \Phi$ and (C, $\phi(sk_R)$, pk_S) $\neq$ (C^*, sk_R, pk_S), and queries (m, ϕ) such that $\phi \in \Phi$ and (m, $\phi(sk_S)$, pk_R) $\neq$ (m^*, sk_S, pk_R).

4.3 Outsider Secure Signcryption under Related-Key Attacks

After introducing the techniques we will use in our signcryption scheme with RKA security, we propose a specific signcryption scheme in the setting of related-key attacks, and reduce its OCC-RKA and OCM-RKA security in the random oracle

model.

4.3.1 Techniques in Our Solution

We make use of a class of functions with an additional input (namely tag), called adaptive trapdoor relations [Wee10, KMO10], which is easy to compute and invert with *tag*, but hard to invert without *tag*. More specifically, our adaptive trapdoor relations F_{pk_u} satisfy the following features.

- Generation. This is a randomized algorithm G that outputs a pair (pk_u, sk_u) on input a security parameter λ.
- Sampling. On input pk_u and tag, this randomized algorithm F outputs $(\theta, F_{pk_u}(tag, \theta))$ for a random θ.
- Inversion. For all tag, y and (pk_u, sk_u), this efficient algorithm F' computes $F'(sk_u, tag, y) = F_{pk_u}^{-1}(tag, y)$.
- One-wayness. For a stateful adversary $\mathcal{A}$, it holds that

$$\Pr\left[\theta = \theta' \;\middle|\; \begin{array}{l} tag^* \leftarrow \mathcal{A}(1^\lambda). \\ (pk_u, sk_u) \leftarrow G(1^\lambda). \\ (\theta, y) \leftarrow F(pk_u, tag^*). \\ \theta' \leftarrow \mathcal{A}^{F_{pk_u}^{-1}(\cdot,\cdot)}(pk_u, y). \end{array}\right]$$

  is a negligible function in the security parameter λ, where adversary $\mathcal{A}$ is allowed to query $F_{pk_u}^{-1}(\cdot, \cdot)$ on any tag different from tag^*.

Key Homomorphism. Let Φ be a set of related-key deriving functions. We say that F_{pk_u} is Φ-key homomorphic if there is a probabilistic polynomial-time algorithm T such that $F'(\phi(sk_u), tag, y) = F'(sk_u, tag, T(\phi, tag, y))$ holds with overwhelming probability for all $\phi \in \Phi$, sk_u, tag and y.

4.3.2 Construction

Let $\hat{e} : G \times G \to G_T$ be a bilinear map over a bilinear group G of prime order p with a generator $g \in G$. The scheme is described in detail as follows. Note that the strong one-time signature scheme Σ to gain chosen ciphertext security that we make use of in the construction was put forward by Groth [Gro06].

- Setup. To generate the system public parameters, this algorithm

  1. chooses random β, $\gamma \in Z_p^*$, and sets $g_1 = g^\beta$, $g_2 = g^\gamma$.
  2. chooses two collision resistant hash functions $H : G^3 \to Z_p^*$, $H' : G^3 \times G_T^2 \to Z_p^*$.
  3. outputs the public parameters (g, g_1, g_2, H, H').

- Keygen. To generate two private and public key pairs for a receiver and a sender respectively, the system chooses random x_R, $x_S \in Z_p^*$ as the private key, and computes $Y_R = g^{x_R}$, $Y_S = g^{x_S}$ as the public keys.

- Signcrypt. To signcrypt a message $m \in G_T$ for receiver R, sender S

  1. chooses a random $r \in Z_p^*$, and computes $\mu = g^r$.
  2. chooses random t_0, t_1, t_2, $t_3 \in Z_p^*$, and computes
  $$v_0 = g^{t_0}, \quad v_1 = g^{t_1}, \quad v_2 = {v_0}^{t_2} {v_1}^{t_3}.$$
  3. sets $\theta = {g_1}^r$, and computes
  $$\tau = (Y_R \cdot {g_1}^{H(v_0, v_1, v_2)})^r, \quad \psi = \hat{e}(\theta, g_2) \cdot m.$$
  4. chooses a random $e \in Z_p^*$, and computes
  $$w = (t_0 \cdot (t_2 - e) + t_1 \cdot t_3 - H'(\mu, \tau, \psi, (Y_R)^{x_S}, m)/t_1.$$
  5. outputs the signcryption $C = (\mu, \tau, \psi, v_0, v_1, v_2, e, w)$.

- Designcrypt. To designcrypt a ciphertext C from sender S, receiver R runs the following procedure.

  1. If $\hat{e}(\mu, Y_R \cdot {g_1}^{H(v_0, v_1, v_2)}) = \hat{e}(\tau, g)$, computes
  $$\theta = (\tau \cdot \mu^{-x_R})^{H(v_0, v_1, v_2)^{-1}}.$$
  2. If $\hat{e}(\theta, g) = \hat{e}(\mu, g_1)$, computes $m = \psi / \hat{e}(\theta, g_2)$, and outputs ($\mu$, τ, ψ, m, v_0, v_1, v_2, e, w, x_R, Y_S).

- Verify. To verify the validity of a signcryption (μ, τ, ψ, m, v_0, v_1, v_2, e, w, x_R, Y_S), this algorithm checks whether
$$v_2 = {v_1}^w \cdot {v_0}^e \cdot g^{H'(\mu, \tau, \psi, {Y_S}^{x_R}, m)}.$$
If the equation holds, it outputs *true*. Otherwise, it outputs *false*.

To be consistent with the adaptive trapdoor relations defined in the beginning of this section, we can let (μ, τ) denote $y = F_{pk_u}(tag, \theta)$. Informally, the verification of $\hat{e}(\theta, g) = \hat{e}(\mu, g_1)$ is the technique we use to warrant one-wayness of the adaptive trapdoor relations, which ensures that only sk_R can compute correct θ.

Remarks. Note that $H(v_0, v_1, v_2)$ is used as *tag* for us to construct a fully OCC-RKA secure and OCM-RKA secure signcrypiton scheme, where v_0, v_1, v_2 are actually the verification keys, and e, w comprise the one-time signature in [Gro06]. Furthermore, our specific construction of RKA secure signcryption fulfils the functions of digital signature and encryption without resulting in extra computational cost to the public-key encryption scheme against related-key attacks in [Wee12].

4.3.3 Proof of Security

We analyze the security of our proposed signcryption scheme against related-key attacks by reducing its OCC-RKA security and OCM-RKA security in the random oracle model.

Theorem 4.1 *Assuming that the decisional BDH assumption holds in G, G_T, the computational DH problem holds in G, and Σ is a strongly unforgeable one-time signature scheme, then our signcryption scheme is OCC-RKA secure regarding linear related-key deriving functions ϕ^+ in the random oracle model.*

Assume that algorithm $\mathcal{A}$ is an adversary algorithm breaking the security of our signcryption scheme, and algorithm $\mathcal{B}$ is a challenger algorithm simulating the security game. Let H' be a random oracle controlled by algorithm $\mathcal{B}$, and let $(VK^*, \mu^*, \tau^*, \psi^*, e^*, w^*)$ be the challenge signcryption of a message M_d $(d \in \{0, 1\})$ given to algorithm $\mathcal{A}$ by algorithm $\mathcal{B}$. Denote Failure by the event that algorithm $\mathcal{A}$ issues $(\mu^*, \tau^*, \psi^*, Y, M_0)$ or $(\mu^*, \tau^*, \psi^*, Y, M_1)$ to random oracle H' where $\hat{e}(Y, g) = \hat{e}(Y_R, Y_S)$.

We prove that if the event Failure does not occur, then our signcryption scheme is OCC-RKA secure. We conclude this proof by showing that the event Failure has negligible probability to occur.

Lemma 4.2 *If the decisional BDH assumption holds in G, Σ is a strongly unforgeable one-time signature scheme, and the event Failure does not happen, then our signcryption scheme is OCC-RKA secure.*

Proof. If algorithm $\mathcal{A}$ is an adversary algorithm against the OCC-RKA security of our signcrypiton scheme, then we can construct a challenger algorithm $\mathcal{B}$ that solves the decisional BDH problem, which is given input a BDH instance (g, g^a, g^b, g^c, Z) and outputs 1 (Z is $\hat{e}(g,g)^{abc}$) or 0 (Z is a random element in G_T).

- **Initialization.** Algorithm $\mathcal{B}$ runs as follows to simulate the system parameters.

  1. Chooses a collision resistant hash function $H : G^3 \to Z_p^*$.
  2. Chooses random $t_0^*, t_1^*, t_2^*, t_3^* \in Z_p^*$, and computes $VK^* = (v_0^*, v_1^*, v_2^*) = (g^{t_0^*}, g^{t_1^*}, {v_0}^{t_2^*} {v_1}^{t_3^*})$.
  3. Chooses a random $x_S \in Z_p^*$, and computes $Y_S = g^{x_S}$.
  4. Chooses a random $x_r \in Z_p^*$, and computes

  $$Y_R = (g^b)^{-H(VK^*)} g^{x_r}.$$

  Algorithm $\mathcal{B}$ sets $g_1 = g^b$, $g_2 = g^a$, and sends to algorithm $\mathcal{A}$ the public parameters (g, g_1, g_2, H, H') where H' is a random oracle controlled by algorithm $\mathcal{B}$, the receiver's public key Y_R and the sender's public key Y_S. Note that $x_R = \log_g Y_R = -b \cdot H(VK^*) + x_r$, which is unknown to algorithm $\mathcal{B}$.

- **H'-query**. At any time algorithm $\mathcal{A}$ can query the random oracle on $(\mu, \tau, \psi, Y, M) \in G^3 \times G_T^2$. Algorithm $\mathcal{B}$ maintains a list $L_{H'}$ of tuples $((\mu, \tau, \psi, Y, M), H'(\mu, \tau, \psi, Y, M))$ which is initially empty. When algorithm $\mathcal{A}$ issues a hash query on (μ, τ, ψ, Y, M), algorithm $\mathcal{B}$ proceeds as follows to respond.

  - If (μ, τ, ψ, Y, M) already appears in list $L_{H'}$, algorithm $\mathcal{B}$ responds with $H'(\mu, \tau, \psi, Y, M)$.
  - Otherwise, algorithm $\mathcal{B}$ chooses a random $s_i \in Z_p^*$, sets $H'(\mu, \tau, \psi, Y, M) = s_i$, sends s_i to algorithm $\mathcal{A}$, and adds $((\mu, \tau, \psi, Y, M), s_i)$ to list $L_{H'}$.

- **Phase 1.** Algorithm $\mathcal{A}$ adaptively issues the RKA signcryption and designcryption queries to algorithm $\mathcal{B}$. Algorithm $\mathcal{B}$ executes as follows to respond.

  - Algorithm $\mathcal{A}$ queries (m, ϕ) to the RKA.Signcrypt oracle.

    1. Chooses a random $r \in Z_p^*$, and computes $\mu = g^r$.

2. Chooses random e, t_0, t_1, t_2, $t_3 \in Z_p^*$, sets $VK = (v_0, v_1, v_2) = (g^{t_0}, g^{t_1}, {v_0}^{t_2}{v_1}^{t_3})$, and computes

$$\tau = {Y_R}^r \cdot ({g_1}^r)^{H(VK)}, \quad \psi = \hat{e}(g_1, g_2)^r \cdot m,$$
$$w = (t_0 \cdot (t_2 - e) + t_1 \cdot t_3 - s_i)/t_1.$$

Algorithm $\mathcal{B}$ adds $((\mu, \tau, \psi, {Y_R}^{x_S} \cdot {Y_R}^{\triangle}, M), s_i)$ to list $L_{H'}$.

3. Outputs the signcryption $C = (\mu, \tau, \psi, v_0, v_1, v_2, e, w)$.

– Algorithm $\mathcal{A}$ queries (C, ϕ) to the RKA.Designcrypt oracle where $C = (VK, \mu, \tau, \psi, e, w)$.

1. If $H(VK) = H(VK^*)$, algorithm $\mathcal{B}$ aborts the simulation.
2. If $\hat{e}(\mu, Y_R \cdot {g_1}^{H(v_0,v_1,v_2)}) \neq \hat{e}(\tau, g)$, algorithm $\mathcal{B}$ outputs $\perp$. Otherwise, it computes θ' with $\phi(x_R)$. To see how algorithm $\mathcal{B}$ obtains θ without x_R, we rewrite τ such that

$$\begin{aligned}\tau &= (Y_R \cdot (g_1)^{H(VK)})^r \\ &= \mu^{-b \cdot H(VK^*) + x_r + b \cdot H(VK)} \\ &= (\mu^b)^{H(VK) - H(VK^*)} \cdot \mu^{x_r} \\ &= \theta^{H(VK) - H(VK^*)} \cdot \mu^{x_r} \\ \Rightarrow \theta &= (\tau/\mu^{x_r})^{\frac{1}{H(VK) - H(VK^*)}}.\end{aligned}$$

On the other hand,

$$\begin{aligned}\theta' &= (\tau \cdot \mu^{-(x_R + \triangle)})^{-\frac{1}{H(VK)}} \\ &= ((\tau \cdot \mu^{-\triangle}) \cdot \mu^{-x_R})^{-\frac{1}{H(VK)}} \\ &= \theta \cdot (\mu^{-\triangle})^{-\frac{1}{H(VK)}}.\end{aligned}$$

3. If $\hat{e}(\theta', g) = \hat{e}(\mu, g_1)$, algorithm $\mathcal{B}$ decrypts C as $m = \psi/\hat{e}(\theta', g_2)$. Otherwise, it outputs $\perp$.

In fact, the results of the RKA.Signcrypt and RKA.Designcrypt oracles reflect how the key homomorphism works in the adaptive trapdoor relations [Wee12] we used in our construction, which is indispensable according to the definitions given in [BC10, Wee12, AHI11].

- **Challenge.** Algorithm $\mathcal{A}$ outputs two messages M_0, $M_1 \in G_T$ on which it wishes to be challenged. Algorithm $\mathcal{B}$ chooses random e^*, $s^* \in Z_p^*$, a random

$d \in \{0,1\}$, sets $\mu^* = g^c$, and computes

$$\tau^* = (g^c)^{x_r}, \quad \psi^* = Z \cdot M_d,$$
$$w^* = (t_0^* \cdot (t_2^* - e^*) + t_1^* \cdot t_3^* - s^*)/t_1^*.$$

Algorithm $\mathcal{B}$ outputs the signcryption $C^* = (VK^*, \mu^*, \tau^*, \psi^*, e^*, w^*)$, and adds $((\mu^*, \tau^*, \psi^*, {Y_R}^{x_S}, M_d^*), s^*)$ to list $L_{H'}$.

- **Phase 2.** Algorithm $\mathcal{A}$ adaptively issues the RKA signcryption and designcryption queries to algorithm $\mathcal{B}$. Algorithm $\mathcal{B}$ performs as follows to make the response.
  - Algorithm $\mathcal{A}$ queries (m, ϕ) to the RKA.Signcrypt oracle. Algorithm $\mathcal{B}$ responds as in Phase 1.
  - Algorithm $\mathcal{A}$ queries (C, ϕ) to the RKA.Designcrypt oracle where $C = (VK, \mu, \tau, \psi, e, w)$.
    1. $H(VK) \neq H(VK^*)$. In this case, algorithm $\mathcal{B}$ responds as in Phase 1.
    2. $H(VK) = H(VK^*)$, and $(\mu, \tau, \psi, e, w) \neq (\mu^*, \tau^*, \psi^*, e^*, w^*)$. If algorithm $\mathcal{B}$ accepts this signcryption, it means that algorithm $\mathcal{A}$ breaks the security of the strong one-time signature scheme Σ, as H is a collision resistant hash function, which guarantees that $VK \neq VK^*$ but $H(VK) = H(VK^*)$ will never happen. Therefore, algorithm $\mathcal{B}$ outputs $\perp$ except with negligible probability.
    3. $H(VK) = H(VK^*)$, and $(\mu, \tau, \psi, e, w) = (\mu^*, \tau^*, \psi^*, e^*, w^*)$ and $\phi(x_R) \neq x_R$. If algorithm $\mathcal{B}$ accepts this signcryption, it means algorithm $\mathcal{A}$ can output $\phi \in \Phi$ such that $(\tau^* \cdot (\mu^*)^{-\phi(x_R)})^{1/H(VK^*)} \neq \perp$. That is, $\hat{e}(\theta, g) = \hat{e}(\mu, g_1)$. Because of the one-wayness property of the adaptive trapdoor relations [Wee12], we have

$$(\tau^* \cdot (\mu^*)^{-x_R})^{\frac{1}{H(VK^*)}} = (\tau^* \cdot (\mu^*)^{-\phi(x_R)})^{\frac{1}{H(VK^*)}}$$
$$\Rightarrow x_R = \phi(x_R).$$

    Therefore, algorithm $\mathcal{B}$ outputs $\perp$ except with negligible probability.

  Note that (C, ϕ) for satisfying $H(VK) = H(VK^*)$, $(\mu, \tau, \psi, e, w) = (\mu^*, \tau^*, \psi^*, e^*, w^*)$ and $\phi(x_R) = x_R$, is not allowed by the definition of the CC-RKA security game.

- **Output.** Algorithm $\mathcal{A}$ outputs a guess $d' \in \{0, 1\}$. If $d = d'$, algorithm $\mathcal{A}$ wins the game, and algorithm $\mathcal{B}$ outputs 1 indicating $Z = \hat{e}(g, g)^{abc}$. Otherwise, algorithm $\mathcal{B}$ outputs 0 indicating Z is random in G_T.

Let abort be the event that algorithm $\mathcal{B}$ aborts in Phase 1 during the security game. We claim that Pr[abort] is negligible, or one can use algorithm $\mathcal{A}$ to forge signatures with at least the same probability. Briefly, we can construct another algorithm $\mathcal{B}'$ to simulate the above security game that is given the signing key as a challenge in an existential forgery game. Algorithm $\mathcal{A}$ causes an abort by querying a signcryption including an existential forgery under the given signing key. Algorithm $\mathcal{B}'$ is able to use this forgery to win the existential forgery game. Note that during the game algorithm $\mathcal{A}$ makes only one chosen message query to generate the signature needed for the challenger signcryption. Thus, Pr[abort] is negligible.

Let ϵ be the advantage that algorithm $\mathcal{A}$ breaks the OCC-RKA security of the above game. We can see that if the input tuple of algorithm $\mathcal{B}$ is $(g,\ g^a,\ g^b,\ g^c,\ Z)$ where $Z = \hat{e}(g, g)^{abc}$, then algorithm $\mathcal{A}$'s view of this simulation is identical to the real attack, and thus the probability of algorithm $\mathcal{A}$ in outputting $d' = d$ must satisfy $\Pr[d = d'] = 1/2 + \epsilon$. On the other hand, if the input tuple of algorithm $\mathcal{B}$ is $(g,\ g^a,\ g^b,\ g^c,\ Z)$ where $Z \in G_T$, then the advantage of algorithm $\mathcal{A}$ is nil and thus $\Pr[d' = d] = 1/2$. In sum, the probability of algorithm $\mathcal{B}$ in solving the decisional BDH problem is

$$\begin{aligned}\Pr[\mathcal{B}(g, g^a, g^b, g^c, Z)] &= 1/2 \cdot (1/2 + \epsilon) + 1/2 \cdot 1/2 \\ &= 1/2 + \epsilon/2.\end{aligned}$$

In the following, we will prove that the event Failure has negligible probability of taking place because of the security of the computational DH problem.

Lemma 4.3 *If the computational DH problem holds in G, then the event Failure happens with negligible probability.*

Proof. Given algorithm $\mathcal{A}$ for which the event Failure happens with noticeable probability, we can construct algorithm $\mathcal{B}'$ that solves the computational DH problem. Specifically, we consider the following game where algorithm $\mathcal{B}'$ solves the computational DH problem. Suppose that algorithm $\mathcal{B}'$ is given a random tuple $(g,\ g^a,\ g^b)$ as input and outputs g^{ab}.

- **Initialization.** Algorithm $\mathcal{B}'$ runs as follows to simulate the system parameters.

1. Chooses a collision resistant hash function $H : G^3 \rightarrow Z_p^*$.
2. Chooses random t_0^*, t_1^*, t_2^*, $t_3^* \in Z_p^*$, and computes $VK^* = (v_0^*, v_1^*, v_2^*) = (g^{t_0^*}, g^{t_1^*}, {v_0}^{t_2^*} {v_1}^{t_3^*})$.
3. Chooses a random $x_s \in Z_p^*$, and computes $Y_S = (g^a)^{x_s}$.
4. Chooses a random $x_r \in Z_p^*$, and computes

$$Y_R = (g^b)^{-H(VK^*)} g^{x_r}.$$

Algorithm $\mathcal{B}'$ sets $g_1 = g^b$, $g_2 = g^a$, and sends the public parameters (g, g_1, g_2, H, H') where H' is a random oracle controlled by algorithm $\mathcal{B}'$, the receiver's public key Y_R and the sender's public key Y_S to algorithm $\mathcal{A}$. Note that $x_S = a \cdot x_s$, $x_R = \log_g Y_R = -b \cdot H(VK^*) + x_r$, which are unknown to algorithm $\mathcal{B}'$.

- **H'-query**. At any time algorithm $\mathcal{A}$ can query the random oracle on (μ, τ, ψ, Y, M) $\in G^3 \times G_T^2$. Algorithm $\mathcal{B}'$ maintains a list $L_{H'}$ of tuples ((μ, τ, ψ, Y, M), $H'(\mu, \tau, \psi, Y, M)$) which is initially empty. When algorithm $\mathcal{A}$ issues a hash query on (μ, τ, ψ, Y, M), algorithm $\mathcal{B}'$ runs the following steps to respond.
  - If $\hat{e}(Y, g) = \hat{e}(Y_S, Y_R)$, algorithm $\mathcal{B}'$ solves the computational DH problem immediately. To see this, we have

$$\begin{aligned} Y &= ((g^b)^{-H(VK^*)} g^{x_r})^{a x_s} \\ &= (g^{ab})^{-H(VK^*) \cdot x_s} {Y_S}^{x_r} \\ &\Rightarrow (g^{ab}) = (\frac{Y}{{Y_S}^{x_r}})^{-\frac{1}{H(VK^*) \cdot x_s}}. \end{aligned}$$

  - If (μ, τ, ψ, Y, M) already appears in list $L_{H'}$, then algorithm $\mathcal{B}'$ responds with $H'(\mu, \tau, \psi, Y, M)$.
  - Otherwise, algorithm $\mathcal{B}'$ chooses a random $s_i \in Z_p^*$, sets $H'(\mu, \tau, \psi, Y, M) = s_i$, sends s_i to algorithm $\mathcal{A}$, and adds ((μ, τ, ψ, Y, M), s_i) to list $L_{H'}$.
- **Phase 1.** The same as in Lemma 4.2.
- **Challenge.** Algorithm $\mathcal{A}$ outputs two messages M_0, $M_1 \in G_T$ on which it wishes to be challenged. Algorithm $\mathcal{B}'$ chooses random r^*, e^*, $s^* \in Z_p^*$, a

random $d \in \{0, 1\}$, and computes

$$\mu^* = g^{r^*}, \quad \tau^* = (Y_R \cdot {g_1}^{H(v_0, v_1, v_2)})^{r^*},$$
$$\psi^* = \hat{e}(g_1, g_2)^{r^*} \cdot M_d,$$
$$w^* = (t_0^* \cdot (t_2^* - e^*) + t_1^* \cdot t_3^* - s^*)/t_1^*.$$

Algorithm $\mathcal{B}'$ outputs the signcryption $C^* = (VK^*, \mu^*, \tau^*, \psi^*, e^*, w^*)$, and adds $((\mu^*, \tau^*, \psi^*, Y^*, M_d), s^*)$ to list $L_{H'}$.

- **Phase 2.** The same as in Lemma 4.2.

Lemma 4.2 makes sure that if the event Failure does not happen, then our signcryption scheme preserves the OCC-RKA security. Lemma 4.3 guarantees that if the event Failure does not happen, algorithm $\mathcal{B}'$ is the same as algorithm $\mathcal{B}$ such that algorithm $\mathcal{A}$ cannot differentiate between algorithm $\mathcal{B}$ and algorithm $\mathcal{B}'$.

This completes the proof of Theorem 4.1.

Theorem 4.4 *Assuming that the computational DH assumption holds in G, then our signcryption scheme is OCM-RKA secure regarding linear related-key deriving functions ϕ^+ in the random oracle model.*

Proof. If there is an adversary algorithm $\mathcal{A}$ that breaks the OCM-RKA security of our proposed signcrypiton scheme, then we can construct a challenger algorithm $\mathcal{B}$ that solves the computational DH problem, which is given input a random tuple (g, g^a, g^b) and outputs g^{ab}.

- **Initialization.** The same as in Lemma 4.3.
- H'**-query**. The same as in Lemma 4.3.
- **Phase 1.** The same as in Lemma 4.3.

If algorithm $\mathcal{A}$ can output a valid signcryption $C^* = (VK, \mu^*, \tau^*, \psi^*, e^*, w^*)$, then algorithm $\mathcal{A}$'s query $(\mu^*, \tau^*, \psi^*, Y, M)$ to random oracle H' must satisfy $\hat{e}(Y, g) = \hat{e}(Y_R, Y_S)$, such that algorithm $\mathcal{B}$ obtains

$$\begin{aligned} Y &= g^{a \cdot x_s \cdot (-b \cdot H(VK^*) + x_r)} \\ &= (g^{ab})^{-x_s \cdot H(VK^*)} (g^a)^{x_s \cdot x_r} \\ &\Rightarrow g^{ab} = (\frac{Y}{(g^a)^{x_s \cdot x_r}})^{-\frac{1}{x_s \cdot H(VK^*)}}, \end{aligned}$$

and solves the computational DH problem.

This completes the proof of Theorem 4.4.

4.3.4 Outsider Anonymous Signcryption from RKA Security

The concept of anonymous signcryption was proposed by Boyen [Boy03], also known as signcryption with ciphertext anonymity [Boy03] or key privacy [LYW+10], where the ciphertext should be anonymous to anyone except the designated receiver. Therefore, an anonymous signcryption scheme hides both the sender's identity and the receiver's identity of the ciphertext. Here, we consider anonymity of signcryption under a setting where both the sender and the receiver are trusted in the system, and call it outsider anonymity.

Obviously, our construction of an RKA secure signcryption scheme only preserves the sender's privacy. For a given signcryption (μ, τ, ψ, v_0, v_1, v_2, e, w), as we compute

$$\tau = {Y_R}^r \cdot ({g_1}^r)^{H(v_0,v_1,v_2)},$$
$$w = (t_0 \cdot (t_2 - e) + t_1 \cdot t_3 - H'(\mu, \tau, \psi, (Y_R)^{x_S}, m)/t_1$$

in the signcryption, which subtly hides the public key Y_S of the sender in the hash function H' (because of the collision resistance property of hash functions). On the other hand, given v_0, v_1, v_2, τ, an adversary can differentiate the public key Y_R of the receiver as

$$\hat{e}(\tau, g) = \hat{e}(\mu, Y_R) \cdot \hat{e}(\mu, g_1)^{H(v_0,v_1,v_2)}.$$

However, with a slight change to this signcryption scheme, we can hide the public key Y_R of the receiver perfectly as

$$\tau = {Y_R}^r \cdot ({g_1}^r)^{H(v_0,v_1,v_2)} \cdot H_0({Y_R}^r),$$

where $H_0 : G \to G$ is a collision resistant hash function. We can see that in this way, only the actual sender can know who is the actual receiver. Finally, our scheme protects the privacy of both the sender and the receiver in a signcryption system, i.e., the adversary cannot directly learn the public keys of both parties from the signcryption.

Theorem 4.5 *Assuming that the decisional BDH assumption holds in G, the computational BDH problem holds in G, and Σ is a strongly unforgeable one-time signature scheme, then our anonymous signcryption scheme resilient against related-key attacks is OANON-RKA secure in the random oracle model.*

Assume that algorithm $\mathcal{A}$ is an adversary algorithm breaking the security of our signcryption scheme, and algorithm $\mathcal{B}$ is a challenger algorithm simulating the security game. Let H', H_0 be two random oracles controlled by algorithm $\mathcal{B}$, and let $(VK^*, \mu^*, \tau^*, \psi^*, e^*, w^*)$ be the challenge signcryption of a message M_d given to algorithm $\mathcal{A}$ by algorithm $\mathcal{B}$. Denote Failure by the event that algorithm $\mathcal{A}$ issues a query to random oracle H' on $(\mu^*, \tau^*, \psi^*, Y_1, M_0)$ or $(\mu^*, \tau^*, \psi^*, Y_1, M_1)$ where $\hat{e}(Y_1, g) = \hat{e}(Y_R, Y_S)$, and a query to random oracle H_0 on (μ, Y_2) where $\hat{e}(Y_2, g) = \hat{e}(Y_R, \mu^*)$.

We prove that if the event Failure does not occur, then our signcryption scheme is OANON-RKA secure. We conclude this proof by showing that the event Failure has negligible probability to occur.

Lemma 4.6 *If the decisional BDH assumption holds in G, Σ is a strongly unforgeable one-time signature scheme, and the event Failure does not happen, then our signcryption scheme is OANON-RKA secure.*

Proof. Assuming that there is an adversary algorithm $\mathcal{A}$ against the OANON-RKA security of our signcryption scheme, then we can construct a challenger algorithm $\mathcal{B}$ that solves the decisional BDH problem, which is given a random tuple (g, g^a, g^b, g^c) as input and outputs 1 (Z is $\hat{e}(g,g)^{abc}$) or 0 (Z is a random element in G_T).

- **Initialization.** Algorithm $\mathcal{B}$ runs as follows to simulate the system parameters.

  1. Chooses a collision resistant hash function $H : G^3 \to Z_p^*$.
  2. Chooses random $t_0^*, t_1^*, t_2^*, t_3^* \in Z_p^*$, and computes $VK^* = (v_0^*, v_1^*, v_2^*) = (g^{t_0^*}, g^{t_1^*}, {v_0}^{t_2^*} {v_1}^{t_3^*})$.
  3. Chooses random $x_{s,0}, x_{s,1}, x_{r,0}, x_{r,1} \in Z_p^*$, and computes

$$Y_{S,0} = (g^a)^{x_{s,0}}, \quad Y_{S,1} = (g^a)^{x_{s,1}},$$
$$Y_{R,0} = (g^b)^{-H(VK^*)} g^{x_{r,0}}, \quad Y_{R,1} = (g^b)^{-H(VK^*)} g^{x_{r,1}}.$$

  Algorithm $\mathcal{B}$ sets $g_1 = g^b$, $g_2 = g^a$, and sends $(g, g_1, g_2, H, H', H_0, Y_{R,0}, Y_{R,1}, Y_{S,0}, Y_{S,1})$ to algorithm $\mathcal{A}$, where H', H_0 are the random oracles controlled by algorithm $\mathcal{B}$.

- H'**-query**. At any time algorithm $\mathcal{A}$ can query the random oracle on (μ, τ, ψ, Y_1, M) $\in G^3 \times G_T^2$. Algorithm $\mathcal{B}$ maintains a list $L_{H'}$ of tuples ((μ, τ, ψ, Y_1, M), $H'(\mu, \tau, \psi, Y_1, M)$) which is initially empty. When algorithm $\mathcal{A}$ issues a hash query on (μ, τ, ψ, Y_1, M), algorithm $\mathcal{B}$ runs the following procedure to respond.
  - If (μ, τ, ψ, Y_1, M) already appears in list $L_{H'}$, algorithm $\mathcal{B}$ responds with $H'(\mu, \tau, \psi, Y_1, M)$.
  - Otherwise, algorithm $\mathcal{B}$ chooses a random $s_i \in Z_p^*$, sets $H'(\mu, \tau, \psi, Y_1, M) = s_i$, sends s_i to algorithm $\mathcal{A}$, and adds ((μ, τ, ψ, Y_1, M), s_i) to list $L_{H'}$.
- H_0**-query**. At any time algorithm $\mathcal{A}$ can query the random oracle on $Y_2 \in G$. Algorithm $\mathcal{B}$ maintains a list L_{H_0} of tuples (Y_2, $H_0(Y_2)$) which is initially empty. When algorithm $\mathcal{A}$ issues a hash query on Y_2, algorithm $\mathcal{B}$ executes as follows to respond.
  - If Y_2 already appears in list L_{H_0}, algorithm $\mathcal{B}$ responds with $H_0(Y_2)$.
  - Otherwise, algorithm $\mathcal{B}$ chooses a random $t_i \in Z_p^*$, sets $H_0(Y_2) = t_i$, sends t_i to algorithm $\mathcal{A}$, and adds (Y_2, t_i) to list L_{H_0}.
- **Phase 1.** Algorithm $\mathcal{A}$ chooses (Y_S, Y_R), where $Y_S \in \{Y_{S,0}, Y_{S,1}\}$, $Y_R \in \{Y_{R,0}, Y_{R,1}\}$, and adaptively issues the RKA signcryption and RKA designcryption queries to algorithm $\mathcal{B}$. Algorithm $\mathcal{B}$ performs the following steps to respond.
  - Algorithm $\mathcal{A}$ queries (m, ϕ) to the RKA.Signcrypt oracle.
    1. Chooses a random $r \in Z_p^*$, and computes $\mu = g^r$.
    2. Chooses random e, t_0, t_1, t_2, t_3, s_i, $t_i \in Z_p^*$, sets $VK = (v_0, v_1, v_2) = (g^{t_0}, g^{t_1}, {v_0}^{t_2} {v_1}^{t_3})$, and computes τ, ψ, w as
    $$\tau = {Y_R}^r \cdot ({g_1}^r)^{H(VK)} \cdot t_i, \quad \psi = \hat{e}(g_1, g_2)^r \cdot m,$$
    $$w = (t_0 \cdot (t_2 - e) + t_1 \cdot t_3 - s_i)/t_1.$$
    3. Outputs the signcryption $C = (\mu, \tau, \psi, v_0, v_1, v_2, e, w)$, and adds (($\mu$, τ, ψ, ${Y_R}^{x_S} \cdot {Y_R}^{\triangle}$, m), s_i) to list $L_{H'}$, (${Y_R}^r$, t_i) to list L_{H_0}.
  - Algorithm $\mathcal{A}$ queries (C, ϕ) to the RKA.Designcrypt oracle where $C = (VK, \mu, \tau, \psi, e, w)$.
    1. If $H(VK) = H(VK^*)$, algorithm $\mathcal{B}$ aborts the simulation.

2. Chooses a random $t_i \in Z_p^*$ satisfying $\hat{e}(Y_R, \mu) = \hat{e}(Y_2, g)$ from list L_{H_0}, and computes

$$\theta = (\frac{\tau}{\mu^{x_r} \cdot t_i})^{\frac{1}{H(VK)-H(VK^*)}}.$$

To see this, we rewrite τ as

$$\begin{aligned}\frac{\tau}{t_i} &= (Y_R \cdot (g_1)^{H(VK)})^r \\ &= \mu^{-b \cdot H(VK^*)+x_r+b \cdot H(VK)} \\ &= (\mu^b)^{H(VK)-H(VK^*)} \cdot \mu^{x_r} \\ &= \theta^{H(VK)-H(VK^*)} \cdot \mu^{x_r}.\end{aligned}$$

3. Decrypts C with $\phi(x_R)$ as $m = \psi/\hat{e}(\theta', g_2)$ if $\hat{e}(\theta', g) = \hat{e}(\mu, g_1)$, where

$$\begin{aligned}\theta' &= (\frac{\tau}{t_i} \cdot \mu^{-(x_R+\triangle)})^{-\frac{1}{H(VK)}} \\ &= (\frac{\tau}{t_i} \cdot \mu^{-\triangle} \cdot \mu^{-x_R})^{-\frac{1}{H(VK)}} \\ &= \theta \cdot (\mu^{-\triangle})^{-\frac{1}{H(VK)}}.\end{aligned}$$

- **Challenge.** Algorithm $\mathcal{A}$ outputs a messages M^* on which it wishes to be challenged. Algorithm $\mathcal{B}$ chooses random e^*, t^*, $s^* \in Z_p^*$, random d, $d' \in \{0, 1\}$, sets $\mu^* = g^c$, and computes

$$\tau^* = (g^c)^{x_{r,d}} \cdot t^*, \quad \psi^* = Z \cdot M^*,$$
$$w^* = (t_0^* \cdot (t_2^* - e^*) + t_1^* \cdot t_3^* - s^*)/t_1^*.$$

Algorithm $\mathcal{B}$ outputs the signcryption $C^* = (VK^*, \mu^*, \tau^*, \psi^*, e^*, w^*)$, and adds $((\mu^*, \tau^*, \psi^*, Y_1^*, M^*), s^*)$ to list $L_{H'}$, (Y_2^*, t^*) to list L_{H_0}.

- **Phase 2.** Algorithm $\mathcal{A}$ chooses (Y_S, Y_R), where $Y_S \in \{Y_{S,0}, Y_{S,1}\}$, $Y_R \in \{Y_{R,0}, Y_{R,1}\}$, and adaptively issues the RKA signcryption and designcryption queries to algorithm $\mathcal{B}$. Algorithm $\mathcal{B}$ responds as in Phase 1.

- **Output.** Algorithm $\mathcal{A}$ outputs a guess k, $k' \in \{0, 1\}$. If $k = d$ and $k' = d'$, algorithm $\mathcal{A}$ wins the game, and algorithm $\mathcal{B}$ outputs 1 indicating $Z = \hat{e}(g, g)^{abc}$. Otherwise, algorithm $\mathcal{B}$ outputs 0 indicating Z is random in G_T.

Let ϵ be the advantage that algorithm $\mathcal{A}$ breaks the OANON-RKA security of the above game. We can see that if the input tuple of algorithm $\mathcal{B}$ is $(g, g^a, g^b, g^c,$

Z) where $Z = \hat{e}(g,g)^{abc}$, then algorithm $\mathcal{A}$'s view of this simulation is identical to the real attack, and thus the probability of algorithm $\mathcal{A}$ in outputting $k = d$ and $k' = d'$ must satisfy $\Pr[k = d \wedge k' = d'] = 1/4 + \epsilon$. On the other hand, if the input tuple of algorithm $\mathcal{B}$ is (g, g^a, g^b, g^c, Z) where $Z \in G_T$, then the advantage of algorithm $\mathcal{A}$ is nil and thus $\Pr[k = d \wedge k' = d'] = 1/4$. To sum up, the probability of algorithm $\mathcal{B}$ in solving the decisional BDH problem is

$$\begin{aligned}\Pr[\mathcal{B}(g, g^a, g^b, g^c, Z)] &= 1/2 \cdot (1/4 + \epsilon) + 1/2 \cdot 1/4 \\ &= 1/4 + \epsilon/2.\end{aligned}$$

In the following, we prove that the event Failure has negligible probability to occur due to the difficulty of the computational DH problem.

Lemma 4.7 *If the computational DH problem holds in G, then the event Failure happens with negligible probability.*

Proof. Given algorithm $\mathcal{A}$ for which the event Failure happens with noticeable probability, we can construct algorithm $\mathcal{B}'$ that solves the computational DH problem. Specifically, we consider the following game where algorithm $\mathcal{B}'$ solves the computational BDH problem. Suppose algorithm $\mathcal{B}'$ is given a random tuple (g, g^a, g^b, g^c) as input and outputs $\hat{e}(g,g)^{abc}$.

- **Initialization.** The same as in Lemma 4.6.
- **H'-query**. At any time algorithm $\mathcal{A}$ can query the random oracle on (μ, τ, ψ, Y_1, M) $\in G^3 \times G_T^2$. Algorithm $\mathcal{B}'$ maintains a list $L_{H'}$ of tuples ((μ, τ, ψ, Y_1, M), $H'(\mu, \tau, \psi, Y_1, M)$) which is initially empty. When algorithm $\mathcal{A}$ issues a hash query on (μ, τ, ψ, Y_1, M), algorithm $\mathcal{B}'$ runs as follows to do the response.
  - If $\hat{e}(Y_1, g) = \hat{e}(Y_{R,d}, Y_{S,d'})$, algorithm $\mathcal{B}'$ solves the computational BDH problem immediately. To see this, we have

$$\begin{aligned} Y_1 = {Y_{R,d}}^{x_{S,d'}} &= ((g^b)^{-H(VK^*)} g^{x_{r,d}})^{a \cdot x_{s,d'}} \\ &= (g^{ab})^{-H(VK^*) \cdot x_{s,d'}} {Y_{S,d'}}^{x_{r,d}} \\ &\Rightarrow (g^{ab}) = (\frac{Y_1}{{Y_{S,d'}}^{x_{r,d}}})^{-\frac{1}{H(VK^*) \cdot x_{s,d'}}} \\ &\Rightarrow \hat{e}(g,g)^{abc} = \hat{e}(g^{ab}, g^c), \end{aligned}$$

    where $\{d, d'\} \in \{0, 1\}$.

- If $(\mu,\ \tau,\ \psi,\ Y_1,\ M)$ already appears in list $L_{H'}$, algorithm $\mathcal{B}'$ responds with $H'(\mu, \tau, \psi, Y_1, M)$.
- Otherwise, algorithm $\mathcal{B}'$ chooses a random $s_i \in Z_p^*$, sets $H'(\mu, \tau, \psi, Y_1, M) = s_i$, sends s_i to algorithm $\mathcal{A}$, and adds $((\mu,\ \tau,\ \psi,\ Y_1,\ M),\ s_i)$ to list $L_{H'}$.

- **H_0-query**. At any time algorithm $\mathcal{A}$ can query the random oracle on $Y_2 \in G$. Algorithm $\mathcal{B}'$ maintains a list L_{H_0} of tuples $(Y_2,\ H_0(Y_2))$ which is initially empty. When algorithm $\mathcal{A}$ issues a hash query on Y_2, algorithm $\mathcal{B}'$ executes as follows to respond.
  - If $\hat{e}(Y_2, g) = \hat{e}(Y_{R,d}, g^c)$, algorithm $\mathcal{B}'$ solves the computational BDH problem immediately. To see this, we have
$$\begin{aligned} Y_2 = {Y_{R,d}}^c &= (g^{bc})^{-H(VK^*)} g^{c \cdot x_{r,d}} \\ \Rightarrow g^{bc} &= (\frac{Y_2}{g^{c \cdot x_{r,d}}})^{-\frac{1}{H(VK^*)}} \\ \Rightarrow \hat{e}(g,g)^{abc} &= \hat{e}(g^a, g^{bc}), \end{aligned}$$
    where $d \in \{0, 1\}$.
  - If Y_2 already appears in list L_{H_0}, algorithm $\mathcal{B}'$ responds with $H_0(Y_2)$.
  - Otherwise, algorithm $\mathcal{B}'$ chooses a random $t_i \in Z_p^*$, sets $H_0(Y_2) = t_i$, sends t_i to algorithm $\mathcal{A}$, and adds $(Y_2,\ t_i)$ to list L_{H_0}.

- **Phase 1.** The same as in Lemma 4.6.

- **Challenge.** Algorithm $\mathcal{A}$ outputs a messages M^* on which it wishes to be challenged. Algorithm $\mathcal{B}'$ chooses random $r^*,\ e^*,\ t^*,\ s^* \in Z_p^*$, random $d,\ d' \in \{0, 1\}$, and computes
$$\begin{aligned} \mu^* = g^{r^*}, \quad \tau^* &= (Y_{R,d} \cdot {g_1}^{H(v_0, v_1, v_2)})^{r^*}, \\ \psi^* &= \hat{e}(g_1, g_2)^{r^*} \cdot M^*, \\ w^* &= (t_0^* \cdot (t_2^* - e^*) + t_1^* \cdot t_3^* - s^*)/t_1^*. \end{aligned}$$
  Algorithm $\mathcal{B}'$ outputs the signcryption $C^* = (VK^*,\ \mu^*,\ \tau^*,\ \psi^*,\ e^*,\ w^*)$, and adds $((\mu^*,\ \tau^*,\ \psi^*,\ {Y_{R,d}}^{x_{S,d'}},\ M^*),\ s^*)$ to list $L_{H'}$, $({Y_{R,d}}^{r^*},\ t^*)$ to list L_{H_0}.

- **Phase 2.** Algorithm $\mathcal{A}$ chooses $(Y_S,\ Y_R)$, where $Y_S \in \{Y_{S,0},\ Y_{S,1}\}$, $Y_R \in \{Y_{R,0},\ Y_{R,1}\}$, and adaptively issues the RKA signcryption and designcryption queries to algorithm $\mathcal{B}'$. Algorithm $\mathcal{B}'$ responds as in Phase 1.

Lemma 4.6 ensures that if the event Failure does not happen, then our signcryption scheme is OANON-RKA secure. Lemma 4.7 guarantees that if the event Failure does not happen, algorithm $\mathcal{B}'$ is the same as algorithm $\mathcal{B}$ such that algorithm $\mathcal{A}$ cannot differentiate between algorithm $\mathcal{B}$ and algorithm $\mathcal{B}'$.

This completes the proof of Theorem 4.5.

4.4 Signcryption from RKA Security

In the previous section, we discussed outsider anonymous signcryption secure against related-key attacks, where we assume that both sender and receiver are honest such that the adversary in the security model will not be given the private keys. In this section, we enhance the security model somewhat by giving the private key of the sender to the adversary. Also, we propose a specific anonymous signcryption scheme in the setting of related-key attacks, and analyze its CC-RKA, CM-RKA and ANON-RKA security.

4.4.1 Security Definitions

Informally, we consider a secure anonymous signcryption scheme against related-key attacks to be semantically secure against chosen ciphertext and related-key attacks (CC-RKA), existentially unforgeable against chosen message and related-key attacks (CM-RKA), and anonymous against related-key attacks in the sense that a signcryption should contain no information that identifies the sender of the signcryption and the receiver of the message (ANON-RKA), and yet be decipherable by the targeted receiver.

CC-RKA Security. A signcryption scheme is semantically secure against chosen ciphertext attacks and related-key attacks (CC-RKA security) if no probabilistic polynomial-time adversary has a non-negligible advantage in the following game.

- Initialization. The challenger algorithm $\mathcal{B}$ runs $params \leftarrow \text{Setup}(1^\lambda)$, and (sk_R, pk_R), $(sk_S, pk_S) \leftarrow \text{Keygen}(1^\lambda, params)$. Algorithm $\mathcal{B}$ gives the public parameters $params$, the public key pk_R, the private and public key pair (sk_S, pk_S) to the adversary algorithm $\mathcal{A}$.

- Phase 1. Algorithm $\mathcal{A}$ adaptively issues queries to the RKA.Designcrypt oracle. On input a signcryption C and a related-key deriving function $\phi \in \Phi$,

algorithm $\mathcal{B}$ runs $(m, \sigma) \leftarrow \text{Designcrypt}(1^\lambda, params, C, \phi(sk_R))$, and sends (m, σ) to algorithm $\mathcal{A}$. Note that as sk_S is given to algorithm $\mathcal{A}$, we remove the queries to the RKA.Signcrypt oracle.

- Challenge. Algorithm $\mathcal{A}$ outputs two messages $M_0^*, M_1^* \in \mathcal{M}$, $|M_0^*| = |M_1^*|$, on which it wishes to be challenged. Algorithm $\mathcal{B}$ chooses a random $d \in \{0, 1\}$, and runs $C^* \leftarrow \text{Signcrypt}(1^\lambda, params, m_d, sk_S, pk_R)$. Algorithm $\mathcal{B}$ sends C^* as the challenge signcryption to algorithm $\mathcal{A}$.

- Phase 2. Algorithm $\mathcal{A}$ continues to issue queries to the RKA.Designcrypt oracle. On input a signcryption C and a related-key deriving function $\phi \in \Phi$ with the constraint $(\phi(sk_R), C) \neq (sk_R, C^*)$, algorithm $\mathcal{B}$ responds as in Phase 1.

- Output. Algorithm $\mathcal{A}$ outputs its guess $d' \in \{0, 1\}$ for d, and it wins the game if $d' = d$.

We define the advantage of algorithm $\mathcal{A}$ in this game to be

$$\text{Adv}_{\mathcal{A}}^{\text{CC-RKA}}(\lambda) \stackrel{\text{def}}{=} |\Pr[d = d'] - 1/2|.$$

CM-RKA Security. A signcryption scheme is existentially unforgeable against chosen message attacks and related-key attacks (CM-RKA security) if no probabilistic polynomial-time adversary has a non-negligible advantage in winning the following game.

- Initialization. The challenger algorithm $\mathcal{B}$ runs $params \leftarrow \text{Setup}(1^\lambda)$, and (sk_R, pk_R), $(sk_S, pk_S) \leftarrow \text{Keygen}(1^\lambda, params)$. Algorithm $\mathcal{B}$ gives the public parameters $params$, the public key pk_S, the private and public key pair (sk_R, pk_R) to the adversary algorithm $\mathcal{A}$.

- Phase 1. Algorithm $\mathcal{A}$ adaptively issues queries to the RKA.Signcrypt oracle. On input a message $m \in \mathcal{M}$ and a related-key deriving function $\phi \in \Phi$, algorithm $\mathcal{B}$ runs $C \leftarrow \text{Signcrypt}(1^\lambda, params, m, \phi(sk_S), pk_R)$, and sends C to algorithm $\mathcal{A}$. Note that as sk_R is given to algorithm $\mathcal{A}$, we remove the queries to the RKA.Designcrypt oracle.

- Output. Algorithm $\mathcal{A}$ outputs a signcryption C^*, and wins the game if $(m^*, \sigma^*) \leftarrow \text{Designcrypt}(1^\lambda, params, C^*, sk_R, pk_S)$, (m^*, sk_S) does not equal to any $(m, \phi(sk_S))$ in Phase 1, and $true \leftarrow \text{Verify}(1^\lambda, params, m^*, \sigma^*, sk_R, pk_S)$.

Unlike the model of the OANON-RKA security, the assumption that the communication parties (the sender and the receiver) are honest will be removed to define ANON-RKA security. In short, the security game will mostly follow the OANON-RKA one except that the adversary considered in the ANON-RKA security will have access to the sender's private key.

ANON-RKA Security. A signcryption scheme is anonymous against chosen ciphertext attacks and related-key attacks (ANON-RKA security) if no probabilistic polynomial-time adversary has a non-negligible advantage in the following game.

- Initialization. The challenger algorithm $\mathcal{B}$ runs $params \leftarrow \text{Setup}(1^\lambda)$, and $(sk_{R,0}, pk_{R,0})$, $(sk_{S,0}, pk_{S,0}) \leftarrow \text{Keygen}(1^\lambda, params)$, $(sk_{R,1}, pk_{R,1})$, $(sk_{S,1}, pk_{S,1}) \leftarrow \text{Keygen}(1^\lambda, params)$, respectively. Algorithm $\mathcal{B}$ gives the public parameters $params$, the public keys $pk_{R,0}$, $pk_{R,1}$, the private and public key pairs $(sk_{S,0}, pk_{S,0})$, $(sk_{S,1}, pk_{S,1})$ to the adversary algorithm $\mathcal{A}$.

- Phase 1. Algorithm $\mathcal{A}$ issues a series of queries to the RKA.Designcrypt oracle. On input $sk_S \in \{sk_{S,0}, sk_{S,1}\}$, $pk_R \in \{pk_{R,0}, pk_{R,1}\}$, a signcryption C and a related-key deriving function $\phi \in \Phi$, algorithm $\mathcal{B}$ runs $(m, \sigma) \leftarrow \text{Designcrypt}(1^\lambda, params, C, \phi(sk_R))$, and sends (m, σ) to algorithm $\mathcal{A}$. Note that as $sk_{S,0}$, $sk_{S,1}$ are given to algorithm $\mathcal{A}$, we remove the queries to the RKA.Signcrypt oracle.

- Challenge. Algorithm $\mathcal{A}$ outputs a message $M^* \in \mathcal{M}$ on which it wishes to be challenged. Algorithm $\mathcal{B}$ chooses random $d, e \in \{0, 1\}$, and runs $C^* \leftarrow \text{Signcrypt}(1^\lambda, params, m, sk_{S,d}, pk_{R,e})$. Algorithm $\mathcal{B}$ sends C^* as the challenge signcryption to algorithm $\mathcal{A}$.

- Phase 2. Algorithm $\mathcal{A}$ continues to issue queries to the RKA.Designcrypt oracle. On input $sk_S \in \{sk_{S,0}, sk_{S,1}\}$, $pk_R \in \{pk_{R,0}, pk_{R,1}\}$, a signcryption C and a related-key deriving function $\phi \in \Phi$, with the constraint $(\phi(sk_{R,d}), C) \neq (sk_{R,d}, C^*)$, algorithm $\mathcal{B}$ responds as in Phase 1.

- Output. Algorithm $\mathcal{A}$ outputs its guess $d', e' \in \{0, 1\}$ for d, e, and it wins the game if $d' = d$ and $e' = e$.

We define the advantage of algorithm $\mathcal{A}$ in this game to be

$$\text{Adv}_{\mathcal{A}}^{\text{ANON-RKA}}(\lambda) \stackrel{\text{def}}{=} |\Pr[d = d' \wedge e = e'] - 1/4|.$$

4.4.2 Construction

Let $\hat{e} : G \times G \to G_T$ be a bilinear map over a bilinear group G of prime order p with a generator $g \in G$. The scheme is described as follows.

- Setup. To generate the system public parameters, this algorithm works as follows.

  1. Chooses random β, $\gamma \in Z_p^*$, and computes $g_1 = g^\beta$, $g_2 = g^\gamma$.
  2. Chooses collision resistant hash functions $H_0 : G^2 \to G$, $H : G^2 \to Z_p^*$, $H' : G^5 \times G_T^2 \to Z_p^*$.
  3. Outputs the public parameters (g, g_1, g_2, H_0, H, H').

- Keygen. To generate two private and public key pairs for receiver R and sender S respectively, the system chooses random x_R, $x_S \in Z_q^*$ as the private keys, and computes $Y_R = g^{x_R}$, $Y_S = g^{x_S}$ as the public keys.

- Signcrypt. To signcrypt a message $m \in G_T$ for receiver R, sender S runs as follows.

  1. Chooses a random $r \in Z_p^*$, and computes $\mu = g^r$, $\theta = {g_1}^r$.
  2. Chooses a random $e \in Z_p^*$, and computes $tag = g^e$.
  3. Computes $\psi = \hat{e}(\theta, g_2) \cdot m$, and

$$\tau = (Y_R \cdot {g_1}^{H(\mu, tag)})^r \cdot H_0(\mu, {Y_R}^r),$$
$$\sigma = e - x_S \cdot H'(\mu, \tau, \psi, {Y_R}^r, {Y_R}^{x_S}, tag, m).$$

  4. Outputs the signcryption $C = (\mu, \tau, \psi, tag, \sigma)$.

- Designcrypt. To designcrypt and verify a signcryption C from sender S, receiver R executes as follows.

  1. Computes θ as

$$\theta = (\frac{\tau}{H_0(\mu, \mu^{x_R})} \cdot \mu^{-x_R})^{\frac{1}{H(\mu, tag)}}.$$

  2. Computes $m = \psi / \hat{e}(\theta, g_2)$ if $\hat{e}(\theta, g) = \hat{e}(\mu, g_1)$, and outputs ($\mu$, τ, ψ, m, tag, σ). Otherwise, it outputs $\perp$.

- Verify. To verify the validity of a signcryption (μ, τ, ψ, m, tag, σ), this algorithm checks the validity of σ via

$$tag = g^{\sigma} \cdot {Y_S}^{H'(\mu,\tau,\psi,\mu^{x_R},{Y_S}^{x_R},tag,m)}.$$

  If the equation holds, it outputs m. Otherwise, it outputs $\perp$.

4.4.3 Proof of Security

We analyze the security of our proposed signcryption scheme against related-key attacks by reducing its CC-RKA security, CM-RKA security and ANON-RKA security in the random oracle model.

Theorem 4.8 *Assuming that the decisional BDH assumption holds in G, G_T, and the computational BDH problem holds in G, G_T, then our signcryption scheme is CC-RKA secure regarding linear related-key deriving functions ϕ^+ in the random oracle model.*

Assume that algorithm $\mathcal{A}$ is an adversary algorithm breaking the security of our signcryption scheme, and algorithm $\mathcal{B}$ is a challenger algorithm simulating the security game. Let H', H_0 be two random oracles controlled by algorithm $\mathcal{B}$, and let (μ^*, τ^*, ψ^*, tag^*, σ^*) be the challenge signcryption of a message M_d given to algorithm $\mathcal{A}$ by algorithm $\mathcal{B}$. Denote Failure by the event that algorithm $\mathcal{A}$ issues (μ^*, τ^*, ψ^*, Y_1, Y_2, M_0) or (μ^*, τ^*, ψ^*, Y_1, Y_2, M_1) to random oracle H', and (μ, Y_1) to random oracle H_0, where $\hat{e}(Y_1, g) = \hat{e}(Y_R, \mu^*)$.

In what follows we prove that if the event Failure does not occur, then our signcryption scheme is CC-RKA secure. We conclude this proof by showing that the event Failure has negligible probability of occurrence.

Lemma 4.9 *If the decisional BDH assumption holds in G, G_T, and the event Failure does not happen, then our signcryption scheme is CC-RKA secure.*

Proof. Assuming that algorithm $\mathcal{A}$ is an adversary algorithm against the CC-RKA security of our signcrypiton scheme, then we can construct a challenger algorithm $\mathcal{B}$ that solves the decisional BDH problem, which is given input a BDH instance (g, g^a, g^b, g^c, Z) and outputs 1 if Z is $\hat{e}(g, g)^{abc}$ or 0 if Z is a random element in G_T.

- **Initialization.** Algorithm $\mathcal{B}$ runs as follows to simulate the system parameters.

1. Chooses a collision resistant hash function $H : G^2 \to Z_p^*$.
2. Chooses a random $e^* \in Z_p^*$, and computes $tag^* = g^{e^*}$.
3. Chooses a random $x_S \in Z_p^*$, and computes computes $Y_S = g^{x_S}$.
4. Chooses a random $x_r \in Z_p^*$, and computes

$$Y_R = (g^b)^{-H(g^c, tag^*)} g^{x_r}.$$

Note that $x_R = \log_g Y_R = -b \cdot H(g^c, tag^*) + x_r$, which is unknown to algorithm $\mathcal{B}$.

Algorithm $\mathcal{B}$ sets $g_1 = g^b$, $g_2 = g^a$, and sends algorithm $\mathcal{A}$ sender S's public and private key pair (x_S, Y_S), receiver R's public key Y_R and the public parameters $(g, g_1, g_2, H_0, H, H')$ of which H_0, H' are the random oracles controlled by algorithm $\mathcal{B}$.

- **H_0-query**. At any time algorithm $\mathcal{A}$ can query the random oracle on (μ, Y_1). Algorithm $\mathcal{B}$ maintains a list L_{H_0} of tuples $((\mu, Y_1), H_0(\mu, Y_1))$ which is initially empty. When algorithm $\mathcal{A}$ issues a hash query on Y, algorithm $\mathcal{B}$ performs the following steps to respond.
  - If (μ, Y_1) already appears in list L_{H_0}, algorithm $\mathcal{B}$ responds with $H_0(\mu, Y_1)$.
  - Otherwise, algorithm $\mathcal{B}$ chooses a random $t_i \in Z_p^*$, sets $H_0(\mu, Y_1) = t_i$, and adds $((\mu, Y_1), t_i)$ to list L_{H_0}.
- **H'-query**. At any time algorithm $\mathcal{A}$ can query the random oracle on $(\mu, \tau, \psi, Y_1, Y_2, tag, m)$. Algorithm $\mathcal{B}$ maintains a list $L_{H'}$ of tuples $((\mu, \tau, \psi, Y_1, Y_2, tag, m), H'(\mu, \tau, \psi, Y_1, Y_2, tag, m))$ which is initially empty. When algorithm $\mathcal{A}$ issues a hash query on $(\mu, \tau, \psi, Y_1, Y_2, tag, m)$, algorithm $\mathcal{B}$ runs the following procedure to respond.
  - If $(\mu, \tau, \psi, Y_1, Y_2, tag, m)$ already appears in list $L_{H'}$, algorithm $\mathcal{B}$ responds with $H'(\mu, \tau, \psi, Y_1, Y_2, tag, m)$.
  - Otherwise, algorithm $\mathcal{B}$ chooses a random $s_i \in Z_p^*$, sets $H'(\mu, \tau, \psi, Y_1, Y_2, tag, m) = s_i$, sends s_i to algorithm $\mathcal{A}$, and adds $((\mu, \tau, \psi, Y_1, Y_2, tag, M), s_i))$ to list $L_{H'}$.

- **Phase 1.** Algorithm $\mathcal{A}$ adaptively issues the RKA designcryption queries to algorithm $\mathcal{B}$. For a query (C, ϕ) to the RKA.Designcrypt oracle where $C = (\mu, \tau, \psi, tag, \sigma)$ and ϕ is a related-key deriving function, algorithm $\mathcal{B}$ executes the following steps to make the response.

  1. Algorithm $\mathcal{B}$ computes θ' with $\phi(x_R)$. To see how algorithm $\mathcal{B}$ obtains θ without x_R, we rewrite τ such that

$$\begin{aligned}\frac{\tau}{t_i} &= (Y_R \cdot {g_1}^{H(\mu,tag)})^r \\ &= \mu^{-b\cdot H(g^c,tag^*)+x_r+b\cdot H(\mu,tag)} \\ &= (\mu^b)^{H(\mu,tag)-H(g^c,tag^*)} \cdot \mu^{x_r} \\ &= \theta^{H(\mu,tag)-H(g^c,tag^*)} \cdot \mu^{x_r} \\ \Rightarrow \theta &= (\frac{\tau}{t_i \cdot \mu^{x_r}})^{\frac{1}{H(\mu,tag)-H(g^c,tag^*)}}.\end{aligned}$$

     On the other hand,

$$\begin{aligned}\theta' &= (\frac{\tau}{t_i} \cdot \mu^{-(x_R+\triangle)})^{\frac{1}{H(\mu,tag)}} \\ &= ((\frac{\tau}{t_i} \cdot \mu^{-\triangle}) \cdot \mu^{-x_R})^{\frac{1}{H(\mu,tag)}} \\ &= \theta \cdot (\mu^{-\triangle})^{\frac{1}{H(\mu,tag)}}.\end{aligned}$$

     Note that this reflects how key homomorphism works in the adaptive trapdoor relation [Wee12].

  2. If $\hat{e}(\theta', g) = \hat{e}(\mu, g_1)$, algorithm $\mathcal{B}$ outputs $m = \psi/\hat{e}(\theta', g_2)$. Otherwise, it outputs $\perp$.

- **Challenge.** Algorithm $\mathcal{A}$ outputs two messages M_0, $M_1 \in G_T$ on which it wishes to be challenged. Algorithm $\mathcal{B}$ chooses random s^*, $t^* \in Z_p^*$, a random $d \in \{0, 1\}$, sets $\mu^* = g^c$, and computes

$$\tau^* = (g^c)^{x_r} \cdot t^*, \quad \psi^* = Z \cdot M_d, \quad \sigma^* = e^* - x_S \cdot s^*.$$

  Algorithm $\mathcal{B}$ outputs the signcryption $C^* = (\mu^*, \tau^*, \psi^*, tag^*, \sigma^*)$, and adds $((\mu^*, \tau^*, \psi^*, {Y_R}^c, {Y_R}^{x_S}, M_d), s^*)$ to list $L_{H'}$ and $((\mu, {Y_R}^c), t^*)$ to list L_{H_0}.

- **Phase 2.** Algorithm $\mathcal{A}$ adaptively issues the RKA designcryption queries to algorithm $\mathcal{B}$. For a query (C, ϕ) to the RKA.Designcrypt oracle where $C = (\mu, \tau, \psi, tag, \sigma)$, algorithm $\mathcal{B}$ runs the following steps to respond.

- $H(\mu, tag) \neq H(g^c, tag^*)$. Algorithm $\mathcal{B}$ responds as in Phase 1.
- $H(\mu, tag) = H(g^c, tag^*)$, and $(\mu, \tau, \psi, \sigma) \neq (\mu^*, \tau^*, \psi^*, \sigma^*)$. If algorithm $\mathcal{B}$ accepts this signcryption, it means that algorithm $\mathcal{A}$ breaks the security of the CM-RKA security of our scheme, which we will analyze later. Therefore, algorithm $\mathcal{B}$ outputs $\perp$ except with negligible probability.
- $H(\mu, tag) = H(g^c, tag^*)$, $(\mu, \tau, \psi, \sigma) = (\mu^*, \tau^*, \psi^*, \sigma^*)$ and $\phi(x_R) \neq x_R$. If algorithm $\mathcal{B}$ accepts this signcryption, it means algorithm $\mathcal{A}$ can output $\phi \in \Phi$ such that $(\frac{\tau^*}{t^*} \cdot (\mu^*)^{-\phi(x_R)})^{\frac{1}{H(g^c, tag^*)}} \neq \perp$. That is, $\hat{e}(\theta', g) = \hat{e}(\mu, g_1)$, and to guarantee this,

$$(\frac{\tau^*}{t^*} \cdot (\mu^*)^{-x_R})^{\frac{1}{H(g^c, tag^*)}} = (\frac{\tau^*}{t^*} \cdot (\mu^*)^{-\phi(x_R)})^{\frac{1}{H(g^c, tag^*)}}$$
$$\Rightarrow x_R = \phi(x_R)$$

  should hold. Therefore, algorithm $\mathcal{B}$ outputs $\perp$ except with negligible probability.

  In fact this is the one-wayness property of the adaptive trapdoor relation, which on the other hand reflects how the key fingerprint property (a indispensable property to achieve RKA security according to the definitions given in [BC10, AHI11, Wee12]) works in our construction.

Note that (C, ϕ) is not allowed by the definition of the CC-RKA security game if $C = (\mu, \tau, \psi, tag, \sigma)$ satisfying $H(\mu, tag) = H(g^c, tag^*)$, $(\mu, \tau, \psi, \sigma) = (\mu^*, \tau^*, \psi^*, \sigma^*)$ and $\phi(x_R) = x_R$.

- **Output.** Algorithm $\mathcal{A}$ outputs a guess $d' \in \{0, 1\}$. If $d = d'$, algorithm $\mathcal{A}$ wins the game, and algorithm $\mathcal{B}$ outputs 1 indicating $Z = \hat{e}(g, g)^{abc}$. Otherwise, algorithm $\mathcal{B}$ outputs 0 indicating Z is random in G_T.

Let ϵ be the advantage that algorithm $\mathcal{A}$ breaks the CC-RKA security of the above game. We can see that if the input tuple of algorithm $\mathcal{B}$ is (g, g^a, g^b, g^c, Z) where $Z = \hat{e}(g, g)^{abc}$, then algorithm $\mathcal{A}$'s view of this simulation is identical to the real attack, and thus the probability of algorithm $\mathcal{A}$ in outputting $d' = d$ must satisfy $\Pr[d = d'] = 1/2 + \epsilon$. On the other hand, if the input tuple of algorithm $\mathcal{B}$ is (g, g^a, g^b, g^c, Z) where $Z \in G_T$, then the advantage of algorithm $\mathcal{A}$ is nil and thus $\Pr[d' = d] = 1/2$. In summary, the probability of algorithm $\mathcal{B}$ in solving the

decisional BDH problem is

$$\begin{aligned}\Pr[\mathcal{B}(g, g^a, g^b, g^c, Z)] &= 1/2 \cdot (1/2 + \epsilon) + 1/2 \cdot 1/2 \\ &= 1/2 + \epsilon/2.\end{aligned}$$

In the following, we will prove that the event Failure has negligible probability to occur due to the security of the computational DH problem.

Lemma 4.10 *If the computational BDH problem holds in G, G_T, then the event Failure happens with negligible probability.*

Proof. Given algorithm $\mathcal{A}$ for which the event Failure happens with noticeable probability, we construct algorithm $\mathcal{B}'$ that solves the computational BDH problem. Specifically, we consider the following game where algorithm $\mathcal{B}'$ solves the computational BDH problem. Suppose that algorithm $\mathcal{B}'$ is given a random tuple (g, g^a, g^b, g^c) as input and outputs $\hat{e}(g, g)^{abc}$.

- **Initialization.** The same as in Lemma 4.9.
- H_0**-query**. At any time algorithm $\mathcal{A}$ can query the random oracle on (μ, Y_1). Algorithm $\mathcal{B}'$ maintains a list L_{H_0} of tuples ((μ, Y_1), $H_0(\mu, Y_1)$) which is initially empty. When algorithm $\mathcal{A}$ issues a hash query on (μ, Y_1), algorithm $\mathcal{B}'$ performs the following steps to respond.
  - If $\hat{e}(Y_1, g) = \hat{e}(Y_R, g^c)$, algorithm $\mathcal{B}'$ solves the computational BDH problem immediately. To see this, we have

$$\begin{aligned} Y_1 = {Y_R}^c &= (g^{bc})^{-H(g^c, tag^*)} g^{c \cdot x_r} \\ \Rightarrow g^{bc} &= (\frac{Y_1}{g^{c \cdot x_r}})^{-\frac{1}{H(g^c, tag^*)}} \\ \Rightarrow \hat{e}(g, g)^{abc} &= \hat{e}(g^a, g^{bc}). \end{aligned}$$

  - If (μ, Y_1) already appears in list L_{H_0}, algorithm $\mathcal{B}'$ responds with $H_0(\mu, Y_1)$.
  - Otherwise, algorithm $\mathcal{B}'$ chooses a random $t_i \in Z_p^*$, sets $H_0(\mu, Y_1) = t_i$, and adds ((μ, Y_1), t_i) to list L_{H_0}.
- H'**-query**. At any time algorithm $\mathcal{A}$ can query the random oracle on (μ, τ, ψ, Y_1, Y_2, tag, m). Algorithm $\mathcal{B}'$ maintains a list $L_{H'}$ of tuples ((μ, τ, ψ, Y_1, Y_2, tag, m), $H'(\mu, \tau, \psi, Y_1, Y_2, tag, m)$) which is initially empty. When algorithm $\mathcal{A}$ issues a hash query on (μ, τ, ψ, Y_1, Y_2, tag, m), algorithm $\mathcal{B}'$ executes as follows to respond.

- If $\hat{e}(Y_1, g) = \hat{e}(Y_R, g^c)$, algorithm $\mathcal{B}'$ responds as in H_0 query.
- If $(\mu,\ \tau,\ \psi,\ Y_1,\ Y_2,\ tag,\ m)$ already appears in list $L_{H'}$, algorithm $\mathcal{B}'$ responds with $H'(\mu,\ \tau,\ \psi,\ Y_1,\ Y_2,\ tag,\ m)$.
- Otherwise, algorithm $\mathcal{B}'$ chooses a random $s_i \in Z_p^*$, sets $H'(\mu,\ \tau,\ \psi,\ Y_1,\ Y_2,\ tag,\ m) = s_i$, sends s_i to algorithm $\mathcal{A}$, and adds $((\mu,\ \tau,\ \psi,\ Y_1,\ Y_2,\ tag,\ m),\ s_i)$ to list $L_{H'}$.

- **Phase 1.** The same as in Lemma 4.9.

- **Challenge.** Algorithm $\mathcal{A}$ outputs two messages $M_0,\ M_1 \in G_T$ on which it wishes to be challenged. Algorithm $\mathcal{B}'$ chooses random $r^*,\ s^*,\ t^* \in Z_p^*$, a random $d \in \{0, 1\}$, and computes

$$\mu^* = g^{r^*}, \quad \tau^* = (Y_R \cdot {g_1}^{H(\mu^*, tag^*)})^{r^*} \cdot t^*,$$
$$\psi^* = \hat{e}(g_1, g_2)^{r^*} \cdot M_d, \quad \sigma^* = e^* - x_S \cdot s^*.$$

  Algorithm $\mathcal{B}'$ outputs the signcryption $C^* = (\mu^*,\ \tau^*,\ \psi^*,\ tag^*,\ \sigma^*)$, and adds $((\mu^*,\ \tau^*,\ \psi^*,\ {Y_R}^{r^*},\ {Y_R}^{x_S},\ tag^*,\ M_d),\ s^*)$ to list $L_{H'}$, $({Y_R}^{r^*},\ t^*)$ to list L_{H_0}.

- **Phase 2.** The same as in Lemma 4.9.

Lemma 4.9 ensures that as long as the event Failure does not happen, then our signcryption scheme preserves the defined CC-RKA security. Lemma 4.10 guarantees that as long as the event Failure does not happen, algorithm $\mathcal{B}'$ is the same as algorithm $\mathcal{B}$ such that algorithm $\mathcal{A}$ cannot differentiate between algorithm $\mathcal{B}$ and algorithm $\mathcal{B}'$.

This completes the proof of Theorem 4.8.

Theorem 4.11 *Assuming that the computational DL problem holds in G, then our signcryption scheme is CM-RKA secure regarding linear related-key deriving functions ϕ^+ in the random oracle.*

Proof. Supposing that algorithm $\mathcal{A}$ is an adversary that breaks the CM-RKA security of our signcrypiton scheme, we can construct algorithm $\mathcal{B}$ that solves the computational DL problem which is given as input a random tuple $(g,\ g^b)$ and outputs b.

- **Initialization.** Algorithm $\mathcal{B}$ runs as follows to simulate the system parameters.

1. Chooses collision resistant hash functions $H_0 : G^2 \to G$, $H : G^2 \to Z_p^*$.
2. Chooses random a, x_R, $x_s \in Z_p^*$, and computes $g_2 = g^a$, $Y_R = g^{x_R}$, $Y_S = (g^b)^{x_s}$. Note that $x_S = \log_g Y_S = b \cdot x_s$, which is unknown to algorithm $\mathcal{B}$.
3. Sets $g_1 = g^b$, and sends to algorithm $\mathcal{A}$ receiver R's public and private key pair (x_R, Y_R), sender S's public key Y_S and the public parameters (g, g_1, g_2, H_0, H, H') of which H' is a random oracle controlled by algorithm $\mathcal{B}$.

- **H'-query**. At any time algorithm $\mathcal{A}$ can query the random oracle on (μ, τ, ψ, Y_1, Y_2, tag, m). Algorithm $\mathcal{B}$ maintains a list $L_{H'}$ of tuples ((μ, τ, ψ, Y_1, Y_2, tag, m), $H'(\mu, \tau, \psi, Y_1, Y_2, tag, m)$) which is initially empty. When algorithm $\mathcal{A}$ issues a hash query on (μ, τ, ψ, Y_1, Y_2, tag, m), algorithm $\mathcal{B}$ executes as follows to respond.
  - If (μ, τ, ψ, Y_1, Y_2, tag, m) already appears in list $L_{H'}$, algorithm $\mathcal{B}$ responds with $H'(\mu, \tau, \psi, Y_1, Y_2, tag, m)$.
  - Otherwise, algorithm $\mathcal{B}$ chooses a random $s_i \in Z_q^*$, sets $H'(\mu, \tau, \psi, Y_1, Y_2, tag, m) = s_i$, sends s_i to algorithm $\mathcal{A}$, and adds ((μ, τ, ψ, Y_1, Y_2, tag, m), s_i) to list $L_{H'}$.
- **Phase 1.** Algorithm $\mathcal{A}$ adaptively issues the RKA signcryption queries to algorithm $\mathcal{B}$. Once algorithm $\mathcal{A}$ queries (m, ϕ) to the RKA.Signcrypt oracle, algorithm $\mathcal{B}$ runs the following steps to respond.
  1. Chooses a random $r \in Z_p^*$, and computes $\mu = g^r$.
  2. Chooses random σ, $s_i \in Z_p^*$, and computes
  $$tag = g^\sigma \cdot {Y_S}^{s_i}, \quad \psi = \hat{e}({g_1}^r, g_2) \cdot m,$$
  $$\tau = (Y_R \cdot {g_1}^{H(\mu, tag)})^r \cdot H_0(\mu, {Y_R}^r).$$
  3. Outputs the signcryption $C = (\mu, \tau, \psi, tag, \sigma)$, and adds (($\mu$, τ, ψ, ${Y_R}^r$, ${Y_R}^{x_S + \triangle}$, tag, m), s_i) to list $L_{H'}$.
- **Output.** Algorithm $\mathcal{A}$ outputs a signcryption $C^* = (\mu^*, \tau^*, \psi^*, tag^*, \sigma^*)$, and algorithm $\mathcal{B}$ designcrypts it following the designcryption algorithm. If this is a valid signcryption, from the Forking Lemma in [PS96], after a polynomial

replay attack of algorithm $\mathcal{A}$, we obtain two valid signcryptions (μ^*, τ^*, ψ^*, tag^*, σ^*) and (μ^*, τ^*, ψ^*, tag^*, σ) with $s_i \neq s^*$, from which we have

$$tag^* = g^{\sigma^*} \cdot {Y_S}^{s^*} = g^{\sigma} \cdot {Y_S}^{s_i}$$
$$\Rightarrow Y_S = g^{\frac{\sigma-\sigma^*}{s^*-s_i}} \Rightarrow b = \frac{\sigma - \sigma^*}{x_s \cdot (s^* - s_i)},$$

and algorithm $\mathcal{B}$ solves the computational DL problem.

This completes the proof of Theorem 4.11.

Theorem 4.12 *Assuming that the computational BDH assumption holds in G, G_T, then our signcryption scheme is ANON-RKA secure regarding linear related-key deriving functions ϕ^+ in the random oracle model.*

Proof. This proof is similar to that of Theorem 4.8. Assume that algorithm $\mathcal{A}$ is an adversary algorithm breaking the security of our signcryption scheme, and algorithm $\mathcal{B}$ is a challenger algorithm simulating the security game. Let H', H_0 be two random oracles controlled by algorithm $\mathcal{B}$, and let (μ^*, τ^*, ψ^*, tag^*, σ^*) be the challenge signcryption of a message M_d given to algorithm $\mathcal{A}$ by algorithm $\mathcal{B}$. Denote Failure by the event that algorithm $\mathcal{A}$ issues (μ^*, τ^*, ψ^*, Y_1, Y_2, M^*) to random oracle H', and (μ, Y_1) to random oracle H_0, where $\hat{e}(Y_1, g) = \hat{e}(Y_R, \mu^*)$. We firstly prove that if the event Failure does not occur, our signcryption scheme is ANON-RKA secure; then conclude it by that the event Failure has negligible probability to occur.

If there is an adversary algorithm $\mathcal{A}$ against the anonymity of our RKA secure signcryption scheme, then we can construct a challenge algorithm $\mathcal{B}$ that solves the computational BDH problem, which is given a random tuple (g, g^a, g^b, g^c) as input and outputs $Z = \hat{e}(g, g)^{abc}$.

- **Initialization.** Algorithm $\mathcal{B}$ runs as follows to simulate the system parameters.

  1. Chooses a collision resistant hash function $H : G^2 \rightarrow Z_p^*$.
  2. Chooses a random $e^* \in Z_p^*$, and computes $tag^* = g^{e^*}$.
  3. Chooses random $x_{S,0}$, $x_{S,1} \in Z_p^*$, and computes $Y_{S,0} = g^{x_{S,0}}$, $Y_{S,1} = g^{x_{S,1}}$.
  4. Chooses random $x_{r,0}$, $x_{r,1} \in Z_p^*$, and computes

$$Y_{R,0} = (g^b)^{-H(g^c, tag^*)} g^{x_{r,0}}, \quad Y_{R,1} = (g^b)^{-H(g^c, tag^*)} g^{x_{r,1}}.$$

Algorithm $\mathcal{B}$ sets $g_1 = g^b$, $g_2 = g^a$, and sends $(g, g_1, g_2, H_0, H, H', (x_{S,0}, Y_{S,0}), (x_{S,0}, Y_{S,1}), Y_{R,0}, Y_{R,1})$ to algorithm $\mathcal{A}$, where H_0, H' are the random oracles controlled by algorithm $\mathcal{B}$.

- **H_0-query**. At any time algorithm $\mathcal{A}$ can query the random oracle on (μ, Y_1). Algorithm $\mathcal{B}$ maintains a list L_{H_0} of tuples $((\mu, Y_1), H_0(\mu, Y_1))$ which is initially empty. When algorithm $\mathcal{A}$ issues a hash query on (μ, Y_1), algorithm $\mathcal{B}$ runs as follows to respond.
  - If $\hat{e}(Y_1, g) = \hat{e}(Y_{R,e}, g^c)$ for $e \in \{0, 1\}$, algorithm $\mathcal{B}$ solves the computational BDH problem immediately. To see this, we have

$$\begin{aligned} Y_1 = {Y_{R,e}}^c &= (g^{bc})^{-H(g^c, tag^*)} g^{c \cdot x_r} \\ \Rightarrow g^{bc} &= (\frac{Y_1}{g^{c \cdot x_r}})^{-\frac{1}{H(g^c, tag^*)}} \\ \Rightarrow \hat{e}(g, g)^{abc} &= \hat{e}(g^a, g^{bc}). \end{aligned}$$

  - If (μ, Y_1) already appears in list L_{H_0}, algorithm $\mathcal{B}$ responds with $H_0(\mu, Y_1)$.
  - Otherwise, algorithm $\mathcal{B}$ chooses a random $t_i \in Z_p^*$, sets $H_0(\mu, Y_1) = t_i$, and adds $((\mu, Y_1), t_i)$ to list L_{H_0}.
- **H'-query**. At any time algorithm $\mathcal{A}$ can query the random oracle on $(\mu, \tau, \psi, Y_1, Y_2, tag, m)$. Algorithm $\mathcal{B}$ maintains a list $L_{H'}$ of tuples $((\mu, \tau, \psi, Y_1, Y_2, tag, m), H'(\mu, \tau, \psi, Y_1, Y_2, tag, m))$ which is initially empty. When algorithm $\mathcal{A}$ issues a hash query on $(\mu, \tau, \psi, Y_1, Y_2, tag, m)$, algorithm $\mathcal{B}$ performs the following steps to respond.
  - If $\hat{e}(Y_1, g) = \hat{e}(Y_{R,e}, g^c)$ for $e \in \{0, 1\}$, algorithm $\mathcal{B}$ responds as in H_0 query.
  - If $(\mu, \tau, \psi, Y_1, Y_2, tag, m)$ already appears in list $L_{H'}$, algorithm $\mathcal{B}$ responds with $H'(\mu, \tau, \psi, Y_1, Y_2, tag, m)$.
  - Otherwise, algorithm $\mathcal{B}$ chooses a random $s_i \in Z_p^*$, sets $H'(\mu, \tau, \psi, Y_1, Y_2, tag, m) = s_i$, sends s_i to algorithm $\mathcal{A}$, and adds $((\mu, \tau, \psi, Y_1, Y_2, tag, m), s_i)$ to list $L_{H'}$.
- **Phase 1.** Algorithm $\mathcal{A}$ chooses (x_S, Y_R), where $x_S \in \{x_{S,0}, x_{S,1}\}$, $Y_R \in \{Y_{R,0}, Y_{R,1}\}$, and adaptively issues the RKA designcryption queries to algorithm $\mathcal{B}$.

For a query (C, ϕ) to the RKA.Designcrypt oracle where $C = (\mu, \tau, \psi, tag, \sigma)$, algorithm $\mathcal{B}$ executes the following procedure to respond.

1. Algorithm $\mathcal{B}$ computes θ' with $\phi(x_R)$. To see how algorithm $\mathcal{B}$ obtains θ without x_R, we rewrite τ such that

$$\begin{aligned}\frac{\tau}{t_i} &= (Y_R \cdot {g_1}^{H(\mu,tag)})^r \\ &= \mu^{-b\cdot H(g^c,tag^*)+x_r+b\cdot H(\mu,tag)} \\ &= (\mu^b)^{H(\mu,tag)-H(g^c,tag^*)} \cdot \mu^{x_r} \\ &= \theta^{H(\mu,tag)-H(g^c,tag^*)} \cdot \mu^{x_r} \\ \Rightarrow \theta &= (\frac{\tau}{t_i \cdot \mu^{x_r}})^{\frac{1}{H(\mu,tag)-H(g^c,tag^*)}}.\end{aligned}$$

   On the other hand,

$$\begin{aligned}\theta' &= (\frac{\tau}{t_i} \cdot \mu^{-(x_R+\triangle)})^{\frac{1}{H(\mu,tag)}} \\ &= ((\frac{\tau}{t_i} \cdot \mu^{-\triangle}) \cdot \mu^{-x_R})^{\frac{1}{H(\mu,tag)}} \\ &= \theta \cdot (\mu^{-\triangle})^{\frac{1}{H(\mu,tag)}}.\end{aligned}$$

2. If $\hat{e}(\theta', g) = \hat{e}(\mu, g_1)$, algorithm $\mathcal{B}$ outputs $m = \psi/\hat{e}(\theta', g_2)$. Otherwise, it outputs $\perp$.

- **Challenge.** Algorithm $\mathcal{A}$ outputs a message $M^* \in G_T$ on which it wishes to be challenged. Algorithm $\mathcal{B}$ chooses random s^*, $t^* \in Z_p^*$, d, $e \in \{0, 1\}$, $Z \in G_T$, sets $\mu^* = g^c$, and computes

$$\tau^* = (g^c)^{x_{r,e}} \cdot t^*, \quad \psi^* = Z \cdot M^*, \quad \sigma^* = e^* - x_{S,d} \cdot s^*.$$

  Algorithm $\mathcal{B}$ outputs the signcryption $C^* = (\mu^*, \tau^*, \psi^*, tag^*, \sigma^*)$, and adds $((\mu^*, \tau^*, \psi^*, (Y_{R,e})^c, {Y_{R,e}}^{x_{S,d}}, tag^*, M^*), s^*)$ to list $L_{H'}$, $((g^c, (Y_{R,e})^c), t^*)$ to list L_{H_0}.

- **Phase 2.** Algorithm $\mathcal{A}$ chooses (x_S, Y_R), where $x_S \in \{x_{S,0}, x_{S,1}\}$, $Y_R \in \{Y_{R,0}, Y_{R,1}\}$, and adaptively issues the RKA designcryption queries to algorithm $\mathcal{B}$. For a query (C, ϕ) to the RKA.Designcrypt oracle where $C = (\mu, \tau, \psi, tag, \sigma)$, algorithm $\mathcal{B}$ performs the following steps to make the response.

  – $H(\mu, tag) \neq H(g^c, tag^*)$. Algorithm $\mathcal{B}$ responds as in Phase 1.

– $H(\mu, tag) = H(g^c, tag^*)$, and $(\mu, \tau, \psi, \sigma) \neq (\mu^*, \tau^*, \psi^*, \sigma^*)$. If algorithm $\mathcal{B}$ accepts this signcryption, it means algorithm $\mathcal{A}$ breaks the security of the CM-RKA security of our scheme. Therefore, algorithm $\mathcal{B}$ outputs $\perp$ except with negligible probability.

– $H(\mu, tag) = H(g^c, tag^*)$, $(\mu, \tau, \psi, \sigma) = (\mu^*, \tau^*, \psi^*, \sigma^*)$ and $\phi(x_R) \neq x_R$. If algorithm $\mathcal{B}$ accepts this signcryption, it means that algorithm $\mathcal{A}$ can output $\phi \in \Phi$ such that

$$(\frac{\tau^*}{t^*} \cdot (\mu^*)^{-\phi(x_R)})^{\frac{1}{H(g^c, tag^*)}} \neq \perp.$$

That is, $\hat{e}(\theta', g) = \hat{e}(\mu, g_1)$. To guarantee this,

$$(\frac{\tau^*}{t^*} \cdot (\mu^*)^{-x_R})^{\frac{1}{H(g^c, tag^*)}} = (\frac{\tau^*}{t^*} \cdot (\mu^*)^{-\phi(x_R)})^{\frac{1}{H(g^c, tag^*)}}$$
$$\Rightarrow x_R = \phi(x_R)$$

should hold.

Therefore, algorithm $\mathcal{B}$ outputs $\perp$ except with negligible probability.

Analysis. Algorithm $\mathcal{A}$ has negligible probability to issue (g^c, Y_1) to the random oracle H_0 such that $\hat{e}(Y_1, g) = \hat{e}(g^c, Y_{R,e})$ for $e \in \{0, 1\}$. If so, algorithm $\mathcal{B}$ can solve the computational BDH problem immediately. On the other hand, without the value of $H_0(\mu, {Y_{R,e}}^c)$, algorithm $\mathcal{A}$ has no idea of the identity of receiver R from the challenge signcryption C^*. Likewise, algorithm $\mathcal{A}$ has negligible probability to issue $(\mu^*, \tau^*, \psi^*, Y_1, Y_2, M^*)$ to random oracle H' such that $\hat{e}(Y_1, g) = \hat{e}(g^c, Y_{R,e})$; otherwise, algorithm $\mathcal{B}$ can solve the computational BDH problem immediately. Obviously, without the value of $H'(\mu^*, \tau^*, \psi^*, {Y_{R,e}}^c, {Y_{R,e}}^{x_{S,d}}, M^*)$, algorithm $\mathcal{A}$ cannot distinguish the identity of sender S from the challenge signcryption C^* via verification.

This completes the proof of Theorem 4.12.

4.5 Summary

Traditional security notions cannot meet the requirements in scenarios where the adversaries might get some partial information about the keys by certain physical methods. For instance, under related-key attacks, an adversary can subsequently observe the outcome of the cryptographic primitives under a series of modified keys

(related to the original key). Motivated by this observation, in this chapter, we focus on establishing secure signcryption schemes under related-key attacks.

Suppose that the parties (both the sender and the receiver) involved are honest in the system. Following the results in [BCM11, AHI11], we define the security notions for a signcryption system resistant to related-key attacks while maintaining the CCA and CMA security, and call them as outsider chosen ciphertext attack security under related-key attacks (OCC-RKA security) and outsider chosen message attack security under related-key attacks (OCM-RKA security), where an adversary is allowed to query the designcryption oracle on linear shifts of the private key of the receiver, and the signcryption oracle on linear shifts of the private key of the sender. In addition, we put forward a specific RKA secure signcryption scheme based on a RKA secure public-key encryption scheme [Wee12].

Furthermore, we extend the above security model for RKA secure signcryption with anonymity, which we call outsider anonymity under related-key attacks (OANON-RKA security), where the ciphertext should be anonymous to anyone except the real receiver. Fortunately, with a slight improvement to our original signcryption scheme, we can obtain an outsider anonymous signcryption scheme which is secure against related-key attacks.

After that, we consider the RKA security in signcryption one step further by removing the requirement that both the sender and the receiver are honest, and call the corresponding security definitions as CC-RKA security, CM-RKA security and ANON-RKA security. Also, we put forward a specific construction of RAK secure signcryption under this improved security model, and prove its CC-RKA security, CM-RKA security and ANON-RKA security in the random oracle model.

Some results of this chapter have been published in "The Seventh International Conference on Provable Security (ProvSec 2013)" and "The Computer Journal".

Chapter 5

Linear Related-Key Attack Secure Public-Key Encryption Schemes

Public-key encryption, also known as asymmetric encryption, refers to a cryptographic algorithm which requires two separate keys, one private and the other public. Although different, the two elements of this key pair are mathematically linked. The public key is used to encrypt plaintexts whereas the private key is used to decrypt ciphertexts. The goal of this chapter is to explore the methods of achieving related-key attack security in the public-key encryption schemes.

5.1 Introduction

In the traditional security model, it is assumed that the adversary is isolated from the internal states of the honest communication parties. However, with the development of information technologies, the security of cryptographic algorithms in modern cryptography is analyzed in the black-box model, where an adversary may view the algorithm's inputs and outputs, but the private key as well as all the internal computation remains perfectly hidden. Unfortunately, this idealized assumption is often hard to satisfy in real systems. In many situations, the adversary might get some partial information about private keys through methods which are not anticipated by the designer of the system and, correspondingly, not taken into account when arguing its security. Take related-key attack (RKA) [BCM11, GLM$^+$04] as an example of such kind of attacks, where an adversary tampers with the private key stored in a cryptographic hardware device, and observes the result of the cryptographic primitive under this modified private key. Here the key could be a signing key of a certificate authority or a decryption key of an encryption scheme. In related-key attacks, the adversary attempts to break an encryption scheme by invoking it with several private keys satisfying some known relations. Due to this drawback, in the

last two decades, the security requirement has been relaxed to capture under the scenarios where some information of the keys is leaked to the adversary.

5.1.1 Related-Key Attack Security for Public-Key Encryption

Wee [Wee12] presented the first public-key encryption schemes resistant to linear related-key attacks under standard assumptions and in the standard model from adaptive trapdoor relations via strong one-time signatures, of which the security is analogous to those for obtaining chosen-ciphertext attack (CCA) security from extractable hash proofs [Wee10] and trapdoor functions [KMO10]. Bellare, Paterson and Thomson [BPT12] provided a framework to enable the construction of identity-based encryption schemes that are secure under related-key attacks. Based on this, they constructed the RKA secure schemes for public-key encryption in the standard model.

Informally, a public-key encryption scheme is secure under related-key attacks, then it is chosen-ciphertext attack secure even when the adversary obtains partial information of the message in the scheme under the modified private keys of the adversary. This is modelled by providing the adversary with access to a related-key attack decryption oracle: the adversary can query the decryption oracle with any function (ϕ, C), and then receive the decryption result of C under $\phi(sk)$, where sk is the private key in the system[1]. The adversary can query the related-key attack decryption oracle adaptively except those where the decryption of a ciphertext C with the private key $\phi(sk)$ equals the decryption of the challenge ciphertext C^* with the original private key sk.

In this chapter, we study public-key encryption schemes secure against related-key attacks, and focus on exploring other ways that are different from those in [BPT12, Wee12] to achieve RKA security in public-key encryption schemes under the definition given in [BCM11]. Because related-key attack is on the private key, we consider related-key attacks for an underlying chosen ciphertext attack secure public-key encryption scheme, and the decryption oracle does not proceed if the given ciphertext matches the challenge ciphertext and the related private key equals the original one.

[1]Note that the related-key deriving functions can be chosen depending on the public key, which is known to the adversary.

We view the system consisting of algorithms, public parameters, encryption key and decryption key, of which the keys are subject to related-key attacks. The public parameters are fixed beforehand and independent of users, and the tampering with them is infeasible or could be easily detected.

5.1.2 Our Results

We present several public-key encryption schemes secure against linear related-key attacks [Wee12] in the standard model of which the private keys are composed of more than one element. Specifically, we obtain public-key encryption schemes secure against linear related-key attacks:

1. from the Cramer-Shoup public-key encryption scheme [CS01];
2. from a selective-identity secure identity-based encryption scheme in [BB04];
3. from a public-key encryption scheme in [Kil06].

Our technique is very simple that we try to make use of the randomness to distort the result from the related-key decryption oracle in an underlying public-key encryption scheme. In this way, the challenge ciphertext becomes an invalid ciphertext from the adversary's view under any private key $\phi(sk) \neq sk$.

In the first construction, we firstly describe an attack different from the one described in [Wee12] on the public-key encryption system of Cramer and Shoup [CS01], where we change all parts of the private key with the same linear shift function ϕ while in [Wee12] the related-key deriving function only changes one part of the private keys. Then, on the practical side, with a minor modification to the basic cryptosystem of Cramer and Shoup [CS01], we obtain an efficient scheme that is RKA secure based on the decisional Diffie-Hellman assumption.

However, the assumption that the modification to every component of the private key is the same is not practical. In the second and third constructions, we strengthen the RKA security by allowing an adversary to adaptively tamper with the different parts of a private key. We firstly show how an adversary breaks a system if given the power to modify every part of a multi-element private key differently, and then we propose a public-key encryption scheme achieving this RKA security requirement from an identity-based encryption scheme based on bilinear pairings. In addition, based on a public-key encryption scheme, we put forth another public-key encryption scheme that achieves this RKA security without pairings.

5.1.3 Organization

The remainder of this chapter is organized as follows. In Section 5.2, we review the concepts associated with this work and the security model of RKA secure public-key key encryption systems. In Section 5.3, after the analysis of a linear attack on the Cramer-Shoup cryptosystem [CS01], we propose an efficient public-key encryption scheme resistant to related-key attacks, and prove its security under the hardness of the DDH problem. In Section 5.4, we point out another related-key attack, different from the one pointed in Section 5.3, on an existing scheme, and present a public-key encryption scheme with the RKA security of which the CC-RKA security is based on the decisional BDH assumption. In Section 5.5, we further propose a public-key encryption scheme secure against related-key attacks without pairings, and demonstrate the CC-RKA security under the decisional LIN assumption. Finally, we conclude this chapter in Section 5.6.

5.2 Modeling Related-Key Attacks under Public-Key Encryption

In this section, we define the notion of a chosen-ciphertext attack; in addition, we present a natural extension of this notion to the setting of related-key attacks, as introduced by Bellare, Cash and Miller [BCM11]. Also, we introduce some notions about related-key attacks, as proposed in [AHI11].

Related-key deriving functions. Our definition follows the notion of related-key deriving functions given in [BK03]. Briefly, a class Φ of related-key deriving functions $\phi : sk \to sk$ is a finite set of functions with the same domain and range, which map a key to a related key. Additionally, Φ should allow an effective membership test, and ϕ should be efficiently computable. Note that in our concrete constructions, we only consider the class Φ^+ as the linear shifts.

CC-RKA Security. A public-key encryption scheme $\mathcal{PKE}$ = (Keygen, Encrypt, Decrypt) is Φ-CC-RKA secure if for a stateful adversary algorithm $\mathcal{A}$, the advantage in the following game is negligible in the security parameter λ.

1. $pars \leftarrow \mathrm{PG}(1^\lambda)$.

2. $(sk, pk) \leftarrow \mathrm{KG}(1^\lambda)$.
3. $(m_0, m_1) \leftarrow \mathcal{A}^{\mathrm{RKA.Dec}(sk,\cdot,\cdot)}(pk)$ such that $|m_0| = |m_1|$.
4. $C^* \leftarrow \mathrm{Enc}(pars, pk, m_d)$ where $d \in \{0, 1\}$.
5. $d' \leftarrow \mathcal{A}^{\mathrm{RKA.Dec}(sk,\cdot,\cdot)}(C^*)$.
6. Output d'.

Here RKA.Dec$(sk, \cdot, \cdot)$ is an oracle that on an input (ϕ, C), it returns $\mathrm{Dec}(\phi(sk), C)$. We constrain that algorithm $\mathcal{A}$ can only make queries (ϕ, C) such that $\phi \in \Phi$ and $(\phi(sk), C) \neq (sk, C^*)$.

We define the advantage of algorithm $\mathcal{A}$ as

$$\mathbf{Adv}^{\Phi\text{-CC-RKA}}_{\mathcal{PKE},\mathcal{A}}(\lambda) \stackrel{\mathrm{def}}{=} |\Pr[d = d'] - 1/2|.$$

5.3 An Efficient Construction of RKA Secure Public-Key Encryption from the Cramer-Shoup Scheme

In this section, we put forward our construction based on the Cramer-Shoup cryptosystem [CS01], and present its security proof under the DDH assumption. To begin with, we describe a simple linear related-key attack on the Cramer-Shoup public-key encryption scheme, which to some extent illustrates the technical obstacles in achieving the RKA security.

5.3.1 A Related-Key Attack on the Cramer-Shoup Cryptosystem

The family Φ^+. Any function $\phi : Z_p \rightarrow Z_p$ in this class is indexed by $\triangle \in Z_p$, where $\phi_\triangle(sk) := sk + \triangle$.

We constrain that if sk is composed of several elements as $(sk_1, \ldots, sk_n)$ with $n \in \mathbb{N}$, for any sk_i where $i \in \{1, \ldots, n\}$, $\phi_\triangle(sk_i) := sk_i + \triangle$ with $\triangle \in Z_p^n$.

We point out a linear related-key attack on the CCA secure encryption scheme based on the DDH assumption proposed by Cramer and Shoup [CS01]. The details of the Cramer-Shoup public-key encryption scheme is given as follows.

- Key generation. Choose random g, $f \in G$, x, y, a, b, α, $\beta \in Z_p$, a collision resistant hash function $H : G^3 \to Z_p$, and sets

$$u_1 = g^x f^y, \quad u_2 = g^a f^b, \quad u_3 = g^\alpha f^\beta.$$

  The public key is $PK = (g, f, u_1, u_2, u_3, H)$, and the private key is $SK = (x, y, a, b, \alpha, \beta)$.

- Encryption. To encrypt message $M \in G$,

  1. choose random $r \in Z_p$, and set

$$C_1 = g^r, \quad C_2 = f^r, \quad C_3 = {u_1}^r \cdot M.$$

  2. compute $t = H(C_1, C_2, C_3)$, $C_4 = (u_2 {u_3}^t)^r$.
  3. output the ciphertext $C = (C_1, C_2, C_3, C_4)$.

- Decryption. To decrypt a ciphertext $C = (C_1, C_2, C_3, C_4)$,

  1. compute $t = H(C_1, C_2, C_3)$, and output $\perp$ if

$$C_4 \neq {C_1}^{a+t\alpha} {C_2}^{b+t\beta}.$$

  2. otherwise, output $M = C_3 \cdot {C_1}^{-x} \cdot {C_2}^{-y}$.

The attack. Suppose we are given a valid ciphertext (C_1, C_2, C_3, C_4) of some message M. We can recover M by making decryption queries to RKA.Decrypt oracle on related secret keys via the following attack. For any $\triangle \in Z_p$, we change the secret key $(x, y, a, b, \alpha, \beta)$ to $(x+\triangle, y+\triangle, a+\triangle, b+\triangle, \alpha+\triangle, \beta+\triangle)$, then $(C_1, C_2, C_3, C_4 \cdot (C_1 \cdot C_2)^{\triangle + t\cdot\triangle})$ can be decrypted to $M \cdot (C_1 \cdot C_2)^{-\triangle}$ under the modified secret keys. As C_1, C_2 and $\triangle$ are known to us, we can obtain M easily by computing $M \cdot (C_1 \cdot C_2)^{-\triangle} \cdot (C_1 \cdot C_2)^{\triangle}$.

Obviously in the above cases, message M can be easily recovered given the output of the decryption algorithm on the modified secret keys.

5.3.2 Our Construction

Let G be a group of prime order p. Below we present a public-key encryption scheme which is CCA secure under the linear related-key attacks.

- Key generation. Choose random elements g, f, $h \in G$, x, y, a, b, α, β, $\gamma \in Z_p$, a collision resistant hash function $H : G^4 \to Z_p$, and sets

$$u_1 = g^x f^y, \quad u_2 = g^a f^b, \quad u_3 = g^\alpha f^\beta, \quad v = h^\gamma.$$

  The public key is $PK = (g, h, f, u_1, u_2, u_3, v)$, and the secret key is $SK = (x, y, a, b, \alpha, \beta, \gamma)$.

- Encryption. To encrypt message $M \in G$,

  1. choose random elements r, $r' \in Z_p$, and set

$$C_1 = g^r v^{r'}, \quad C_2 = f^r v^{r'}, \quad C_3 = h^{r'}, \quad C_4 = {u_1}^r \cdot M.$$

  2. compute $t = H(C_1, C_2, C_3, C_4)$, $C_5 = (u_2 {u_3}^t)^r$.
  3. output the ciphertext $C = (C_1, C_2, C_3, C_4, C_5)$.

- Decryption. To decrypt a ciphertext $C = (C_1, C_2, C_3, C_4, C_5)$,

  1. compute $t = H(C_1, C_2, C_3, C_4)$, and output $\perp$ if

$$C_5 \neq (C_1 \cdot {C_3}^{-\gamma})^{a+t\alpha} (C_2 \cdot {C_3}^{-\gamma})^{b+t\beta}.$$

  2. otherwise, output M as

$$M = C_4 \cdot (C_1 \cdot {C_3}^{-\gamma})^{-x} \cdot (C_2 \cdot {C_3}^{-\gamma})^{-y}.$$

Correctness. For any sequence of the key generation and encryption algorithms, it holds that

$$\begin{aligned}
(u_2 {u_3}^t)^r &= (C_1 \cdot {C_3}^{-\gamma})^{a+t\alpha} (C_2 \cdot {C_3}^{-\gamma})^{b+t\beta} \\
&= (g^a f^b (g^\alpha f^\beta)^t)^r, \\
M &= C_4 \cdot (C_1 \cdot {C_3}^{-\gamma})^{-x} \cdot (C_2 \cdot {C_3}^{-\gamma})^{-y} \\
&= C_4 \cdot (g^x f^y)^{-r},
\end{aligned}$$

and therefore the decryption algorithm is always correct.

Remarks. Note that compared to the scheme proposed in [Wee12], our construction is more efficient. The CCA-RKA secure public-key encryption schemes in [Wee12]

are built from adaptive trapdoor relations [KMO10] to generate a *tag* for every ciphertext via a strong one-time signature scheme, which implies a trick in it such that the adversary cannot obtain more information if *tag* of a ciphertext C equals tag^* of the challenge ciphertext C^*, not to mention $C = C^*$; while in our construction, we use the Cramer-Shoup public-key encryption scheme [CS01] as the basis, and the strong one-time signature schemes are replaced by the ciphertext to generate *tag*, such that the RKA.Decrypt oracle will still not facilitate the adversary when a given ciphertext C matches the challenge one C^*, as long as SK is not equal to $\phi(SK)$ for any $\phi \in \Phi$.

5.3.3 Security Proof

Theorem 5.1 *Assuming the hardness of the decisional DH problem, the above public-key encryption scheme is secure in the CC-RKA security game regarding the linear related-key deriving function ϕ^+.*

Proof. The proof of security is based on augmenting the proof of Cramer and Shoup with the ideas of generating a generic construction. Specifically, we show that for any adversary algorithm $\mathcal{A}$ that breaks the security of the scheme, we can build a challenger algorithm $\mathcal{B}$ that can distinguish between a DH instance and a non-DH instance, which is given a random tuple $(g, f, Z_1 = g^r, Z_2 = f^r) \in G^4$ as input.

- **Setup.** Algorithm $\mathcal{B}$ chooses random elements $h \in G$, $x, y, a, b, \alpha, \beta, \gamma \in Z_p$, and sets

  $$u_1 = g^x f^y, \quad u_2 = g^a f^b, \quad u_3 = g^\alpha f^\beta, \quad v = h^\gamma.$$

  Then it chooses a collision resistant hash function H: $G^4 \to Z_p$.

  Algorithm $\mathcal{B}$ sends the public key $PK = (g, h, f, u_1, u_2, u_3, v)$ to algorithm $\mathcal{A}$, and keeps the private key $SK = (x, y, a, b, \alpha, \beta, \gamma)$.

- **Phase 1.** Algorithm $\mathcal{A}$ queries (ϕ, C) to the RKA.Decrypt oracle. Algorithm $\mathcal{B}$ responds using the private key $\phi(SK)$.

- **Challenge.** Algorithm $\mathcal{A}$ outputs two messages M_0, M_1 on which it wishes to be challenged. Algorithm $\mathcal{B}$ chooses a random bit $d \in \{0, 1\}$, a random element $r' \in Z_p$, and then responds with the ciphertext $C^* = (C_1^*, C_2^*, C_3^*, C_4^*,$

C_5^*) where

$$C_1^* = Z_1 v^{r'}, \quad C_2^* = Z_2 v^{r'}, \quad C_4^* = {Z_1}^x {Z_2}^y \cdot M_d,$$
$$C_3^* = h^{r'}, \quad t^* = H(C_1^*, C_2^*, C_3^*, C_4^*), \quad C_5^* = {Z_1}^{a+\alpha t^*} {Z_2}^{b+\beta t^*}.$$

- **Phase 2.** Algorithm $\mathcal{A}$ continues to adaptively issue queries (ϕ, C) to the RKA.Decrypt oracle.
  - If $\phi(SK) = SK$ and $C = C^*$, such queries are ruled out by the definition of the CC-RKA security game, so algorithm $\mathcal{B}$ responds with $\perp$.
  - Otherwise, algorithm $\mathcal{B}$ responds as in Phase 1.
- **Output.** Algorithm $\mathcal{A}$ outputs a guess $d' \in \{0, 1\}$. If $d' = d$, algorithm $\mathcal{B}$ outputs 1; otherwise, algorithm $\mathcal{B}$ outputs 0.

Obviously, if (g, f, Z_1, Z_2) is a DH instance, then the simulation will be identical to the actual attack, such that algorithm $\mathcal{A}$ has a non-negligible advantage in outputting the bit $d' = d$.

Lemma 5.2 *If (g, f, Z_1, Z_2) is a DH instance then algorithm $\mathcal{A}$'s view is identical to the actual attack.*

Proof. The actual attack and simulated attack are identical except for the challenge ciphertext. It remains to be proved that the challenge ciphertext has the correct distribution when (g, f, Z_1, Z_2) is a DH instance. Actually, in this case, for a random $r \in Z_p$, $Z_1 = g^r$ and $Z_2 = f^r$, the ciphertext $C^* = (C_1^*, C_2^*, C_3^*, C_4^*, C_5^*)$ is as it should be. Assume that the advantage of algorithm $\mathcal{A}$ in breaking the CC-RKA security of the above scheme is ϵ, then we can see that the probability of algorithm $\mathcal{A}$ in outputting the bit $d = d'$ could be $1/2 + \epsilon$.

Next, we show that if (g, f, Z_1, Z_2) is a non-DH instance, then algorithm $\mathcal{A}$ has a negligible advantage in outputting the bit $d' = d$. We assume that (g, f, Z_1, Z_2) is a non-DH instance, where $\log_g Z_1 = r_1$, $\log_f Z_2 = r_2$, and $r_1 \neq r_2$.

Let (C_1^*, C_2^*, C_3^*, C_4^*) be the challenge ciphertext given to algorithm $\mathcal{A}$ by algorithm $\mathcal{B}$. We use Failure to denote the event where for RKA decryption queries (ϕ, C) it holds that (C_1, C_2, C_3, C_4) $\neq$ (C_1^*, C_2^*, C_3^*, C_4^*), and $H(C_1, C_2, C_3, C_4) = H(C_1^*, C_2^*, C_3^*, C_4^*)$. Note that the event Failure has negligible probability to occur

because the hash function H is collision resistant. We say that a ciphertext C is invalid if for any $\triangle \in Z_p^n$,

$$\log_g \frac{C_1}{{C_3}^{\gamma+\triangle}} \neq \log_f \frac{C_2}{{C_3}^{\gamma+\triangle}}.$$

Below we prove that algorithm $\mathcal{A}$ has a negligible advantage in outputting the bit $d' = d$ if the event Failure does not happen. Specifically, we perform it in two cases:

- if the event Failure does not happen, then the RKA decryption oracle rejects all invalid ciphertexts except with negligible probability;
- if the RKA decryption oracle rejects all invalid ciphertexts, then algorithm $\mathcal{A}$ has a negligible advantage in outputting the bit $d' = d$.

We conclude by the fact that the event Failure occurs with negligible probability.

Lemma 5.3 *If (g, f, Z_1, Z_2) is a non-DH instance and the event Failure does not happen, then the RKA decryption algorithm rejects all invalid ciphertexts except with negligible probability.*

Proof. The probability of the invalid ciphertexts happening in our security game is analogous to that in the Cramer-Shoup public-key encryption scheme [CS01] except that for the RKA decryption oracles, some invalid ciphertexts which will be rejected in the security game of the Cramer-Shoup scheme will be accepted in our security game. Suppose that algorithm $\mathcal{A}$ is given the public key $PK = (g, h, f, u_1, u_2, u_3, v)$, and the challenge ciphertext $C^* = (C_1^*, C_2^*, C_3^*, C_4^*, C_5^*)$. We prove this lemma via considering $(a, b, \alpha, \beta) \in Z_p$ from algorithm $\mathcal{A}$'s point of view, such that for $k = \log_g f$, (a, b, α, β) is uniformly random subject to

$$\begin{cases} \log_g u_2 = a + kb \\ \log_g u_3 = \alpha + k\beta \\ \log_g C_5^* = r_1 a + r_2 kb + t^* r_1 \alpha + t^* r_2 k\beta \end{cases}.$$

Note that algorithm $\mathcal{A}$ learns nothing on (a, b, α, β) by querying valid ciphertexts to the decryption oracle. Actually, from submitting a valid ciphertext, algorithm $\mathcal{A}$ learns only a linear combination of the constraint $\log_g u_1 = x + ky$, which is known from the public key.

We denote $(C_1, C_2, C_3, C_4, C_5) \neq (C_1^*, C_2^*, C_3^*, C_4^*, C_5^*)$ as the first invalid ciphertext queried by algorithm $\mathcal{A}$, where $C_1 = g^{r_1}v^{r'}$, $C_2 = f^{r_2}v^{r'}$, $r_1 \neq r_2$, and $t = H(C_1, C_2, C_3, C_4)$. In this case, there are three cases we need to take into consideration.

- $(C_1, C_2, C_3, C_4) \neq (C_1^*, C_2^*, C_3^*, C_4^*)$ and $t = t^*$. This is impossible since we assume that the event Failure does not happen.

  Note that the event Failure will never happen because the hash function H in our construction is collision resistant.

- $(C_1, C_2, C_3, C_4) \neq (C_1^*, C_2^*, C_3^*, C_4^*)$ and $t \neq t^*$. In this case, if the RKA decryption algorithm accepts the invalid ciphertext, we obtain

$$\begin{cases} \log_g u_2 = a + kb \\ \log_g u_3 = \alpha + k\beta \\ \log_g C_5^* = r_1 a + r_2 kb + t^* r_1 \alpha + t^* r_2 k\beta \\ \log_g C_5 = r_1'(a + \triangle) + r_2' k(b + \triangle) + t r_1'(\alpha + \triangle) + t r_2' k(\beta + \triangle) \end{cases},$$

  where $w = \log_g h$.

  These equations are linearly independent as long as $k^2(r_1 - r_2)(r_1' - r_2')(t - t^*) \neq 0$, so algorithm $\mathcal{A}$ can be used to guess (a, b, α, β). Therefore, the probability that the decryption algorithm accepts the first invalid ciphertexts is, at most, $1/p$.

- $(C_1, C_2, C_3, C_4) = (C_1^*, C_2^*, C_3^*, C_4^*)$, $t = t^*$ but $C_5 \neq C_5^*$, In this case, if the RKA decryption algorithm accepts the invalid ciphertext, we obtain

$$\begin{cases} \log_g u_2 = a + kb \\ \log_g u_3 = \alpha + k\beta \\ \log_g C_5^* = r_1 a + r_2 kb + t^* r_1 \alpha + t^* r_2 k\beta \\ \log_g C_5 = r_1(a + \triangle) + r_2 k(b + \triangle) + t^* r_1(\alpha + \triangle) + t^* r_2 k(\beta + \triangle) \\ \qquad\qquad - r' w \triangle(a + \triangle + t^*(\alpha + \triangle) + b + \triangle + t^*(\beta + \triangle)) \end{cases},$$

  where $w = \log_g h$.

  These equations are linearly independent as long as $\triangle \neq 0$, which is ruled out by the definition of the CC-RKA security, so algorithm $\mathcal{A}$ can be used to guess (a, b, α, β).

For all the subsequent invalid decryption queries, the above analysis holds except that each time the RKA decryption oracle rejects an invalid ciphertext, algorithm $\mathcal{A}$ can rule out one more value of (a, b, α, β).

Lemma 5.4 *If (g, f, Z_1, Z_2) is a non-DH instance and the RKA decryption algorithm rejects all invalid ciphertexts, then algorithm $\mathcal{A}$ has a negligible advantage in outputting the bit $d' = d$.*

Proof. We prove this lemma by considering the distribution of $(x, y, \gamma) \in Z_p$ from the view of algorithm $\mathcal{A}$. Algorithm $\mathcal{A}$ is given the public key $PK = (g, h, f, u_1, u_2, u_3, v)$, such that algorithm $\mathcal{A}$'s point of view, (x, y, γ) is uniformly random subject to $\log_g u_1 = x + ky$ where $k = \log_g f$ and $\log_g v = k'\gamma$ where $k' = \log_g h$. We suppose that the RKA decryption algorithm rejects all invalid ciphertexts, and note that by querying valid ciphertexts to the RKA decryption oracle, algorithm $\mathcal{A}$ does not learn any more information about (x, y, γ) except the relations of the constraint $\log_g u_1 = x + ky$ and $\log_g v = k'\gamma$. Hence, algorithm $\mathcal{A}$ cannot learn any information about (x, y, γ) through the RKA decryption queries.

Let $C_1 = Z_1 v^{r'}$, $C_2 = Z_2 v^{r'}$, $C_3 = h^{r'}$. Note that as long as $k'k(r_1 - r_2) \neq 0$,

$$\begin{cases} \log_g u_1 = x + ky \\ \log_g v = k'\gamma \\ \log_g {Z_1}^x {Z_2}^y = r_1 x + k r_2 y \end{cases}$$

are linearly independent. In the following, we consider two cases.

- $\phi(SK) = SK$ and $(C_1, C_2, C_3, C_4, C_5) = (C_1^*, C_2^*, C_3^*, C_4^*, C_5^*)$. In this case, from the definition of the CC-RKA security game, such queries will be ruled out, therefore the RKA decryption algorithm outputs $\perp$ with noticeable probability.

- $\phi(SK) \neq SK$ and $(C_1, C_2, C_3, C_4) = (C_1^*, C_2^*, C_3^*, C_4^*)$. If the verification of C_5 on (C_1, C_2, C_3, C_4) with $\phi(SK)$ fails, the RKA decryption algorithm outputs $\perp$. Otherwise, the RKA decryption algorithm responds as

$$\begin{aligned} M' &= C_4^* \cdot (C_1^* \cdot C_3^{*-\gamma-\triangle})^{-x-\triangle} \cdot (C_2^* \cdot C_3^{*-\gamma-\triangle})^{-y-\triangle} \\ &= M_d \cdot g^{-r\cdot\triangle} \cdot h^{r'\cdot\triangle\cdot(x+\triangle)} \cdot f^{-r\cdot\triangle} \cdot h^{r'\cdot\triangle\cdot(y+\triangle)} \\ &= M_d \cdot g^{-r\cdot\triangle} \cdot f^{-r\cdot\triangle} \cdot h^{r'\cdot\triangle\cdot(x+y+\triangle+\triangle)}. \end{aligned}$$

We can see that even if all the ciphertexts submitted to the RKA.Decrypt oracle are exactly the same as the challenge ciphertext, algorithm $\mathcal{A}$ learns nothing about (x, y, γ) from the RKA decryption queries under $(x+\triangle, y+\triangle, \gamma+\triangle)$, as long as $(x+\triangle, y+\triangle, \gamma+\triangle) \neq (x, y, \gamma)$. On the one hand, without the values of (x, y, γ) algorithm $\mathcal{A}$ fails to compute $d' = d$ under the modified secret keys $(x+\triangle, y+\triangle, \gamma+\triangle)$. Therefore the probability of algorithm $\mathcal{A}$ in outputting the bit $d' = d$ is $1/2$.

Lemma 5.3 makes sure that as long as the event Failure does not happen, the RKA decryption algorithm rejects all invalid ciphertexts except with negligible probability. Lemma 5.4 proves that as long as the RKA decryption algorithm rejects all the invalid ciphertexts, algorithm $\mathcal{A}$ has a negligible advantage in outputting the bit $d' = d$. Therefore, we can say that the probability of algorithm $\mathcal{A}$ in outputting the bit $d' = d$ is $1/2$.

To sum up, we can see that if (g, f, Z_1, Z_2) is a DH tuple, algorithm $\mathcal{A}$ wins the CC-RKA game with the probability $1/2 + \epsilon$, such that the probability of algorithm $\mathcal{B}$ in solving the decisional DH problem is $1/2 + \epsilon$; if (g, f, Z_1, Z_2) is a non-DH tuple, algorithm $\mathcal{A}$ wins the CC-RKA game with the probability $1/2$, such that the probability of algorithm $\mathcal{B}$ in solving the decisional DH problem is $1/2$. Denote by $\mathcal{B}(g, f, Z_1, Z_2) = 1$ the event that algorithm $\mathcal{B}$ solves the decisional DH problem. Hence, the probability of algorithm $\mathcal{B}$ in solving the decisional DH problem is

$$\begin{aligned}\Pr[\mathcal{B}(g, f, Z_1, Z_2) = 1] &= 1/2 \cdot (1/2 + \epsilon) + 1/2 \cdot 1/2 \\ &= 1/2 + \epsilon/2.\end{aligned}$$

This concludes the proof of Theorem 5.1.

5.3.4 Efficiency

We compare Wee's CC-RKA secure public-key encryption scheme from factoring, from BDH, from DDH with weaker security and ours from DDH in Table 1.

In this table, "Pairing-E" means the sum of pairing computation executed during the encryption phase, and "Pairing-D" means the sum of pairing computation executed during the decryption phase. "Ex-E" means the the sum of exponentiation computation executed during the encryption phase, and "Ex-D" means the the sum of exponentiation computation executed during the decryption phase.

Table 5.1: Comparison between some existing CC-RKA secure public-key encryption schemes and ours

Scheme	Ciphertext Size	Pairing-E	Pairing-D	Ex-E	Ex-D
Factoring[Wee12]	6	0	0	9	7
BDH[Wee12]	6	1	3	7	5
DDH[Wee12]	7	0	0	9	9
Ours	5	0	0	7	5

5.4 An RKA Secure Public-Key Encryption Scheme from Bilinear Pairings

In this section, we show a related-key attack on an existing public-key encryption scheme where each element of the private key has its own linear related-key deriving function, and propose a public-key encryption scheme secure in this setting of related-key attacks from bilinear pairings, which is CC-RKA secure based on the decisional BDH assumption.

5.4.1 A Related-Key Attack on An Existing Public-Key Encryption Scheme

The family Φ^+. Any function $\phi : Z_p^* \to Z_p^*$ in this class is indexed by $\triangle \in Z_p^*$, where $\phi_\triangle(sk) = sk + \triangle$. Note that if sk is composed of several elements as $(sk_1, \ldots, sk_n)$ with $n \in \mathbb{N}$, for any sk_i where $i \in \{1, \ldots, n\}$, $\triangle_i \in Z_p^*$.

The algorithms of the CCA secure public-key encryption scheme in [BCHK07] are given as follows, which is claimed to be CCA secure under the related-key attacks where each component of the private key shares the same linear related-key deriving function in [BCM11]. Let (Mac, Vrfy) denote the message authentication code [BCHK07]. Let $H : \{0,1\}^* \to \{0,1\}^*$ represent a hash function to be second-preimage resistant [BCHK07]. Let G be a group of order p, and let $\hat{e} : G \times G \to G_T$ be a bilinear group.

- Key generation. To generate the public key and secret key of the system,

  1. choose random $g \in G$, x_1, x_2, $y \in Z_p^*$, and set

$$g_1 = g^{x_1}, \quad g_2 = g^{x_2}, \quad g_3 = g^y, \quad \Omega = \hat{e}(g_1, g_3).$$

2. choose a hash function h from a family of pairwise-independent hash functions [BCHK07].

The public key is $PK = (g, h, g_1, g_2, g_3, \Omega)$, and the secret key is $SK = (x_1, x_2, y)$.

- Encryption. To encrypt a message $M \in G_T$,
  1. choose a random $r \in \{0,1\}^*$, and set $k_1 = h(r)$, $ID = H(r)$.
  2. choose a random $s \in Z_p^*$, and set $C = (C_1, C_2, C_3)$ for
  $$C_1 = g^s, \quad C_2 = {g_2}^s {g_3}^{s \cdot ID}, \quad C_3 = Z^s \oplus (M \circ r)).$$
  3. output the ciphertext $(ID, C, \mathrm{Mac}_{k_1}(C))$.
- Decryption. To decrypt a ciphertext $(ID, C, \mathrm{Mac}_{k_1}(C))$,
  1. parse C as (C_1, C_2, C_3), choose a random $t \in Z_p^*$, and compute
  $$(M \circ r) = C_3 \oplus \hat{e}({C_1}^{x_1 \cdot y + t \cdot (x_2 + y \cdot ID)} {C_2}^{-t}, g).$$
  2. set $k_1 = h(r)$. If $\mathrm{Vrfy}_{k_1}(C, tag) = 1$ and $H(r) = ID$, output M; otherwise, output $\perp$.

The attack. We point out a related-key attack on the above scheme where each element of the private key has its own linear related-key deriving function. Note that this does not contradict the result in [BCHK07], since we consider a different related-key deriving function from theirs. Suppose we are given a valid ciphertext $(ID, C, \mathrm{Mac}_{k_1}(C))$ of some message M. We can recover M by making decryption queries to RKA.Decrypt oracle on a related secret key via the following attack. For any $\triangle_i \in Z_p^*$, $i \in \{1,2,3\}$, we change the secret key (x_1, x_2, y) to $(x_1, x_2 + \triangle_2, y + \triangle_3)$ where $\triangle_2 = -ID \cdot \triangle_3$, then $(ID, C, \mathrm{Mac}_{k_1}(C))$ can be decrypted to M directly.

5.4.2 Construction

Let G be a group of order p, and let $\hat{e} : G \times G \to G_T$ be a bilinear group. Based on the selective-identity secure identity-based encryption scheme in [BB04], we construct a CC-RKA secure public-key encrypiton scheme as follows.

- Key generation. To generate the public key and private key of the system,

  1. choose random g, $h \in G$, x_1, x_2, $y \in Z_p^*$, and set

  $$g_1 = g^{x_1}, \quad g_2 = g^{x_2}, \quad h_1 = h^{y_1}, \quad h_2 = h^{y_2}, \quad \Omega = \hat{e}(g,g)^{x_1 \cdot x_2}.$$

  2. choose two collision resistant hash functions $H_1 : G^3 \to Z_p^*$, $H_2 : G^3 \times G_T \to Z_p^*$.

  The public key is $PK = (g, h, g_1, g_2, h_1, h_2, \Omega, H_1, H_2)$, and the private key is $SK = (x_1, x_2, y_1, y_2)$.

- Encryption. To encrypt a message $M \in G_T$,

  1. choose random a, b, $c \in Z_p^*$ independently, and set $u_1 = g^a$, $u_2 = g^b$, $u_3 = g^c$.
  2. choose a random $r \in Z_p^*$, compute $t = H_1(u_1, u_2, u_3)$, and set $C = (C_1, C_2, C_3, C_4, C_5)$, where

  $$C_1 = g^r, \quad C_2 = h^r, \quad C_3 = {h_1}^r {g_2}^{r \cdot t},$$
  $$C_4 = {h_2}^r {g_1}^{r \cdot t}, \quad C_5 = \Omega^r \cdot M.$$

  3. choose a random $e \in Z_p^*$, and compute

  $$\sigma = c + e \cdot a + (H_2(C) + e) \cdot b.$$

  4. output the ciphertext (u, e, σ, C), where $u = (u_1, u_2, u_3)$.

- Decryption. To decrypt a ciphertext (u, e, σ, C),

  1. parse u as (u_1, u_2, u_3), and compute $t = H_1(u_1, u_2, u_3)$.
  2. parse C as $(C_1, C_2, C_3, C_4, C_5)$, and output $\perp$ if $g^\sigma \neq u_3 {u_1}^e {u_2}^{H_2(C)+e}$.
  3. if $\hat{e}(C_3, g) = \hat{e}(C_1, h_1) \cdot \hat{e}(C_1, g_2)^t$, $\hat{e}(C_3, h) = \hat{e}(C_2, h_1) \cdot \hat{e}(C_2, g_2)^t$, and $\hat{e}(C_4, g) = \hat{e}(C_1, h_2) \cdot \hat{e}(C_1, g_1)^t$, $\hat{e}(C_4, h) = \hat{e}(C_2, h_2) \cdot \hat{e}(C_2, g_1)^t$, choose random s_1, $s_2 \in Z_p^*$, and compute

  $$M = \frac{C_5}{\hat{e}({C_1}^{x_1 \cdot x_2 + s_1 \cdot x_2 \cdot t + s_2 \cdot x_1 \cdot t} {C_2}^{s_1 \cdot y_1 + s_2 \cdot y_2} {C_3}^{-s_1} {C_4}^{-s_2}, g)}.$$

Correctness. For any sequence of the key generation and encryption algorithms, it holds that

$$\begin{aligned} M &= \frac{C_5}{\hat{e}({C_1}^{x_1 \cdot x_2 + s_1 \cdot x_2 \cdot t + s_2 \cdot x_1 \cdot t} {C_2}^{s_1 \cdot y_1 + s_2 \cdot y_2} {C_3}^{-s_1} {C_4}^{-s_2}, g)} \\ &= \frac{\Omega^r \cdot M}{\hat{e}((g^r)^{x_1 \cdot x_2 + s_1 \cdot x_2 \cdot t + s_2 \cdot x_1 \cdot t} (h^r)^{s_1 \cdot y_1 + s_2 \cdot y_2} ({h_1}^r {g_2}^{r \cdot t})^{-s_1} ({h_2}^r {g_1}^{r \cdot t})^{-s_2}, g)} \\ &= \frac{\Omega^r \cdot M}{\hat{e}(g^{x_1 \cdot x_2 \cdot r}, g)}, \end{aligned}$$

and therefore the decryption algorithm is always correct.

5.4.3 Security Proof

Theorem 5.5 *The above public-key encryption scheme is secure in the CC-RKA security game regarding linear related-key deriving function ϕ^+ under the decisional BDH assumption.*

Proof. We show that given an adversary algorithm $\mathcal{A}$ that breaks the security of the CC-RKA secure public-key encryption scheme, we can build a challenger algorithm $\mathcal{B}$ that solves the decisional BDH problem, which is given a random tuple (g, g^{x_1}, g^{x_2}, g^{x_3}, Z) as input. Algorithm $\mathcal{B}$'s goal is to determine whether $Z = \hat{e}(g^{x_1}, g^{x_2})^{x_3}$ or Z is a random element in G_T.

- **Setup.** Algorithm $\mathcal{B}$ chooses a random $\gamma \in Z_p^*$ such that $h = g^\gamma$, two hash functions $H_1 : G^3 \to Z_p^*$, $H_2 : G^4 \to Z_p^*$, and sets $g_1 = g^{x_1}$, $g_2 = g^{x_2}$, and $\Omega = \hat{e}(g_1, g_2)$. Also, algorithm $\mathcal{B}$ chooses random a, b, $c \in Z_p^*$, and sets $t^* = H_1(u_1^*, u_2^*, u_3^*)$, where $u_1^* = g^a$, $u_2^* = g^b$, $u_3^* = g^c$. Then algorithm $\mathcal{B}$ chooses random y_1, $y_2 \in Z_p^*$, and sets $h_1 = {g_2}^{-t^*} h^{y_1}$, $h_2 = {g_1}^{-t^*} h^{y_2}$. Algorithm $\mathcal{B}$ sends the public key $PK = (g, h, g_1, g_2, h_1, h_2, \Omega, H_1, H_2)$ to algorithm $\mathcal{A}$.

- **Phase 1.** Algorithm $\mathcal{A}$ queries $(\phi, (u, e, \sigma, C))$ to the RKA.Decrypt oracle. Algorithm $\mathcal{B}$

  - parses C as $(C_1, C_2, C_3, C_4, C_5)$. If $\hat{e}(C_3, g) \neq \hat{e}(C_2, h_1) \cdot \hat{e}(C_1, g_1)^t$, or $\hat{e}(C_4, g) \neq \hat{e}(C_2, h_2) \cdot \hat{e}(C_1, g_2)^t$, algorithm $\mathcal{B}$ outputs $\perp$.
  - parses u as (u_1, u_2, u_3), and checks the signature (e, σ) on C. If $g^\sigma \neq u_3 {u_1}^e {u_2}^{H_2(C)+e}$, algorithm $\mathcal{B}$ outputs $\perp$.
  - computes $t = H_1(u_1, u_2, u_3)$. If $t = t^*$, algorithm $\mathcal{B}$ aborts the simulation. Otherwise, algorithm $\mathcal{B}$ runs the following steps.

1. Chooses random s_1, $s_2 \in Z_p^*$, and sets

$$d_1 = g_1^{\frac{-\gamma \cdot y_1}{t-t^*}}(g_2{}^t h_1)^{s_1}, \quad d_2 = g_2^{\frac{-\gamma \cdot y_2}{t-t^*}}(g_1{}^t h_2)^{s_2}.$$

Let $s_1' = s_1 - x_1/(t-t^*)$, $s_2' = s_2 - x_2/(t-t^*)$, so we have

$$\begin{aligned} d_1 &= g_1^{\frac{-\gamma \cdot y_1}{t-t^*}}(g_2{}^t h_1)^{s_1} = g_1^{\frac{-\gamma \cdot y_1}{t-t^*}}(g_2{}^t h_1)^{s_1' + \frac{x_1}{t-t^*}} \\ &= g^{x_1 \cdot x_2}(g_2{}^t h_1)^{s_1'}, \\ d_2 &= g_2^{\frac{-\gamma \cdot y_2}{t-t^*}}(g_1{}^t h_2)^{s_2} = g_2^{\frac{-\gamma \cdot y_2}{t-t^*}}(g_1{}^t h_2)^{s_2' + \frac{x_2}{t-t^*}} \\ &= g^{x_1 \cdot x_2}(g_1{}^t h_2)^{s_2'}. \end{aligned}$$

2. Computes M as

$$M = \frac{C_5}{[\hat{e}(d_1, C_1) \cdot \hat{e}(C_3^{-s_1'}, g) \cdot \hat{e}(d_2, C_1) \cdot \hat{e}(C_4^{-s_2'}, g)]^{\frac{1}{2}}}.$$

To see this, we rewrite

$$\begin{aligned} M &= \frac{C_5}{[\hat{e}(d_1, C_1) \cdot \hat{e}(C_3^{-s_1'}, g) \cdot \hat{e}(d_2, C_1) \cdot \hat{e}(C_4^{-s_2'}, g)]^{1/2}} = \frac{C_5}{\hat{e}(g^{x_1}, g^{x_2})^r} \\ &= \frac{C_5}{[\hat{e}(d_1, C_1) \cdot \hat{e}(C_3, g^{-s_1'}) \cdot \hat{e}(d_2, C_1) \cdot \hat{e}(C_4, g^{-s_2'})]^{1/2}} \\ &= \frac{C_5}{[\hat{e}(d_1, C_1) \cdot \hat{e}(C_3, g_1^{\frac{1}{t-t^*}} \cdot g^{-s_1}) \cdot \hat{e}(d_2, C_1) \cdot \hat{e}(C_4, g_2^{\frac{1}{t-t^*}} \cdot g^{-s_2}]^{\frac{1}{2}}}. \end{aligned}$$

3. Outputs M' as

$$M' = \frac{M}{\hat{e}(C_1, g_1^{\triangle_{x_2}} g_2^{\triangle_{x_1}} g^{\triangle_{x_1} \cdot \triangle_{x_2}} g^{(s_1 \cdot t \cdot \triangle_{x_2} + s_2 \cdot t \cdot \triangle_{x_1}) + \gamma \cdot (s_1 \cdot \triangle_{y_1} + s_2 \cdot \triangle_{y_2})})}.$$

From the RKA decryption algorithm, we have

$$\begin{aligned} M' &= \frac{C_5}{\hat{e}(C_1{}^{\phi(x_1) \cdot \phi(x_2) + s_1 \cdot \phi(x_2) \cdot t + s_2 \cdot \phi(x_1) \cdot t} C_2{}^{s_1 \cdot \phi(y_1) + s_2 \cdot \phi(y_2)} C_3{}^{-s_1} C_4{}^{-s_2}, g)} \\ &= \frac{M}{\hat{e}(C_1{}^{x_1 \cdot \triangle_{x_2} + x_2 \cdot \triangle_{x_1} + \triangle_{x_1} \cdot \triangle_{x_2} + (s_1 \cdot t \cdot \triangle_{x_2} + s_2 \cdot t \cdot \triangle_{x_1})} C_2{}^{s_1 \cdot \triangle_{y_1} + s_2 \cdot \triangle_{y_2}}, g)} \\ &= \frac{M}{\hat{e}(C_1, g_1^{\triangle_{x_2}} g_2^{\triangle_{x_1}} g^{\triangle_{x_1} \cdot \triangle_{x_2}} g^{(s_1 \cdot t \cdot \triangle_{x_2} + s_2 \cdot t \cdot \triangle_{x_1}) + \gamma \cdot (s_1 \cdot \triangle_{y_1} + s_2 \cdot \triangle_{y_2})})}, \end{aligned}$$

where $\phi(x_1) = x_1 + \triangle_{x_1}$, $\phi(x_2) = x_2 + \triangle_{x_2}$, $\phi(y_1) = y_1 + \triangle_{y_1}$, and $\phi(y_2) = y_2 + \triangle_{y_2}$.

Thus, as long as the ciphertext from algorithm $\mathcal{A}$ is correctly formulated, the response of algorithm $\mathcal{B}$ is always identical to the RKA decryption algorithm as required.

Note that algorithm $\mathcal{B}$ responds the RKA decryption queries without using the private key SK.

- **Challenge.** Algorithm $\mathcal{A}$ outputs two messages M_0, $M_1 \in G_T$ on which it wishes to be challenged. Algorithm $\mathcal{B}$ chooses a random $d \in \{0,1\}$, and responds with the ciphertext $C^* = (C_1^*, C_2^*, C_3^*, C_4^*, C_5^*)$, where

$$C_1^* = g^{x_3}, \quad C_2^* = (g^{x_3})^{\gamma}, \quad C_3^* = (g^{x_3})^{\gamma \cdot y_1},$$
$$C_4^* = (g^{x_3})^{\gamma \cdot y_2}, \quad C_5^* = Z \cdot M_d.$$

  Hence, if $Z = \hat{e}(g,g)^{x_1 x_2 x_3} = \hat{e}(g_1, g_2)^{x_3}$. then C^* is a valid encryption of M_d with respect to t^*. On the other hand, when Z is a random element in G_T, then C^* is independent of d in the view of algorithm $\mathcal{A}$.

- **Phase 2.** Algorithm $\mathcal{A}$ continues to adaptively issue queries $(\phi, (u, e, \sigma, C))$ to the RKA.Decrypt oracle.
  - If $\phi(SK) = SK$ and $C = C^*$, algorithm $\mathcal{B}$ responds with $\perp$.
  - Otherwise, algorithm $\mathcal{B}$ responds as in Phase 1 except that when $H_1(u_1, u_2, u_3) = t^*$, algorithm $\mathcal{B}$ outputs

$$M' = \frac{M_d}{\hat{e}\big(C_1, {g_1}^{\triangle_{x_2}} {g_2}^{\triangle_{x_1}} g^{\triangle_{x_1} \cdot \triangle_{x_2}} g^{(s_1 \cdot t^* \cdot \triangle_{x_2} + s_2 \cdot t^* \cdot \triangle_{x_1}) + \gamma \cdot (s_1 \cdot \triangle_{y_1} + s_2 \cdot \triangle_{y_2})}\big)}.$$

  Note that without the randomness chosen by algorithm $\mathcal{B}$, algorithm $\mathcal{A}$ has negligible probability in outputting M_d. In other words, M' is independent of d from the view of algorithm $\mathcal{A}$.

- **Output.** Algorithm $\mathcal{A}$ outputs a guess $d' \in \{0,1\}$. If $d' = d$, algorithm $\mathcal{B}$ outputs 1; otherwise, algorithm $\mathcal{B}$ outputs 0.

Let abort be the event that algorithm $\mathcal{B}$ aborts during the simulation. To conclude the proof of Theorem 5.5, it remains to be analyzed the probability that algorithm $\mathcal{B}$ aborts the simulation for one of the RKA decryption queries from algorithm $\mathcal{A}$. We claim that Pr[abort] is negligible, or one can use algorithm $\mathcal{A}$ to forge signatures with at least the same probability. Briefly, we can construct another algorithm $\mathcal{B}'$ to simulate the above security game that is given the private key, but is given the signing key as a challenge in an existential forgery game. Algorithm $\mathcal{A}$ causes an abort by querying a ciphertext including an existential forgery under

the given signing key. Algorithm $\mathcal{B}'$ is able to use this forgery to win the existential forgery game. Note that during the game algorithm $\mathcal{A}$ makes only one chosen message query to generate the signature needed for the challenge ciphertext. Thus, Pr[abort] is negligible.

From the analysis of Phase 2, algorithm $\mathcal{A}$ definitely has negligible probability in outputting $d' = d$ via the RKA decryption queries, as M' is always consistent with two random s_1, $s_2 \in Z_p^*$ chosen by algorithm $\mathcal{B}$ such that M' is independent of d from the view of algorithm $\mathcal{A}$. Let ϵ be the advantage that algorithm $\mathcal{A}$ breaks the CC-RKA security of the above game. Therefore, we can see that if the input tuple of algorithm $\mathcal{B}$ is (g, g^a, g^b, g^c, Z) where $Z = \hat{e}(g,g)^{abc}$, then algorithm $\mathcal{A}$'s view of this simulation is identical to the real attack, so the probability of algorithm $\mathcal{A}$ in outputting $d' = d$ must satisfy $\Pr[d = d'] = 1/2 + \epsilon$. On the other hand, if the input tuple of algorithm $\mathcal{B}$ is (g, g^a, g^b, g^c, Z) where $Z \in G_T$, then the advantage of algorithm $\mathcal{A}$ is nil, and algorithm $\mathcal{A}$'s view of the challenge ciphertext is independent of d, so the probability of algorithm $\mathcal{A}$ in outputting $d' = d$ is $\Pr[d' = d] = 1/2$. In sum, the probability of algorithm $\mathcal{B}$ in solving the decisional BDH problem is

$$\begin{aligned}\Pr[\mathcal{B}(g, g^a, g^b, g^c, Z)] &= 1/2 \cdot (1/2 + \epsilon) + 1/2 \cdot 1/2 \\ &= 1/2 + \epsilon/2.\end{aligned}$$

This completes the proof of Theorem 5.5.

5.5 A Public-Key Encryption Scheme with RKA Security without Pairings

In this section, we put forward a public-key encryption scheme against related-key attacks without pairings, and reduce its CC-RKA security under the decisional Linear assumption.

5.5.1 Construction

Assume that the scheme is parameterized by a gap parameter generator $\mathcal{G}$ [OP01, Kil06], which on input 1^k, returns the description of a multiplicative cyclic group G of prime order p, and the description of a Diffie-Hellman oracle that outputs 1 if an input $(g, g^x, g^y, g^z) \in G^4$ is a Diffie-Hellman tuple; otherwise, it outputs 0.

We present our construction based on the public-key encryption scheme in [Kil06] as follows.

- Key generation. To generate the public key and the private key of the system,

  1. choose random $h \in G$, y_1, $y_2 \in Z_p^*$, and set $h_1 = h^{y_1}$, $h_2 = h^{y_2}$.
  2. choose two collision resistant hash functions $H_1 : G^3 \to Z_p^*$, $H_2 : G^7 \to Z_p^*$.
  3. choose random g, f, $z \in G$, x_1, $x_2 \in Z_p^*$ such that $z = g^{x_1} = f^{x_2}$.

  The public key is $PK = (g, f, h, z, h_1, h_2, H_1, H_2)$, and the private key is $SK = (x_1, x_2, y_1, y_2)$.

- Encryption. To encrypt a message $M \in G$,

  1. choose random a, b, $c \in Z_p^*$ independently, and set $u_1 = g^a$, $u_2 = g^b$, $u_3 = g^c$.
  2. choose random r_1, $r_2 \in Z_p^*$, compute $t = H_1(u_1, u_2, u_3)$, and set $C = (C_1, C_2, C_3, C_4, C_5, C_6, C_7)$ where

  $$C_1 = g^{r_1}, \quad C_2 = h^{r_1}, \quad C_3 = f^{r_2}, \quad C_4 = h^{r_2},$$
  $$C_5 = z^{t \cdot r_1} {h_1}^{r_1}, \quad C_6 = z^{t \cdot r_2} {h_2}^{r_2}, \quad C_7 = z^{r_1 + r_2} \cdot M.$$

  3. choose a random $e \in Z_p^*$, and compute

  $$\sigma = c + e \cdot a + (H_2(C) + e) \cdot b.$$

  4. output the ciphertext (u, e, σ, C), where $u = (u_1, u_2, u_3)$.

- Decryption. To decrypt a ciphertext (u, e, σ, C),

  1. output $\perp$ if $g^\sigma \neq u_3 {u_1}^e {u_2}^{H_2(C)+e}$.
  2. parse C as $(C_1, C_2, C_3, C_4, C_5, C_6, C_7)$, and output $\perp$ if either ${C_1}^{x_1 \cdot t} {C_2}^{y_1} \neq C_5$, or ${C_3}^{x_2 \cdot t} {C_4}^{y_2} \neq C_6$.
  3. parse u as (u_1, u_2, u_3), and compute $t = H_1(u_1, u_2, u_3)$.
  4. choose random s_1, $s_2 \in Z_p$, and output

  $$M = C_7 \cdot \left(\frac{{C_1}^{x_1 + s_1 \cdot t \cdot x_1} {C_2}^{s_1 \cdot y_1} {C_3}^{x_2 + s_2 \cdot t \cdot x_2} {C_4}^{s_2 \cdot y_2}}{{C_5}^{s_1} {C_6}^{s_2}}\right)^{-1}.$$

  Otherwise, output $\perp$.

Correctness. For any sequence of the key generation and encryption algorithms, it holds that

$$\begin{aligned} M &= C_7 \cdot (\frac{{C_1}^{x_1+s_1\cdot t\cdot x_1} {C_2}^{s_1\cdot y_1} {C_3}^{x_2+s_2\cdot t\cdot x_2} {C_4}^{s_2\cdot y_2}}{{C_5}^{s_1} {C_6}^{s_2}})^{-1} \\ &= z^{r_1+r_2} \cdot M \cdot (\frac{(g^{r_1})^{x_1} {g_1}^{t\cdot r_1\cdot s_1} {h_1}^{r_1\cdot s_1} (g^{r_2})^{x_2} {g_2}^{t\cdot r_2\cdot s_2} {h_2}^{r_2\cdot s_2}}{{C_5}^{s_1} {C_6}^{s_2}})^{-1} \\ &= z^{r_1+r_2} \cdot M \cdot ((g^{r_1})^{x_1} (f^{r_2})^{x_2})^{-1}, \end{aligned}$$

and therefore the decryption algorithm is always correct.

5.5.2 Security Proof

Theorem 5.6 *The above public-key encryption scheme is secure in the CC-RKA security game regarding linear related-key deriving function ϕ^+ under the decisional LIN assumption relative to a gap parameter generator $\mathcal{G}$.*

Proof. We show that given an adversary algorithm $\mathcal{A}$ that breaks the security of the CC-RKA secure public-key encryption scheme, we can build a challenger algorithm $\mathcal{B}$ that can solves the decisional LIN problem, which is given a random tuple (g, f, z, g^{r_1}, f^{r_2}, w) relative to a gap parameter generator $\mathcal{G}$ as input. Algorithm $\mathcal{B}$'s goal is to determine whether $w = z^{r_1+r_2}$ or w is a random group element.

- **Setup.** Algorithm $\mathcal{B}$ runs as follows to generate the public parameters.

  1. Chooses random γ_1, $\gamma_2 \in Z_p^*$, and computes $h = g^{\gamma_1} = f^{\gamma_2}$.
  2. Chooses two hash functions $H_1 : G^3 \to Z_p^*$, $H_2 : G^7 \to Z_p^*$.
  3. Chooses random a, b, $c \in Z_p^*$, sets $u_1 = g^a$, $u_2 = g^b$, $u_3 = g^c$, and computes $t^* = H_1(u_1, u_2, u_3)$.
  4. Chooses c_1, $c_2 \in Z_p^*$, and computes $h_1 = z^{-t^*} g^{c_1}$, $h_2 = z^{-t^*} f^{c_2}$.

  Algorithm $\mathcal{B}$ publishes the public key $PK = (g, f, h, z, h_1, h_2, H_1, H_2)$.

  Let $x_1 = \log_g z$, $x_2 = \log_f z$, such that y_1, y_2 can be defined as

  $$\begin{aligned} y_1 = \log_h h_1 = \frac{1}{\gamma_1} \cdot (-t^* \cdot x_1 + c_1), \\ y_2 = \log_h h_2 = \frac{1}{\gamma_2} \cdot (-t^* \cdot x_2 + c_2). \end{aligned}$$

  Note that the value of the private key $SK = (x_1, x_2, y_1, y_2)$ is unknown to algorithm $\mathcal{B}$.

- **Phase 1.** Algorithm $\mathcal{A}$ queries $(\phi, (u, e, \sigma, C))$ to the RKA.Decrypt oracle. To make it easier to understand, we ignore the related-key deriving functions for now. Algorithm $\mathcal{B}$ executes as follows to respond.

  1. Checks the signature (e, σ) on C using the verification key u. If the signature is invalid, i.e., $g^{\sigma} \neq u_3 {u_1}^{e} {u_2}^{H_2(C)+e}$, algorithm $\mathcal{B}$ outputs $\perp$.
  2. Parses C as $(C_1, C_2, C_3, C_4, C_5, C_6, C_7)$. If neither $(g, z^t h_1, C_1, C_5)$, $(h, z^t h_1, C_2, C_5)$, $(f, z^t h_2, C_3, C_6)$ nor $(h, z^t h_2, C_4, C_6)$ is a Diffie-Hellman tuple, algorithm $\mathcal{B}$ outputs $\perp$.

     Note that the Diffie-Hellman oracle of gap parameter generator $\mathcal{G}$ can be obtained using bilinear pairings [BF01].
  3. Parses u as (u_1, u_2, u_3). If $H_1(u_1, u_2, u_3) = t^*$, algorithm $\mathcal{B}$ aborts the simulation. Otherwise, algorithm $\mathcal{B}$ computes $t = H_1(u_1, u_2, u_3)$, and outputs

$$M = C_7 \cdot \left(\frac{C_5 C_6}{{C_1}^{c_1} {C_3}^{c_2}}\right)^{\frac{1}{t-t^*}}.$$

  To see that algorithm $\mathcal{B}$'s response of M is the same as in a real attack, we rewrite

$$\begin{aligned} C_5 &= {C_1}^{x_1 \cdot t} {C_1}^{-x_1 \cdot t^* + c_1} = {C_1}^{x_1 \cdot (t-t^*)} {C_1}^{c_1}, \\ C_6 &= {C_3}^{x_2 \cdot t} {C_3}^{-x_2 \cdot t^* + c_2} = {C_3}^{x_2 \cdot (t-t^*)} {C_3}^{c_2}. \end{aligned}$$

  Thus, algorithm $\mathcal{B}$'s response is identical to the decryption algorithm as required.

  Denote by M' the result returned from algorithm $\mathcal{B}$ using $\phi(SK)$ to decrypt (u, e, σ, C). Algorithm $\mathcal{B}$ chooses random $s_1, s_2 \in Z_p^*$, and outputs M' as

$$\begin{aligned} M' &= C_7 \cdot \left(\frac{{C_1}^{X} {C_2}^{s_1 \cdot (y_1 + \triangle_{y_1})} {C_3}^{Y} {C_4}^{s_2 \cdot (y_2 + \triangle_{y_2})}}{{C_5}^{s_1} {C_6}^{s_2}}\right)^{-1} \\ &= \frac{M}{{C_1}^{\triangle_{x_1} + s_1 \cdot t \cdot \triangle_{x_1}} {C_2}^{s_1 \cdot \triangle_{y_1}} {C_3}^{\triangle_{x_2} + s_2 \cdot t \cdot \triangle_{x_2}} {C_4}^{s_2 \cdot \triangle_{x_2}}}. \end{aligned}$$

  where $X = (x_1 + \triangle_{x_1}) + s_1 \cdot t \cdot (x_1 + \triangle_{x_1})$, $Y = (x_2 + \triangle_{x_2}) + s_2 \cdot t \cdot (x_2 + \triangle_{x_2})$.

  Note that algorithm $\mathcal{B}$ responds the RKA decryption queries without using the private key SK.

- **Challenge.** Algorithm $\mathcal{A}$ outputs two messages M_0, $M_1 \in G$ on which it wishes to be challenged. Algorithm $\mathcal{B}$ executes the following procedures to respond.

  1. Chooses a random $d \in \{0,1\}$, and computes $C^* = (C_1^*, C_2^*, C_3^*, C_4^*, C_5^*, C_6^*, C_7^*)$, where

  $$C_1^* = g^{r_1}, \quad C_2^* = C_1^{1/\gamma_1}, \quad C_3^* = f^{r_2}, \quad C_4^* = C_3^{1/\gamma_2},$$
  $$C_5^* = (C_1^*)^{c_1}, \quad C_6^* = (C_2^*)^{c_2}, \quad C_7^* = M_d \cdot w.$$

  2. Chooses a random $e^* \in Z_p^*$, and computes

  $$\sigma^* = c + e^* \cdot a + (H_2(C^*) + e^*) \cdot b.$$

  3. Outputs ciphertext (u, e^*, σ^*, C^*), where $u = (u_1, u_2, u_3)$.

  From the analysis in Phase 1, we can see that (u, e^*, σ^*, C^*) is always consistent with the encryption algorithm as required. We can see that if $w = z^{r_1+r_2}$, then C^* is a valid encryption of M_d with respect to t^*; if w is a random element in G, then C^* is independent of d in the view of algorithm $\mathcal{A}$.

- **Phase 2.** Algorithm $\mathcal{A}$ continues to adaptively issue queries $(\phi, (u, e, \sigma, C))$ to the RKA.Decrypt oracle.

  – If $\phi(SK) = SK$ and $C = C^*$, algorithm $\mathcal{B}$ responds with $\perp$.

  – Otherwise, algorithm $\mathcal{B}$ responds as in Phase 1 except that when $H_1(u_1, u_2, u_3) = t^*$, algorithm $\mathcal{B}$ outputs

  $$M' = \frac{M_d}{C_1^{\triangle_{x_1} + s_1 \cdot t^* \cdot \triangle_{x_1}} C_2^{s_1 \cdot \triangle_{y_1}} C_3^{\triangle_{x_2} + s_2 \cdot t^* \cdot \triangle_{x_2}} C_4^{s_2 \cdot \triangle_{x_2}}}.$$

  Note that without the randomness chosen by algorithm $\mathcal{B}$, algorithm $\mathcal{A}$ has a negligible probability in outputting M_d.

- **Output.** Algorithm $\mathcal{A}$ outputs a guess $d' \in \{0,1\}$. If $d' = d$, algorithm $\mathcal{B}$ outputs 1; otherwise, algorithm $\mathcal{B}$ outputs 0.

Let abort be the event that algorithm $\mathcal{B}$ aborts during the simulation. To conclude the proof of Theorem 5.6, it remains to be analyzed the probability that algorithm $\mathcal{B}$ aborts the simulation for one of the RKA decryption queries from algorithm $\mathcal{A}$. We claim that Pr[abort] is negligible, or one can use algorithm $\mathcal{A}$ to

forge signatures with at least the same probability. Briefly, we can construct another algorithm $\mathcal{B}'$ to simulate the above security game that is given the private key, but is given the signing key as a challenge in an existential forgery game. Algorithm $\mathcal{A}$ causes an abort by querying a ciphertext including an existential forgery under the given signing key. Algorithm $\mathcal{B}'$ is able to use this forgery to win the existential forgery game. Note that during the game algorithm $\mathcal{A}$ makes only one chosen message query to generate the signature needed for the challenge ciphertext. Thus, Pr[abort] is negligible.

From the analysis of Phase 2, algorithm $\mathcal{A}$ has negligible probability in outputting $d' = d$ via the RKA decryption queries, as M' is always consistent with two random numbers s_1, $s_2 \in Z_p^*$ chosen by algorithm $\mathcal{B}$ such that M' is independent of d from the view of algorithm $\mathcal{A}$. Let ϵ be the advantage that algorithm $\mathcal{A}$ breaks the CC-RKA security of the above game. We can see that if the input tuple of algorithm $\mathcal{B}$ is $(g,\ f,\ z,\ g^{r_1},\ f^{r_2},\ w)$ where $w = z^{r_1+r_2}$, then algorithm $\mathcal{A}$'s view of this simulation is identical to the real attack, so the probability of algorithm $\mathcal{A}$ in outputting $d' = d$ must satisfy $\Pr[d = d'] = 1/2 + \epsilon$. On the other hand, if the input tuple of algorithm $\mathcal{B}$ is $(g,\ f,\ z,\ g^{r_1},\ f^{r_2},\ w)$ where $w \in G$, then algorithm the advantage of $\mathcal{A}$ is nil and algorithm $\mathcal{A}$'s view of the challenge ciphertext is independent of d, so the probability of algorithm $\mathcal{A}$ in outputting $d' = d$ is $\Pr[d' = d] = 1/2$. In summary, the probability of algorithm $\mathcal{B}$ in solving the decisional LIN problem is

$$\begin{aligned}\Pr[\mathcal{B}(g, f, z, g^{r_1}, f^{r_2}, w)] &= 1/2 \cdot (1/2 + \epsilon) + 1/2 \cdot 1/2 \\ &= 1/2 + \epsilon/2.\end{aligned}$$

This completes the proof of Theorem 5.6.

5.6 Summary

Following the work in [BCM11], Wee [Wee12] proposed the first public-key encryption schemes against related-key attacks using adaptive trapdoor relations [Wee10] while paying a small overhead in efficiency, of which the existing public-key set-ups can be maintained without changing. Later, based on a framework to enable the construction of identity-based encryption schemes that are secure under related-key attacks, Bellare, Paterson and Thomson [BPT12] provided a framework to enable the construction of public-key encryption schemes that are secure under related-key attacks. Public-key encryption schemes in [BPT12] are achieved in the standard

model, and they hold the CC-RKA security under reasonable hardness assumptions in the standard model.

In this chapter, we consider whether there are other methods to achieve the CC-RKA security for an underlying public-key encryption scheme, and we mainly concentrate on the setting of the multi-element private key, meaning that the private key is made up of more than one component. Our technique is very straightforward in that we use randomness to disguise the information obtained by the adversary from the related-key decryption oracle.

After pointing out a simple linear related-key attack on the Cramer-Shoup basic CCA secure public-key encryption scheme [CS01], we firstly put forward an efficient public-key encryption scheme with a slight modification to the Cramer-Shoup scheme, and demonstrate its CC-RKA security based on the difficulty of solving the DDH problem under a weaker RKA security model where all the parts of the private key use the same linear shift.

Next, we show that the assumption of weaker RKA security does not make much sense in practice, and the system's security could be easily broken if different linear shifts are allowed to the private key, which means a stronger notion of RKA security is needed to get around this problem. Based on an efficient selective-identity secure identity-based encryption scheme without random oracles in [BB04], we obtain an RKA secure public-key encryption scheme from bilinear pairings where different parts of the private key can be modified under different linear shifts. In addition, we present another RKA secure public-key encryption scheme without pairings from a public-key encryption scheme in [Kil06], which is secure against this kind of related-key attacks as well.

Some results of this chapter have been published in "The 9th International Conference on Security and Privacy in Communication Networks (SecureComm 2014)" and "The 13h IEEE International Conference on Trust, Security and Privacy in Computing and Communications (IEEE TrustCom-14)".

Chapter 6

New Attacks to Publicly Verifiable Proof of Retrievability

The notion of proof of retrievability is introduced to allow users to verify whether their data is still available and can be retrieved when they are stored by third parties. The aim of this chapter is to provide stronger security for publicly verifiable proof of retrievability schemes.

6.1 Introduction

In order to improve network bandwidth and reliability and reduce user reliance on local resources, computing systems are resorting to delegating computing services to outside entities as forms of outsourcing. Meanwhile, users are increasingly using software and data that reside thousands of miles away in machines that they do not own. Cloud computing, grid computing and software as a service calling for data, both personal and business, to be stored by third parties are now important parts in the internet technology. In these scenarios, classical cryptographic security primitives for the purpose of data privacy cannot be applied, and it is more important to achieve retrievability so that the integrity of the stored data can be checked rather than achieve confidentiality. However, because the deployment of outsourced storage has fallen behind, users of outsourced storage are in great danger of the continued availability from their storage providers. As a result, various proof of retrievability systems are introduced in order to allow users to verify whether their data is still available and can be retrieved [FB06, SM06, SDFMB$^+$08, ZX11]. From the role of the verifier in the model, these schemes can be divided into two categories: public verifiability and private verifiability. In these systems, the user and the server engage in a protocol where the user attempts to be convinced by the protocol interaction with the server that its file is being stored. This capability is very important to

storage providers, since users may be reluctant to store their data in an unknown startup, and they need an auditing mechanism to reassure them that their data are indeed still available. Regarding the formal security model, the proof of storage systems all attempt to achieve the following criteria [SW08]:

1. The system should be as efficient as possible in computational complexity and communication complexity of the proof-of-storage protocol, and the storage overhead on the server should be as small as possible.

2. The system should allow unbounded use rather than imposing a priori bound on the number of audit protocol interactions.

3. Verifiers should be stateless and not have to maintain and update state between audits, since such a state is difficult to maintain if the verifier's machine crashes or if the verifier's role is delegated to third parties or distributed among multiple machines.

4. The system establishes that any server that passes a verification check for a file is actually storing the file, even for a dishonest server that exhibits possibly arbitrary and malicious behavior.

Statelessness and unbounded use are required for proof of storage systems with public verifiability, in which anyone can undertake the role of verifier in the proof of storage protocol, not just the user who originally stored the file. In fact, it is more rational to equip the verification protocol with public verifiability, which is expected to play a more important role in achieving economies of scale for outsourcing, as the clients themselves might not be able to afford the overhead of performing frequent integrity checks. Unfortunately, none of these existing works has taken physical attacks like side-channel [MR04] attacks into consideration, where given physical access to a hardware device, an adversary would observe some "physical output" of a computation (such as radiation, power, temperature, and running time) and the "logical output" of the computation with methods like fault injection techniques.

6.1.1 Our Contributions

Inspired by the idea of "Cryptography Secure Against Related-Key Attacks and Tampering" [BCM11], where an adversary tampers with the private key stored in a

cryptographic hardware device and subsequently observes the outcome of the cryptographic primitive under this modified key, we take into account a special attack in proof of retrievability systems with public verification, where the adversary captures the relations among the public keys of the users in an outsourcing infrastructure. We still call this attack as a related-key attack, and we use the related-key deriving functions to describe the relations obtained by the adversary among different public keys.

It is an important ability for the publicly verifiable proof of retrievability protocols to be resistant to this kind of attacks, as related-key attacks could happen in practice. However, most existing proof of retrievability protocols with public verifiability [WWRL10, WWR+11, WCW+13], following the structure of Shacham and Waters [SW08], do not take related-key attacks into consideration during the initial design phase. In other words, if related-key attacks happen in these systems, they cannot guarantee the availability of user's data. For example, the server stores the data of user A and user B, and it knows the related-key deriving function between the public keys of user A and user B. Now under related-key attacks, the server could assure a verifier that user A's (or user B's) file is kept intact while the fact maybe that user A's (or user B's) file has been erased. Another instance is in a situation where a user, who modifies its private key at regular intervals such that these modified keys may exhibit some known relations if they are derived from some related-key deriving function; however, the server, under related-key attacks, can convince a verifier that it updates its storage of this user's information regularly and correspondingly while the truth is that it does not. From this point of view, it becomes of interest, accordingly, to achieve RKA security in the proof of retrievability systems. In this chapter, we propose a primitive called public verification resistant to related-key attacks, in particular, in the proof of retrievability systems. We first point out a simple linear related-key attack on an existing proof of retrievability scheme with public verifiability, then we describe our concrete construction of a public verification scheme in the setting of related-key attacks, and show its security in the random oracle model.

6.1.2 Related Work

Blum et al. [BEG+94] first addressed the task of efficiently checking the correctness of a memory management program in broad generality, and the following work

has been done on dynamic memory checking in a range of settings. Clarke et al. [CSG+05] considered the case of a trusted entity with a small amount of state and applied a Merkle hash-tree over the contents of this memory. Noar and Rothblum [NR05] first considered the formal model of proof of storage for "authenticators". Juels and Kaliski [JJ07] focused on memory integrity checking in the static files, and they presented the first security model for "proof of retrievability". Shacham and Waters [SW08] gave the first proof-of-retrievability schemes with full proofs of security against arbitrary adversaries in the strongest model (compared to that of Juels and Kaliski), relying on homomorphic properties to aggregate a proof into one small authenticator value. Filho and Barreto [FB06] described a cryptographic protocol based on similar principles, through which a prover can demonstrate the possession of an arbitrary set of data known to the verifier without the requirement of having to have this data at hand during the protocol execution. Schwarz and Miller [SM06] used m/n erasure-correcting coding to safeguard the stored data and used algebraic signatures with algebraic properties for verification in which large amounts of storage can be verified with minimal network bandwidth. Sebé et al. [SDFMB+08] presented a remote data possession checking protocol which allows unlimited amount of file integrity verification. Zheng and Xu [ZX11] introduced the concept of fair and dynamic proof of retrievability, a useful extension of static proof of retrievability in practice.

6.1.3 Organization

The remainder of this chapter is organized as follows. In Section 6.2, we briefly present the concepts associated with this work. In Section 6.3, we define the security model of the RKA secure publicly verifiable proof of retrievability systems. In Section 6.4, we describe a publicly verifiable system and related-key attacks on it. In Section 6.5, we propose a public verification scheme secure against related-key attacks. In Section 6.6, we prove its security in the random oracle model. Finally, we conclude this chapter in Section 6.7.

6.2 Preliminaries

In this section, we briefly describe the framework of public verification systems, and then define its security model under the setting of related-key attacks.

6.2.1 Public Verification Scheme

A public verification scheme Pub is composed of the following four algorithms: Pub.Kg, Pub.St, Pub.P and Pub.V [JJ07, SW08].

- Pub.Kg(1^λ). This algorithm takes a security parameter λ as input, and outputs a public and private key pair (pk, sk).
- Pub.St(sk, M). This file-storing algorithm takes a private key sk and a file $M \in \{0,1\}^*$ as input, it processes M to output M^*, which will be stored on the server side, as well as a tag t, which contains information that names the file being stored (it could also contain additional secret information encrypted under the private key).
- Pub.P, Pub.V. The proving and verifying algorithms define a protocol for proving file retrievability called the proof of retrievability protocol, which can be denoted as

  $$\{0,1\} \leftarrow (\text{Pub.V}(pk,t) \rightleftharpoons \text{Pub.P}(pk,t,M^*)).$$

  Both algorithms take a public key pk, a tag t as input during the protocol run. The proving algorithm also takes a processed file description M^* as input. After the protocol execution, Pub.V outputs 1 meaning that the file is being stored on the server side, or 0 meaning that the file is not being stored on the server side.

We require a public verification scheme to be correct if for all key pairs (pk, sk) output by Pub.Kg, for all files $M \in \{0,1\}^*$, and for all (M^*, t) output by Pub.St(sk, M), the verifying algorithm outputs 1 when communicating with the proving algorithm

$$(\text{Pub.V}(pk,t) \rightleftharpoons \text{Pub.P}(pk,t,M^*)) = 1.$$

6.2.2 RKA Secure Signature

Assume that a signature scheme is composed of the following three algorithms: key generation algorithm SKg, signing algorithm SSig and verifying algorithm SVer.

1. SKg(1^λ) $\rightarrow$ (spk, ssk): Taking a security parameter λ as input, this algorithm outputs a verifying (public) key spk and a signing (private) key ssk.

2. SSig(m, ssk) $\rightarrow \sigma$: Taking a message m, and the signing key ssk as input, this algorithm outputs a signature σ.

3. SVer(spk, m, σ) $\rightarrow$ *true*/*false*: Taking the public parameters $params$, the verifying key spk, a pair (m, σ) of message and signature as input, this algorithm outputs *true* for a valid signature or *false* for an invalid signature.

We require that a signature scheme $\mathcal{SIG}$ is correct if for any $\lambda \in \mathbb{N}$, (spk, ssk) $\leftarrow$ SKg(1^λ), and $\sigma \leftarrow$ SSig(m, ssk), we have SVer(spk, m, σ) = *true*.

Related-key deriving functions. Our definition follows the notion of related-key deriving functions given in [BK03]. Briefly speaking, a class Φ of related-key deriving functions $\phi : sk \rightarrow sk$ is a finite set of functions with the same domain and range, mapping a key to a related key. Additionally, Φ should allow an efficient membership test, and ϕ should be efficiently computable. Note that in this chapter we regard the class Φ^+ as linear shifts.

The family Φ^+. Any function $\phi : Z_p^* \rightarrow Z_p^*$ in this class is indexed by $\triangle \in Z_p^*$, where $\phi(sk) = sk + \triangle$.

We revisit the RKA security game to a signature scheme $\mathcal{SIG}$ and the RKA specification Φ in Figure 6.1. We say $\mathcal{SIG}$ is RKA secure if the advantage function

$$\mathbf{Adv}^{\mathrm{RKA}}_{\mathcal{SIG},\Phi}(\mathcal{A}) = \Pr[\mathcal{SIG}^{\mathcal{A}} \Rightarrow \mathit{true}]$$

is negligible in the security parameter λ for any adversary algorithm $\mathcal{A}$.

proc Initialize	proc Sign(ϕ, m)
$M \leftarrow \emptyset$	$ssk' \leftarrow \phi(ssk)$
(spk, ssk) $\leftarrow$ SKg(1^λ)	If $ssk' = \perp$ then return $\perp$
Return spk	If $ssk' = ssk$ then $M \leftarrow M \cup \{m\}$
proc Finalize(m, σ)	$\sigma \leftarrow$ SSig(m, ssk')
Return SVer(spk, m, σ) = *true* $\wedge$ ($m \notin M$)	Return σ

Figure 6.1: Game defining RKA security for $\mathcal{SIG}$ = (SKg, SSig, SVer).

6.3 Security Model for RKA Secure Proof of Retrievability with Public Verification

In this section, we detail the security definition of the RKA security in the proof of retrievability systems with public verifiability.

Following the security definition given in [SW08], we provide a modular proof framework for the security of public verification schemes secure against related-key attacks. This framework enables us to argue about unforgeability, extractability and retrievability with three parts in the setting of related-key attacks on the basis of cryptographic and mathematical techniques.

RKA-Soundness. A public verification scheme is sound if any cheating prover that convinces the verifier that it is storing a file M of a user, called Alice, is actually storing the file (together with the latest information) of Alice (not Bob or other users). To give a precise definition of soundness, we formalize an extraction algorithm Extr(pk, t, Pub.$\mathcal{P}$) which takes as input the public key pk, the file tag t, and the description of a mechanism Pub.$\mathcal{P}$ playing the role of the proving algorithm in the public verification system, and outputs the file $M \in \{0,1\}^*$. We consider the following game between an adversary algorithm $\mathcal{A}$ and a simulator algorithm $\mathcal{B}$.

1. Algorithm $\mathcal{B}$ generates a public and private key pair (pk, sk) by running Pub.Kg, and sends pk to algorithm $\mathcal{A}$.

2. Algorithm $\mathcal{A}$ makes related-key queries to algorithm $\mathcal{B}$. For each query, a file M and a related-key deriving function $\phi \in \Phi$. Algorithm $\mathcal{B}$ runs (M^*, t) $\leftarrow$ Pub.St($\phi(sk)$, M) to obtain a processed version M^* of M and a file tag t, and sends both of them to algorithm $\mathcal{A}$.

3. For any M which has been made a query, algorithm $\mathcal{A}$ can execute the proof of retrievability protocol with algorithm $\mathcal{B}$, with the corresponding tag t, where algorithm $\mathcal{A}$ plays the part of the proving algorithm while algorithm $\mathcal{B}$ plays the part of the verifying algorithm: Pub.V(pk, t) $\rightleftharpoons$ $\mathcal{A}$. When this is completed, algorithm $\mathcal{A}$ is given the output of Pub.V.

4. Finally, algorithm $\mathcal{A}$ outputs a challenge tag t return from some query and the description of a proving algorithm Pub.$\mathcal{P}$.

The cheating proving algorithm Pub.$\mathcal{P}$ is ϵ-admissible if it definitely responds an ϵ fraction of verification challenge:

$$\Pr[(\text{Pub.V}(pk, t) \rightleftharpoons Pub.\mathcal{P}) = 1] \geq \epsilon.$$

Here the probability is over the randomness of the proving and verifying algorithms. Let M be the message input to the query that responds to the challenge tag t, as well as a M's processed version M^*.

Definition 6.1 *We say a publicly verifiable proof of retrievability system is RKA-sound if there is an extraction algorithm* Extr *such that, for any algorithm $\mathcal{A}$ playing the above game, whenever algorithm $\mathcal{A}$ outputs an ϵ admissible cheating prover* Pub.$\mathcal{P}$ *for a file M then the probability that extraction fails is negligible. That is, the equation* Extr(pk, t, Pub.$\mathcal{P}$) $= M$ *holds, except with negligible probability.*

Note that algorithm $\mathcal{A}$ is allowed to engage in the proof of retrievability protocol for M in its interaction with algorithm $\mathcal{B}$. We require that algorithm $\mathcal{A}$ succeeds when it causes Pub.V to accept with any non-negligible probability.

Actually, in publicly verifiable proof of retrievability systems, for a successful related-key attack, it means that the adversary generates a prover that is ϵ-admissible but for which extraction fails.

6.4 A Related-Key Attack on the Shacham-Waters Scheme

In this section, we point out a linear attack on the publicly verifiable scheme based on the computational DH assumption proposed by Shacham and Waters [SW08].

To clarify the presentation of the attack, we simplify the Shacham-Waters publicly verifiable scheme as follows. Let $\hat{e} : G \times G \rightarrow G_T$ be a bilinear map over a bilinear group G of prime order p with a generator $g \in G$. Let $H : \{0,1\}^* \rightarrow G$ be a collision resistant hash function. Let $m_1, \ldots, m_n \in Z_p^*$ for some large prime q be n blocks of an erasure encoded file. A user's private key is $x \in Z_p^*$, and its public key is $v = g^x \in G$ along with another generator $u \in G$. The authenticator on block i is $\sigma_i = (H(i)u^{m_i})^x$. The blocks $\{m_i\}$ and authenticators $\{\sigma_i\}$ are stored on the server side. The proof of retrievability protocol runs as follows. The verifier chooses a random challenge set I of l indices along with l random coefficients in Z_p^* . Let Q be the set $\{(i, \nu_i)\}$ of challenge index-coefficient pairs. The verifier sends Q to the prover. On receiving a query $Q = \{(i, \nu_i)\}$, the prover computes and sends back the

response, a pair (σ, μ), as

$$\sigma = \prod_{(i,\nu_i)\in Q} {\sigma_i}^{\nu_i}, \quad \mu = \sum_{(i,\nu_i)\in Q} \nu_i \cdot m_i.$$

The verifier can check that the response is correctly formed by checking that

$$\hat{e}(\sigma, g) = \hat{e}(\prod_{(i,\nu_i)\in Q} H(i)^{\nu_i} \cdot u^{\mu}, v).$$

This scheme has public verifiability as the private key x is required for generating the authenticators $\{\sigma_i\}$, but the public key v is sufficient for the verifier in the proof-of-retrievability protocol.

The attack. We show that under related-key attacks, using authenticators σ' for the blocks of a message m' generated under a related key $sk + \triangle$, an adversary can generate a proof which the verifier will think is valid for a different message m under the key sk. Let $(m_1, \ldots, m_n)$ be blocks of an n-block file. Let the related-key deriving functions $\phi : sk \to sk$ be indexed by $\triangle \in Z_p^*$, where $\phi(sk) = sk + \triangle$. A query will consist of l indices along with l random coefficients in Z_p^* ; the authenticator σ_i' on the block m_i' of a user with public key $v = g^{x+\triangle}$ is $\sigma_i' = (H(i)u^{m_i'})^{x+\triangle}$. The adversary stores the blocks $\{m_i'\}$ and the authenticators $\{\sigma_i'\}$. Now, consider query $Q = \{(i, \nu_i)\}$ on a user with public key $v = g^x$, and the adversary responds with

$$\sigma = \prod_{(i,\nu_i)\in Q} \left(\frac{\sigma_i'}{(H(i)u^{m_i'})^{\triangle}}\right)^{\nu_i}, \quad \mu = \sum_{(i,\nu_i)\in Q} \nu_i \cdot m_i'.$$

Obviously, the adversary makes

$$\hat{e}(\sigma, g) = \hat{e}(\prod_{(i,\nu_i)\in Q} H(i)^{\nu_i} \cdot u^{\mu}, v)$$

hold without storing the actual authenticators $\{\sigma_i\}$ on the blocks $\{m_i\}$, where $\sigma_i = (H(i)u^{m_i})^x$.

We would like to remark that this attack is outside the security model considered in the Shacham-Waters system [SW08].

6.5 A Public Verification System Secure Against Related-Key Attacks

In this section, we put forward a public verification scheme based on the Shacham-Waters scheme [SW08], and present its security proof under the computational DH assumption.

Our construction works in the group Z_p^*. When we work in the bilinear setting, the group Z_p^* is the support of the bilinear group G, i.e., $\#G = p$. In queries, coefficients will come from a set $B \subseteq Z_p^*$.

To make the system more efficient in certain situations, we assume that after preliminary processing of a file, the file is split into $n \in Z_p^*$ blocks for some large prime p, and each block is split into $s \in Z_p^*$ sectors. We will refer to individual file sectors as $\{m_{ij}\}$, with $1 \leq i \leq n$ and $1 \leq j \leq s$.

6.5.1 Queries and Aggregation

Queries. Let coefficients in queries come from a set $B \subseteq Z_p^*$. A query is an l-element set $Q = \{(i, \nu_i)\}$. Each entry $(i, \nu_i) \in Q$ is such that i is a block index in the range $[1, n]$, and ν_i is a multiplier in B. The size l of Q is a system parameter, as is the choice of the set B.

When the verifier chooses a random query. First, it chooses an l-element subset I of $[1, n]$ uniformly at random. Next, for each element $i \in I$, it chooses an element $\nu_i \in B$ uniformly at random. This procedure implies selection of l elements from $[1, n]$ without replacement but a selection of l elements from B with replacement.

Though the set notation $Q = \{(i, \nu_i)\}$ is space-efficient and convenient for implementation, we will also make use of a vector notation in the analysis. A query Q over indices $I \subset [1, n]$ is denoted by a vector $\mathbf{q} \in Z_p^*$ where $\mathbf{q}_i = \nu_i$ for $i \in I$ and $\mathbf{q}_i = 0$ for all $i \notin I$. Equally, let $\mathbf{u}_1, \ldots, \mathbf{u}_n$ be the usual basis for Z_p^*, we have

$$\mathbf{q} = \sum_{(i,\nu_i) \in Q} \nu_i \mathbf{u}_i.$$

If the set B does not contain 0 then a random query is a random weight-l vector in Z_p^* with coefficients in B. If B does contain 0, then a similar argument can be made, but care must be taken to distinguish the case "$i \in I$ and $\nu_i = 0$" from the case "$i \notin I$".

Aggregation. To respond to a query Q, the server computes the value

$$\mu_j = \sum_{(i,\nu_i) \in Q} \nu_i m_{ij},$$

where $1 \leq j \leq s$.

That is, by combining sectorwise the blocks named in Q, each with its multiplier ν_i. The response is $(\mu_1, \ldots, \mu_s) \in \mathbb{Z}_p^*$.

Supposing that the message blocks on the server are viewed as an $n \times s$ element matrix $M = (m_{ij})$, then using the vector notation for queries given above, the server's response is given by $\mathbf{q}M$.

6.5.2 Parameter Selection

Let λ be the security parameter; typically, $\lambda = 80$. p should be 2λ-bit prime, and the curve should be chosen so that discrete logarithm is 2^λ-secure. For values of λ up to 128, Barreto-Naehrig curves [BN05] are the correct choice.

Let n be the number of blocks in the file. We assume that $n \gg \lambda$. Suppose we use a rate-ρ erasure code, i.e., one in which any ρ-fraction of the blocks suffices for decoding. Let l be the number of indices in the query Q, and $B \subseteq Z_p^*$ be the set from which the challenge weight ν_i is drawn. It is the requirement, to guarantee that extraction will succeed from any adversary convincingly answering an ϵ-fraction of queries, provided that $\epsilon - \rho^l - 1/\#B$ is non-negligible in the security parameter λ, that guides the choice of parameters.

A conservative choice is $\rho = 1/2$, $l = \lambda$, $B = \{0,1\}^\lambda$, which guarantees extraction against any adversary. It has been shown in [SW08] that for 80-bit security, the challenge coefficient ν_i can be 80 bits long. The smaller these coefficients, the more efficient the multiplications or exponentiations that involve them.

6.5.3 Construction

Let $\hat{e} : G \times G \to G_T$ be a bilinear map over a bilinear group G of prime order p with a generator $g \in G$. Let $H : \{0,1\}^* \to G$ be a collision resistant hash function. Let (SKg, SSig, SVer) be a signature scheme secure against related-key attacks, of which the details about the construction can be found in [BPT12]. Note that our construction essentially follows that of the Shacham-Waters scheme with the exception of the inclusion of g^α within the hash in the definition of σ_i, thereby addressing well the aforementioned related-key attack problem, which we will detail later in this section. We give the formal description for our public verification system resistant to related-key attacks Pub as follows.

- Pub.Kg(1^λ). Taking a security parameter λ as input, this algorithm first generates a random signing key pair (spk, ssk) $\leftarrow$ SKg(1^λ), and then chooses a random $\alpha \in Z_p^*$, and computes $v = g^\alpha$. The secret key is $sk = (\alpha, ssk)$; the

public key is $pk = (v, spk)$.

- Pub.St(sk, M). Given a file M, this algorithm first splits M into n blocks, each s sectors long: $\{m_{ij}\}$ ($1 \leq i \leq n$ and $1 \leq j \leq s$). Then it chooses a random file name $name \in Z_p^*$ and s random elements $u_1, \ldots, u_s \in G$. Let t_0 be "$name||n||u_1|| \cdots ||u_s$", and t be a file tag of t_0 together with a signature on t_0 under the private key ssk : $t = t_0||\text{SSig}(ssk, t_0)$. For each i, $1 \leq i \leq n$, it computes

$$\sigma_i = (H(name||i||g^{\alpha}) \cdot \textstyle\prod_{j=1}^{s} {u_j}^{m_{ij}})^{\alpha}.$$

  The processed file M^* is $\{m_{ij}\}$, $1 \leq i \leq n$, $1 \leq j \leq s$ together with $\{\sigma_i\}$, $1 \leq i \leq n$.

- Pub.P(pk, t, M^*). For a query Q, an l-element set $\{(i, \nu_i)\}$ ($i \in [1, n]$ and $\mu_i \in B$), sent by the verifier, this algorithm computes

$$\mu_j = \sum_{(i,\nu_i) \in Q} \nu_i m_{ij} \in Z_q^*, \quad \sigma = \prod_{(i,\nu_i) \in Q} {\sigma_i}^{\nu_i} \in G,$$

  where $1 \leq j \leq s$. The response is the values $\mu_1, \ldots, \mu_s$ and σ.

- Pub.V(pk, t). Firstly, this algorithm verifies the signature on t by SVer(spk, t_0, SSig(ssk, t_0)). If this is a invalid signature, it outputs 0; otherwise, it checks whether

$$\hat{e}(\sigma, g) = \hat{e}(\prod_{(i,\nu_i) \in Q} H(name||i||v)^{\nu_i} \cdot \prod_{j=1}^{s} {u_j}^{\mu_j}, v).$$

  If the equation holds, it outputs 1; otherwise, it outputs 0.

6.6 Security Proofs

Under the model which we provided in Section 6.3, below we prove the security of our system in three parts.

1. Prove that the verifying algorithm will reject except when the prover's $\{\mu_j\}$ $= \sum_{(i,\nu_i) \in Q} \nu_i m_{ij}$ are correctly computed.

2. Prove that a ρ fraction of the file blocks can be efficiently reconstructed when interacting with a prover providing correctly-computed $\{\mu_j\}$ response for a fraction of the query space.

3. Prove that a ρ fraction of the file blocks is sufficient for reconstructing the original file.

The first part shows that an adversary can never give a forged response back to a verifier. The second part shows that from any adversary that passes the check we can extract a constant fraction of the blocks. The second step uses the fact that all verified responses must be legal. Lastly, we show that if this constant fraction of blocks is recovered we are able to reconstruct the original file.

Intuitively, the crucial point is under related-key attacks, only the first part of the proof is different from that given in [SW08], the second and third parts of the proof are roughly identical. Consequently, we focus only on the proof of the first part, and review the proofs of the other two.

6.6.1 Part-One Proof

Theorem 6.1 *Assuming that the signature scheme used for file tags is existentially unforgeable against the chosen message and related-key attacks, and the computational DH problem holds in G, then, in the random oracle, except with negligible probability no adversary against the soundness of our public verification scheme ever causes* Pub.V *to output* 1 *in one proof of retrievability protocol instance, except by responding with the correct values of* $\{\mu_j\}$ *and* σ *generated by* Pub.P.

Proof. We prove this theorem in a series of games. Note that our proof follows the structure of the proof of Shacham and Waters [SW08] for their scheme, but the reason that their proof errs in a related-key setting while our proof can address this is that the computation of the hash function in their scheme is independent of the key means that their simulation in Game 2 (and similarly Game 3) does not work in a related-key setting, since the adversary can distinguish the simulated hash values from a true random oracle; while in our scheme, the fact that the hash value depends on g^{α} as well allows us to avoid this issue, and then the rest of the proof can follow as before.

- Game 0. This game is simply the security game as defined.
- Game 1. Game 1 is the same as Game 0 except that algorithm $\mathcal{B}$ keeps a list of all signed file tags including those as part of a storing protocol query under a related-key deriving function $\phi(ssk)$ chosen by algorithm $\mathcal{A}$. If algorithm $\mathcal{A}$

ever submits a file tag t either in initiating a proof of retrievability protocol or as the challenge file tag that has a valid signature under ssk but is not a file tag signed by algorithm $\mathcal{B}$ under ssk, algorithm $\mathcal{B}$ aborts.

Obviously, if algorithm $\mathcal{A}$ can distinguish between Game 0 and Game 1, algorithm $\mathcal{A}$ can be used to forge the signature scheme under related-key attacks.

- Game 2. Game 2 is the same as Game 1 except that algorithm $\mathcal{B}$ keeps a list of its responses to file storing queries under related-key deriving functions from algorithm $\mathcal{A}$. Here algorithm $\mathcal{B}$ runs Pub.St under $\phi(sk)$ for the results. Algorithm $\mathcal{B}$ observes all the instances of the proof of retrievability protocol with algorithm $\mathcal{A}$, whether because of proof of retrievability queries from algorithm $\mathcal{A}$, or in the test from Pub.$\mathcal{P}$, or as part of the extraction attempt from Extr. If in any of these instances algorithm $\mathcal{A}$ is successful, but the aggregate signature σ from algorithm $\mathcal{A}$ and $\prod_{(i,\nu_i)\in Q} {\sigma_i}^{\nu_i}$ are not equal, where Q is the challenge issued by the verifier and σ_i are the signatures on the blocks of the file considered in the protocol, algorithm $\mathcal{B}$ aborts.

  In the following we first describe some notations and obtain some conclusions, and then analyze the probability of success that algorithm $\mathcal{A}$ can differentiate between Game 1 and Game 2. Suppose the file that causes algorithm $\mathcal{B}$ to abort is n blocks long, has name $name$ and generating exponents $\{u_j\}$, and contains sectors $\{m_{ij}\}$, and that the signatures on blocks issues by Pub.St are $\{\sigma_i\}$. Let $Q = \{(i, \nu_i)\}$ be the query that causes algorithm $\mathcal{B}$ to abort, and algorithm $\mathcal{A}$'s response to this query be $\mu'_1, \ldots, \mu'_s$ together with σ'. Assume that the valid response from an honest prover is $\mu_1, \ldots, \mu_s$ for

$$\mu_j = \sum_{(i,\nu_i)\in Q} \nu_i m_{ij},$$

  $1 \leq j \leq s$, together with

$$\sigma = \prod_{(i,\nu_i)\in Q} {\sigma_i}^{\nu_i}.$$

  From the verifying algorithm, we have

$$\hat{e}(\sigma, g) = \hat{e}(\prod_{(i,\nu_i)\in Q} H(name||i||v)^{\nu_i} \cdot \prod_{j=1}^{s} {u_j}^{\mu_j}, v).$$

Because algorithm $\mathcal{B}$ aborted, such that $\sigma' \neq \sigma$ but

$$\hat{e}(\sigma', g) = \hat{e}(\prod_{(i,\nu_i)\in Q} H(name||i||v)^{\nu_i} \cdot \prod_{j=1}^{s} {u_j}^{\mu'_j}, v)$$

holds, where $v = g^{\alpha}$ is part of the public key. We can see that $\sigma' = \sigma$ if $\mu'_j = \mu_j$ for $1 \leq j \leq s$, which contradicts the above assumption. Hence, for $1 \leq j \leq s$, there must be at least one of $\{\mu'_j - \mu_j\}$ satisfying $\mu'_j - \mu_j \neq 0$.

Next, we show that if algorithm $\mathcal{A}$ can distinguish between Game 1 and Game 2, algorithm $\mathcal{B}$ can solve the computational DH problem.

Algorithm $\mathcal{B}$ is given $g, h, g^{\alpha} \in G$ as input, and it aims to output h^{α}. Algorithm $\mathcal{B}$ behaves as in Game 1 except that:

- Algorithm $\mathcal{B}$ set $v = g^{\alpha}$ as the public key, and it has no idea about the value of the private key α.
- Algorithm $\mathcal{B}$ keeps a list of queries and responses to a random oracle H. For a query from algorithm $\mathcal{A}$, it chooses a random $r_i \in Z_p^*$ and answers with $g^{r_i} \in G$. In addition, it answers queries of the form $H(name||i||V)$, $V \in G$, in a special way, which we will see below.
- Algorithm $\mathcal{A}$ issues a storing file query on a related-key deriving function $\phi(\alpha) = \alpha + \triangle$, which comprises n blocks $\{m_{ij}\}$, $1 \leq i \leq n$, $1 \leq j \leq s$,
  1. algorithm $\mathcal{B}$ chooses a random name $name \in Z_p^*$.
  2. algorithm $\mathcal{B}$ chooses random $\beta_j, \gamma_j \in Z_p^*$ for $1 \leq j \leq s$, and sets $u_j = g^{\beta_j} \cdot h^{\gamma^j}$.
  3. algorithm $\mathcal{B}$ chooses a random $r_i \in Z_p^*$, and responds to the random oracle of $name||i||V$ as

$$H(name||i||V) = g^{r_i}/(g^{\sum_{j=1}^{s} \beta_j m_{ij}} \cdot h^{\sum_{j=1}^{s} \gamma_j m_{ij}}).$$

Since

$$\begin{aligned} H(name||i||V) \cdot \prod_{j=1}^{s} {u_j}^{m_{ij}} &= \prod_{j=1}^{s} {u_j}^{m_{ij}} \cdot g^{r_i}/g^{\sum_{j=1}^{s} \beta_j m_{ij}} \cdot h^{\sum_{j=1}^{s} \gamma_j m_{ij}} \\ &= \frac{g^{\sum_{j=1}^{s} \beta_j m_{ij}} \cdot h^{\sum_{j=1}^{s} \gamma_j m_{ij}} \cdot g^{r_i}}{g^{\sum_{j=1}^{s} \beta_j m_{ij}} \cdot h^{\sum_{j=1}^{s} \gamma_j m_{ij}}} = g^{r_i}, \end{aligned}$$

algorithm $\mathcal{B}$ can compute

$$\sigma_i = (H(name||i||g^{\alpha} \cdot g^{\triangle}) \cdot \prod_{j=1}^{s} {u_j}^{m_{ij}})^{\alpha+\triangle}$$
$$= (g^{\alpha} \cdot g^{\triangle})^{r_i}.$$

Note that the adversary cannot distinguish the simulated hash value from the real random oracle because the hash value also depends on g^{α}, such that when a related-key deriving function $\phi(\alpha)$ is queried, the hash value will be correspondingly changed.

– Algorithm $\mathcal{A}$ continues issuing file storing queries to algorithm $\mathcal{B}$ until algorithm $\mathcal{A}$ succeeds in responding with a signature σ' that is different from the valid signature σ.

The changes made from Game 0 to Game 1 show that the parameters associated with this protocol instance are generated by algorithm $\mathcal{B}$ as part of a Pub.St query; otherwise, execution will have already aborted. In other words, these parameters are yielded according to algorithm $\mathcal{B}$'s procedure described above. Therefore, algorithm $\mathcal{B}$ obtains

$$\hat{e}(\sigma'/\sigma, g) = \hat{e}(\prod_{j=1}^{s} {u_j}^{\mu'_j - \mu_j}, v) = \hat{e}(\prod_{j=1}^{s} (g^{\beta_j} \cdot h^{\gamma_j})^{\mu'_j - \mu_j}, v)$$
$$\Rightarrow \hat{e}(\frac{\sigma'}{\sigma} \cdot v^{-\sum_{j=1}^{s} \beta_j(\mu'_j - \mu_j)}, g) = \hat{e}(h, v)^{\sum_{j=1}^{s} \gamma_j(\mu'_j - \mu_j)}$$
$$\Rightarrow h^{\alpha} = (\frac{\sigma'}{\sigma} \cdot v^{-\sum_{j=1}^{s} \beta_j(\mu'_j - \mu_j)})^{\frac{1}{\sum_{j=1}^{s} \gamma_j(\mu'_j - \mu_j)}}.$$

We can see that algorithm $\mathcal{B}$ solves the computational DH problem.

Thus if algorithm $\mathcal{A}$ can distinguish Game 1 and Game 2, algorithm $\mathcal{B}$ can solve the computational DH problem.

- Game 3. Game 3 is the same as Game 2 except that if in any of the instances of the proof of retrievability protocol, algorithm $\mathcal{A}$ is successful but there exists at least one of the aggregate messages m_j that is not equal to the valid $\sum_{(i,\nu_i) \in Q} \nu_i m_{ij}$, where Q is the challenge issued by the verifier, algorithm $\mathcal{B}$ aborts.

Suppose the file that causes algorithm $\mathcal{B}$ to abort is n blocks long, has name *name* and generating exponents $\{\mu_j\}$, and contains sectors $\{m_{ij}\}$, and that

the signatures on blocks issues by Pub.St are $\{\sigma_i\}$. Let $Q = \{(i, \nu_i)\}$ be the query that causes algorithm $\mathcal{B}$ to abort, and algorithm $\mathcal{A}$'s response to this query be $\mu'_1, \ldots, \mu'_s$ together with σ'. Assume that the valid response from an honest prover is $\mu_1, \ldots, \mu_s$ for

$$\mu_j = \sum_{(i,\nu_i)\in Q} \nu_i m_{ij},$$

$1 \leq j \leq s$, together with

$$\sigma = \prod_{(i,\nu_i)\in Q} {\sigma_i}^{\nu_i}.$$

Game 2 assures that $\sigma' = \sigma$ such that only $\{\mu'_j\}$ and $\{\mu_j\}$ can make a difference. Again, for $1 \leq j \leq s$, there must be at least one of $\{\mu'_j - \mu_j\}$ satisfying $\mu'_j - \mu_j \neq 0$.

In the following, we prove that if algorithm $\mathcal{A}$ has non-negligible probability in distinguishing Game 2 from Game 3, then algorithm $\mathcal{B}$ can solve the DL problem.

Algorithm $\mathcal{B}$ is given as input $g, h \in G$, and its aim is to output x such that $h = g^x$. Algorithm $\mathcal{B}$ behaves the same as in Game 2 except with the following differences:

- Upon receiving the storing file queries comprising n blocks $\{m_{ij}\}$ for $1 \leq i \leq n$ and $1 \leq j \leq s$ from algorithm $\mathcal{A}$, on a related-key deriving function $\phi(\alpha) = \alpha + \triangle$, algorithm $\mathcal{B}$ runs Pub.St to respond except that algorithm $\mathcal{B}$ chooses random β_j, $\gamma_j \in Z_p^*$, and sets $u_j = g^{\beta_j} \cdot h^{\gamma_j}$, for each j, $1 \leq j \leq s$.

  Note that as the aggregate messages $\{\mu_j\}$ have no relation to the private key α, so the related-key deriving function in this case cannot facilitate algorithm $\mathcal{A}$ any more.

- Algorithm $\mathcal{A}$ continues issuing file storing queries to algorithm $\mathcal{B}$ until algorithm $\mathcal{A}$ succeeds in responding with the aggregate messages $\{\mu'_j\}$ that are not equal to the valid aggregate messages $\{\mu_j\}$.

  From Game 2 we already know that $\sigma' = \sigma$, so the following equation

holds.

$$\begin{aligned}\hat{e}(\sigma, g) &= \hat{e}(\prod_{(i,\nu_i)\in Q} H(name||i||g^{\alpha})^{\nu_i} \cdot \prod_{j=1}^{s} {u_j}^{\mu_j}, v) = \hat{e}(\sigma', g) \\ &= \hat{e}(\prod_{(i,\nu_i)\in Q} H(name||i||g^{\alpha})^{\nu_i} \cdot \prod_{j=1}^{s} {u_j}^{\mu'_j}, v) \\ &\Rightarrow \prod_{j=1}^{s} {u_j}^{\mu_j} = \prod_{j=1}^{s} {u_j}^{\mu'_j}.\end{aligned}$$

Therefore,

$$\begin{aligned}1 &= \prod_{j=1}^{s} {u_j}^{\mu'_j - \mu_j} = \prod_{j=1}^{s} (g^{\beta_j} \cdot h^{\gamma_j})^{\mu'_j - \mu_j} \\ &= g^{\sum_{j=1}^{s} \beta_j(\mu'_j - \mu_j)} \cdot h^{\sum_{j=1}^{s} \gamma_j(\mu'_j - \mu_j)} \\ &\Rightarrow h = g^{-\frac{\sum_{j=1}^{s} \beta_j(\mu'_j - \mu_j)}{\sum_{j=1}^{s} \gamma_j(\mu'_j - \mu_j)}}\end{aligned}$$

Clearly, algorithm $\mathcal{B}$ solves the discrete logarithm problem.

Thus if algorithm $\mathcal{A}$ can distinguish Game 2 and Game 3, algorithm $\mathcal{B}$ can solve the DL problem.

In Game 3, algorithm $\mathcal{A}$ is restricted from answering any verification query with values other than those that would have been computed by Pub.P. As we have assumed that the signature scheme is secure and computational Diffie-Hellman and discrete logarithm are hard in bilinear groups, there is only a negligible difference in the success probability of algorithm $\mathcal{A}$ in this game compared to Game 1, where the adversary is not constrained in this manner. Moreover, the difficulty of the CDH problem implies the difficulty of the discrete logarithm problem. This completes the proof of Theorem 6.1, the first part of the security.

Remarks. We discuss only the linear related-key deriving function in the above proof. In fact, the related-key deriving function can also be non-linear such as affine, polynomial. Because for the file storing queries, we require Pub.St to recompute g^{α} to obtain the result of the hash function H, this gadget prevents the adversary from forging authenticators from related-key deriving functions without the value of α.

6.6.2 Part-Two Proof

Below we prove the second part of the security. A cheating prover Pub.$\mathcal{P}$ is said to be well behaved if it never causes Pub.V to accept in a proof of retrievability protocol instance except by responding with values $\{\mu_j\}$ and σ that are computed correctly [SW08]. Theorem 6.1 guarantees that the cheating provers outputted by the adversaries that win the soundness game with non-negligible probability are well behaved, provided that the employed cryptographic primitives are secure. In Theorem 6.2, we show that the extraction algorithm always succeeds against a well behaved cheating prover. As the file storing queries on related-key deriving functions cannot give further help to Pub.$\mathcal{P}$, this part is roughly the same as in [SW08].

Theorem 6.2 *Supposing that a cheating prover* Pub.$\mathcal{P}$ *on an n block file M is well behaved, and it is ϵ admissible, then it is possible to recover a ρ fraction of the file blocks in interactions with* Pub.$\mathcal{P}$.

Proof. We rewrite Theorem 6.2 as follows. Supposing that a cheating prover Pub.$\mathcal{P}$ on an n block file M convincingly answers an ϵ fraction of verification queries, then provided that $\epsilon - \omega$ is positive and non-negligible where

$$\omega = 1/\#B + (\rho n)^l/(n - l + 1)^l,$$

it is possible to recover a ρ fraction of the encoded file blocks in $O(n/(\epsilon - \omega))$ interactions with Pub.$\mathcal{P}$ and in $O(n^2 s + (1 + \epsilon n^2)(n)/(\epsilon - \omega))$ time overall [SW08].

In [SW08], it first makes a definition: assume that an adversary algorithm $\mathcal{B}$, implemented as a probabilistic polynomial-time Turing machine, that, given a query Q on its input tape, outputs either the correct response ($\mathbf{q}M$ in vector notation) or a special symbol $\perp$ to its output tape. Suppose that algorithm $\mathcal{B}$ responds with probability ϵ, i.e., on an ϵ fraction of the query-and-random-tape space. Such algorithm $\mathcal{B}$ is said to be ϵ-polite [SW08].

Then it claims that the proof of Theorem 6.2 depends upon the lemma provided below.

Lemma 6.3 *Suppose that algorithm $\mathcal{B}$ is an ϵ-polite adversary as defined above. Let ω equal $1/\#B + (\rho n)^l/(n - l + 1)^l$. If $\epsilon > \omega$ then it is possible to recover a ρ fraction of the encoded file blocks in $O(n/(\epsilon - \omega))$ interactions with algorithm $\mathcal{B}$ and in $O(n^2 s + (1 + \epsilon n^2)(n)/(\epsilon - \omega))$ time overall.*

To apply Lemma 6.3, it is necessary to show that a well-behaved ϵ-admissible cheating prover Pub.$\mathcal{P}$, as output by a setup-game adversary algorithm $\mathcal{A}$, can be turned into an ϵ-polite algorithm $\mathcal{B}$. Below is how algorithm $\mathcal{B}$ is implemented, where Pub.$\mathcal{P}$ is used to construct ϵ-algorithm $\mathcal{B}$ [SW08].

1. Given a query Q, interacts with Pub.$\mathcal{P}$ according to Pub.V(pk, t) $\rightleftharpoons$ Pub.$\mathcal{P}$, playing the role of verifier.

2. If the output of the interaction is 1, writes (μ_1, ..., μ_s) to the output tape; otherwise, writes $\perp$.

Every time algorithm $\mathcal{B}$ runs Pub.$\mathcal{P}$, it provides it with a clean scratch tape and a new randomness tape, effectively rewinding it [SW08]. Since Pub.$\mathcal{P}$ is well-behaved, a successful response will compute (μ_1, ..., μ_s) as prescribed for an honest prover. Since (μ_1, ..., μ_s) is ϵ-admissible, on an ϵ fraction of interactions it answers correctly. Thus algorithm $\mathcal{B}$ that we have constructed is an ϵ-polite adversary.

All that remains to guarantee is that

$$\omega = 1/\#B + (\rho n)^l/(n-l+1)^l$$

is such that $\epsilon - \omega$ is positive and non-negligible. This simply requires each of $1/\#B$ and $(\rho n)^l/(n-l+1)^l$ to be negligible in the security parameter.

To prove Lemma 6.3, some arguments in linear algebra are introduced in [SW08]. For a subspace $\mathbb{D} \in Z_p^*$, denote the dimension of $\mathbb{D}$ by dim $\mathbb{D}$. Let the free variables of a space, free $\mathbb{D}$, be the indices of the basis vector $\{\mathbf{u}_i\}$ include in $\mathbb{D}$, i.e.,

$$\text{free } \mathbb{D} \stackrel{\text{def}}{=} \{i \in [1,n] : \mathbb{D} \cap \mathbf{u}_i = \mathbf{u}_i\}.$$

Here if $\mathbb{D}$ is represented by means of a basis matrix in the row-reduced echelon form, then dim $\mathbb{D}$ and free $\mathbb{D}$ can be efficiently computed [SW08].

Next the following two claims are given in [SW08] to support the proof.

Claim 6.4 *Let $\mathbb{D}$ be a subspace of Z_p^*, and let I be an l-element subset of $[1,n]$. If $I \nsubseteq$ free $\mathbb{D}$, then a random query over indices I with coefficients in B is in $\mathbb{D}$ with probability at most $1/\#B$.*

Proof. The proof is exactly the same as in [SW08]. Let $\mathbb{I}$ be the subspace spanned by the unit vectors in I, i.e., by $\{\mathbf{u}_i\}_{i\in I}$. Clearly, dim $\mathbb{D} \cap \mathbb{I}$ is at most $l-1$; if it equalled l, then $\mathbb{D} \cap \mathbb{I} = \mathbb{I}$ and each of the vectors $\{\mathbf{u}_i\}_{i\in I}$ would be in $\mathbb{D}$, contradicting

the claim statement. Suppose that dim $\mathbb{D} \cap \mathbb{I}$ equals r. Then there exist r indices in I such that a choice of values for the coordinates at these indices determines the values of the remaining $l - r$ coordinates. This means that there are at most $(\#B)^r$ vectors in $\mathbb{D} \cap \mathbb{I}$ with coordinated values in B: a choice of one of $\#B$ values for each of the r coordinates above determines the value to each of the other $l-r$ coordinates; if the values of these coordinates are all in B, then this vector contributes 1 to the count; otherwise it contributes 0. The maximum possible count is thus $(\#B)^r$. By contract, there are $(\#B)^l$ vectors in $\mathbb{I}$ with coordinates in B, and these are exactly the vectors corresponding to each random query with indices I. Thus the probability that a random query is in $\mathbb{D}$ is at most

$$1/(\#B)^{l-r} \leq 1/(\#B),$$

which proves the claim.

Claim 6.5 *Let $\mathbb{D}$ be a subspace of Z_p^*, and suppose that* #free $\mathbb{D} = m$. *Then for a random l-element subset I of $[1, n]$ the probability that $I \subseteq$ free $\mathbb{D}$ is at most $m^l/(n - l + 1)^l$.*

Proof. The proof is exactly the same as in [SW08]. Color the m indices included in free $\mathbb{D}$ black, and color the remaining $n - m$ indices white. A query I corresponds to a choice of l indices out of all these, without replacement. A query satisfies the condition that $I \subseteq$ free $\mathbb{D}$ exactly if every element of I is in free $\mathbb{D}$, i.e., is colored black. Thus the probability that a random query satisfies the condition is just the probability of drawing l black balls, without replacement, from a jar containing m black balls and $n - m$ white balls, and this probability is

$$\binom{m}{l} \Big/ \binom{n}{l} = \frac{m!/(m-l)!}{n!/(n-l)!} < \frac{m^l}{(n-l+1)^l},$$

as required. Note that if $m > n - l$ then $I \subset$ free $\mathbb{D}$ with probability 1; the upper bound given by this claim is a probability greater than 1 in this case, but of course still correct.

Now we begin to prove Lemma 6.3. Assume that the extractor's knowledge at each point is a subspace $\mathbb{D}$, represented by a $t \times n$ matrix A in the row-reduced echelon form [SW08]. Suppose that the query-response pairs contributing to the extractor's knowledge are

$$\mathbf{q}^{(1)}M = (\mu_1^{(1)}, \ldots, \mu_s^{(1)}) \ \cdots \ \mathbf{q}^{(t)}M = (\mu_1^{(t)}, \ldots, \mu_s^{(t)}),$$

or $VM = W$, where V is the $t \times n$ matrix whose rows are $\{\mathbf{q}^i\}$ and W is the $t \times s$ matrix whose rows are $(\mu_1^{(i)}, \ldots, \mu_s^{(i)})$. The row-reduced echelon matrix A is related to V by $A = UV$, where U is a $t \times t$ matrix with a nonzero determinant computed in applying Gaussian elimination to V.

The extractor's knowledge is initially empty, i.e., $\mathbb{D} = \emptyset$. The extractor repeats the following behaviour until $\#(\text{free } \mathbb{D}) \geq \rho n$: the extractor chooses a random query Q, and it turns algorithm $\mathcal{B}$ on Q. Suppose that algorithm $\mathcal{B}$ chooses to respond, giving answer $(\mu_1, \ldots, \mu_s)$; obviously this happens with probability ϵ. Let Q be over indices $I \in [1, n]$, and denote it in vector notation as $\mathbf{q}$. Q can be divided into three types [SW08]: (1) $\mathbf{q} \notin \mathbb{D}$; (2) $\mathbf{q} \in \mathbb{D}$ but $I \nsubseteq \text{free } \mathbb{D}$; (3) $\mathbf{q} \in \mathbb{D}$ and $I \subseteq \text{free } \mathbb{D}$. For queries of the first type, the extractor adds Q to its knowledge $\mathbb{D}$, obtaining new knowledge $\mathbb{D}'$, as follows. It adds a row corresponding to the query to V, obtaining V', and a row corresponding to the response to W, obtaining W'; it modifies the transform matrix U, obtaining U', so that $A' = U'V'$ is again in row-reduced echelon form and spans $\mathbf{q}$. The primed versions $\mathbb{D}'$, A', U', V' and W' replace the unprimed versions in the extractor's state. For queries of type 2 or 3, the extractor does not add to its knowledge. Regardless, the extractor continues with another query.

Obviously, a type-1 query increases dim $\mathbb{D}$ by 1. If dim $\mathbb{D}$ equals n then free $\mathbb{D}$ $= [1, n]$ and $\#(\text{free } \mathbb{D}) = n \geq \rho n$, so the extractor's query phase is guaranteed to terminate by the time it has encountered n type-1 queries.

It is not difficult to see that when the simulator is in its query phase, type-1 queries make up at least a $1 - \omega$ fraction of the query space [SW08]. From Claim 6.4, type-2 queries make up at most a $\#B$ fraction of the query space, since

$$\begin{aligned}
\Pr[Q \text{ is type} - 2] &= \Pr[\mathbf{q} \in \mathbb{D} \wedge I \nsubseteq \text{free } \mathbb{D}] \\
&= \Pr[\mathbf{q} \in \mathbb{D} | I \nsubseteq \text{free } \mathbb{D}] \cdot \Pr[I \nsubseteq \mathbb{D}] \\
&\leq \Pr[\mathbf{q} \in \mathbb{D} | I \nsubseteq \text{free } \mathbb{D}] \\
&\leq 1/\#B,
\end{aligned}$$

where it is the last inequality that follows from the claim (the claim gives a condition for a single I satisfying the condition $I \nsubseteq \text{free } \mathbb{D}$; the inequality here is over all such I; but if the probability never exceeds $1/\#B$ for any specific I then it does not exceed $1/\#B$ over a random choice of I, either). Here the probability expressions are all over a random choice of query Q, and I and $\mathbf{q}$ are the index set and vector form corresponding to the chosen query.

Similarly, suppose that $\#(\text{free } \mathbb{D}) = m$. Then by Claim 6.5, type-3 queries make up at most an $m^l/(n-l+1)^l$ fraction of the query space, since $m < \rho n$[1], this fraction is at most $(\rho n)^l/(n-l+1)^l$ [SW08].

Therefore, the fraction of the query space consisting of type-2 and type-3 queries is at most

$$1/\#B + (\rho n)^l/(n-l+1)^l = \omega.$$

Since query type depends on the query and not on the randomness supplied to algorithm $\mathcal{B}$, it follows that the fraction of query-and-randomness-tape space consisting of type-1 and type-2 queries is also at most ω. Now, algorithm $\mathcal{B}$ must respond correctly to an ϵ fraction of the query-and-randomness-tape space. Even if the adversary is as unhelpful as it can be and this ϵ fraction includes the entire ω fraction of type-2 and type-3 queries, there remains at least an $(\epsilon - \omega)$ fraction of the query-and-randomness-tape space to which the adversary will respond correctly and in which the query is of type 1 and thereby helpful to the extractor [SW08]. Observe that this fraction is nonempty assuming that $\epsilon > \omega$.

Since the extractor needs at most n successful type-1 queries to complete the query phase, and it obtains a successful type-1 query from an interaction with algorithm $\mathcal{B}$ with probability $O(\epsilon - \omega)$, it follows that the extractor will require at most $O(n/(\epsilon - \omega))$ interactions.

With $\mathbb{D}$ represented by a basis matrix A row-reduced echelon form, it is possible, given a query $\mathbf{q}$ to which the adversary has responded, to determine efficiently which type it is [SW08]. The extractor appends $\mathbf{q}$ to A, and runs the Gaussian elimination algorithm on the new row, a process that takes $O(n^2)$ time (more specifically, $O(tn)$ time if A is a $t \times n$ matrix for $t \leq n$). If the reduced row is not all zeros, then the query is type 1; the reduction also means that augmented matrix A' is again in row-reduced echelon form, and the steps of the reduction also give the appropriate updates to the transform matrix U'. Since the reduction need only be performed for the ϵ fraction of queries to which algorithm $\mathcal{B}$ correctly responds, the overall running time of the query phase is $O((1 + \epsilon n^2)n/(\epsilon - \omega))$ [SW08].

Once the query phase is completed, the extractor has matrices A, U, V and W such that $VM = W$ where $M = (m_{ij})$ is the matrix consisting of encoded file blocks, $A = UV$, and A is in row-reduced echelon form. Moreover, there are at least ρn free dimensions in the subspace $\mathbb{D}$ spanned by A and by V [SW08]. Suppose that i is in

[1]Otherwise, the extractor would have ended the query phase.

free $\mathbb{D}$. Since A is in row-reduced echelon form, there must be a row in A, say row t, that equals the i-th basis vector $\mathbf{u}_i$. Multiplying both sides of $VM = W$ by U on the left gives the equation $AM = UW$. For any j, $1 \leq j \leq s$, consider the entry at row t and column j in the matrix AM. It is equal to

$$\mathbf{u}_i \cdot (m_{1,j}, m_{2,j}, \ldots, m_{n,j}) = m_{i,j}.$$

If the matrix product UW is computed, every block of every sector for $i \in$ free $\mathbb{D}$ can thus be read off from it. Computing the matrix multiplication takes $O(n^2 s)$ time. The extractor computes the relevant rows, outputs them, and halts.

Note that in the actual proof of retrievability with public verification systems, it does not expect the extractor algorithm to be used in the outsourced storage deployments, so issues such as efficiency of this algorithm are not very important in practice.

This completes the proof of Lemma 6.3, the second part of the security.

6.6.3 Part-Three Proof

Theorem 6.6 *Given a ρ fraction of the n blocks of a file M^*, it is possible to recover the entire original file M with all but negligible probability.*

Proof. To complete the proof of Theorem 6.6, the third part of the security, we need to use coding theory techniques referred in [AL96, Rab89, Riz97].

- For rate-ρ Reed-Solomon codes this is of slight importance, since any ρ fraction of encoded file blocks suffices for decoding[2].

  Erasure codes. It is easier to verify that a server is storing half the blocks of a file or any other constant fraction r than to verify that it is storing all the blocks of a file: probabilistic checks are unlikely to uncover a single file block's being dropped. Therefore, before storing it on the server, it is preferred to encode an n-block file into a $2n$-block file or more generally (n/r)-block file with the encoding done in such a way that any n blocks suffice for recovering the original file.

  Erasure codes are the codes that provide this property [AL96, Rab89]. The parameter r is called the rate. Some erasure codes are rateless in that for an

[2]For more details about this claim, please refer to Appendix A of [SW08].

n-block file, they allow the generation of arbitrarily many blocks, any n of which suffice for decoding. Some erasure codes, called nearly-optimal, require somewhat mare than n blocks for decoding: $n(1+\epsilon)$ blocks, where ϵ is a parameter; a choice of ϵ effects other code parameters.

A important property of erasure codes is the efficiency of their encoding and decoding procedures [SW08]. Ideally, one would like both procedures to have performance linear in n. This is especially important for our application, where n can be very large. For instance, if a block is 1000 bytes and the file being stored is 1 GB, the we have $n \approx 2^{20}$. A code where encoding and decoding take $O(n^2)$ time would be extremely slow.

Another important property of erasure codes is the sort of erasure they can correct [SW08]. Ideally, it should correct against arbitrary erasures: any n blocks should suffice for recovering the original file. Some codes, however, correct only against random erasures: any n blocks suffice for decoding with overwhelming probability, whereas a maliciously selected n block set will not suffice.

Unfortunately, no codes are known that provide linear decoding time in the presence of arbitrary erasure. Note that codes that correct random erasures only can still be made use of in private retrievability, but in this case additional secret preprocessing is required that makes public retrievability impossible.

- For rate-ρ linear-time codes, the additional measures described in [SW08][3] guarantee that the ρ fraction of blocks retrieved will allow decoding with overwhelming probability. Traditional Reed-Solomon-style erasure codes can be constructed for arbitrary rates allowing recovery of the original file from any r fraction of the encoded file blocks [Riz97]. The encoding and decoding procedures will take $O(n^2)$ time. The code matrix used can be made public and any user can apply the decoding procedure.

  It is recommended in [SW08] to use a systematic code in which the first m blocks of the encoded file are in fact the encoded file itself. This can make recovering the file from a server that is not adversarial much more efficient, as it can be achieved by simply asking for the first m blocks.

[3] See Appendix A.1 of [SW08] for the explicit description.

6.7 Summary

Cloud computing, grid computing and software as a service are becoming increasingly popular, but there are many security and privacy problems that have not been well addressed. For example, a specific problem encountered in the context of cloud storage, where users outsource their data (or files) to untrusted cloud storage servers, is to convince the users that their data are kept intact at the storage servers. An important approach to achieve this goal is called proof of retrievability, by which a storage server can convince a user via a concise proof that its data can be recovered. However, under a special attack called related-key attack where the adversary is given the relations among the public keys of the users in an outsourcing infrastructure, for the public verification systems, the existing protocols fail to achieve the soundness property as required by the proof of retrievability systems. With this problem in mind, we come up with the notion of public verification secure against related-key attacks. After defining the security model of publicly verifiable proof of retrievability in the setting of related-key attacks, we put forward the first public verification scheme secure against related-key attacks, and prove its soundness against related-key attacks under the random oracle model.

Some results of this chapter have been published in "IET Information Security".

Chapter 7

Conclusions

We briefly review the main contributions in this thesis.

1. Achieving security in encryption systems involving third parties. Confidentiality and anonymity under a practical scenario are emphasized in Chapter 3, where a gateway is present for checking the identity of the receiver when a sender communicates with receivers in an anonymous encryption system. To solve this problem, the notion of verifiable and anonymous encryption is put forth. A gateway-based verifiable anonymous encryption is a three-party protocol to enable an outside sender to securely and anonymously transmit a ciphertext to an inside receiver who belongs to a large group of users, taking into account the presence of an untrusted gateway. In addition to the semantic security of the plaintext, anonymity with respect to a malicious gateway and some curious receivers is also provided in the specific constructions. The first scheme is achieved in a broadcast environment, but it is ill designed and vulnerable in the real world, so another two schemes without broadcasting are given from the point of improving the security model step by step.

2. Signcryption secure against related-key attacks. In Chapter 4, the security definitions as indistinguishability, unforgeability and anonymity of signcryption [Zhe97] are considered one step further in the setting of related-key attacks. Under related-key attacks, the adversary in the security games is now allowed to obtain the outcome of the signcryption scheme under the modified private keys of both sides — the sender and the receiver. In order to design signcryption schemes secure against related-key attacks, the corresponding security notions of signcryption secure against related-key attacks are formally defined, and then assuming that the communication parties in the system are honest, a related-key attack secure signcryption scheme without considering

anonymity is proposed. Based on this scheme, anonymity is later achieved as well. After that, the requirement of trusted sender and receiver is removed, and a fully related-key attack resistant signcryption scheme is presented.

3. Related-key attack secure public-key encryption schemes. How to build public-key encryption schemes secure against related-key attacks has been systematically studied in [Wee12, BPT12], and their methods to achieve RKA security in public-key encryption are efficient and easy to perform. However, the constructions of public-key encryption with related-key attack security given in [Wee12, BPT12] are executed in the setting of single-element private keys. Due to this observation, in Chapter 5, other ways different from those described in [Wee12, BPT12] are found to make public-key encryption schemes with multi-element private keys secure against related-key attacks. The first scheme is based on the Cramer-Shoup cryptosystem [CS01], and the second and third schemes with stronger security are on the basis of the systems in [BB04] and [Kil06], respectively. They all achieve linear related-key attack security in public-key encryption with additional randomness.

4. New attacks to publicly verifiable proof of retrievability. As users using outsourcing infrastructures to preserve the data no longer physically possess the storage of their data, it is crucial for the integrity of their data to be guaranteed. Proof of retrievability (PoR) [JJ07] is a protocol from which users can verify the integrity of the data stored at the remote servers without downloading all the data. In Chapter 6, it is claimed that in most existing PoR systems, if an adversary learns the relations among the keys of the users, users' data privacy will be in danger as the integrity of their data cannot be vouched any more. This statement is then convinced by exposing a related-key attack on the Shacham-Waters proof of retrievability scheme, where an adversary, given the relations among the keys, assures that it owns the data without storing the required data of users. Also, a straightforward and simple method is described to resist such attacks in the proof of retrievability systems.

Bibliography

[ABC+05] 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. In *CRYPTO*, volume 3621 of *Lecture Notes in Computer Science*, pages 205–222. Springer, 2005.

[ACFP05] Michel Abdalla, Olivier Chevassut, Pierre-Alain Fouque, and David Pointcheval. A simple threshold authenticated key exchange from short secrets. In *ASIACRYPT*, volume 3788 of *Lecture Notes in Computer Science*, pages 566–584. Springer, 2005.

[AGHO11] Masayuki Abe, Jens Groth, Kristiyan Haralambiev, and Miyako Ohkubo. Optimal structure-preserving signatures in asymmetric bilinear groups. In *CRYPTO*, volume 6841 of *Lecture Notes in Computer Science*, pages 649–666. Springer, 2011.

[AGV09] Adi Akavia, Shafi Goldwasser, and Vinod Vaikuntanathan. Simultaneous hardcore bits and cryptography against memory attacks. In *TCC*, volume 5444 of *Lecture Notes in Computer Science*, pages 474–495. Springer, 2009.

[AHI11] Benny Applebaum, Danny Harnik, and Yuval Ishai. Semantic security under related-key attacks and applications. In *ICS*. Tsinghua University Press, 2011.

[AL96] Noga Alon and Michael Luby. A linear time erasure-resilient code with nearly optimal recovery. *IEEE Transactions on Information Theory*, 42(6):1732–1736, 1996.

[ARP03] Sattam S. Al-Riyami and Kenneth G. Paterson. Certificateless public key cryptography. In *ASIACRYPT*, volume 2894 of *Lecture Notes in Computer Science*, pages 452–473. Springer, 2003.

[ASW98] N. Asokan, Victor Shoup, and Michael Waidner. Optimistic fair exchange of digital signatures (extended abstract). In *EUROCRYPT*, volume 1403 of *Lecture Notes in Computer Science*, pages 591–606. Springer, 1998.

[Bao98] Feng Bao. An efficient verifiable encryption scheme for encryption of discrete logarithms. In *CARDIS*, volume 1820 of *Lecture Notes in Computer Science*, pages 213–220. Springer, 1998.

[BB04] Dan Boneh and Xavier Boyen. Efficient selective-id secure identity-based encryption without random oracles. In *EUROCRYPT*, volume 3027 of *Lecture Notes in Computer Science*, pages 223–238. Springer, 2004.

[BBDP01] Mihir Bellare, Alexandra Boldyreva, Anand Desai, and David Pointcheval. Key-privacy in public-key encryption. In *ASIACRYPT*, volume 2248 of *Lecture Notes in Computer Science*, pages 566–582. Springer, 2001.

[BBW06] Adam Barth, Dan Boneh, and Brent Waters. Privacy in encrypted content distribution using private broadcast encryption. In *Financial Cryptography*, volume 4107 of *Lecture Notes in Computer Science*, pages 52–64. Springer, 2006.

[BC10] Mihir Bellare and David Cash. Pseudorandom functions and permutations provably secure against related-key attacks. In *CRYPTO*, volume 6223 of *Lecture Notes in Computer Science*, pages 666–684. Springer, 2010.

[BCHK07] Dan Boneh, Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from identity-based encryption. *SIAM J. Comput.*, 36(5):1301–1328, 2007.

[BCM11] Mihir Bellare, David Cash, and Rachel Miller. Cryptography secure against related-key attacks and tampering. In *ASIACRYPT*, volume

7073 of *Lecture Notes in Computer Science*, pages 486–503. Springer, 2011.

[BCOP04] Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. Public key encryption with keyword search. In *EUROCRYPT*, volume 3027 of *Lecture Notes in Computer Science*, pages 506–522. Springer, 2004.

[BEG+94] Manuel Blum, William S. Evans, Peter Gemmell, Sampath Kannan, and Moni Naor. Checking the correctness of memories. *Algorithmica*, 12(2/3):225–244, 1994.

[BF01] Dan Boneh and Matt Franklin. Identity-based encryption from the weil pairing. In *CRYPTO*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–219. Springer-Verlag, 2001.

[BG92] Mihir Bellare and Oded Goldreich. On defining proofs of knowledge. In *CRYPTO*, volume 740 of *Lecture Notes in Computer Science*, pages 390–420. Springer, 1992.

[BGLS03] Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In *EUROCRYPT*, volume 2656 of *Lecture Notes in Computer Science*, pages 416–432. Springer, 2003.

[BGW05] Dan Boneh, Craig Gentry, and Brent Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. In *CRYPTO*, volume 3621 of *Lecture Notes in Computer Science*, pages 258–275. Springer, 2005.

[Bih93] Eli Biham. New types of cryptoanalytic attacks using related keys (extended abstract). In *EUROCRYPT*, volume 765 of *Lecture Notes in Computer Science*, pages 398–409. Springer, 1993.

[BK03] Mihir Bellare and Tadayoshi Kohno. A theoretical treatment of related-key attacks: Rka-prps, rka-prfs, and applications. In *EUROCRYPT*, volume 2656 of *Lecture Notes in Computer Science*, pages 491–506. Springer, 2003.

[BK05] Dan Boneh and Jonathan Katz. Improved efficiency for cca-secure cryptosystems built using identity-based encryption. In *CT-RSA*, volume 3376 of *Lecture Notes in Computer Science*, pages 87–103. Springer, 2005.

[BLMQ05] Paulo S. L. M. Barreto, Benoît Libert, Noel McCullagh, and Jean-Jacques Quisquater. Efficient and provably-secure identity-based signatures and signcryption from bilinear maps. In *ASIACRYPT*, volume 3788 of *Lecture Notes in Computer Science*, pages 515–532. Springer, 2005.

[BN05] Paulo S. L. M. Barreto and Michael Naehrig. Pairing-friendly elliptic curves of prime order. In *Selected Areas in Cryptography*, volume 3897 of *Lecture Notes in Computer Science*, pages 319–331. Springer, 2005.

[Bon98] Dan Boneh. The decision diffie-hellman problem. In *ANTS*, volume 1423 of *Lecture Notes in Computer Science*, pages 48–63. Springer, 1998.

[Bou00] Fabrice Boudot. Efficient proofs that a committed number lies in an interval. In *EUROCRYPT*, volume 1807 of *Lecture Notes in Computer Science*, pages 431–444. Springer, 2000.

[Boy03] Xavier Boyen. Multipurpose identity-based signcryption (a swiss army knife for identity-based cryptography). In *CRYPTO*, volume 2729 of *Lecture Notes in Computer Science*, pages 383–399. Springer, 2003.

[BPT12] Mihir Bellare, Kenneth G. Paterson, and Susan Thomson. Rka security beyond the linear barrier: Ibe, encryption and signatures. In *ASIACRYPT*, volume 7658 of *Lecture Notes in Computer Science*, pages 331–348. Springer, 2012.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM Conference on Computer and Communications Security*, pages 62–73. ACM, 1993.

[BS97] Eli Biham and Adi Shamir. Differential fault analysis of secret key cryptosystems. In *CRYPTO*, volume 1294 of *Lecture Notes in Computer Science*, pages 513–525. Springer, 1997.

[BSNS05] Joonsang Baek, Reihaneh Safavi-Naini, and Willy Susilo. Public key encryption with keyword search revisited. *IACR Cryptology ePrint Archive*, 2005:191, 2005.

[BSNS08] Joonsang Baek, Reihaneh Safavi-Naini, and Willy Susilo. Public key encryption with keyword search revisited. In *ICCSA (1)*, volume 5072 of *Lecture Notes in Computer Science*, pages 1249–1259. Springer, 2008.

[BSZ07] Joonsang Baek, Ron Steinfeld, and Yuliang Zheng. Formal proofs for the security of signcryption. *J. Cryptology*, 20(2):203–235, 2007.

[BTV12] Mihir Bellare, Stefano Tessaro, and Alexander Vardy. Semantic security for the wiretap channel. In *CRYPTO*, volume 7417 of *Lecture Notes in Computer Science*, pages 294–311. Springer, 2012.

[CGKS95] Benny Chor, Oded Goldreich, Eyal Kushilevitz, and Madhu Sudan. Private information retrieval. In *FOCS*, pages 41–50. IEEE Computer Society, 1995.

[CHK04] Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from identity-based encryption. In *EUROCRYPT*, volume 3027 of *Lecture Notes in Computer Science*, pages 207–222. Springer, 2004.

[CL01] Jan Camenisch and Anna Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In *EUROCRYPT*, volume 2045 of *Lecture Notes in Computer Science*, pages 93–118. Springer, 2001.

[CS97] Jan Camenisch and Markus Stadler. Efficient group signature schemes for large groups (extended abstract). In *CRYPTO*, volume 1294 of *Lecture Notes in Computer Science*, pages 410–424. Springer, 1997.

[CS01] Ronald Cramer and Victor Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *IACR Cryptology ePrint Archive*, 2001:108, 2001.

[CS03] Jan Camenisch and Victor Shoup. Practical verifiable encryption and decryption of discrete logarithms. In *CRYPTO*, volume 2729 of *Lecture Notes in Computer Science*, pages 126–144. Springer, 2003.

[CSG+05] Dwaine E. Clarke, G. Edward Suh, Blaise Gassend, Ajay Sudan, Marten van Dijk, and Srinivas Devadas. Towards constant bandwidth overhead integrity checking of untrusted data. In *IEEE Symposium on Security and Privacy*, pages 139–153. IEEE Computer Society, 2005.

[CYHC03] Sherman S. M. Chow, Siu-Ming Yiu, Lucas Chi Kwong Hui, and K. P. Chow. Efficient forward and provably secure id-based signcryption scheme with public verifiability and public ciphertext authenticity. In *ICISC*, volume 2971 of *Lecture Notes in Computer Science*, pages 352–369. Springer, 2003.

[Dam87] Ivan Damgård. Collision free hash functions and public key signature schemes. In *EUROCRYPT*, volume 304 of *Lecture Notes in Computer Science*, pages 203–216. Springer, 1987.

[DDN91] Danny Dolev, Cynthia Dwork, and Moni Naor. Non-malleable cryptography (extended abstract). In *STOC*, pages 542–552. ACM, 1991.

[Del07] Cécile Delerablée. Identity-based broadcast encryption with constant size ciphertexts and private keys. In *ASIACRYPT*, volume 4833 of *Lecture Notes in Computer Science*, pages 200–215. Springer, 2007.

[DF] Yevgeniy Dodis and Nelly Fazio. Public key broadcast encryption for stateless receivers. In *Digital Rights Management Workshop*.

[DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[DP08] Stefan Dziembowski and Krzysztof Pietrzak. Leakage-resilient cryptography. In *FOCS*, pages 293–302. IEEE Computer Society, 2008.

[DPP07] Cécile Delerablée, Pascal Paillier, and David Pointcheval. Fully collusion secure dynamic broadcast encryption with constant-size ciphertexts or decryption keys. In *Pairing*, volume 4575 of *Lecture Notes in Computer Science*, pages 39–59. Springer, 2007.

[ES02] Edith Elkind and Amit Sahai. A unified methodology for constructing public-key encryption schemes secure against adaptive chosen-ciphertext attack. *IACR Cryptology ePrint Archive*, 2002:42, 2002.

[FB06] Décio Luiz Gazzoni Filho and Paulo Sérgio Licciardi Messeder Barreto. Demonstrating data possession and uncheatable data transfer. *IACR Cryptology ePrint Archive*, 2006:150, 2006.

[FN94] Amos Fiat and Moni Naor. Broadcast encryption. In *CRYPTO*, volume 773 of *Lecture Notes in Computer Science*, pages 480–491. Springer, 1994.

[FR95] Matthew K. Franklin and Michael K. Reiter. Verifiable signature sharing. In *EUROCRYPT*, volume 921 of *Lecture Notes in Computer Science*, pages 50–63. Springer, 1995.

[FS86] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO*, pages 186–194, 1986.

[Gen03] Craig Gentry. Certificate-based encryption and the certificate revocation problem. In *EUROCRYPT*, volume 2656 of *Lecture Notes in Computer Science*, pages 272–293. Springer, 2003.

[GLM+04] Rosario Gennaro, Anna Lysyanskaya, Tal Malkin, Silvio Micali, and Tal Rabin. Algorithmic tamper-proof (atp) security: Theoretical foundations for security against hardware tampering. In *TCC*, volume 2951 of *Lecture Notes in Computer Science*, pages 258–277. Springer, 2004.

[GM82] Shafi Goldwasser and Silvio Micali. Probabilistic encryption and how to play mental poker keeping secret all partial information. In *STOC*, pages 365–377. ACM, 1982.

[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.

[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. Comput.*, 18(1):186–208, 1989.

[GOR11] Vipul Goyal, Adam O'Neill, and Vanishree Rao. Correlated-input secure hash functions. In *TCC*, volume 6597 of *Lecture Notes in Computer Science*, pages 182–200. Springer, 2011.

[GPS08] Steven D. Galbraith, Kenneth G. Paterson, and Nigel P. Smart. Pairings for cryptographers. *Discrete Applied Mathematics*, 156(16):3113–3121, 2008.

[Gro06] Jens Groth. Simulation-sound nizk proofs for a practical language and constant size group signatures. In *ASIACRYPT*, volume 4284 of *Lecture Notes in Computer Science*, pages 444–459. Springer, 2006.

[GW09] Craig Gentry and Brent Waters. Adaptive security in broadcast encryption systems (with short ciphertexts). In *EUROCRYPT*, volume 5479 of *Lecture Notes in Computer Science*, pages 171–188. Springer, 2009.

[HSH+08] J. Alex Halderman, Seth D. Schoen, Nadia Heninger, William Clarkson, William Paul, Joseph A. Calandrino, Ariel J. Feldman, Jacob Appelbaum, and Edward W. Felten. Lest we remember: Cold boot attacks on encryption keys. In *USENIX Security Symposium*, pages 45–60. USENIX Association, 2008.

[JJ07] Ari Juels and Burton S. Kaliski Jr. Pors: proofs of retrievability for large files. In *ACM Conference on Computer and Communications Security*, pages 584–597. ACM, 2007.

[JN03] Antoine Joux and Kim Nguyen. Separating decision diffie-hellman from computational diffie-hellman in cryptographic groups. *J. Cryptology*, 16(4):239–247, 2003.

[Jou00] Antoine Joux. A one round protocol for tripartite diffie-hellman. In *ANTS*, volume 1838 of *Lecture Notes in Computer Science*, pages 385–394. Springer-Verlag, 2000.

[Kil06] Eike Kiltz. Chosen-ciphertext security from tag-based encryption. In *TCC*, volume 3876 of *Lecture Notes in Computer Science*, pages 581–600. Springer, 2006.

[KMO10] Eike Kiltz, Payman Mohassel, and Adam O'Neill. Adaptive trapdoor functions and chosen-ciphertext security. In *EUROCRYPT*, volume 6110 of *Lecture Notes in Computer Science*, pages 673–692. Springer, 2010.

[Koc96] Paul C. Kocher. Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In *CRYPTO*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113. Springer, 1996.

[LPQ11] Benoît Libert, Kenneth G. Paterson, and Elizabeth A. Quaglia. Anonymous broadcast encryption. *IACR Cryptology ePrint Archive*, 2011:476, 2011.

[LQ03] Benoît Libert and Jean-Jacques Quisquater. New identity based signcryption schemes from pairings. *IACR Cryptology ePrint Archive*, 2003:23, 2003.

[LQ04] Benoît Libert and Jean-Jacques Quisquater. Efficient signcryption with key privacy from gap diffie-hellman groups. In *Public Key Cryptography*, volume 2947 of *Lecture Notes in Computer Science*, pages 187–200. Springer, 2004.

[Luc04] Stefan Lucks. Ciphers secure against related-key attacks. In *FSE*, volume 3017 of *Lecture Notes in Computer Science*, pages 359–370. Springer, 2004.

[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 *EuroPKI*, volume 4582 of *Lecture Notes in Computer Science*, pages 78–93. Springer, 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.

[MKI88] Tsutomu Matsumoto, Koki Kato, and Hideki Imai. Speeding up secret computations with insecure auxiliary devices. In *CRYPTO*, volume 403 of *Lecture Notes in Computer Science*, pages 497–506. Springer, 1988.

[ML02] John Malone-Lee. Identity-based signcryption. *IACR Cryptology ePrint Archive*, 2002:98, 2002.

[MR04] Silvio Micali and Leonid Reyzin. Physically observable cryptography (extended abstract). In *TCC*, volume 2951 of *Lecture Notes in Computer Science*, pages 278–296. Springer, 2004.

[Nac05] David Naccache. Secure and practical identity-based encryption. *IACR Cryptology ePrint Archive*, 2005:369, 2005.

[NR05] Moni Naor and Guy N. Rothblum. The complexity of online memory checking. In *FOCS*. IEEE Computer Society, 2005.

[NY90] Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *STOC*, pages 427–437. ACM, 1990.

[OP01] Tatsuaki Okamoto and David Pointcheval. The gap-problems: A new class of problems for the security of cryptographic schemes. In *Public Key Cryptography*, volume 1992 of *Lecture Notes in Computer Science*, pages 104–118. Springer, 2001.

[Pai99] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *EUROCRYPT*, volume 1592 of *Lecture Notes in Computer Science*, pages 223–238. Springer, 1999.

[PS96] David Pointcheval and Jacques Stern. Security proofs for signature schemes. In *EUROCRYPT*, volume 1070 of *Lecture Notes in Computer Science*, pages 387–398. Springer, 1996.

[PS00] Guillaume Poupard and Jacques Stern. Fair encryption of rsa keys. In *EUROCRYPT*, volume 1807 of *Lecture Notes in Computer Science*, pages 172–189. Springer, 2000.

[Rab89] Michael O. Rabin. Efficient dispersal of information for security, load balancing, and fault tolerance. *J. ACM*, 36(2):335–348, 1989.

[Riz97] Luigi Rizzo. Effective erasure codes for reliable computer communication protocols. *Computer Communication Review*, 27(2):24–36, 1997.

[RTSS09] Thomas Ristenpart, Eran Tromer, Hovav Shacham, and Stefan Savage. Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds. In *ACM Conference on Computer and Communications Security*, pages 199–212. ACM, 2009.

[Sch90] Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In *CRYPTO*, volume 435 of *Lecture Notes in Computer Science*, pages 239–252. Springer, 1990.

[Sch91] Claus-Peter Schnorr. Efficient signature generation by smart cards. *J. Cryptology*, 4(3):161–174, 1991.

[SDFMB+08] Francesc Sebé, Josep Domingo-Ferrer, Antoni Martínez-Ballesté, Yves Deswarte, and Jean-Jacques Quisquater. Efficient remote data possession checking in critical information infrastructures. *IEEE Trans. Knowl. Data Eng.*, 20(8):1034–1038, 2008.

[SM06] Thomas J. E. Schwarz and Ethan L. Miller. Store, forget, and check: Using algebraic signatures to check remotely administered storage. In *ICDCS*, page 12. IEEE Computer Society, 2006.

[Sta96] Markus Stadler. Publicly verifiable secret sharing. In *EUROCRYPT*, volume 1070 of *Lecture Notes in Computer Science*, pages 190–199. Springer, 1996.

[SW08] Hovav Shacham and Brent Waters. Compact proofs of retrievability. In *ASIACRYPT*, volume 5350 of *Lecture Notes in Computer Science*, pages 90–107. Springer, 2008.

[WCW+13] Cong Wang, Sherman S. M. Chow, Qian Wang, Kui Ren, and Wenjing Lou. Privacy-preserving public auditing for secure cloud storage. *IEEE Trans. Computers*, 62(2):362–375, 2013.

[Wee10] Hoeteck Wee. Efficient chosen-ciphertext security via extractable hash proofs. In *CRYPTO*, volume 6223 of *Lecture Notes in Computer Science*, pages 314–332. Springer, 2010.

[Wee12] Hoeteck Wee. Public key encryption against related key attacks. In *Public Key Cryptography*, volume 7293 of *Lecture Notes in Computer Science*, pages 262–279. Springer, 2012.

[WWR+11] Qian Wang, Cong Wang, Kui Ren, Wenjing Lou, and Jin Li. Enabling public auditability and data dynamics for storage security in cloud computing. *IEEE Trans. Parallel Distrib. Syst.*, 22(5):847–859, 2011.

[WWRL10] Cong Wang, Qian Wang, Kui Ren, and Wenjing Lou. Privacy-preserving public auditing for data storage security in cloud computing. In *INFOCOM*, pages 525–533. IEEE, 2010.

[YWD05] Guomin Yang, Duncan S. Wong, and Xiaotie Deng. Analysis and improvement of a signcryption scheme with key privacy. In *ISC*, volume 3650 of *Lecture Notes in Computer Science*, pages 218–232. Springer, 2005.

[Zhe97] Yuliang Zheng. Digital signcryption or how to achieve cost(signature & encryption) << cost(signature) + cost(encryption). In *CRYPTO*, volume 1294 of *Lecture Notes in Computer Science*, pages 165–179. Springer, 1997.

[ZX11] Qingji Zheng and Shouhuai Xu. Fair and dynamic proofs of retrievability. In *CODASPY*, pages 237–248. ACM, 2011.