

2011

Cryptographic protocols involving partially trusted third parties

Wei Wu

University of Wollongong

Recommended Citation

Wu, Wei, Cryptographic protocols involving partially trusted third parties, Doctor of Philosophy thesis, School of Computer Science and Software Engineering, University of Wollongong, 2011. <http://ro.uow.edu.au/theses/3221>



Cryptographic Protocols Involving Partially Trusted Third Parties

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

Doctor of Philosophy

from

UNIVERSITY OF WOLLONGONG

by

Wei Wu

School of Computer Science and Software Engineering
February 2011

© Copyright 2011

by

Wei Wu

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.

Wei Wu
February 18, 2011

Abstract

Third parties are very common in modern cryptography, both in symmetric cryptographic applications (e.g., a key distribution protocol and a non-repudiation protocol) and public key cryptographic systems (e.g., the certificate authority and the private key generator in identity-based cryptography). A widely used approach is to assume that a third party has a certain level of trust, but is not fully trusted. In this thesis, we present several new results about cryptosystems, which do not require any fully trusted third party. We provide detailed schemes and security analysis on server-aided verification signatures, certificate-based signatures and encryption, and optimistic fair exchange.

The computational cost required by cryptographic protocols is a significant burden to power-constrained devices such as smart cards and mobile terminals. Server-aided computation is a promising solution to mitigate this issue by employing a powerful server to carry out costly cryptographic computation. Such techniques require an additional care since servers cannot be fully trusted. In this thesis, we investigate the issues of server-aided verification signatures, i.e., using server(s) to assist signature verification. Several notions in digital signatures with server-aided verification are formally defined, based on different trust levels of the server. As instances, we provide server-aided verification protocols for Boneh-Lynn-Shacham (BLS) signature (with the random oracle assumption) and Waters signature (without the random oracle assumption).

As cryptographic primitives, certificate-based public key cryptography (CB-PKC) and certificateless public key cryptography (CL-PKC) are used to ease certificate management and solve key escrow problems due to the trust third party in identity-based public key cryptography. As one of features, the signature verification (resp. encryption) does not require public key authenticity check. In this thesis, we take a closer look at the relationship between certificateless cryptosystem and certificate-based cryptosystem, present the security definitions of certificate-based signatures

and certificate-based encryption, and provide generic constructions of certificate-based signatures from certificateless signatures and certificate-based encryption from certificateless encryption, respectively. The proposed generic constructions are proven to be secure in the random oracle model.

An off-line trusted third party (arbitrator) plays an important role in optimistic fair exchange (OFE), which allows two parties to exchange their digital items in a fair way. As one of the fundamental problems in secure electronic business and digital rights management, OFE has been studied intensively since its introduction. This thesis introduces and defines a new property for OFE: Strong Resolution-Ambiguity. We show that many existing OFE protocols have the new property, but its formal investigation has been missing in those protocols. We prove that in the certified-key model, an OFE protocol is secure in the multi-user setting if it is secure in the single-user setting and has the property of strong resolution-ambiguity. An OFE protocol has the property of strong resolution-ambiguity if one can transform a partial signature σ' into a full signature σ using signer's private key or arbitrator's private key, and given such a pair (σ', σ) , it is infeasible to tell which key is used in the conversion. Our result not only simplifies the security analysis of OFE protocols in the multi-user setting but also provides a new approach for the design of multi-user secure OFE protocols. In addition, a new OFE protocol with strong resolution-ambiguity is proposed. Our analysis shows that the protocol is setup-free, stand-alone and multi-user secure without the random oracle assumption.

Acknowledgement

I would like to express my deep and sincere gratitude to my supervisor Professor Yi Mu, for his motivation, patience, enthusiasm, and immense knowledge. He has provided invaluable suggestions and encouragement from the beginning of my research career. This thesis would have been impossible without his support.

I would like to thank my co-supervisor, Professor Willy Susilo. He has made available his support in a number of ways. My thanks also go to Professor Futai Zhang for his advice and support since 2003. I am indebted to my colleagues and friends, a non-exhaustive list of whom includes: Man Ho Au, Xiaofeng Chen, Fuchun Guo, Jinguang Han, Shekh Faisal ABDUL LATIP, Jiguo Li, Ching Yu Ng, Angela Piper, Shams Ud Din Qazi, Mohammad Reza Reyhanitabar, Siamak Fayyaz Shahandashti, Pairat Thorncharoensri, Raylin Tso, Rungrat Wiangsripanawan, Duncan S. Wong, Qianhong Wu, Yong Yu, Tsz Hon Yuen, Fangguo Zhang and Miao Zhou, as well as the anonymous referees who reviewed the papers included in this thesis.

It is an honour for me to be a student of Centre for Computer and Information Security Research and the School of Computer Science and Software Engineering. I would like to thank the University of Wollongong for providing the International Postgraduate Research Scholarship and the University Postgraduate Award.

I would like to thank my husband Xinyi Huang, for his patience and love. Without him, this work would never be possible.

Publications

I published the following papers related to this thesis.

1. Wei Wu, Yi Mu, Willy Susilo and Xinyi Huang. *Provably Secure Server-Aided Verification Signatures,*” *Journal of Computers and Mathematics with Applications, 2011* (accepted, 27 Jan. 2011).
2. Xinyi Huang, Yi Mu, Willy Susilo, Wei Wu, and Yang Xiang. *Further Observations on Optimistic Fair Exchange Protocols in the Multi-user Setting.* In Phong Q. Nguyen and David Pointcheval, editors, 13th International Conference on Practice and Theory in Public Key Cryptography, PKC 2010, Lecture Notes in Computer Science 6056, pages 124-141. Springer, 2010.
3. Wei Wu, Yi Mu, Willy Susilo and Xinyi Huang. *Certificate-Based Signatures: New Definitions and A Generic Construction from Certificateless Signatures.* In Kyo-Il Chung, Kiwook Sohn and Moti Yung, editors, Information Security Applications, 9th International Workshop, WISA 2008, Lecture Notes in Computer Science 5379, pages 99-114. Springer, 2009.
4. Wei Wu, Yi Mu, Willy Susilo and Xinyi Huang. *Certificate-Based Signatures Revisited.* *Journal of Universal Computer Science* 15(8), pages 1659-1684. 2009.
5. Xinyi Huang, Willy Susilo, Yi Mu and Wei Wu. *Delegating Authentication Power in Mobile Networks.* Book Chapter in S. Gritzalis, T. Karygiannis and C. Skianis, editors, Security and Privacy in Wireless and Mobile Computing. Troubador Publishing, 2009.
6. Wei Wu, Yi Mu, Willy Susilo and Xinyi Huang. *Server-Aided Verification Signatures: Definitions and New Constructions.* In Joonsang Baek, Feng

- Bao, Kefei Chen and Xuejia Lai, editors, Provable Security, Second International Conference, ProvSec 2008, Lecture Notes in Computer Science 5324, pages 141-155. Springer, 2008.
7. Wei Wu, Yi Mu, Willy Susilo and Xinyi Huang. *Convertible Undeniable Proxy Signatures: Security Models and Efficient Construction*. In Sehun Kim, Moti Yung and Hyung-Woo Lee, editors, Information Security Applications, 8th International Workshop, WISA 2007, Lecture Notes in Computer Science 4867, pages 16-29. Springer, 2008.
 8. Wei Wu, Yi Mu, Willy Susilo and Xinyi Huang. *Provably Secure Identity-Based Undeniable Signatures with Selective and Universal Convertibility*. In Dingyi Pei, Moti Yung, Dongdai Lin and Chuankun Wu, editors, Information Security and Cryptology, Third SKLOIS Conference, Inscrypt 2007, Lecture Notes in Computer Science 4990, pages 25-39. Springer, 2008.
 9. Jiguo Li, Xinyi Huang, Yi Mu and Wei Wu. *Cryptanalysis and Improvement of An Efficient Certificateless Signature Scheme*. Journal of Communications and Networks 10(1), pages 10-17. 2008.
 10. Wei Wu, Yi Mu, Willy Susilo, Jennifer Seberry and Xinyi Huang. *Identity-Based Proxy Signature from Pairings*. In Bin Xiao, Laurence Tianruo Yang, Jianhua Ma, Christian Müller-Schloer, and Yu Hua, editors, Autonomic and Trusted Computing, 4th International Conference, ATC 2007, Lecture Notes in Computer Science 4610, pages 22-31. Springer, 2007.
 11. Xinyi Huang, Yi Mu, Willy Susilo, Duncan S. Wong and Wei Wu. *Certificateless Signature Revisited*. In Josef Pieprzyk, Hossein Ghodosi and Ed Dawson, editors, Information Security and Privacy, 12th Australasian Conference, ACISP 2007, Lecture Notes in Computer Science 4586, pages 308-322. Springer, 2007.
 12. Xinyi Huang, Willy Susilo, Yi Mu and Wei Wu. *Proxy Signature Without Random Oracles*. In Jiannong Cao, Ivan Stojmenovic, Xiaohua Jia and Sajal K. Das, editors, Mobile Ad-hoc and Sensor Networks, Second International Conference, MSN 2006, Lecture Notes in Computer Science 4325, pages 473-484. Springer, 2006.

Other Publications.

1. Xinyi Huang, Willy Susilo, Yi Mu and Wei Wu. *Secure Universal Designated Verifier Signature without Random Oracles*. International Journal of Information Security 7(3), pages 171-183. Springer, 2008.
2. Xinyi Huang, Yi Mu, Willy Susilo and Wei Wu. *Provably Secure Pairing-Based Convertible Undeniable Signature with Short Signature Length*. In Tsuyoshi Takagi, Tatsuaki Okamoto, Eiji Okamoto and Takeshi Okamoto, editors, Pairing-Based Cryptography-Pairing 2007, First International Conference, Lecture Notes in Computer Science 4575, pages 367-391. Springer, 2007.
3. Xinyi Huang, Yi Mu, Willy Susilo and Wei Wu. *A Generic Construction for Universally-Convertible Undeniable Signatures*. In Feng Bao, San Ling, Tatsuaki Okamoto, Huaxiong Wang and Chaoping Xing, editors, Cryptology and Network Security, 6th International Conference, CANS 2007, Lecture Notes in Computer Science 4856, pages 15-33. Springer, 2007.
4. Xinyi Huang, Willy Susilo, Yi Mu and Wei Wu. *Universal Designated Verifier Signature Without Delegatability*. In Peng Ning, Sihan Qing and Ninghui Li, editors, Information and Communications Security, 8th International Conference, ICICS 2006, Lecture Notes in Computer Science 4307, pages 479-498. Springer, 2006.

Contents

Abstract	v
Acknowledgement	vii
Publications	viii
1 Introduction	1
1.1 Background	1
1.1.1 Server-aided Verification Signatures	2
1.1.2 Certificateless PKC and Certificate-based PKC	3
1.1.3 Optimistic Fair Exchange	4
1.2 Aims and Objectives	6
1.3 Structure of This Thesis and Contributions	7
2 Preliminaries	9
2.1 Complexity Assumptions on Bilinear Group	9
2.2 Digital Signatures	11
2.3 Public Key Encryption	13
2.4 Proofs of Knowledge	14
3 Digital Signatures with Server-aided Verification	16
3.1 Introduction	16
3.2 Server-Aided Verification Signatures	19
3.3 Existentially Unforgeable SAV- Σ	20
3.3.1 Definition of Existential Unforgeability of SAV- Σ	21
3.3.2 Further Observations on EUF-SAV- Σ	22
3.3.3 Analysis of the SAV- Σ in Asiacrypt'05	23
3.4 Existentially Unforgeable SAV-BLS and SAV-Waters	26

3.4.1	Existentially Unforgeable SAV-BLS	26
3.4.2	Existentially Unforgeable SAV-Waters	32
3.5	SAV- Σ Secure Against Collusion and Adaptive Chosen Message Attacks	39
3.5.1	Security of SAV- Σ Against Collusion and Adaptive Chosen Message Attacks	39
3.5.2	SAV-BLS Secure Against Collusion and Adaptive Chosen Mes- sage Attacks	40
3.5.3	SAV-Waters Secure Against Collusion and Adaptive Chosen Message Attacks	43
3.6	SAV- Σ Secure Against Strong Collusion and Adaptive Chosen Mes- sage Attacks	45
3.6.1	SAV-BLS Secure Against Strong Collusion and Adaptive Cho- sen Message Attacks	46
3.7	Conclusion	48
4	Generic Constructions of Signatures and Encryption in Certificate- based PKC	50
4.1	Introduction	50
4.1.1	Related Work	53
4.1.2	Motivations and Contributions	54
4.2	Certificate-based Signatures	56
4.2.1	Syntax of Certificate-Based Signatures	57
4.3	Adversaries and Oracles	58
4.4	Security Models of Certificate-based Signatures	60
4.4.1	Security Against Normal Type I Adversary	61
4.4.2	Security Against Strong Type I Adversary	62
4.4.3	Security Against Super Type I Adversary	62
4.4.4	Security Against Type II Adversary	63
4.4.5	Security Against Malicious-but-Passive Type II Adversary	64
4.5	Generic Construction of Certificate-based Signatures	65
4.5.1	Syntax of Certificateless Signatures	65
4.5.2	Generic Construction: CLS-2-CBS	66
4.6	Concrete Examples of CLS-2-CBS	80
4.6.1	Scheme I	80
4.6.2	Scheme II	81

4.6.3	Efficiency Comparison	82
4.7	Certificate-based Encryption	83
4.7.1	Syntax of Certificate-Based Encryption	83
4.7.2	Adversaries and Oracles	84
4.8	Security Models of Certificate-based Encryption	86
4.8.1	Security against Normal Type I Adversary	86
4.8.2	Security against Strong Type I Adversary	88
4.8.3	Security against Super Type I Adversary	88
4.8.4	Security Against Type II Adversary	89
4.8.5	Security Against Malicious—but—Passive Type II Adversary	90
4.9	Generic Construction of Certificate-based Encryption	92
4.9.1	Syntax of Certificateless Encryption	92
4.9.2	Generic Construction: CLE-2-CBE	93
4.10	A Concrete Example of CLE-2-CBE	105
4.10.1	A Concrete Scheme	105
4.11	Conclusion	107
5	Optimistic Fair Exchange with Strong Resolution-Ambiguity	108
5.1	Introduction	108
5.1.1	Previous Work	109
5.1.2	Motivation	111
5.1.3	Our Contributions	112
5.2	Definitions of OFE in the Multi-user Setting	113
5.2.1	Syntax of OFE	113
5.2.2	Security against Signer(s)	115
5.2.3	Security against Verifier(s)	116
5.2.4	Security against the Arbitrator	117
5.3	Strong Resolution-Ambiguity	119
5.3.1	Definition of Strong Resolution-Ambiguity	120
5.3.2	Strong Resolution-Ambiguity in Concrete OFE Protocols	121
5.3.3	Security of OFE with Strong Resolution-Ambiguity	123
5.4	New OFE Protocol with Strong Resolution-Ambiguity	127
5.4.1	The Proposed Protocol	127
5.4.2	Scheme Analysis	128
5.4.3	Comparison to Previous Protocols	132

5.5 Conclusion	133
6 Conclusions	135
Bibliography	137

List of Tables

4.1	Comparison of CL Cryptography and CB Cryptography	53
4.2	Sign Oracles	60
4.3	Efficiency Comparison	82
4.4	Security Level Comparison	82
5.1	Multi-user Secure Stand-Alone and Setup-Free OFE Protocols with- out Random Oracles	133

List of Figures

3.1	SA-Verify in SAV-ZSS [GL05]	24
3.2	SA-Verify in SAV-BLS with EUF	27
3.3	The SAV Protocol For Waters Signature	33
3.4	SA-Verify in SAV-BLS with Soundness I	41
3.5	SA-Verify in SAV-Waters with Soundness	43
3.6	SA-Verify in SAV-BLS with Soundness II	47

Chapter 1

Introduction

Third parties, like judges in the court and referees in sports games, play an important role in our daily life. In public key cryptography, third parties are important for protecting information and communications in digital world and essential in many public key cryptographic systems. A good example is identity-based public key cryptography (or, ID-PKC for short). The concept of ID-PKC was introduced by Shamir in [Sha84], where the user's public key is his/her identity information (e.g., email address) and the associated private key is generated by a third party called Private Key Generator (PKG). Since PKG has all users' private keys, it is able to carry out all cryptographic operations (e.g., decryption and signing) on behalf of any user. Thus, a necessary assumption in ID-PKC is that the third party must be fully trusted. This assumption, from a practical point of view, does not easily hold in applications with a large number of users: It is hard to find a third party who is fully trusted by all users. For this reason, some public key cryptographic protocols make use of the third party in a different way, namely only assuming a certain level of trust (rather than the full trust) on the third party. This is a weaker and more practical assumption compared with the assumption of a fully trusted third party. As shown in the literature, similar approaches have led to the introduction of several new cryptographic primitives, such as server-aided verification signatures, certificateless public key cryptography, certificate-based public key cryptography and optimistic fair exchange. These are also the focus of this thesis.

1.1 Background

Cryptographic protocols with untrusted third parties have been investigated intensively in the literature. In this section, we describe the notions on which our thesis is

focused, namely server-aided verification signatures, certificateless public key cryptography, certificate-based public key cryptography and optimistic fair exchange.

1.1.1 Server-aided Verification Signatures

In public key cryptography [DH76], a user is equipped with a pair of cryptographic keys: a public key and a private key. The private key is kept secret, but the public key can be widely distributed and published. One of the most important applications of public key cryptography is digital signatures. Digital signatures are equivalent to traditional handwritten signatures in many aspects, such as integrity check, non-repudiation evidence, and authenticity purpose, etc. In particular, a digital signature generated by a user on a message guarantees that it was indeed the user who signed the message and the message cannot be altered. A digital signature scheme typically consists of three algorithms: a *Key-Generation* algorithm that outputs a private key and the associated public key; a *Signature-Generation* algorithm that, taking as input a message and a private key, produces a signature on that message; and a *Signature-Verification* algorithm that verifies the validity of a message-signature pair under a given public key.

As an advantage over a symmetric key cryptography, communication using public key cryptography does not need any secure channels for key sharing. However, public key cryptography has introduced more computational overheads to computer systems. This is a burden to computer systems, especially for low-power devices such as smart cards and mobile terminals. Several techniques (e.g., pre-computation and off-line computation) have therefore been introduced and adopted to improve the efficiency of those protocols. While such techniques can reduce the computational load, the computational requirement of many public key cryptographic algorithms and protocols, especially those with excellent security features, still remains too heavy for power-constrained devices.

A promising solution is to employ a powerful server to assist power-constrained devices to carry out cryptographic computations. This is known as “server-aided computation”. If the server can be fully-trusted, computations can be easily conducted between client and server. As an example, the client can send his/her private key to the server, who then acts on behalf of the client to decrypt ciphertexts or sign messages, and returns the result to the client. However, the assumption of a fully-trusted server seems too strong to be practical, since most likely clients will

face an untrusted server which may be interested in extracting the secret of the client (in decryption or signing) or responding with a false result (in encryption or signature verification). In the scenario of digital signatures, the motivation of server-aided verification is to use a powerful server to alleviate the computation load on the verifier who does not fully trust the server. The notion of server-aided verification was introduced by Quisquater and De Soete [QS89] in order to speed up RSA verification with a small exponent, while Giraul and Lefrance [GL05] made the first attempt to define the security of server-aided verification signatures. This thesis will investigate the formal security definitions and concrete protocols of digital signatures with server-aided verification.

1.1.2 Certificateless PKC and Certificate-based PKC

The central problem in a public key system is to prove that a public key is genuine and authentic, and has not been tampered with or replaced by a malicious third party. The usual approach to ensure the authenticity of a public key is to use a certificate. A (digital) certificate is a signature of a certificate authority (CA) that binds together the identity of a user, its public key and other information. This kind of systems is referred to as traditional public key infrastructure (PKI), which is generally considered to be costly to use and manage.

The original motivation of identity-based public key cryptography (ID-PKC) introduced by Shamir [Sha84] is to ease certificate management in the e-mail system. A user's public key in ID-PKC is some unique information about the identity of the user (e.g., email address). The private key in ID-PKC is generated by a trusted third party called Private Key Generator (PKG) who holds a master key. Although certificate is eliminated, ID-PKC within the structure of [Sha84] has an inherent problem, i.e., key escrow, since PKG has any user's private key and must be fully trusted by all users. The key escrow problem can be partially solved by the introduction of multiple PKGs and the use of threshold techniques, but this is at the cost of extra communications and infrastructures.

Al-Riyami and Paterson proposed a new paradigm called certificateless public key cryptography [ARP03] (or, CL-PKC for short). The original motivation of CL-PKC is to find a public key system that does not require the use of certificates and does not have the key escrow problem. Each user in CL-PKC holds two secrets: a secret value and a partial private key. The secret value SV is generated by the user,

and a third party Key Generating Center (KGC), holding a master key, generates the partial private key PPK from the user's identity information. The user's actual private key is the output of some function with the input SV and PPK . In this way, the key escrow problem is eliminated and KGC does not need to be fully trusted. The user can use the actual private key to generate the public key, which can be available to other entities by transmitting it along with messages or by placing it in a public directory. However, there is no public key certificate in CL-PKC.

In Eurocrypt 2003, Gentry [Gen03] introduced the notion of certificate-based encryption. As in the traditional PKI, each client generates its own public/private key pair and requests a certificate from the CA. The difference is that, a certificate in certificate-based encryption, or more generally, a signature from the third party acts not only as a certificate (as in the traditional PKI) but also as a decryption key (as in ID-PKC and CL-PKC). The sender can encrypt a message without obtaining explicit information other than the recipient's public key and the parameters of CA. To decrypt a message, a keyholder needs both its secret key and an up-to-date certificate from its CA (or a signature from an authority). Therefore, CA does not need to make the certificate status information available among the whole system, and only needs to contact the certificate holder for revocation and update. A similar idea was later used to design other cryptographic protocols within the same infrastructure of [Gen03]. In this thesis, we will investigate two fundamental notions-encryption and signatures-in certificate-based public key cryptography, and demonstrate the relationship with their counterparts in certificateless public key cryptography.

1.1.3 Optimistic Fair Exchange

The objective of a fair exchange protocol is to allow two parties to fairly exchange their items so that no one can gain any advantage. A straightforward solution is to introduce a fully trusted third party as a mediator: each party sends the item to the trusted third party, who upon verifying the correctness of both items, forwards each item to the other party. Such protocols require that the third party be always online, as each exchange needs the assistance from the third party. In order to alleviate the requirement of an always-online third party, Asokan, Schunter and Waidner [ASW97] introduced the notion of *Optimistic Fair Exchange* (hereinafter referred to as "OFE"). OFE also makes use of a third party called *arbitrator*, but

it does not need to be always online; instead, the arbitrator only gets involved if something goes wrong (e.g., one party attempts to cheat or other faults occur).

A typical example of OFE is as follows: Two participants (say, Alice and Bob) first agree on the items to be exchanged: Alice's item is a valid *full signature* (e.g., signature on a credit card purchase) and Bob's item is denoted by $item_B$ (e.g., a book). At the end of the exchange, Alice should have $item_B$ and Bob should have the *full signature*.

Step 1. Alice starts the exchange by generating a *partial signature* and sending it to Bob. The partial signature only shows Alice's willingness to send her full signature to Bob if Bob sends her $item_B$.

Step 2. Bob verifies the *partial signature* and sends $item_B$ to Alice if the *partial signature* is valid.

Step 3. Upon receiving $item_B$, Alice generates a *full signature* and sends it to Bob.

Step 4. If Bob does not receive the *full signature*, he can obtain it from the arbitrator who is able to convert the *partial signature* to the *full signature*. This property guarantees that Bob will obtain the *full signature* if Alice refuses to send it after Step 2.

In the above scenario, Alice is usually called as “signer” and Bob is called as “verifier”. The exchange between the signer and the verifier has attracted many attentions from researchers on OFE, and it is also the case which OFE refers to in the remainder of this thesis. (But readers should bear in mind that OFE is a broad notion and includes many other types of exchanges.) It has been widely accepted that an OFE protocol must satisfy the following security requirements:

Security against Signers: The signer should not be able to produce a valid partial signature which cannot be transformed into a valid full signature by an honest arbitrator.

Security against Verifiers: The verifier should not be able to transform a valid partial signature into a valid full signature, without explicitly asking the arbitrator to do so.

Security against the Arbitrator: The arbitrator should not be able to produce a valid full signature on message m without explicitly asking the signer to produce a partial signature on m .

1.2 Aims and Objectives

This thesis is focused on the design of aforementioned cryptographic protocols with untrusted third parties. The aim of this thesis is to address three aspects defined as follows.

1. Several server-aided verification protocols have been proposed but the notion has never been formally investigated. Giraul and Lefrance [GL05] had made the first attempt to define the security of server-aided verification in digital signatures. Their definition however is only heuristic, and it is still worthwhile to provide more elaborated models for the further research on server-aided verification signatures. The first aim of this thesis is to formally define server-aided verification in digital signatures based on different trust levels assumed on the third party, and design concrete server-aided verification protocols in different settings.
2. As mentioned in [ARP03], certificate-based public cryptography and certificateless public cryptography share some commonalities. In both systems, message signing (resp. decryption) requires two pieces of information: one is generated by the third party and the other one is generated by the public key owner. In addition, explicit verification of the authenticity of a public key is not required in signature verification (resp. encryption). Motivated by those similarities, we believe it is worthwhile to investigate the relationship of digital signatures (resp. encryption) between certificateless public cryptography and certificate-based public key cryptography. This is the second aim of our thesis.
3. Most OFE protocols are designed only in the single-user setting, namely there is only one signer. However, OFE protocols are most likely used in the multi-user setting where there are two or more signers in the system (but items are still exchanged between two parties). In [DLY07], Dodis, Lee and Yum pointed out that the single-user security of OFE cannot guarantee the multi-user security. A similar result was also discovered independently by Zhu, Susilo and Mu [ZSM07]. On the other hand, it is known that several single-user secure OFE protocols can be proven secure in the multi-user setting [DLY07]. However, it remains unknown *under which conditions single-user secure OFE protocols will be secure in the multi-user setting?* We believe the investigation of this question not only will provide a further understanding on the security

of OFE in the multi-user setting, but also can introduce new constructions of multi-user secure optimistic fair exchange.

1.3 Structure of This Thesis and Contributions

Chapter 1 briefly describes the background of this thesis and gives a short introduction of each chapter and its contribution. The basic cryptographic notions, mathematical tools and complexity assumptions required in this thesis are reviewed in Chapter 2.

Chapter 3 is focused on digital signatures with server-aided verification (SAV). We first introduce and define the existential unforgeability of server-aided verification signatures. We prove that the new notion includes the existential unforgeability of digital signature schemes and the soundness of server-aided verification protocols under the same assumption in [GL05]. We then introduce the server-aided verification to BLS signature [BLS01] and Waters signature [Wat05], by providing the first constructions of existentially unforgeable SAV-BLS and SAV-Waters. Additionally, we investigate the collusion between the signer and the server, and define the security models to capture the collusion attack and its stronger version on server-aided verification signatures. Concrete server-aided verification signature schemes secure against collusion attacks are also proposed. The result described in Chapter 3 was presented at Provable Security 2008, Second International Conference [WMSH08].

Chapter 4 presents several new results of digital signatures and encryption in certificate-based public key cryptography. We first provide new definitions of the potential adversaries in certificate-based public key cryptography, which we believe will establish a systematic approach for constructing and proving secure cryptographic protocols in certificate-based public key cryptography. After that, we show how to, in a generic way, build a certificate-based signature scheme from a certificateless signature scheme and a certificate-based encryption scheme from a certificateless encryption scheme, respectively. The security of our generic constructions is proven in the random oracle model. Concrete protocols are also provided to demonstrate the application of our generic constructions. Part of this chapter was published in the Journal of Universal Computer Science [WMSH09].

Chapter 5 gives a sufficient condition for single-user secure OFE protocols remaining secure in the multi-user setting. We first introduce and define a new property of OFE called Strong Resolution-Ambiguity. This property requires that one

can transform a partial signature σ' into a full signature σ using signer's private key or arbitrator's private key, and given such a pair (σ', σ) , it is infeasible to tell which key is used in the conversion. For an OFE protocol with strong resolution-ambiguity, we prove that its security in the single-user setting is preserved in the multi-user setting. More precisely, we show that: (1) the security against the signer and the security against the verifier in the single-user setting are preserved in the multi-user setting for OFE protocols with strong resolution-ambiguity, and (2) the security against the arbitrator in the single-user setting is preserved in the multi-user setting (for OFE protocols either with or without strong resolution-ambiguity). Additionally, we provide a new construction of OFE with strong resolution-ambiguity, which possesses several desirable properties and is multi-user secure without the random oracle assumption. The result of this chapter was presented at PKC 2010 [HMS⁺10].

Chapter 6 concludes this thesis.

Chapter 2

Preliminaries

In this chapter, we review some fundamental backgrounds required in the thesis.

2.1 Complexity Assumptions on Bilinear Group

Bilinear Map. Let \mathbb{G}_1 and \mathbb{G}_T be two groups of prime order p and let g be a generator of \mathbb{G}_1 . The map $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$ is said to be an admissible bilinear map if the following three conditions hold true:

- e is bilinear, i.e., $e(g^a, g^b) = e(g, g)^{ab}$ for all $a, b \in \mathbb{Z}_p$.
- e is non-degenerate, i.e., $e(g, g) \neq 1_{\mathbb{G}_T}$.
- e is efficiently computable.

Definition 2.1 *We say that $(\mathbb{G}_1, \mathbb{G}_T)$ are bilinear groups if there exists a non-degenerate bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$, such that the group order $p = |\mathbb{G}_1| = |\mathbb{G}_T|$ is prime, and the pairing e and the group operations in \mathbb{G}_1 and \mathbb{G}_T are all efficiently computable.*

Bilinear maps can be constructed from modified Weil pairing [Lan73] and Tate pairing [FMR99] on suitable elliptic curves. Weil and Tate pairings were first used to break cryptographic algorithms (e.g., MOV reduction [MOV93]) and later on used to design cryptographic protocols, such as one-round 3-party Diffie-Hellman [Jou04], practical identity-based encryption [BF03], etc.

Remark 2.1 *A more general form of bilinear map is*

$$e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T,$$

where $\mathbb{G}_1, \mathbb{G}_2$ and \mathbb{G}_T are groups of the same prime order p . There are three basic types:

Type 1: $\mathbb{G}_1 = \mathbb{G}_2$;

Type 2: $\mathbb{G}_1 \neq \mathbb{G}_2$ but there is an efficiently computable homomorphism $\phi : \mathbb{G}_1 \rightarrow \mathbb{G}_2$; and

Type 3: $\mathbb{G}_1 \neq \mathbb{G}_2$ and there are no efficiently computable homomorphisms between \mathbb{G}_1 and \mathbb{G}_2 .

The Type I bilinear map provides advantages in protocol design, but Type III has the most flexible options (in the sense of suitable elliptic curves satisfying various security levels) [GPS08]. As one can see, the bilinear map we will use in this thesis is Type I.

Discrete Logarithm (DL) on \mathbb{G}_1 . Given $g, g^a \in \mathbb{G}_1$, compute $a \in \mathbb{Z}_p$.

An algorithm \mathcal{B} solves the DL problem on \mathbb{G}_1 with advantage ε if $\text{Adv DL}_{\mathcal{B}} = \Pr[\mathcal{B}(\mathbb{G}_1, \mathbb{G}_T, e, g, g^a) = a : a \in_R \mathbb{Z}_p] \geq \varepsilon$. The probability is over the uniform random choice of a from \mathbb{Z}_p , and over the coin tosses of \mathcal{B} . We say that an algorithm \mathcal{B} (t, ε) -solves DL on \mathbb{G}_1 if \mathcal{B} runs in time at most t , and $\text{Adv DL}_{\mathcal{B}}$ is at least ε .

Definition 2.2 *The Discrete Logarithm problem on \mathbb{G}_1 is said to be (t, ε) -hard if there exists no algorithm that (t, ε) -solves it.*

Computational Diffie-Hellman (CDH) on \mathbb{G}_1 [BLS01]. Given $g, g^a, g^b \in \mathbb{G}_1$, compute $g^{ab} \in \mathbb{G}_1$.

An algorithm \mathcal{B} solves the CDH problem on \mathbb{G}_1 with advantage ε if $\text{Adv CDH}_{\mathcal{B}} = \Pr[\mathcal{B}(\mathbb{G}_1, \mathbb{G}_T, e, g, g^a, g^b) = g^{ab} : a, b \in_R \mathbb{Z}_p] \geq \varepsilon$. The probability is over the uniform random choice of a, b from \mathbb{Z}_p , and over the coin tosses of \mathcal{B} . We say that an algorithm \mathcal{B} (t, ε) -solves CDH on \mathbb{G}_1 if \mathcal{B} runs in time at most t , and $\text{Adv CDH}_{\mathcal{B}}$ is at least ε .

Definition 2.3 *The Computational Diffie-Hellman problem on \mathbb{G}_1 is said to be (t, ε) -hard if there exists no algorithm that (t, ε) -solves it.*

Bilinear Diffie-Hellman Problem (BDH) on $(\mathbb{G}_1, \mathbb{G}_T)$ [BF01]. Given $(g, g^a, g^b, g^c) \in \mathbb{G}_1$, compute $e(g, g)^{abc} \in \mathbb{G}_T$.

An algorithm \mathcal{B} solves the BDH problem on $(\mathbb{G}_1, \mathbb{G}_T)$ with advantage ε if

$$\text{Adv BDH}_{\mathcal{B}} = \Pr[\mathcal{B}(g, g^a, g^b, g^c) = e(g, g)^{abc}] \geq \varepsilon.$$

The probability is over the uniform random choice of $\mathbf{a}, \mathbf{b}, \mathbf{c}$ from \mathbb{Z}_p , and over the coin tosses of \mathcal{B} . We say that an algorithm \mathcal{B} (t, ε) -solves BDH on $(\mathbb{G}_1, \mathbb{G}_T)$ if \mathcal{B} runs in time at most t , and $\text{Adv BDH}_{\mathcal{B}}$ is at least ε .

Definition 2.4 *The Bilinear Diffie-Hellman problem on $(\mathbb{G}_1, \mathbb{G}_T)$ is said to be (t, ε) -hard if there exists no algorithm that (t, ε) -solves it.*

Decisional Bilinear Diffie-Hellman Oracle $\mathcal{O}_{\text{DBDH}}$ on $(\mathbb{G}_1, \mathbb{G}_T)$ [MW96]. Given $(g, g^{\mathbf{a}}, g^{\mathbf{b}}, g^{\mathbf{c}}) \in \mathbb{G}_1$ and $\mathbf{h} \in \mathbb{G}_T$, this oracle outputs “1” if $\mathbf{h} = e(g, g)^{\mathbf{a}\mathbf{b}\mathbf{c}}$ or “0” otherwise.

Gap Bilinear Diffie-Hellman Problem (GBDH) on $(\mathbb{G}_1, \mathbb{G}_T)$. Given $(g, g^{\mathbf{a}}, g^{\mathbf{b}}, g^{\mathbf{c}}) \in \mathbb{G}_1$, compute $e(g, g)^{\mathbf{a}\mathbf{b}\mathbf{c}}$ with the help of Decisional Bilinear Diffie-Hellman Oracle $\mathcal{O}_{\text{DBDH}}$.

An algorithm \mathcal{B} solves the GBDH on $(\mathbb{G}_1, \mathbb{G}_T)$ with advantage ε if

$$\text{Adv GBDH}_{\mathcal{B}} = \Pr[\mathcal{B}(g, g^{\mathbf{a}}, g^{\mathbf{b}}, g^{\mathbf{c}}, \mathcal{O}_{\text{DBDH}}) = e(g, g)^{\mathbf{a}\mathbf{b}\mathbf{c}} : \mathbf{a}, \mathbf{b}, \mathbf{c} \in_R \mathbb{Z}_p] \geq \varepsilon.$$

The probability is over the uniform random choice of $\mathbf{a}, \mathbf{b}, \mathbf{c}$ from \mathbb{Z}_p , and over the coin tosses of \mathcal{B} . We say that an algorithm \mathcal{B} (t, ε) -solves GBDH on $(\mathbb{G}_1, \mathbb{G}_T)$ if \mathcal{B} runs in time at most t , and $\text{Adv GBDH}_{\mathcal{B}}$ is at least ε .

Definition 2.5 *The Gap Bilinear Diffie-Hellman (GBDH) problem on $(\mathbb{G}_1, \mathbb{G}_T)$ is said to be (t, ε) -hard if there exists no algorithm that (t, ε) -solves it.*

2.2 Digital Signatures

The notion of digital signatures was first proposed by Diffie and Hellman [DH76] in 1976. A digital signature scheme Σ considered in this thesis consists of the following algorithms:

Parameter-Generation: $\text{ParamGen}(1^k) \rightarrow \text{param}$.

This algorithm takes as input a security parameter k and returns a string param which denotes the common scheme parameters, including the description of the message space \mathbb{M} and the signature space Ω , etc. param is shared among all the users in the system.

Key-Generation: $\mathbf{KeyGen}(param) \rightarrow (sk, pk)$.

This algorithm takes as input the system parameter $param$ and returns a private/public key-pair (sk, pk) for a user in the system.

Signature-Generation: $\mathbf{Sign}(param, m, sk, pk) = \sigma$.

This algorithm takes as input the system parameter $param$, the message m and the key pair (sk, pk) , and returns a signature σ .

Signature-Verification: $\mathbf{Verify}(param, m, \sigma, pk) \rightarrow \{\mathbf{Valid}, \mathbf{Invalid}\}$.

This algorithm takes as input the system parameter $param$, a pair (m, σ) and the public key pk , and returns **Valid** or **Invalid**. σ is said to be a valid signature of m under pk if $\mathbf{Verify}(param, m, \sigma, pk)$ outputs **Valid**. Otherwise, σ is said to be invalid.

Completeness. Any signature properly generated by **Sign** can always pass through the verification in **Verify**. That is, $\mathbf{Verify}(param, m, \mathbf{Sign}(param, m, sk, pk), pk) = \mathbf{Valid}$.

Existential unforgeability of Σ . The standard notion of security for a signature scheme is called existential unforgeability under adaptive chosen message attacks [GMR88], which is defined using the following game between a challenger \mathcal{C} and an adversary \mathcal{A} .

Setup. The challenger \mathcal{C} runs the algorithm **ParamGen** and **KeyGen** to obtain system parameter $param$ and one key pair (sk, pk) . The adversary \mathcal{A} is given $param$ and pk .

Queries. Proceeding adaptively, the adversary \mathcal{A} can request signatures of at most q_s messages. For each sign query $m_i \in \{m_1, \dots, m_{q_s}\}$, the challenger \mathcal{C} returns $\sigma_i = \mathbf{Sign}(param, m_i, sk, pk)$ as the response.

Output. Eventually, the adversary \mathcal{A} outputs a pair (m^*, σ^*) and wins the game if:

1. $m^* \notin \{m_1, \dots, m_{q_s}\}$; and
2. $\mathbf{Verify}(param, m^*, \sigma^*, pk) = \mathbf{Valid}$.

We define $\Sigma\text{-Adv}_{\mathcal{A}}$ to be the probability that the adversary \mathcal{A} wins in the above game, taken over the coin tosses made by \mathcal{A} and the challenger.

Definition 2.6 A forger \mathcal{A} is said to (t, q_s, ε) -break a signature scheme Σ if \mathcal{A} runs in time at most t , \mathcal{A} makes at most q_s signature queries, and $\Sigma\text{-Adv}_{\mathcal{A}}$ is at least ε . Σ is (t, q_s, ε) -existentially unforgeable against adaptive chosen message attacks if there exists no forger that (t, q_s, ε) -breaks it.

2.3 Public Key Encryption

A public key encryption scheme, **first introduced in [DH76]**, is made up of the following algorithms:

Parameter-Generation: $\mathbf{ParamGen}(1^k) \rightarrow param$.

This algorithm takes as input a security parameter k and returns a string $param$ which denotes the common scheme parameters, including the description of the message space \mathbb{M} and the ciphertext space \mathbb{C} , etc. $param$ is shared among all the users in the system.

Key-Generation: $\mathbf{KeyGen}(param) \rightarrow (sk, pk)$.

This algorithm takes as input the system parameter $param$ and returns a private/public key-pair (sk, pk) for a user in the system.

Encrypt: $\mathbf{Encrypt}(param, pk, m) \rightarrow cipher$.

This algorithm takes as input the system parameter $param$, a public key pk and a message $m \in \mathbb{M}$, and returns a ciphertext $cipher$.

Decrypt: $\mathbf{Decrypt}(param, sk, cipher) \rightarrow \{m, \perp\}$.

This algorithm takes as input the system parameter $param$, a private key sk and a ciphertext $cipher$, and returns a message m or the symbol “ \perp ” indicating a decryption failure.

Correctness. Ciphertexts generated by the algorithm $\mathbf{Encrypt}$ can be successfully decrypted by $\mathbf{Decrypt}$: $\mathbf{Decrypt}(param, sk, \mathbf{Encrypt}(param, pk, m)) = m$.

IND-CCA. The commonly accepted security notion for a public key encryption scheme is indistinguishability against adaptive chosen ciphertext attacks (known as IND-CCA), which is defined using the following game between a challenger \mathcal{C} and an adversary \mathcal{A} .

Setup. The challenger \mathcal{C} runs the algorithm **ParamGen** and **KeyGen** to obtain system parameter $param$ and one key pair (sk, pk) . The adversary \mathcal{A} is given $param$ and pk .

Queries-I. Proceeding adaptively, the adversary \mathcal{A} can make decryption queries. For each query $cipher_i$, the challenger \mathcal{C} responds with the output of **Decrypt** $(param, sk, cipher_i)$.

Challenge. The adversary \mathcal{A} submits two (equal length) messages m_0, m_1 . The challenger selects a random bit $b \in \{0, 1\}$, sets $C^* = \mathbf{Encrypt}(param, pk, m_b)$ and sends C^* to the adversary as its challenge ciphertext.

Queries-II. The adversary continues to adaptively issue decryption queries as in Queries-I, but with the constraint that the adversary does not request the decryption of C^* .

Output: After all queries, the adversary outputs a guess $b' \in \{0, 1\}$ and wins the game if $b = b'$.

The success probability that an adaptive chosen ciphertext adversary has in the above game is $\Pr[b = b']$.

Definition 2.7 *A public key encryption scheme is (t, q, ϵ) -indistinguishable against adaptive chosen ciphertext attacks if no adversary, making at most q decryption queries in time t , can have success probability at least $\epsilon + 1/2$ in the above game.*

2.4 Proofs of Knowledge

An NP-relation R is a subset of $\{0, 1\}^* \times \{0, 1\}^*$ for which there exists a polynomial f such that $|y| \leq f(|x|)$ for all $(x, y) \in R$, and there exists a polynomial-time algorithm for deciding membership in R . Let $R(x) = \{y : (x, y) \in R\}$ and $L_R = \{x : \exists y \text{ such that } (x, y) \in R\}$. A proof of knowledge [BG92] for an NP-relation R is a protocol whereby there is a common input x to the prover and the verifier, and a private input y to the prover. The prover tries to convince the verifier that $(x, y) \in R$.

Definition 2.8 (Systems of proofs of knowledge) [BG92] *Let R be a binary*

relation, and $\kappa : \{0, 1\}^* \rightarrow [0, 1]$. Let V be an interactive function which is computable in probabilistic polynomial time. We say that a V is a knowledge verifier for the relation R with knowledge error κ if the following two conditions hold.

- *Non-triviality:* There exists an interactive function P^* so that for all $x \in L_R$, all possible interactions of V with P^* on common input x are accepting.
- *Validity (with error κ):* There exists a constant $c > 0$ and a probabilistic oracle machine K such that for every interaction function P and every $x \in L_R$, machine K satisfies the following condition: let $p(x) > \kappa(x)$ be the probability that the interaction of V with P on common input x are accepting, then on input x and access to oracle P_x , machine K outputs a string from the set $R(x)$ within an expected number of steps bounded by $\frac{|x|^c}{p(x) - \kappa(x)}$. The oracle machine K is called a universal knowledge extractor, and κ is called the knowledge error function.

Chapter 3

Digital Signatures with Server-aided Verification

This chapter investigates digital signatures with server-aided verification, namely a third party called server is employed to assist signature verification. According to different trust levels assumed on the server, we provide formal definitions and concrete constructions of server-aided verification signatures. Part of this work will be published in Journal of Computers and Mathematics with Applications [WMSH11].

3.1 Introduction

Cryptographic protocols introduce extra computational costs to computer systems. Although it is not a significant burden to normal computer systems, low power devices such as smart cards and mobile terminals require an additional care when using cryptographic protocols. Several techniques, e.g., pre-computation and off-line computation, have been introduced and adopted in order to improve the efficiency of cryptographic protocols. While such techniques can reduce the computational load, the computational requirement of many cryptographic systems (especially those with excellent security features) still remains too heavy for low-power devices. Pairing computation on elliptic curves is an example. Due to its elegant properties, pairing has been widely employed as building blocks for lots of cryptographic schemes, in particular in the design of identity-based encryption and short signatures. However, a pairing operation on elliptic curve requires much more computational cost than carrying out a modular exponentiation. How to reduce the computational cost of pairing-based cryptography is a challenging task.

A promising solution is to employ a powerful server assisting the low-power device (we refer it to as *client*) to carry out costly cryptographic operations. This is known as “server-aided computation”. If the server is fully-trusted, computations

can be easily done through a secure channel between the client and the server. As an example, the client can send his/her private key to the server, who then can act on behalf of the client to decrypt ciphertexts or sign messages, and return the result to the client. However, the assumption of a fully-trusted server seems too strong to be practical, since more likely clients will face an untrusted server which could try to extract the secret of the client (in decryption or signing) or respond with a false result (in encryption or signature verification).

Many server-aided computation schemes [And92, BM94, BQ95, GP03, GRK00, iKS93, MILY92, MKI88, NS98, PW92, YL92, YL93] have been proposed in the literature. The first server-aided computation scheme was proposed by Matsumoto, Kato and Imai [MKI88] in the scenario of RSA signature. However, Phitzmann and Waidner [PW92] showed that all protocols in [MKI88] are insecure due to several passive attacks. Béguin and Quisquater's protocol [BQ95] was based on the fast exponentiation algorithm [BGMW92], which does not require expensive pre-computations. Unfortunately, an effective lattice-based passive attack against their protocol was proposed in [NS98]. The focus of this chapter is on the server-aided verification signature $\text{SAV-}\Sigma$. In general, $\text{SAV-}\Sigma$ consists of a digital signature scheme and a server-aided verification protocol. The purpose of $\text{SAV-}\Sigma$ is to enable signature verifier to perform signature verification with less computational cost, by executing the server-aided verification protocol with the server. The notion of server-aided verification was introduced by Quisquater and De Soete [QS89] in order to speed up RSA verification with a small exponent. In Eurocrypt 1995, Lim and Lee proposed efficient protocols for speeding up the verification of identity proofs and signatures in discrete-logarithm-based identification schemes, based on the "randomization" of the verification equation [LL95]. Girault and Quisquater [GQ02] proposed another different approach, which does not require pre-computation or randomization. Their server-aided verification protocol is computationally secure based on the hardness of a sub-problem of the initial underlying problem of the signature scheme. Hohenberger and Lysyanskaya considered the sever-aided verification under the situation that the server is made of two untrusted software packages, which are assumed not to communicate with each other [LL95]. Under this assumption, it allows a very light public computation task (typically one modular multiplication in the Schnorr scheme). Girault and Lefranc [GL05] proposed a more generalized model of server-aided verification without the assumption in [LL95]. A generic server-aided verification protocol for digital signatures from bilinear maps was also proposed in [GL05].

Their protocol can be applied to signature schemes with similar constructions to the BB signature [BB04] and the ZSS signature [ZSNS04].

The purposes of this chapter are to formally define the security of server-aided verification signatures and construct new schemes that are secure under realistic security models. Giraul and Lefrance [GL05] had made the first attempt to define the security of server-aided verification signatures. Their definition consists of two aspects, namely the existential unforgeability of the signature scheme and the soundness of server-aided verification protocol. The former notion is the same as that in digital signatures, and the latter requires that the server be unable to prove an invalid signature as valid using the server-aided verification protocol. Although their definition captures the essence of server-aided verification signatures, the adaptive attacks on server-aided verification are not clearly described and it is still worthwhile to define more elaborated models for the further research on server-aided verification signatures. The main results of this chapter are summarized as follows.

First, we introduce and define the existential unforgeability of server-aided verification signatures (or, $\text{EUF-SAV-}\Sigma$ for short). For server-aided verification signatures, we prove that $\text{EUF-SAV-}\Sigma$ includes the existential unforgeability of signature schemes and the soundness of server-aided verification protocols under the same assumption in [GL05], i.e., the server does not have any valid signature of the message when it tries to prove a signature of that message as valid. An existentially unforgeable server-aided verification signature scheme ensures that even the server (without colluding the signer) is not able to forge a signature which can be proven to be valid using the server-aided verification protocol.

Second, we analyze the existential unforgeability of ZSS signature [ZSNS04] with the server-aided verification protocol proposed in [GL05]. The analysis shows that the server-aided verification ZSS in [GL05] can be made secure in our model, but requires more computational cost than that claimed in [GL05]. This is due to the difference between the security model defined in this chapter and that in [GL05]. In our model, before proving an invalid signature as valid, the server is allowed to execute the server-aided verification protocol with the verifier before proving to the verifier that an invalid signature is valid. This, however, is not allowed in [GL05] when making the security analysis of server-aided verification ZSS. We believe that our model reflects a realistic case in real life.

Third, we introduce the server-aided verification to BLS signature [BLS01] and

Waters signature [Wat05], respectively. We provide the first constructions of existentially unforgeable SAV-BLS and SAV-Waters. The existential unforgeability of SAV-BLS can be reduced to the hardness of BDH problem in the random oracle model. SAV-Waters inherits the desirable property of Waters signature, which can be proven to be existentially unforgeable without random oracles under GBDH assumption.

Last, we investigate the collusion between the signer and the server in server-aided verification signatures, who collaboratively prove an invalid signature to be valid. Such attacks were first sketched in [GL05] in the definition of “auxiliary non-repudiation”. Previous definitions (including EUF-SAV- Σ) are all based on the assumption that the malicious server does not have any valid signature of the message when it tries to prove an invalid signature of that message to be valid. We formally define the security models to capture the collusion attack and its stronger version in server-aided verification signatures, and provide concrete server-aided verification signature schemes secure against collusion attacks.

Organization of the Chapter

The rest of this chapter is organized as follows. In Section 3.2, we define the notion of server-aided verification signature scheme (SAV- Σ). We then analyze the existential unforgeability of a previously proposed SAV- Σ in Section 3.3.3. New constructions of existentially unforgeable SAV- Σ and their security proofs are given in Section 3.4. The security of server-aided verification signatures under collusion attacks is investigated in Section 3.5 and Section 3.6. Finally, we conclude this chapter in Section 3.7.

3.2 Server-Aided Verification Signatures

In this section, we provide the definition of server-aided verification signatures.

A server-aided verification signature scheme SAV- Σ consists of six algorithms: **ParamGen**, **KeyGen**, **Sign**, **Verify**, **SA-Verifier-Setup**, and **SA-Verify**. The first four algorithms are the same as those in an ordinary signature scheme Σ defined in Section 2.2, and the last two are defined as follows:

Server-Aided-Verifier-Setup: **SA-Verifier-Setup**($param$) \rightarrow **VString**.

This algorithm takes as input the system parameter $param$ and returns the bit string **VString**, which contains the information that can be pre-computed

by the verifier. Note that `VString` might be the same as `param` if no pre-computation is required.

Server-Aided Verification:

$$\mathbf{SA-Verify}(\mathbf{Server}^{(param)}, \mathbf{Verifier}^{(m, \sigma, pk, VString)}) \rightarrow \{\mathbf{Valid}, \mathbf{Invalid}\}.$$

SA-Verify is an interactive protocol between **Server** and **Verifier**, who only has a limited computational ability and is not able to perform all computations in **Verify** alone. Given the message/signature pair (m, σ) , as well as the public key pk and the inner information `VString`, **Verifier** checks the validity of σ with the help of **Server** by running **SA-Verify**. **SA-Verify** returns **Valid** if **Server** can convince **Verifier** that σ is valid. Otherwise, σ is said to be invalid.

Completeness. There are two types of completeness in SAV- Σ :

1. **Completeness of Σ .** Any signature properly generated by **Sign** can always pass through the verification in **Verify**. That is,

$$\mathbf{Verify}(param, m, \mathbf{Sign}(param, m, sk, pk), pk) = \mathbf{Valid}.$$

2. **Completeness of SA-Verify.** An honest server can correctly convince the verifier about the validity of a signature. That is,

$$\begin{aligned} & \mathbf{SA-Verify}(\mathbf{Server}^{(param)}, \mathbf{Verifier}^{(m, \sigma, pk, VString)}) \\ &= \mathbf{Verify}(param, m, \sigma, pk). \end{aligned}$$

3.3 Existentially Unforgeable SAV- Σ

The security of SAV- Σ must include two security notions: existential unforgeability of Σ (EUF- Σ) and the soundness of **SA-Verify** (**Soundness-SA-Verify**). The former is the same as that in Definition 2.6, while the latter is a new notion and only appears in the scenario of SAV- Σ . As usual, the soundness notion requires that the server should not be able to use **SA-Verify** to convince the verifier that an invalid signature is valid. The formal definition of the soundness depends on the assumption about the server. We will give the first security model of SAV- Σ under the same assumption in [GL05]. We will define security models under different assumptions in Section 3.5 and Section 3.6.

3.3.1 Definition of Existential Unforgeability of SAV- Σ

Our first model follows the assumption in [GL05], namely, the server does not have the valid signature of the message when it tries to use **SA-Verify** to convince the verifier that an invalid signature of that message is valid. Under this assumption, it is not necessary to consider EUF- Σ and **Soundness-SA-Verify** separately. Instead, we will give a unified notion, called the *existential unforgeability of SAV- Σ* (or, EUF-SAV- Σ for short), which implies EUF- Σ and **Soundness-SA-Verify**.

Briefly speaking, EUF-SAV- Σ requires that the adversary should not be (computationally) capable of producing a signature of a new message which can be proven as **Valid** by **SA-Verify**, even the adversary acts as **Server**. A formal game-based definition is described as follows.

Setup. The challenger \mathcal{C} runs the algorithm **ParamGen**, **KeyGen** and **SA-Verifier-Setup** to obtain system parameter $param$, one key pair (sk, pk) and **VString**. The adversary \mathcal{A} is given $param$ and pk .

Queries. The adversary \mathcal{A} can make the following queries:

Signature Queries. Proceeding adaptively, the adversary \mathcal{A} can request signatures of at most q_s messages. For each sign query $m_i \in \{m_1, \dots, m_{q_s}\}$, the challenger \mathcal{C} returns $\sigma_i = \mathbf{Sign}(param, m_i, sk, pk)$ as response.

Server-Aided Verification Queries. Proceeding adaptively, the adversary \mathcal{A} can make at most q_v server-aided verification queries. For each query (m, σ) , the challenger \mathcal{C} responds by executing **SA-Verify** with the adversary \mathcal{A} , where the adversary \mathcal{A} acts as **Server** and the challenger \mathcal{C} acts as **Verifier**. At the end of each execution, the challenger returns the output of **SA-Verify** to the adversary \mathcal{A} .

Output. Eventually, the adversary \mathcal{A} outputs a pair (m^*, σ^*) and wins the game if:

1. $m^* \notin \{m_1, \dots, m_{q_s}\}$; and
2. $\mathbf{SA-Verify}(\mathcal{A}^{(param, \mathbf{InnerInfo})}, \mathcal{C}^{(m^*, \sigma^*, pk, \mathbf{VString})}) = \mathbf{Valid}$, where **InnerInfo** refers to the inner information of \mathcal{A} (e.g., the random element) in the generation of σ^* .

We define $\text{SAV-}\Sigma\text{-Adv}_{\mathcal{A}}$ to be the probability that the adversary \mathcal{A} wins in the above game, taken over the coin tosses made by \mathcal{A} and the challenger.

Definition 3.1 *A forger \mathcal{A} is said to $(t, q_s, q_v, \varepsilon)$ -break a SAV- Σ , if \mathcal{A} runs in time at most t , makes at most q_s signature queries, q_v server-aided verification queries, and $\text{SAV-}\Sigma\text{-Adv}_{\mathcal{A}}$ is at least ε . A SAV- Σ is $(t, q_s, q_v, \varepsilon)$ -existentially unforgeable under adaptive chosen message attacks if there exists no forger that $(t, q_s, q_v, \varepsilon)$ -breaks it.*

When discussing security in the random oracle model, we add a fifth parameter q_h to denote an upper bound on the number of queries that the adversary makes to the random oracle.

Remark 3.1 (EUF-SAV- Σ .) *We note that in **Setup**, **VString** is not provided to the adversary who now is acting as **Server**. This is due to the concern that **VString** might contain some private information of the verifier, which must be kept as secret in server-aided verification signatures. We can see that in the definition, adversary \mathcal{A} acts as the server and the challenger \mathcal{C} acts as the verifier. It simulates a practical attack where the adversary might be able to extract some information of **VString** during the interactions with the verifier. There is no need to consider the other case where \mathcal{A} acts as the verifier and \mathcal{C} acts as the (honest) server, respectively, as \mathcal{A} can perform all of the computations of an honest server.*

We will show in Section 3.3.3 that the adversary defined in the above model is stronger than that in [GL05].

3.3.2 Further Observations on EUF-SAV- Σ

We will show the relationship among EUF-SAV- Σ , EUF- Σ and Soundness-SA-Verify.

It is self-evident that EUF-SAV- Σ guarantees Soundness-SA-Verify. Otherwise, if there is an adversary who can prove an invalid signature is valid by SA-Verify with success probability ε , then it can also break the existential unforgeability of SAV- Σ with the same probability. We now prove that EUF-SAV- Σ also implies EUF- Σ .

Theorem 3.1 *If SAV- Σ is $(t, q_s, q_v, \varepsilon)$ -existentially unforgeable, then Σ is (t, q_s, ε) -existentially unforgeable.*

Proof: Let the ordinary signature scheme $\Sigma = (\mathbf{ParamGen}, \mathbf{KeyGen}, \mathbf{Sign}, \mathbf{Verify})$, and its server-aided verification counterpart $\mathbf{SAV}\text{-}\Sigma = (\Sigma, \mathbf{SA}\text{-}\mathbf{Verifier}\text{-}\mathbf{Setup}, \mathbf{SA}\text{-}\mathbf{Verify})$. We prove the correctness of this theorem by converting a (t, q_s, ε) forger $\Sigma_{\mathcal{A}}$ to a $(t, q_s, 0, \varepsilon)$ forger $\mathbf{SAV}\text{-}\Sigma_{\mathcal{A}}$.

As defined in the game in Section 3.3.1, $\mathbf{SAV}\text{-}\Sigma_{\mathcal{A}}$ will obtain $(param, pk)$ from its challenger of $\mathbf{SAV}\text{-}\Sigma$. Then, $\mathbf{SAV}\text{-}\Sigma_{\mathcal{A}}$ acts as the challenger of $\Sigma_{\mathcal{A}}$ as follows.

Setup. $(param, pk)$ is given to $\Sigma_{\mathcal{A}}$.

Queries. For each signature query m_i from $\Sigma_{\mathcal{A}}$, $\mathbf{SAV}\text{-}\Sigma_{\mathcal{A}}$ forwards m_i to its challenger as a signature query of $\mathbf{SAV}\text{-}\Sigma$. As defined, $\sigma_i = \mathbf{Sign}(param, m, sk, pk)$ will be returned as the answer. $\mathbf{SAV}\text{-}\Sigma_{\mathcal{A}}$ then forwards σ_i to $\Sigma_{\mathcal{A}}$. It is clear that each signature query from $\Sigma_{\mathcal{A}}$ can be correctly answered.

Output. After making queries, $\Sigma_{\mathcal{A}}$ will output a pair (m^*, σ^*) . $\mathbf{SAV}\text{-}\Sigma_{\mathcal{A}}$ sets (m^*, σ^*) as its own output.

If $\Sigma_{\mathcal{A}}$ (t, q_s, ε) -breaks the signature scheme Σ , then m^* is not one of the signature queries and $\Pr[\mathbf{Verify}(param, m^*, \sigma^*, pk) = \mathbf{Valid}] \geq \varepsilon$. Due to the completeness of $\mathbf{SAV}\text{-}\Sigma$, if $\mathbf{Verify}(param, m^*, \sigma^*, pk) = \mathbf{Valid}$, then $\mathbf{SA}\text{-}\mathbf{Verify}$ will return \mathbf{Valid} as well. Therefore, $\mathbf{SAV}\text{-}\Sigma_{\mathcal{A}}$ wins the game with the same probability ε , without making any server-aided verification queries. This completes the proof. \square

3.3.3 Analysis of the SAV- Σ in Asiacrypt'05

In this section, we consider the existential unforgeability of the generic SAV- Σ proposed by Girault and Lefranc [GL05]. Their server-aided verification protocol applies to signature schemes whose verification algorithms are similar to those in ZSS [ZSNS04] and BB [BB04] signatures.

The Description of SAV-ZSS [GL05]

1. **ParamGen:** Let $(\mathbb{G}_1, \mathbb{G}_T)$ be bilinear groups described in Section 2.1, where $|\mathbb{G}_1| = |\mathbb{G}_T| = p$, for some prime number $p \geq 2^k$, k be the system security number and g_1 be the generator of \mathbb{G}_1 . e denotes the bilinear map $\mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$. There is one cryptographic hash function $h : \{0, 1\}^* \rightarrow \mathbb{Z}_p$. The system parameter $param = (\mathbb{G}_1, \mathbb{G}_T, k, g_1, p, e, h)$.
2. **KeyGen:** The signer picks a random number $x \in \mathbb{Z}_p^*$ and keeps it as the secret key. The public key is set as $pk = g_1^x$.

Figure 3.1: SA-Verify in SAV-ZSS [GL05]

Verifier ($\mathbf{VString}: (t, K_1)$)	Server ($param$)
Input: $R = (g_1^{h(m)} \cdot pk)^t$	$K_2 = e(\sigma, R)$
Output: Valid , if $K_1 = K_2$ Invalid , otherwise.	$\xrightarrow{\sigma, R}$ $\xleftarrow{K_2}$

3. **Sign:** For a message m to be signed, the signer uses its secret key to generate the signature $\sigma = g_1^{\frac{1}{h(m)+x}}$.
4. **Verify:** For a message/signature pair (m, σ) , everyone can check whether $e(\sigma, g_1^{h(m)} \cdot pk) \stackrel{?}{=} e(g_1, g_1)$. If the equation holds, output **Valid**. Otherwise, output **Invalid**.
5. **SA-Verifier-Setup:** Given the system parameter $param = (\mathbb{G}_1, \mathbb{G}_T, k, g_1, p, e, h)$, the verifier picks a random integer t in \mathbb{Z}_p and computes $K_1 = e(g_1, g_1)^t$. The **VString** is (t, K_1) .
6. **SA-Verify:** The verifier and the server interact with each other using the protocol described in Figure 3.1.

Security of SAV-ZSS [GL05]. We now show that SAV-ZSS [GL05] is insecure in the model defined in Section 3.3.1, if the same (t, K_1) is used in each execution of **SA-Verify** described in Figure 3.1.

We first briefly review the security conclusion of SAV-ZSS proven in [GL05]:

1. A malicious server is not able to convince a verifier that an invalid signature of a message m is valid by using **SA-Verify** in Figure 3.1, if
2. The server does not know the ZSS signature of m and k -BCAA problem is hard (Please refer to [GL05] for the definition of k -BCAA).

However, the malicious server considered in [GL05] is not allowed to execute **SA-Verify** with the verifier, before it tries to prove the verifier that an invalid signature is valid. We believe this restriction is not reasonable as the verifier in the real world would execute **SA-Verify** with the server for several times. In the model defined in Section 3.3.1, we allow the adversary (acting as the server) to choose any message-signature pair, and execute **SA-Verify** with the challenger (acting as the verifier). This is analogous to the definition of existentially unforgeability, where the forger is allowed to obtain valid signatures of messages chosen by itself. Under this model, SAV-ZSS [GL05] will be insecure¹ if the same (t, K_1) is used in **SA-Verify** in Figure 3.1. The following shows how the adversary in our model can break the existential unforgeability of SAV-ZSS [GL05]:

1. The adversary \mathcal{A} first issues a signature query on a message m . Let the response from the challenger be σ .
2. \mathcal{A} makes a server-aided verification request (m, σ) . As shown in **SA-Verify** in Figure 3.1, the challenger will send the adversary $R = (g_1^{h(m)} \cdot pk)^t$.
3. \mathcal{A} computes $K_2 = e(\sigma, R)$. As σ is a valid ZSS signature of m , $K_1 = K_2 = e(g_1, g_1)^t$.
4. With the knowledge of K_1 , \mathcal{A} is able to prove that any invalid signature is valid if the same (t, K_1) is used in **SA-Verify**. To do that, \mathcal{A} just sends K_1 to the challenger in every execution of **SA-Verify**. Thus, \mathcal{A} can always win the game defined in Section 3.3.1.

The above attack will not work if the verifier pre-computes $q_v + 1$ pairs (t, K_1) in SAV-ZSS [GL05] and the adversary is allowed to make at most q_v server-aided verification queries. This will require more storage space for the verifier. Alternatively, the verifier can choose different t , and compute $(g_1^{H(m)} \cdot pk)^t$ and $e(g_1, g_1)^t$ in each execution of **SA-Verify**. This will however lead to one more exponentiation in \mathbb{G}_T than the computational cost of the verifier claimed in [GL05].

¹SAV-ZSS in [GL05] is still secure against the adversary defined in [GL05]. However, the adversary in [GL05] is weaker than the one defined in this chapter.

3.4 Existentially Unforgeable SAV-BLS and SAV-Waters

This section describes new server-aided verification signature schemes: SAV-BLS and SAV-Waters, respectively.

3.4.1 Existentially Unforgeable SAV-BLS

Our first protocol is based on the BLS signature [BLS01]. The description of our protocol is as following.

1. **ParamGen:** Let $(\mathbb{G}_1, \mathbb{G}_T)$ be bilinear groups described in Section 2.1, where $|\mathbb{G}_1| = |\mathbb{G}_T| = p$, for some prime number $p \geq 2^k$, k be the system security number and g_1 be the generator of \mathbb{G}_1 . e denotes the bilinear map $\mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$. There is one cryptographic hash function $H : \{0, 1\}^* \rightarrow \mathbb{G}_1$. The system parameter $param = (\mathbb{G}_1, \mathbb{G}_T, k, g_1, p, e, H)$.
2. **KeyGen:** The signer picks a random number $x \in \mathbb{Z}_p^*$ and keeps it as the secret key. The public key is set as $pk = g_1^x$.
3. **Sign:** For a message m to be signed, the signer uses its secret key to generate the signature $\sigma = H(m)^x$.
4. **Verify:** For a message/signature pair (m, σ) , everyone can check whether $e(\sigma, g_1) \stackrel{?}{=} e(H(m), pk)$. If the equation holds, output `Valid`. Otherwise, output `Invalid`.
5. **SA-Verifier-Setup:** Given the system parameter $(\mathbb{G}_1, \mathbb{G}_T, k, g_1, p, e, H)$, the verifier V randomly chooses $r \in \mathbb{Z}_p$ and sets $R = g_1^r$. The `VString` is (r, R) .
6. **SA-Verify:** The verifier V and the server S interact with each other using the protocol described in Figure 3.2.

Computational-Saving. The verifier in SAV-BLS described above needs to compute one pairing, one exponentiation on \mathbb{G}_T , and one map-to-point hash. It is obvious that Φ - **SA-Verify** < Φ -**Verify**.

Security Proof of Existentially Unforgeable SAV-BLS

Figure 3.2: SA-Verify in SAV-BLS with EUF

Verifier (VString: (r, R))	Server ($param$)
Input:	
$(m, \sigma), pk, param$	$\xrightarrow{\sigma, R}$
	$\xleftarrow{K_1}$
	$K_1 = e(\sigma, R)$
Compute:	
$K_2 = e(H(m), pk)^r$	
Output:	
Valid, if $K_1 = K_2$;	
Invalid, otherwise.	

Note that R is precomputed and the verifier sends the same R to the server in server-aided verification of different message-signature pairs.

Theorem 3.2 *The SAV-BLS signature scheme is $(t, q_s, q_v, q_h, \epsilon)$ -existentially unforgeable against adaptive chosen message attacks, if the Bilinear Diffie-Hellman problem is $(t + c_{(\mathbb{G}_1, \mathbb{G}_T)}(q_h + 2q_s + 2q_v + 1), \frac{\epsilon}{\epsilon q_v(q_s + 1)})$ -hard on $(\mathbb{G}_1, \mathbb{G}_T)$. Here, $c_{(\mathbb{G}_1, \mathbb{G}_T)}$ is a constant that depends on $(\mathbb{G}_1, \mathbb{G}_T)$ and ϵ is the base of natural logarithm.*

Proof: We will prove if there is a (t, q_s, q_v, q_h) adaptively chosen message adversary \mathcal{A} wins the game defined in Section 3.3.1 with probability ϵ , then there exists another algorithm \mathcal{B} which can solve a random instance of Bilinear Diffie-Hellman (BDH) problem in time $t' = t + c_{(\mathbb{G}_1, \mathbb{G}_T)}(q_h + 2q_s + 2q_v + 1)$ with success probability $\frac{\epsilon}{\epsilon q_v(q_s + 1)}$. This contradicts the assumption that the BDH problem is $(t', \frac{\epsilon}{\epsilon q_v(q_s + 1)})$ -hard on $(\mathbb{G}_1, \mathbb{G}_T)$.

We employ a similar technique in [BLS01] and regard the hash functions H as the random oracle. In the game, \mathcal{A} can adaptively make H **queries**, **Signature Queries** and **Server-Aided Verification Queries**. To make the proof more clearly, we introduce the notion of “*special pair*”. Let M_v be the set of \mathcal{A} 's server-aided verification queries in the game. A message-signature pair $(m, \sigma) \in M_v$ is a special pair if

1. By running **SA-Verify** \mathcal{A} can convince the challenger that (m, σ) is a valid message-signature pair; and

2. m has not appeared as one of the signature queries when \mathcal{A} makes the server-aided verification query (m, σ) .

We now define the following two events:

- **E1**: There is a special pair in M_v .
- **E2**: There is no special pair in M_v .

It is clear that either **E1** or **E2** happens in the game, and thus $\Pr[\mathbf{E1}] + \Pr[\mathbf{E2}] = 1$.

Let $(\mathbb{G}_1, \mathbb{G}_T)$ be bilinear groups of prime order p . Algorithm \mathcal{B} is given $g_1, g_1^a, g_1^b, g_1^c \in \mathbb{G}_1$ which is a random instance of the BDH problem. Its goal is to compute $e(g_1, g_1)^{abc}$. Algorithm \mathcal{B} will simulate the challenger and interact with the adversary \mathcal{A} as described below. Let **Succ** be the event that \mathcal{B} solves the given instance of the BDH problem, then we have

$$\Pr[\mathbf{Succ}] = \Pr[\mathbf{Succ} \wedge \mathbf{E1}] + \Pr[\mathbf{Succ} \wedge \mathbf{E2}] = \Pr[\mathbf{Succ}|\mathbf{E1}] \Pr[\mathbf{E1}] + \Pr[\mathbf{Succ}|\mathbf{E2}] \Pr[\mathbf{E2}].$$

We now consider each probability individually.

The Event Succ|E1: If the event **E1** happens, then there is a special pair in \mathcal{A} 's server-aided verification queries. At the beginning of the simulation, \mathcal{B} picks a random integer j in $\{1, 2, \dots, q_v\}$, and lets j be its guess of the index of the first special pair.

1. **Setup**: \mathcal{B} starts by setting $pk = g_1^a$ and $R = g_1^c$, where g_1^a, g_1^c are the inputs of the BDH problem and returns $(\mathbb{G}_1, \mathbb{G}_T, k, g_1, p, e, pk)$ to \mathcal{A} .
2. **H queries**: At any time the adversary \mathcal{A} can request the hash function H of the input m_i . To respond to these queries, algorithm \mathcal{B} will maintain an *H-List* which consists of tuple $(m_i, H(m_i), \alpha_i, coin_i)$ as explained later. For a query m_i , \mathcal{B} responds as follows.
 - (a) If there exists a tuple $(m_i, H(m_i), \alpha_i, coin_i)$ on the *H-List*, \mathcal{B} will return $H(m_i)$ to \mathcal{A} .
 - (b) Otherwise, \mathcal{B} will generate a random coin $coin_i \in \{0, 1\}$ such that $\Pr[coin_i = 1] = \frac{1}{q_s+1}$.
 - (c) If $coin_i = 1$, \mathcal{B} chooses $\alpha_i \in_R \mathbb{Z}_p$ and computes $H(m_i) = g_1^b \cdot g_1^{\alpha_i}$ where g_1^b is the input of the BDH problem.

- (d) Otherwise, $coin_i = 0$ and \mathcal{B} chooses $\alpha_i \in_R \mathbb{Z}_p$ and computes $H(m_i) = g_1^{\alpha_i}$.
- (e) At last \mathcal{B} returns $H(m_i)$ to \mathcal{A} and adds $(m_i, H(m_i), \alpha_i, coin_i)$ to the list $H\text{-List}$.

3. **Signature Queries:** At any time the adversary \mathcal{A} can request the signature of the message m_i . We assume that m_i already appears on the $H\text{-List}$ in a tuple $(m_i, H(m_i), \alpha_i, coin_i)$. Otherwise, \mathcal{B} makes an H query itself to ensure such tuple exists on the $H\text{-List}$.

- (a) If $coin_i = 0$, then \mathcal{B} responds \mathcal{A} with the signature $\sigma_i = pk^{\alpha_i}$. Note that σ_i is a valid signature as $e(\sigma_i, g_1) = e(pk^{\alpha_i}, g_1) = e(pk, H(m_i))$.
- (b) Otherwise, $coin_i = 1$, \mathcal{B} terminates the simulation and reports failure.

4. **Server-Aided Verification Queries:** At any time the adversary \mathcal{A} can make the server-aided verification of a message-signature pair (m_i, σ_i) . We assume that m_i already appears on the $H\text{-List}$. Otherwise, \mathcal{B} makes an H query itself to ensure such tuple exists on the $H\text{-List}$. Recall that \mathcal{B} has made a guess at the beginning of the game that the first special pair is the j^{th} server-aided verification query.

- (a) For the i^{th} query (m_i, σ_i) where $i < j$,
 - If m_i has never appeared as one of signature queries before this server-aided verification query, then \mathcal{B} will execute the server-aided verification protocol with \mathcal{A} but output `Invalid` at the end of the protocol, no matter what \mathcal{A} 's response is in the protocol. If \mathcal{B} 's guess of the first special pair is correct, then \mathcal{B} 's output will be correct as well.
 - Otherwise, \mathcal{A} has issued m_i as one of signature queries and \mathcal{B} answered with a valid signature which is denoted as σ'_i . In this case, \mathcal{B} will execute the protocol with \mathcal{A} . First, \mathcal{B} sends (σ_i, R) to \mathcal{A} . Then, \mathcal{A} responds with K_1 . At last, \mathcal{B} will output `Valid` if $K_1 = e(\sigma'_i, R)$. Otherwise, output `Invalid`.
- (b) Otherwise, $i = j$ and (m_j, σ_j) is a special pair. Let the corresponding tuple on the $H\text{-List}$ be $(m_j, H(m_j), \alpha_j, coin_j)$. If $coin_j = 0$, \mathcal{B} reports failure and aborts. Otherwise, $coin_j = 1$ and $H(m_j) = g_1^b \cdot g_1^{\alpha_j}$. \mathcal{B}

executes the server-aided verification protocol with \mathcal{A} , by sending (σ_j, R) to \mathcal{A} . As response, \mathcal{A} will send K_1 to \mathcal{B} . As (m_j, σ_j) is a special pair, the server-aided verification protocol **SA-Verify** will output **Valid**. We have $K_1 = e(g_1, g_1)^{abc} \cdot e(g_1^a, g_1^c)^{\alpha_j}$ as $pk = g_1^a$, $H(m_j) = g_1^b \cdot g_1^{\alpha_j}$ and $R = g_1^c$. \mathcal{B} terminates the simulation and outputs $K_1 \cdot e(g_1^a, g_1^c)^{-\alpha_j}$ as the solution to the given instance of the BDH problem.

We now compute the probability that \mathcal{B} solves the BDH problem if **E1** happens. All the following events are required for \mathcal{B} 's success.

1. \mathcal{B} does not abort as the result of \mathcal{A} 's signature queries. This happens with probability $(1 - 1/(q_s + 1))^{q_s} \geq 1/\epsilon$. Here, ϵ is the base of natural logarithm.
2. \mathcal{B} makes a correct guess of special pair, which happens with probability $1/q_v$. This also guarantees that before \mathcal{B} terminates the simulation, all server-aided verification queries from \mathcal{A} can be correctly answered.
3. The above two events happens and $coin_j = 1$ for the special pair (m_j, σ_j) . This happens with probability at least $1/(q_s + 1)$.

Therefore, if the event **E1** happens, the probability that \mathcal{B} can solve the random instance of BDH problem is $\Pr[\text{Succ}|\text{E1}] \geq \frac{1}{\epsilon q_v (q_s + 1)}$.

The Event Succ|E2: If the event **E2** happens, then there is no special pair in \mathcal{A} 's server-aided verification queries. \mathcal{B} responds \mathcal{A} 's queries as following.

1. **Setup, H queries, Signature Queries.** \mathcal{B} responds to these queries in the same way as described in the case **Succ|E1**.
2. **Server-Aided Verification Queries:** At any time the adversary \mathcal{A} can make a server-aided verification query of (m_i, σ_i) .
 - (a) If m_i has never appeared as one of signature queries before this query, then \mathcal{B} will execute the server-aided verification protocol with \mathcal{A} but output **Invalid** at the end of the protocol, no matter what \mathcal{A} 's response is in the protocol. It is clear that if **E2** happens, then \mathcal{B} 's output will be correct as well.

- (b) Otherwise, \mathcal{A} has issued m_i as one of signature queries and \mathcal{B} answered with a valid signature which is denoted as σ'_i . In this case, \mathcal{B} will execute the protocol with \mathcal{A} . First, \mathcal{B} sends (σ_i, R) to \mathcal{A} . Then, \mathcal{A} responds with K_1 . At last, \mathcal{B} will output **Valid** if $K_1 = e(\sigma'_i, R)$. Otherwise, output **Invalid**.

If \mathcal{B} does not abort the simulation, \mathcal{A} will output a message-signature pair (m^*, σ^*) with the restriction described in Section 3.3.1 and convince \mathcal{B} that σ^* is valid by **SA-Verify**. Let the corresponding tuple on the *H-List* be $(m^*, H(m^*), \alpha^*, coin^*)$. If $coin^* = 0$, \mathcal{B} reports failure and aborts. Otherwise, $coin^* = 1$ and $H(m^*) = g_1^b \cdot g_1^{\alpha^*}$. \mathcal{B} then executes the server-aided verification protocol **SA-Verify** with \mathcal{A} . Let K_1^* be the response of \mathcal{A} in **SA-Verify**. If \mathcal{A} can successfully prove that σ^* is a valid signature, then $K_1^* = e(g_1, g_1)^{abc} \cdot e(g_1^a, g_1^c)^{\alpha^*}$ as $pk = g_1^a$, $H(m^*) = g_1^b \cdot g_1^{\alpha^*}$ and $R = g_1^c$. \mathcal{B} thus can output $K_1^* \cdot e(g_1^a, g_1^c)^{-\alpha^*}$ as the solution to the given instance of the BDH problem.

We now compute the probability that \mathcal{B} solves the BDH problem if E2 happens. All the following events are required for \mathcal{B} 's success.

1. \mathcal{B} does not abort as the result of \mathcal{A} 's signature queries. This happens with probability $(1 - 1/(q_s + 1))^{q_s} \geq 1/\mathbf{e}$. Here, \mathbf{e} is the base of natural logarithm.
2. \mathcal{B} correctly answers all server-aided verification queries from \mathcal{A} . This happens with probability 1 if E2 happens.
3. \mathcal{A} can successfully prove the validity of σ^* by **SA-Verify**. This happens with the probability ε if \mathcal{B} 's simulation does not fail.
4. The above three events happen and $coin^* = 1$ for the pair (m^*, σ^*) . This happens with probability at least $1/(q_s + 1)$.

Therefore, if the event E2 happens, the probability that \mathcal{B} can solve the given instance of BDH problem is

$$\Pr[\text{Succ}|\text{E2}] \geq \frac{\varepsilon}{\mathbf{e}(q_s + 1)}.$$

Recall that $\Pr[\text{Succ}|\text{E1}] \geq \frac{1}{\epsilon q_v(q_s+1)}$ and $\Pr[\text{E1}] + \Pr[\text{E2}] = 1$, then the probability that \mathcal{B} can solve the given instance of BDH problem is

$$\begin{aligned} \Pr[\text{Succ}] &= \Pr[\text{Succ}|\text{E1}] \Pr[\text{E1}] + \Pr[\text{Succ}|\text{E2}] \Pr[\text{E2}] \\ &= \frac{1}{\epsilon q_v(q_s+1)} \Pr[\text{E1}] + \frac{\epsilon}{\epsilon(q_s+1)} \Pr[\text{E2}] \\ &\geq \frac{\epsilon}{\epsilon q_v(q_s+1)} (\Pr[\text{E1}] + \Pr[\text{E2}]) \\ &= \frac{\epsilon}{\epsilon q_v(q_s+1)}. \end{aligned}$$

Algorithm \mathcal{B} 's running time is the same as \mathcal{A} 's running time plus the time it takes to respond to $(q_h + q_s + q_v)$ random oracle queries, q_s signature queries, q_v verification queries and compute $e(g_1, g_1)^{\text{abc}}$ from \mathcal{A} 's output. Each requires at most one pairing operation and one exponentiation which we assume takes time $c_{(\mathbb{G}_1, \mathbb{G}_T)}$. Hence, the total running time is at most $t + c_{(\mathbb{G}_1, \mathbb{G}_T)}(q_h + 2q_s + 2q_v + 1)$. This completes the proof of Theorem 3.2. \square

3.4.2 Existentially Unforgeable SAV-Waters

This subsection provides a server-aided verification protocol for Waters signature [Wat05]. The details are given as follows.

1. **ParamGen:** Let $(\mathbb{G}_1, \mathbb{G}_T)$ be bilinear groups described in Section 2.1, where $|\mathbb{G}_1| = |\mathbb{G}_T| = p$, for some prime number $p \geq 2^k$, k be the system security number and g_1 be the generator of \mathbb{G}_1 . e denotes the bilinear mapping $\mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$. The system parameter $param = (\mathbb{G}_1, \mathbb{G}_T, k, g_1, p, e)$. The message space $\mathbb{M} = \{0, 1\}^n$.
2. **KeyGen:** Given the system parameter $(\mathbb{G}_1, \mathbb{G}_T, k, g_1, p, e)$, the signer generates the public key pk as (\vec{v}, PK) and $sk = x$, here \vec{v} is a vector consisting of $n+1$ elements $V_0, V_1, V_2, \dots, V_n$ randomly selected in \mathbb{G}_1 and $PK = e(g_1, g_1)^x$, where x is a random element in \mathbb{Z}_p .
3. **Sign:** For an n -bit message m in $\{0, 1\}^n$, let $\mathcal{M} \subseteq \{1, 2, \dots, n\}$ be the set of all i for which the i^{th} bit of m is 1. The signature σ is constructed as: $\sigma = (\sigma_1, \sigma_2) = (g_1^x (V_0 \prod_{i \in \mathcal{M}} V_i)^r, g_1^r)$, where $r \in_R \mathbb{Z}_p$.
4. **Verify:** For a claimed signature $\sigma = (\sigma_1, \sigma_2)$ of a message m , this algorithm outputs **Valid** if and only if $e(\sigma_1, g_1) = PK \cdot e(V_0 \prod_{i \in \mathcal{M}} V_i, \sigma_2)$. Otherwise, output **Invalid**.

Figure 3.3: The SAV Protocol For Waters Signature

Verifier (VString: (d, D))	Server
Input:	
$(m, \sigma = (\sigma_1, \sigma_2)), pk, param$	$\xrightarrow{\sigma_1, D}$
	$\xleftarrow{K_1}$
$K_2 = e(\sigma_2, V_0 \prod_{i \in \mathcal{M}} V_i)$	$K_1 = e(\sigma_1, D)$
Output:	
Valid, if $K_1 = (PK \cdot K_2)^d$	
Invalid, otherwise.	

Note that the value D is precomputed, at each execution of **SV-Verify**, verifier sends the same D to the server.

5. **SA-Verifier-Setup**: Given the system parameter $(\mathbb{G}_1, \mathbb{G}_T, k, g_1, p, e)$, the verifier randomly chooses $d \in \mathbb{Z}_p^*$, then calculates $D = g_1^d$. The VString is (d, D) .
6. **SA-Verify**: The verifier and the server interact with each other using the protocol described in Figure 3.3.

Computational-Saving. We replace the pairing operation $e(\sigma_1, g_1)$ in the **Verify** algorithm with one exponentiation on \mathbb{G}_T . Thus, we have Φ -**SA-Verify** $<$ Φ -**Verify**.

Theorem 3.3 *The SAV-Waters described above is $(t, q_s, q_v, \varepsilon)$ -existentially unforgeable against chosen message attacks, if the Gap Bilinear Diffie-Hellman problem is $(t + c_{(\mathbb{G}_1, \mathbb{G}_T)}(q_s + q_v + 1), \frac{\varepsilon}{8q_s q_v (n+1)})$ -hard on $(\mathbb{G}_1, \mathbb{G}_T)$. Here, $c_{(\mathbb{G}_1, \mathbb{G}_T)}$ is a constant that depends on $(\mathbb{G}_1, \mathbb{G}_T)$.*

Proof: We prove that if there is a (t, q_s, q_v) adaptively chosen message adversary \mathcal{A} wins the game defined in Section 3.3.1 with probability ε , then there exists another algorithm \mathcal{B} which can solve a random instance of Gap Bilinear Diffie-Hellman (GBDH) problem in time $t + c_{(\mathbb{G}_1, \mathbb{G}_T)}(q_s + q_v + 1)$ with success probability $\frac{\varepsilon}{8q_s q_v (n+1)}$. This contradicts the assumption that GBDH is $(t + c_{(\mathbb{G}_1, \mathbb{G}_T)}(q_s + q_v + 1), \frac{\varepsilon}{8q_s q_v (n+1)})$ -hard on $(\mathbb{G}_1, \mathbb{G}_T)$.

Let $(\mathbb{G}_1, \mathbb{G}_T)$ be bilinear groups of prime order p . Algorithm \mathcal{B} is given $g_1, g_1^a, g_1^b, g_1^c \in \mathbb{G}_1$ which is a random instance of the GBDH problem. Its goal is to compute $e(g_1, g_1)^{abc}$. Algorithm \mathcal{B} will simulate the challenger and interact with the adversary

\mathcal{A} as described below. \mathcal{A} can adaptively make **Signature Queries** and **Server-Aided-Verification Queries**. To make the proof more clearly, we introduce the notion of “*special pair*”. Let M_v be the set of \mathcal{A} 's server-aided verification queries in the game. A message-signature pair $(m, \sigma) \in M_v$ is a special pair if

1. By running **SA-Verify** \mathcal{A} can convince the challenger that (m, σ) is a valid message-signature pair; and
2. m has not appeared as one of the signature queries when \mathcal{A} makes the server-aided verification query (m, σ) .

We now define the following two events:

- **E1**: There is a special pair in M_v .
- **E2**: There is no special pair in M_v .

Either **E1** or **E2** happens in the game, and thus $\Pr[\mathbf{E1}] + \Pr[\mathbf{E2}] = 1$.

The Event Succ|E1: If the event **E1** happens, then there is a special pair in \mathcal{A} 's server-aided-verification queries. \mathcal{B} picks a random integer j in $\{1, 2, \dots, q_v\}$, and guesses this is the index of the first special pair.

1. **Setup**: \mathcal{B} first sets an integer $z = 4q_s$, and chooses an integer k , where $k \in_R \{0, 1, \dots, n\}$. Then \mathcal{B} chooses random $(n + 1)$ -length vectors $\vec{\alpha} = (\alpha_i)$ and $\vec{\beta} = (\beta_i)$, where $\alpha_i \in_R \{0, 1, \dots, z\}$ and $\beta_i \in_R \mathbb{Z}_p$, respectively.

Meanwhile, for a message $m \in \{0, 1\}^n$, we set $\mathcal{M} \in \{1, 2, \dots, n\}$ be the set of all i for which the i^{th} bit of m is 1. Then we define three functions.

$$\begin{aligned} F(m) &= (p - zk) + \alpha_0 + \sum_{i \in \mathcal{M}} \alpha_i; \\ J(m) &= \beta_0 + \sum_{i \in \mathcal{M}} \beta_i; \\ K(m) &= \begin{cases} 0, & \text{if } \alpha_0 + \sum_{i \in \mathcal{M}} \alpha_i \equiv 0 \pmod{z}, \\ 1, & \text{otherwise.} \end{cases} \end{aligned}$$

\mathcal{B} sets $g' = g_1^a$, $g'' = g_1^b$ and $D = g_1^c$ where g_1^a , g_1^b and g_1^c are the inputs of the GBDH problem. \mathcal{B} then calculates $V_0 = g'^{p-kz+\alpha_0} g_1^{\beta_0}$, $V_i = g'^{\alpha_i} g_1^{\beta_i}$, $i = 1, 2, \dots, n$, $PK = e(g', g'')$, and sets $\vec{v} = (V_i)$, the public key $pk = (\vec{v}, PK)$.

2. **Signature Queries**: At any time, the adversary \mathcal{A} can request the signature of the input m .

- (a) If $K(m) = 0$, \mathcal{B} terminates the simulation and reports failure.
- (b) Otherwise, $K(m) \neq 0$ which implies $F(m) \neq 0$, \mathcal{B} chooses $r \in_R \mathbb{Z}_p$, and generates the signature as

$$\sigma = (\sigma_1, \sigma_2) = (g''^{\frac{-J(m)}{F(m)}} (V_0 \prod_{i \in \mathcal{M}} V_i)^r, g''^{\frac{-1}{F(m)}} g^r).$$

σ is a valid signature as shown below.

Let $\tilde{r} = r - \frac{b}{F(m)}$, we have

$$\begin{aligned} \sigma_1 &= g''^{\frac{-J(m)}{F(m)}} (V_0 \prod_{i \in \mathcal{M}} V_i)^r \\ &= g''^{\frac{-J(m)}{F(m)}} (g'^{F(m)} g_1^{J(m)})^r \\ &= g_1^{\text{ab}} g_1^{-\text{ab}} g''^{\frac{-J(m)}{F(m)}} (g'^{F(m)} g_1^{J(m)})^r \\ &= g_1^{\text{ab}} (g'^{F(m)} g_1^{J(m)})^{\frac{-b}{F(m)}} (g'^{F(m)} g_1^{J(m)})^r \\ &= g_1^{\text{ab}} (g'^{F(m)} g_1^{J(m)})^{r - \frac{b}{F(m)}} \\ &= g_1^{\text{ab}} (V_0 \prod_{i \in \mathcal{M}} V_i)^{\tilde{r}}. \end{aligned}$$

$$\text{And } \sigma_2 = g''^{\frac{-1}{F(m)}} g_1^r = g_1^{r - \frac{b}{F(m)}} = g_1^{\tilde{r}}.$$

3. Server-Aided-Verification Queries: At any time, the adversary \mathcal{A} can make a Server-Aided-Verification query of (m_l, σ_l) , where $\sigma_l = (\sigma_{l1}, \sigma_{l2})$.

- (a) For $l \in \{1, 2, \dots, q_v\}$ and $l < j$,
- If m_l has not appeared as one of **Signature Queries**, \mathcal{B} will execute the server-aided-verification protocol with \mathcal{A} but output **Invalid** at the end of the protocol, no matter \mathcal{A} 's response in the protocol. It is clear that if \mathcal{B} 's guess of special pair is correct, then \mathcal{B} 's output will be correct as well.
 - Otherwise, \mathcal{A} has made a signature query of m_l and let \mathcal{B} 's answer be $\sigma'_l = (\sigma'_{l1}, \sigma'_{l2})$. In this case, \mathcal{B} will execute the protocol with \mathcal{A} . First, \mathcal{B} sends (σ_{l1}, D) to \mathcal{A} . Then, \mathcal{A} responds with K_{l1} . After that \mathcal{B} computes $\theta = \frac{\sigma'_{l2}}{\sigma'_{l1}}$, $\vartheta = V_0 \prod_{i \in \mathcal{M}} V_i$, $\lambda = \frac{K_{l1}}{e(\sigma'_{l1}, D)}$, and issues the query $(g, \theta, \vartheta, D, \lambda)$ to $\mathcal{O}_{\text{GBDH}}$. At last, \mathcal{B} will output **Valid** if $\mathcal{O}_{\text{GBDH}}$ returns 1. Otherwise, the output is **Invalid**.

Note that if $\mathcal{O}_{\text{GBDH}}$ returns 1, i.e. $\lambda = e(\theta, \vartheta)^c$, then we have

$$\begin{aligned}
\frac{K_{1l}}{e(\sigma'_{l1}, D)} &= e\left(\frac{\sigma_{l2}}{\sigma'_{l2}}, V_0 \prod_{i \in \mathcal{M}} V_i\right)^c \\
K_{1l} &= e(\sigma'_{l1}, D) \cdot e\left(\frac{\sigma_{l2}}{\sigma'_{l2}}, V_0 \prod_{i \in \mathcal{M}} V_i\right)^c \\
&= e(g_1, g_1)^{abc} \cdot e(\sigma'_{l2}, V_0 \prod_{i \in \mathcal{M}} V_i)^c \cdot e\left(\frac{\sigma_{l2}}{\sigma'_{l2}}, V_0 \prod_{i \in \mathcal{M}} V_i\right)^c \\
&= e(g_1, g_1)^{abc} \cdot e(\sigma_{l2}, V_0 \prod_{i \in \mathcal{M}} V_i)^c \\
&= [PK \cdot e(\sigma_{l2}, V_0 \prod_{i \in \mathcal{M}} V_i)]^c,
\end{aligned}$$

which means σ_l is valid according to the protocol described in Figure 3.3. Otherwise $\mathcal{O}_{\text{GBDH}}$ returns 0, σ_l is invalid.

- (b) For $l \in \{1, 2, \dots, q_v\}$ and $l = j$, let (m^*, σ^*) be the l^{th} query where $\sigma^* = (\sigma_1^*, \sigma_2^*)$. After receiving D from \mathcal{B} , \mathcal{A} will send K_1^* to \mathcal{B} . As (m^*, σ^*) is a special pair, \mathcal{A} can prove it as valid by using the protocol **SA-Verify** described in Figure 3.3. \mathcal{B} will receive K_1^* such that

$$\begin{aligned}
K_1^* &= (PK \cdot K_2^*)^c = e(g_1, g_1)^{abc} e(\sigma_{l2}, V_0 \prod_{i \in \mathcal{M}} V_i)^c \\
&= e(g_1, g_1)^{abc} e((g^{F(m^*)} g_1^{J(m^*)})^{r^*}, D).
\end{aligned}$$

If $\alpha_0 + \sum_{i \in \mathcal{M}^*} \alpha_i = kz$, then $K_1^* = e(g_1, g_1)^{abc} e((\sigma_2^*)^{J(m^*)}, D)$. \mathcal{B} can solve the GBGDH problem by computing $e(g_1, g_1)^{abc} = K_1^* / e((\sigma_2^*)^{J(m^*)}, D)$ and terminate the simulation.

\mathcal{B} can output $e(g_1, g_1)^{abc}$ if and only if

- (a) \mathcal{B} does not abort during **Signature Queries**, the probability that this event happens is $\Pr[\bigwedge_{i=1}^{q_s} K(m_i) = 1]$.
- (b) \mathcal{B} makes a correct guess of special pair, which happens with the probability $\frac{1}{q_v}$. This also guarantees that before \mathcal{B} terminates the simulation, all server-aided-verification queries from \mathcal{A} can be correctly answered.
- (c) The above two events happen and $\alpha_0 + \sum_{i \in \mathcal{M}^*} \alpha_i = kz$ in the special pair (m^*, σ^*) .

Therefore, the probability that \mathcal{B} can successfully output $e(g_1, g_1)^{abc}$ is

$$\frac{1}{q_v} \Pr\left[\bigwedge_{i=1}^{q_s} K(m_i) = 1 \wedge \alpha_0 + \sum_{i \in \mathcal{M}^*} \alpha_i = kz\right].$$

We have

$$\begin{aligned}
& \Pr\left[\bigwedge_{i=1}^{q_s} K(m_i) = 1 \wedge \alpha_0 + \sum_{i \in \mathcal{M}^*} \alpha_i = kz\right] \\
&= (1 - \Pr\left[\bigvee_{i=1}^{q_s} K(m_i) = 0\right]) \Pr\left[\alpha_0 + \sum_{i \in \mathcal{M}^*} \alpha_i = kz \mid \bigwedge_{i=1}^{q_s} K(m_i) = 1\right] \\
&\geq (1 - \sum_{i=1}^{q_s} \Pr[K(m_i) = 0]) \Pr\left[\alpha_0 + \sum_{i \in \mathcal{M}^*} \alpha_i = kz \mid \bigwedge_{i=1}^{q_s} K(m_i) = 1\right] \\
&= (1 - \frac{q_s}{z}) \Pr\left[\alpha_0 + \sum_{i \in \mathcal{M}^*} \alpha_i = kz \mid \bigwedge_{i=1}^{q_s} K(m_i) = 1\right] \\
&= (1 - \frac{q_s}{z}) \frac{1}{(n+1)} \Pr[K(m^*) = 0 \mid \bigwedge_{i=1}^{q_s} K(m_i) = 1] \\
&= (1 - \frac{q_s}{z}) \frac{1}{(n+1)} \frac{\Pr[K(m^*) = 0]}{\Pr[\bigwedge_{i=1}^{q_s} K(m_i) = 1]} \Pr\left[\bigwedge_{i=1}^{q_s} K(m_i) = 1 \mid K(m^*) = 0\right] \\
&\geq (1 - \frac{q_s}{z}) \frac{1}{(n+1)} \frac{1}{z} \Pr\left[\bigwedge_{i=1}^{q_s} K(m_i) = 1 \mid K(m^*) = 0\right] \\
&= (1 - \frac{q_s}{z}) \frac{1}{z(n+1)} (1 - \Pr\left[\bigvee_{i=1}^{q_s} K(m_i) = 0 \mid K(m^*) = 0\right]) \\
&\geq (1 - \frac{q_s}{z}) \frac{1}{z(n+1)} (1 - \sum_{i=1}^{q_s} \Pr[K(m_i) = 0 \mid K(m^*) = 0]) \\
&= (1 - \frac{q_s}{z}) \frac{1}{z(n+1)} (1 - \frac{q_s}{z}) \\
&\geq \frac{1}{z(n+1)} (1 - 2\frac{q_s}{z}) \\
&= \frac{1}{8q_s(n+1)}.
\end{aligned}$$

Therefore, $\Pr[\text{Succ}|\text{E1}] \geq \frac{1}{8q_s q_v (n+1)}$.

The Event Succ|E2: If the event E2 happens, then there is no special pair in \mathcal{A} 's server-aided-verification queries.

1. **Setup, Signature Queries.** \mathcal{B} responds to these queries in the same way as described in the case Succ|E1.
2. **Server-Aided-Verification Queries:** At any time, the adversary \mathcal{A} can make a Server-Aided-Verification query of the input (m_l, σ_l) , \mathcal{B} answers the query in the same way described in the 3a of the first event E1.

If \mathcal{B} does not abort during the simulation, \mathcal{A} will output a message/signature pair (m^*, σ^*) where $\sigma^* = (\sigma_1^*, \sigma_2^*)$ with the restriction described in Section 3.3.1. If \mathcal{A} can prove σ^* as a valid signature by **SA-Verify** protocol described in Figure 3.3. Then \mathcal{A} will return D^* to \mathcal{B} such that $K_1^* = e(g_1, g_1)^{\text{abc}} e(\sigma_2^*, V_0 \prod_{i \in \mathcal{M}} V_i)^c$. \mathcal{B} can calculate $e(g_1, g_1)^{\text{abc}} = K_1^*/e(\sigma_2^*, V_0 \prod_{i \in \mathcal{M}} V_i)^c$. \mathcal{B} can calculate $e(g_1, g_1)^{\text{abc}}$ if and only if

- (a) \mathcal{B} does not abort during **Signature Queries**, the probability that this event happens is $\Pr[\bigwedge_{i=1}^{q_s} K(m_i) = 1]$.
- (b) \mathcal{B} makes a correct guess of special pair, which happens with the probability $\frac{1}{q_v}$. This also guarantees that before \mathcal{B} terminates the simulation, all server-aided-verification queries from \mathcal{A} can be correctly answered.
- (c) The above two events happen and $\alpha_0 + \sum_{i \in \mathcal{M}^*} \alpha_i = kz$ in the pair (m^*, σ^*) .

Therefore, in this event, the probability that \mathcal{B} can successfully output $e(g_1, g_1)^{\text{abc}}$ is

$$\begin{aligned} \Pr[\text{Succ}|\text{E2}] &= \Pr[\bigwedge_{i=1}^{q_s} K(m_i) = 1 \wedge \alpha_0 + \sum_{i \in \mathcal{M}^*} \alpha_i = kz] \cdot \varepsilon \\ &\geq \frac{1}{8q_s(n+1)} \cdot \varepsilon. \end{aligned}$$

Above all, \mathcal{B} can solve the GBDH problem with the probability

$$\begin{aligned} \Pr[\text{Succ}] &= \Pr[\text{Succ}|\text{E1}]\Pr[\text{E1}] + \Pr[\text{Succ}|\text{E2}]\Pr[\text{E2}] \\ &\geq \frac{1}{q_v} \frac{1}{8q_s(n+1)} \cdot \Pr[\text{E1}] + \frac{1}{8q_s(n+1)} \cdot \varepsilon \cdot \Pr[\text{E2}] \\ &\geq \frac{1}{q_v} \frac{1}{8q_s(n+1)} \cdot \varepsilon \cdot (\Pr[\text{E1}] + \Pr[\text{E2}]) \\ &= \frac{\varepsilon}{8q_s q_v (n+1)}. \end{aligned}$$

Algorithm \mathcal{B} 's running time is the same as \mathcal{A} 's running time plus the time it takes to respond to q_s signature queries, q_v verification queries and compute $e(g_1, g_1)^{\text{abc}}$ from \mathcal{A} 's output. Assume each takes time $c_{(\mathbb{G}_1, \mathbb{G}_T)}$. Hence, the total running time is at most $t + c_{(\mathbb{G}_1, \mathbb{G}_T)}(q_s + q_v + 1)$. This completes the proof of Theorem 3.3. \square

3.5 SAV- Σ Secure Against Collusion and Adaptive Chosen Message Attacks

We pay our attention on investigating the security of SAV- Σ against the collusion between the server and the signer in this section, and propose server-aided verification protocols secure against this attack. The definition of SAV- Σ and all the notation used in this chapter are the same as what we have described previously.

3.5.1 Security of SAV- Σ Against Collusion and Adaptive Chosen Message Attacks

If we allow the server and the signer to collude, the server will have valid signatures of any messages. Thus, it is impossible to give a unified security notion to capture both EUF- Σ and **Soundness-SA-Verify** simultaneously. With this in mind, we now define the soundness of the server-aided verification protocol **SA-Verify** against collusion and adaptive chosen message attacks. In the game, the adversary is given the secret key of the signer.

Setup. The challenger \mathcal{C} runs the algorithm **ParamGen**, **KeyGen** and **SA-Verifier-Setup** to obtain system parameter $param$, one key pair (sk, pk) and $VString$. The adversary \mathcal{A} is given $param$ and (sk, pk) .

Queries. The adversary \mathcal{A} only needs to make **Server-Aided Verification Queries**. Proceeding adaptively, the adversary \mathcal{A} can make at most q_v such queries. The challenger \mathcal{C} responds each query in the same way as described in Definition 3.1.

Output. The adversary \mathcal{A} will output a message m^* . We denote Ω_{m^*} as the set of valid signatures of m^* . The challenger \mathcal{C} chooses a random element σ^* in $\Omega \setminus \Omega_{m^*}$. That is, σ^* is a random invalid signature of m^* . We say \mathcal{A} wins the game if

$$\mathbf{SA-Verify}(\mathcal{A}, \mathcal{C}^{(m^*, \sigma^*, pk, VString)}) = \text{Valid}.$$

Note that the challenge signature is chosen by the challenger, and is not given to the adversary. This is different from the collusion and adaptive chosen message attacks defined in [WMSH08]. Our definition ensures that even the server has the knowledge of user's private key, it cannot prove an invalid signature to be valid.

We define **Soundness-SA-Verify-Adv \mathcal{A}** to be the probability that the adversary \mathcal{A} wins in the above game, taken over the coin tosses made by \mathcal{A} and the challenger.

Definition 3.2 *An adversary \mathcal{A} is said to (t, q_v, ε) -break the soundness of **SA-Verify** in a SAV- Σ if \mathcal{A} runs in time at most t , makes at most q_v server-aided verification queries, and **Soundness-SA-Verify-Adv \mathcal{A}** is at least ε . The **SA-Verify** in a SAV- Σ is (t, q_v, ε) -sound against collusion and adaptive chosen message attacks if there exists no adversary that (t, q_v, ε) -breaks it.*

Definition 3.3 *SAV- Σ is $(t, q_s, q_v, \varepsilon)$ -secure against collusion and adaptive chosen message attacks if Σ is (t, q_s, ε) -existentially unforgeable against adaptive chosen message attacks and its server-aided verification protocol **SA-Verify** is (t, q_v, ε) -sound against collusion and adaptive chosen message attacks.*

3.5.2 SAV-BLS Secure Against Collusion and Adaptive Chosen Message Attacks

We now give a server-aided verification protocol for the BLS signature [BLS01], which is secure against the collusion and adaptive chosen message attacks.

1. **ParamGen, KeyGen, Sign, Verify**: these algorithms are the same as defined in Section 3.4.1.
2. **SA-Verifier-Setup**: Given the system parameter $(\mathbb{G}_1, \mathbb{G}_T, k, g_1, p, e, H)$, the verifier V computes $K_1 = e(g_1, g_1)$. The **VString** is K_1 .
3. **SA-Verify**: The verifier V and the server S interact with each other using the protocol described in Figure 3.4.

Computational-Saving. The verifier in SAV-BLS described above only needs to compute one multiplication on \mathbb{G}_1 , one (fixed-base) exponentiation on \mathbb{G}_1 , one multiplication on \mathbb{G}_T and one (fixed-based) exponentiation on \mathbb{G}_T . In particular, there is no pairing or map-to-point operation. Thus, Φ -**SA-Verify** $<$ Φ -**Verify**.

Security Proof of SAV-BLS Against Collusion and Chosen Message Attacks

We only need to show that the server-aided verification protocol in Figure 3.4 is sound against collusion and adaptive chosen message attacks.

Figure 3.4: SA-Verify in SAV-BLS with Soundness I

Verifier (VString: K_1)	Server (param)
Input: $(m, \sigma), pk, param$	
Compute: $r \in_R \mathbb{Z}_p, \sigma' = \sigma \cdot g_1^r$	$\xrightarrow{m, \sigma', pk}$ $K_2 = e(\sigma', g_1)$ $\xleftarrow{K_2, K_3}$ $K_3 = e(H(m), pk)$
Output: Valid, if $K_2 = K_3 \cdot K_1^r$ Invalid, otherwise.	

Theorem 3.4 *The server-aided verification protocol described in Figure 3.4 is $(t, q_v, \frac{1}{p-1})$ -sound against collusion and adaptive chosen message attacks.*

Proof: We prove that the adversary's probability to prove an invalid signature as valid is $\frac{1}{p-1}$, without any complexity assumption.

1. **Setup:** The challenger starts by choosing the secret key $sk = x \in_R \mathbb{Z}_p$, sets the public key as $pk = g_1^x$ and computes $K_1 = e(g_1, g_1)$, then returns $(\mathbb{G}_1, \mathbb{G}_T, k, g_1, p, e, H)$ and (sk, pk) to \mathcal{A} .
2. **Server-Aided Verification Queries:** At any time the adversary \mathcal{A} can make the server-aided verification query (m_i, σ_i) . The challenger executes the protocol **SA-Verify** with \mathcal{A} as described in Figure 3.4.
3. **Output:** After all the queries, \mathcal{A} will output a message m^* . As response, the challenger will choose a random element σ^* in $\mathbb{G}_1 \setminus \{H(m^*)^{sk}\}$, that is, σ^* is a random element in invalid signature space of m^* .

The challenger then executes **SA-Verify** with \mathcal{A} as described in Figure 3.4. The challenger selects $r^* \in_R \mathbb{Z}_p$ and computes $\sigma'^* = \sigma^* \cdot g_1^{r^*}$. After that, σ'^* is sent to \mathcal{A} , who will return K_2^* and K_3^* as the response. We now show that $K_2^* = K_3^* \cdot K_1^{r^*}$ happens with probability $\frac{1}{p-1}$. The following analysis use the similar technique in the proof of Cramer-Shoup encryption scheme [CS98].

- (a) The element σ'^* sent to \mathcal{A} does not constrain the distribution of (σ^*, r^*) . This is due to the reason that given σ'^* , there are $(p - 1)$ pairs (σ_i, r_i) satisfy the equation $\sigma'^* = \sigma_i \cdot g_1^{r_i}$ and (σ^*, r^*) chosen by the challenger is just a random one among these $p - 1$ pairs. In other words, from the adversary's view, each (σ_i, r_i) has the equal probability to be (σ^*, r^*) . To make our analysis more clear, we rewrite the the equation $\sigma'^* = \sigma^* \cdot g_1^{r^*}$ as

$$DL_{g_1}\sigma'^* = DL_{g_1}\sigma^* + r^* \quad (1)$$

- (b) Suppose \mathcal{A} returns K_2^* and K_3^* such that $K_2^* = K_3^* \cdot K_1^{r^*}$. We rewrite this equation as

$$DL_{e(g_1, g_1)}K_2^* = DL_{e(g_1, g_1)}K_3^* + r^* \quad (2)$$

If $DL_g\sigma^* \neq DL_{e(g_1, g_1)}K_3^*$, then Equations (1) and (2) are linearly independent. It follows that r^* satisfies Equation (2) with probability $\frac{1}{p-1}$, as (σ^*, r^*) is randomly chosen from $p-1$ pairs from the view of the adversary. Otherwise, $DL_{g_1}\sigma^* = DL_{e(g_1, g_1)}K_3^*$, that is, $e(\sigma^*, g_1) = K_3^*$. This means that σ^* is uniquely determined by K_3^* . As σ'^* sent to \mathcal{A} does not constrain the distribution of σ^* , this happens also with probability $\frac{1}{p-1}$.

Therefore, the probability that **SA-Verify** will output **Valid** is $\frac{1}{p-1}$. This completes the proof of this theorem. \square

Remark 3.2 *Recently, a new type of “collusion attackers” was defined in [WWYH10]. In the new definition, an attacker is given a key pair (sk_f, pk_f) and a public key pk , while the challenger keeps the corresponding private key sk as secret. The attacker is said to break the soundness of the SAV protocol if he/she can find a message/signature pair (m^*, σ^*) which is valid under the public key pk_f and can be proven as valid under pk via SAV protocol. As one can see, the adversary defined in [WWYH10] actually belongs to those defined in Section 3.3.1, i.e., adversary does not have the private key but can choose the challenge message/signature pair. (Also notice that the key pair (sk_f, pk_f) can also be generated by the adversaries defined in Section 3.5.1.) This is different from the collusion attacks defined in Section 3.5.1, where the adversary is given the private sk but is not allowed to choose the challenge signature σ^* . As shown in [WWYH10], SAV protocols secure against collusion attacks defined in this section might be existentially forgeable, since adversaries in these two notions are different. It is certainly more desirable if SAV protocols are secure*

Figure 3.5: SA-Verify in SAV-Waters with Soundness

Verifier (VString: K_1)	Server
Input: $(m, \sigma = (\sigma_1, \sigma_2)), pk, param$ $d \in_R \mathbb{Z}_p, \sigma'_1 = \sigma_1 \cdot g_1^d$	$\xrightarrow{m, \sigma'_1, \sigma_2}$ $K_2 = e(\sigma'_1, g_1)$ $\xleftarrow{K_2, K_3}$ $K_3 = e(V_0 \prod_{i \in \mathcal{M}} V_i, \sigma_2)$
Output: Valid, if $K_2 = PK \cdot K_3 \cdot K_1^d$ Invalid, otherwise.	

against collusion attacks where adversaries are also allowed to choose the challenge message/signature pair. These protocols will be investigated in Section 3.6.

3.5.3 SAV-Waters Secure Against Collusion and Adaptive Chosen Message Attacks

In this section, we provide another server-aided verification protocol, which is based on Waters signature [Wat05] and secure against collusion and adaptive chosen message attacks.

1. **ParamGen, KeyGen, Sign, Verify:** these algorithms are the same as defined in Section 3.4.2.
2. **SA-Verifier-Setup:** Given the system parameter $(\mathbb{G}_1, \mathbb{G}_T, k, g_1, p, e)$, V computes $K_1 = e(g_1, g_1)$. Then set $D = g_1^d$. The VString is K_1 .
3. **SA-Verify:** The verifier V and the server S interact with each other using the protocol described in Figure 3.5.

The verifier in SAV-Waters described above only needs to compute one multiplication on \mathbb{G}_1 , one (fixed-base) exponentiation on \mathbb{G}_1 , two multiplications on \mathbb{G}_T and one (fixed-based) exponentiation on \mathbb{G}_T . In particular, there is no pairing operation. Thus, Φ -SA-Verify $<$ Φ -Verify.

Security Proof of SAV-Waters Against Collusion and Chosen Message Attacks

We will show that the server-aided-verification protocol is sound against collusion and adaptive chosen message attacks.

Theorem 3.5 *The server-aided-verification protocol described in Figure 3.5 is $(t, q_v, \frac{1}{p-1})$ -sound against collusion and adaptive chosen message attacks.*

Proof: We will prove that the adversary's probability to prove an invalid signature as valid is $\frac{1}{p-1}$.

1. **Setup:** The challenger starts by choosing the secret key $sk = x \in_R \mathbb{Z}_p$, sets the public key as $pk = (\vec{v}, PK)$ where $\vec{v} = (V_i)$, $i = 0, 1, \dots, n$ and $V_i \in_R \mathbb{G}_1$. The challenger also computes $K_1 = e(g_1, g_1)$, then returns $(\mathbb{G}_1, \mathbb{G}_T, e, p, sk, pk)$ to \mathcal{A} .
2. **Server-Aided-Verification Queries:** At any time the adversary \mathcal{A} can make the server-aided-verification query (m_i, σ_i) . The challenger executes the protocol **SA-Verify** with \mathcal{A} as described in Figure 3.5.
3. **Output:** After all queries, \mathcal{A} will output a message m^* . Let Ω_{m^*} be the valid signature of m^* . As response, the challenger will choose a random element $\sigma^* = (\sigma_1^*, \sigma_2^*)$ in $\mathbb{G}_1^2 \setminus \Omega_{m^*}$, that is, σ^* is a random element in the invalid signature space of m^* .

The challenger then executes **SA-Verify** with \mathcal{A} as described in Figure 3.5. The challenger selects $d^* \in_R \mathbb{Z}_p$ and computes $\sigma_1'^* = \sigma_1^* \cdot g_1^{d^*}$. After that, $\sigma_1'^*$ and σ_2^* are sent to \mathcal{A} , who will return K_2^* and K_3^* as the response. We now show that $K_2^* = PK \cdot K_3^* \cdot K_1^{d^*}$ happens with probability $\frac{1}{p-1}$. The following analysis has the similar idea in the proof of Cramer-Shoup encryption scheme [CS98].

- (a) The element $\sigma_1'^*$ sent to \mathcal{A} does not constrain (σ_1^*, d^*) . This is due to the reason that given $\sigma_1'^*$, there are $(p-1)$ pairs (σ_{1i}, d_i) satisfy the equation $\sigma_1'^* = \sigma_{1i} \cdot g_1^{d_i}$ and (σ^*, d^*) chosen by the challenger is just a random one among these $p-1$ pairs. In other words, from the adversary's view, each (σ_{1i}, d_i) has the equal probability to be (σ^*, d^*) . To make our analysis more clear, we rewrite the the equation $\sigma_1'^* = \sigma_1^* \cdot g_1^{d^*}$ as

$$DL_{g_1} \sigma_1'^* = DL_{g_1} \sigma_1^* + d^* \quad (1)$$

- (b) Suppose \mathcal{A} returns K_2^* and K_3^* such that $K_2^* = PK \cdot K_3^* \cdot K_1^{d^*}$. We rewrite this equation as

$$DL_{e(g_1, g_1)} K_2^* = DL_{e(g_1, g_1)}(PK \cdot K_3^*) + d^* \quad (2)$$

If Equations (1) and (2) are linearly independent, then (σ_1^*, d^*) will be uniquely determined, which happens with probability $\frac{1}{p-1}$ as (σ_1^*, d^*) is randomly chosen from $p-1$ pairs from the view of the adversary.

Otherwise, Equations (1) and (2) are linearly dependent. This requires that $DL_{g_1} \sigma_1^* = DL_{e(g_1, g_1)}(PK \cdot K_3^*)$, that is, $e(\sigma_1^*, g_1) = PK \cdot K_3^*$. This means that σ_1^* is uniquely determined by K_3^* . As $\sigma_1^{I^*}$ sent to \mathcal{A} does not constrain σ_1^* , this happens also with probability $\frac{1}{p-1}$.

Therefore, the probability that **SA-Verify** will output **Valid** is $\frac{1}{p-1}$. This completes the proof of this theorem. \square

3.6 SAV- Σ Secure Against Strong Collusion and Adaptive Chosen Message Attacks

As we can see in Section 3.5.1, it is the challenger who chooses an invalid signature of the message m^* (where m^* is chosen by the adversary). This section considers strong collusion and adaptive chosen message attacks, where the adversary has the ability to choose the invalid signature under the message m^* . The concrete game is defined as follows.

Setup. The challenger \mathcal{C} runs the algorithm **ParamGen**, **KeyGen** and **SA-Verifier-Setup** to obtain system parameter $param$, one key pair (sk, pk) and **VString**. The adversary \mathcal{A} is given $param$ and (sk, pk) .

Queries. The adversary \mathcal{A} only needs to make **Server-Aided Verification Queries**. Proceeding adaptively, the adversary \mathcal{A} can make at most q_v such queries. The challenger \mathcal{C} responds each query in the same way as described in Definition 3.1.

Output. The adversary \mathcal{A} will output a message m^* and choose a random element σ^* in $\Omega \setminus \Omega_{m^*}$, where Ω_{m^*} denotes the set of valid signatures of m^* . We say \mathcal{A} wins the game if **SA-Verify** $(\mathcal{A}, \mathcal{C}^{(m^*, \sigma^*, pk, \text{VString})}) = \text{Valid}$.

We define **Strong-Soundness-SA-Verify-Adv \mathcal{A}** to be the probability that the adversary \mathcal{A} wins in the above game, taken over the coin tosses made by \mathcal{A} and the challenger.

Definition 3.4 *An adversary \mathcal{A} is said to (t, q_v, ε) -strongly-break the soundness of **SA-Verify** in a SAV- Σ if \mathcal{A} runs in time at most t , makes at most q_v server-aided verification queries, and **Strong-Soundness-SA-Verify-Adv \mathcal{A}** is at least ε . The **SA-Verify** in a SAV- Σ is (t, q_v, ε) -sound against strong collusion and adaptive chosen message attacks if there exists no adversary that (t, q_v, ε) -breaks it.*

Definition 3.5 *SAV- Σ is $(t, q_s, q_v, \varepsilon)$ -secure against strong collusion and adaptive chosen message attacks if Σ is (t, q_s, ε) -existentially unforgeable against adaptive chosen message attacks and its server-aided verification protocol **SA-Verify** is (t, q_v, ε) -sound against strong collusion and adaptive chosen message attacks.*

Remark 3.3 *Like the collusion and chosen messages attacks, the game defined in this section simulates a malicious signer acting as the server. It is called strong collusion and chosen message attacks since the adversary is allowed to choose the challenging message.*

3.6.1 SAV-BLS Secure Against Strong Collusion and Adaptive Chosen Message Attacks

We now give a server-aided verification protocol for BLS signature [BLS01], which is secure against the strong collusion and adaptive chosen message attacks.

1. **ParamGen, KeyGen, Sign, Verify**: these algorithms are the same as defined in Section 3.4.1.
2. **SA-Verifier-Setup**: Given the system parameter $(\mathbb{G}_1, \mathbb{G}_T, k, g_1, p, e, H)$, the verifier V computes $K_1 = e(g_1, g_1)$. The **VString** is K_1 .
3. **SA-Verify**: The verifier V and the server S interact with each other using the protocol described in Figure 3.4.

The verifier in SAV-BLS described above needs to compute one multiplication on \mathbb{G}_1 , one (fixed-base) exponentiation on \mathbb{G}_1 , one exponentiation on \mathbb{G}_1 , one multiplication on \mathbb{G}_T , one (fixed-based) exponentiation on \mathbb{G}_T , one exponentiation on

Figure 3.6: SA-Verify in SAV-BLS with Soundness II

Verifier (VString: K_1)	Server (param)
<p>Input: $(m, \sigma), pk, param$</p> <p>Compute: $r_1, r_2 \in_R \mathbb{Z}_p, \sigma' = \sigma^{r_1} \cdot g_1^{r_2}$</p> <p>$K_3 = e(H(m), pk)$</p> <p>Output: Valid, if $K_2 = K_3^{r_1} \cdot K_1^{r_2}$ Invalid, otherwise.</p>	$K_2 = e(\sigma', g_1)$

\mathbb{G}_T and one pairing. This is also less than the computational cost required in the original verify algorithm.

Theorem 3.6 *The server-aided verification protocol described in Figure 3.6 is $(t, q_v, \frac{1}{p})$ -strong-sound against collusion and adaptive chosen message attacks.*

Proof: We prove that the adversary's probability to prove an invalid signature as valid is $\frac{1}{p}$, without any complexity assumption.

1. **Setup:** The challenger starts by choosing the secret key $sk = x \in_R \mathbb{Z}_p$, sets the public key as $pk = g_1^x$ and computes $K_1 = e(g_1, g_1)$, then returns $(\mathbb{G}_1, \mathbb{G}_T, k, g_1, p, e, H)$ and (sk, pk) to \mathcal{A} .
2. **Server-Aided Verification Queries:** At any time the adversary \mathcal{A} can make the server-aided verification query (m_i, σ_i) . The challenger executes the protocol **SA-Verify** with \mathcal{A} as described in Figure 3.6.
3. **Output:** After all queries, \mathcal{A} will output a message m^* along with a random element σ^* in $\mathbb{G}_1 \setminus \{H(m^*)^{sk}\}$, that is, σ^* is a random element in the invalid signature space of m^* .

The challenger then executes **SA-Verify** with \mathcal{A} as described in Figure 3.6. The challenger selects $r_1^*, r_2^* \in_R \mathbb{Z}_p$ and computes $\sigma'^* = \sigma^{*r_1^*} \cdot g_1^{r_2^*}$. After that,

σ'^* is sent to \mathcal{A} , who will return K_2^* as the response. We now show that $K_2^* = K_3^{*r_1^*} \cdot K_1^{r_2^*}$ happens with probability $\frac{1}{p}$. The following analysis uses the similar technique in the proof of Cramer-Shoup encryption scheme [CS98].

- (a) The element σ'^* sent to \mathcal{A} does not constrain the distribution of (r_1^*, r_2^*) . This is due to the reason that given σ'^* , there are p pairs (r_{1i}, r_{2i}) satisfy the equation $\sigma'^* = \sigma_i^{r_{1i}} \cdot g_1^{r_{2i}}$ and (r_1^*, r_2^*) chosen by the challenger is just a random one among these p pairs. In other words, from the adversary's view, each (r_{1i}, r_{2i}) has the equal probability to be (r_1^*, r_2^*) . To make our analysis more clear, we rewrite the the equation $\sigma'^* = \sigma^{*r_1^*} \cdot g_1^{r_2^*}$ as

$$DL_{g_1} \sigma'^* = r_1^* \cdot DL_{g_1} \sigma^* + r_2^* \quad (1)$$

- (b) Suppose \mathcal{A} returns K_2^* such that $K_2^* = K_3^{*r_1^*} \cdot K_1^{r_2^*}$. We rewrite this equation as

$$DL_{e(g_1, g_1)} K_2^* = r_1^* \cdot DL_{e(g_1, g_1)} K_3^* + r_2^* \quad (2)$$

As required, σ^* is an invalid signature of m^* , i.e. $DL_{g_1} \sigma^* \neq DL_{e(g_1, g_1)} K_3^*$. It follows that Equations (1) and (2) are linearly independent and (r_1^*, r_2^*) will be uniquely determined. This happens with probability $\frac{1}{p}$ as (r_1^*, r_2^*) is randomly chosen from p pairs from the view of the adversary.

Therefore, the probability that **SA-Verify** will output **Valid** is $\frac{1}{p}$. This completes the proof of this theorem. \square

3.7 Conclusion

We formally defined the existential unforgeability of server-aided verification signatures and generalized the existing security requirements in server-aided verification signatures. We analyzed the Girault-Lefranc scheme from Asiacypt 2005 and proposed the first server-aided verification BLS signature (whose existential unforgeability is proven in the random oracle model) and server-aided verification Waters signature (whose existential unforgeability does not rely on the random oracle model). Another contribution of this chapter is the formal security definition of server-aided verification signatures under collusion attacks and strong collusion attacks. Concrete constructions secure against those attacks were also presented.

Signature schemes with the server-aided verification protocols proposed in this chapter reduce the computational cost at the verifier side and can be applied to situations with power-constrained devices.

Chapter 4

Generic Constructions of Signatures and Encryption in Certificate-based PKC

Certificate-based public key cryptography is proposed to ease the certificate management problem in traditional PKI. The essential idea is to use a third party to generate a certificate, which not only ensures the public key authenticity but is also necessary for signature signing and decryption. This chapter presents several new results of digital signatures and encryption in certificate-based public key cryptography. Part of this chapter was published in the Journal of Universal Computer Science [WMSH09].

4.1 Introduction

In a public-key cryptosystem, each user has a pair of keys: public key and private key. The public key is usually published and publicly accessible, while the private key is kept secret by the owner. The central problem in a public key system is to prove that a public key is genuine and authentic, and has not been tampered with or replaced by a malicious third party. The usual approach to ensure the authenticity of a public key is to use a certificate. A (digital) certificate is a signature of a trusted certificate authority (CA) that binds together the identity of an entity A , its public key PK and other information. This kind of systems is referred to as public key infrastructure (PKI). The PKI however is generally considered to be costly to use and manage.

Shamir [Sha84] introduced the concept of identity-based public key cryptography (or, ID-PKC for short), where the original motivation is to ease certificate management in the e-mail system. A user's public key in ID-PKC is some unique information about the identity of the user (e.g., email address). The private key in ID-PKC is generated by a trusted third party called Private Key Generator (PKG)

who holds a master key. Thus, key escrow is an inherent problem in this kind of ID-PKC (e.g., [Sha84, BF01]), as the PKG has any user's private key. The key escrow problem can be partially solved by the introduction of multiple PKGs and the use of threshold techniques, which requires extra communications and infrastructures.

Al-Riyami and Paterson proposed a new paradigm called certificateless public key cryptography [ARP03] (or, CL-PKC for short), where the original motivation is to find a public key system that does not require the use of certificates and does not have the key escrow problem. Each entity in CL-PKC holds two secrets: a secret value and a partial private key. The secret value SV is generated by the entity itself, while a third party Key Generating Center (KGC), holding a master key, generates the partial private key PPK from the user's identity information¹. The entity's actual private key is the output of some function with the input SV and PPK . In this way, KGC does not know the actual private key and the key escrow problem is eliminated. The entity can use the actual private key to generate the public key, which is no longer only computed from the identity. This makes the certificateless system non-identity-based. The entity's public key could be available to other entities by transmitting it along with messages (for example, in a signing application) or by placing it in a public directory (this would be more appropriate for an encryption setting). However, there is no certificate to ensure the authenticity of the entity's public key in CL-PKC. Therefore, it is necessary to assume that an adversary is able to replace the entity's public key with a false key of its choice, which is also known as *key replacement attack* [HSMZ05]. One assumption in CL-PKC is that KGC never mounts the key replacement attack. In the traditional PKI, however, one does not need to make the same assumption on the third party CA, who if replaces the entity's public key with a false key of its choice, can be implicitly proven due to the existence of the certificate for that false key.

In Eurocrypt 2003, Gentry [Gen03] introduced the notion of certificate-based encryption. As in the traditional PKI, each client generates its own public/private key pair and requests a certificate from the CA. The difference is that, a certificate in the certificate-based cryptography, or more generally, a signature from the third party acts not only as a certificate (as in the traditional PKI) but also as a decryption key

¹In Section 5.1 of [ARP03], the authors sketched an alternative partial private key generation technique. In this chapter, when we mention a cryptographic protocol in CL-PKC, we mean it is a protocol with the classic private key generation technique used in Section 4.1 of [ARP03], which has been adopted by most researchers in CL-PKC.

(as in ID-PKC and CL-PKC). The sender can encrypt a message without obtaining explicit information other than the recipient's public key and the parameters of CA. To decrypt a message, a keyholder needs both its secret key and an up-to-date certificate from its CA (or a signature from an authority). Therefore, CA does not need to make the certificate status information available among the whole system, and only needs to contact the certificate holder for revocation and update. As the sender is not required to verify the certificate of the recipient's public key, the sender could be duped to encrypt messages with an uncertified public key. This could be due to the recipient has not yet had his/her public key certified, or the encryption key that the sender holds is not the recipient's authentic public key. In this sense, certificate-based encryption works in a similar way to certificateless encryption, but the difference is that certificates do exist in certificate-based encryption.

Certificate-based cryptography was introduced to solve the certificate management problem in the traditional PKI, but only in the scenario of encryption. The notion of certificate-based encryption was extended to certificate-based signature in [KPH04, LHM⁺07]. However, as mentioned in [Gen03], if we only consider signing and verification signatures in a public key cryptosystem, then the certificate management problem is not as challenging as in the scenario of encryption and decryption. For example, the signer can send its public key and the proof of certificate status to the verifier simultaneously with its signature, thus the verifier can obtain the certificate without referring to a public directory or issuing a third-party query to CA. This, however, will require more bandwidth for signature transmitting. Public key cryptosystems like ID-PKC [Sha84, BF01] and CL-PKC [ARP03] can eliminate the certificate management problem as one can directly use the entity A 's public key to verify signatures, without checking the certificate of A 's public key. However, this is achieved at the cost of assuming certain trust on the authority, who is able to impersonate any user in an undetectable way. In certificate-based cryptosystem, the certificate management problem can be eased in a different way. To generate valid certificate-based signatures of a user with the identity information ID and the public key PK , one needs two pieces of secret information, namely a valid certificate of (ID, PK) and the secret key of PK . In other words, a valid certificate-based signature ensures the existence of a valid certificate. Thus, the signer does not need to send the certificate along with the message and the signature. This is achieved with the cost of more computational operations in signature verification, which implies the verification of the certificate. If one replaces PK with PK' and

Table 4.1: Comparison of CL Cryptography and CB Cryptography

	CL Cryptography	CB Cryptography
Certificate	No	Yes
Certificate manage problem	No	Yes
Trust Level [Gir91]	2	3

generates a valid signature under ID and PK' , he/she must have a certificate of (ID, PK') . This can prove that the third party CA is dishonest, as there is only one party with the ability to generate certificates. Therefore, the third party in certificate-based signatures has the Trust Level 3 in the definition in [Gir91], which is similar as CA in the traditional PKI and a few constructions of identity-based signatures [BNN04, GHK06]. To summarize, (1) The authority in certificate-based signatures and traditional PKI-based signatures is at Trust Level 3 in the definition given in [Gir91], which is higher than the authority in the ID-PKC and the CL-PKC, and (2) To ease the problem of certificate management, certificate-based signatures consume (in general) less bandwidth in signature transmitting but might require more computational cost than traditional-PKI-based signatures. The table 4.1 below gives a brief comparison of certificateless Cryptography and certificate-based Cryptography.

4.1.1 Related Work

Kang, Park and Hahn proposed the notion and the first construction of certificate-based signatures in [KPH04], by extending the idea of certificate-based encryption in [Gen03]. That is, to generate a valid signature under the public key PK , the entity needs to know both the corresponding private key SK and the up-to-date certificate of PK . To verify a claimed signature, one only needs the signer's public key and the parameter of CA (particularly, no need to check the certificate of that public key). As the verifier is not required to check the certificate about a claimed public key, key replacement attacks also exist in certificate-based cryptography. Key replacement attacks in certificate-based signatures were first addressed in [KPH04] and formally defined in [LHM⁺07]. As introduced in [LHM⁺07], adversaries in certificate-based signatures can be divided into two types: **CB- \mathcal{A}_I** and **CB- \mathcal{A}_{II}** . **CB- \mathcal{A}_I** can replace any entity's public key PK with a new public key PK' chosen by itself, and is trying to forge a valid signature under PK' whose certificate is not

available to $\mathbf{CB}\text{-}\mathcal{A}_I$. $\mathbf{CB}\text{-}\mathcal{A}_{II}$ has the knowledge of CA's master key and thus can generate the certificate for any user. $\mathbf{CB}\text{-}\mathcal{A}_{II}$ is trying to forge a valid signature under an entity's authentic public key PK (that is, PK is chosen by that entity), whose private key is not available to $\mathbf{CB}\text{-}\mathcal{A}_{II}$. In addition to the security models, a certificate-based signature scheme secure against key replacement attacks was also proposed in [LHM⁺07]. Very recently, Liu, Baek, Susilo and Zhou proposed two new certificate-based signature schemes [LBSZ08]. The first one does not require any pairing operation and the security of their second scheme can be proven without random oracles. Some variants of certificate-based signatures (e.g., certificate-based proxy signature [KPH04] and certificate-based linkable ring signature [ALSY07]) have also been proposed.

Since its seminal introduction in [Gen03], several concrete certificate-based encryption schemes have been proposed (e.g., [LZ08, MR06]). The relationship between certificate-based encryption and certificateless encryption has been investigated in [ARP05], where a generic construction of certificate-based encryption from certificateless encryption was introduced. But Kang and Park [KP05] pointed out a flaw in the security proof of [ARP05], and this leaves the security of the construction in [ARP05] questionable.

4.1.2 Motivations and Contributions

As mentioned in [ARP03], certificate-based cryptography and certificateless cryptography are quite similar and there could be a possible method to convert a certificateless cryptographical protocol to a certificate-based cryptographical protocol. In particular, there are four similarities in certificateless public key cryptography and certificate-based public key cryptography.

1. In both public key cryptosystems, signature signing (or, decryption) requires two pieces of information. In certificateless cryptography, one needs a partial private key and a secret value of a public key to produce a valid signature or perform a correct decryption. Similarly, in certificate-based cryptography, one needs the certificate and the private key of a public key to generate valid signatures or decrypt ciphertexts.
2. The partial private key is generated by KGC in certificateless cryptography, and the certificate is generated by Certifier in certificate-based cryptography.

3. The secret value is chosen by the user in certificateless cryptography, and the private key in certificate-based cryptography is also chosen by the user.
4. In both notions, explicit verification of the authenticity of a public key is not required when one verifies signatures or encrypts messages.

Motivated by those similarities, we believe that certificate-based signatures and certificateless signatures (resp. certificate-based encryption and certificateless encryption) are closely related, and the investigation of the relationship between those two notions is worthwhile. The contributions of this chapter are threefold.

1. *New Security Models of Certificate-based Signatures*

A reasonable and elaborated security model is necessary for constructing provably secure cryptographic protocols. For example, although the key replacement attack has been widely accepted in certificateless cryptography, there is no consensus on the precise meaning of that term in the early research of certificateless cryptography and several certificateless signature schemes have been broken [ARP03, GS05, HWZD06, HSMZ05, Par06, YL04, ZF06]. Although some security models [KPH04, LHM⁺07] have been proposed so far, the security definition of certificate-based signatures is not satisfactory, especially in the exact meaning of key replacement attacks. In this chapter, we provide elaborated definitions of certificate-based signatures, which will allow us to establish a systematic approach for constructing and proving secure certificate-based signature schemes. Our definitions are inspired by and modified from the security notions in certificateless signatures. This is not only because certificateless signatures and certificate-based signatures are analogous in many ways, but also due to the fact that security definitions of certificateless signatures have been formalized recently.

2. *A Generic Construction of Certificate-based Signatures from Certificateless Signatures*

After giving new security models of certificate-based signatures, we describe a generic construction of certificate-based signatures which is secure in the proposed models. We show how to design a certificate-based signature scheme from a certificateless signature scheme, by treating partial private keys in certificateless signatures as certificates in certificate-based signatures. Our method can be used to build certificate-based signature schemes secure (in the random oracle model) against any type of adversaries defined in this chapter, assuming that the underlying certificateless signature schemes satisfy certain security notions. We also give two concrete instances

of our generic construction and compare them with other existing ones.

3. *A Generic Construction of Certificate-based Encryption from Certificateless Encryption*

We extend the generic construction of certificate-based signatures to certificate-based encryption. In order to further classify the potential attacks on certificate-based encryption, we provide an elaborate security model for certificate-based encryption. The security model is inspired by and modified from the security notions of both certificateless public key encryption [Den08] and certificate-based encryption [Gen03]. Under the new security model, we provide a new generic construction of certificate-based encryption from certificateless encryption, which can be viewed as an analogue of our generic construction of certificate-based signatures. Assuming that the underlying certificateless encryption schemes satisfy certain security notions, our method can be applied to build certificate-based encryption schemes secure (in the random oracle model) against any type of adversaries defined in this chapter.

Organization of this Chapter

The outline of a certificate-based signature (CBS) scheme is presented in the next section, and the description of the oracles accessible by the adversaries is given in Section 4.3. We then redefine the security of CBS against different types of attacks in Section 4.4. The generic construction of certificate-based signatures from certificateless signatures is proposed in Section 4.5. Section 4.6 demonstrates the application of our generic construction by showing two concrete certificate-based signature schemes.

The outline of a certificate-based encryption (CBE) scheme is given in Section 4.7. After that, we define the security of CBE against different types of attacks in Section 4.8. This is followed by a generic construction of certificate-based encryption from certificateless encryption in Section 4.9. Section 4.10 describes a concrete instance of our generic construction of certificate-based encryption. Finally, Section 4.11 concludes this chapter.

4.2 Certificate-based Signatures

In this section, we will first review the definitions of certificate-based signatures. After that, we will describe oracles used in our security model.

4.2.1 Syntax of Certificate-Based Signatures

In a certificate-based cryptosystem, a certificate generator, which is called as the “certifier”, will first generate the system parameter and a master public/private key pair. The certifier will use that key pair to generate certificates for users in the system. Users then will generate their own public/secret key pairs and contact the certifier to obtain the corresponding certificates. A user can use the secret key and the certificate to generate a signature on a message. In this case, that user is also called as the signer. A signature recipient is called as the verifier if he/she performs the signature verification.

A certificate-based signature (CBS) scheme consists of the following five algorithms:

1. $\text{CB-Setup}(1^k) \rightarrow (CB\text{-}msk, CB\text{-}mpk, CB\text{-}params)$. By taking as input a security parameter 1^k , the certifier runs the algorithm **CB-Setup** to generate the certifier’s master secret key $CB\text{-}msk$, master public key $CB\text{-}mpk$ and the system parameter $CB\text{-}params$. $CB\text{-}params$ includes the description of a string space Γ , which can be any subset of $\{0, 1\}^*$.
2. $\text{CB-UserKeyGen}(CB\text{-}mpk, CB\text{-}params, \text{ID}) \rightarrow (CB\text{-}SK_{\text{ID}}, CB\text{-}PK_{\text{ID}})$. The user with the identity information ID runs the algorithm **CB-UserKeyGen** to generate the user ID ’s secret/public key pair $(CB\text{-}SK_{\text{ID}}, CB\text{-}PK_{\text{ID}}) \in \mathcal{SK}^{CB} \times \mathcal{PK}^{CB}$, by taking as input $CB\text{-}mpk$ and $CB\text{-}params$. Here, \mathcal{SK}^{CB} denotes the set of valid secret key values and \mathcal{PK}^{CB} denotes the set of valid public key values.
3. $\text{CB-CertGen}(CB\text{-}msk, CB\text{-}mpk, CB\text{-}params, \text{ID}, CB\text{-}PK_{\text{ID}}) \rightarrow Cert_{\text{ID}}$. The certifier runs the algorithm **CB-CertGen** to generate the certificate $Cert_{\text{ID}}$, by taking as input $CB\text{-}msk$, $CB\text{-}mpk$, $CB\text{-}params$, ID and its public key $CB\text{-}PK_{\text{ID}}$.
4. $\text{CB-Sign}(m, CB\text{-}params, CB\text{-}mpk, \text{ID}, Cert_{\text{ID}}, CB\text{-}SK_{\text{ID}}, CB\text{-}PK_{\text{ID}}) \rightarrow CB\text{-}\sigma$. The prospective signer runs the algorithm **CB-Sign** to generate the signature $CB\text{-}\sigma$, by taking as input a message m , $CB\text{-}params$, $CB\text{-}mpk$, the user’s identity ID , its $Cert_{\text{ID}}$ and key pair $(CB\text{-}SK_{\text{ID}}, CB\text{-}PK_{\text{ID}})$.
5. $\text{CB-Verify}(m, CB\text{-}\sigma, CB\text{-}mpk, CB\text{-}params, \text{ID}, CB\text{-}PK_{\text{ID}}) \rightarrow \{true, false\}$. Anyone can run the algorithm **CB-Verify** to check the validity of the signature.

By taking as input a message/signature pair $(m, CB-\sigma)$, ID , $CB-PK_{ID}$, $CB-mpk$, $CB-params$, this algorithm outputs *true* if $CB-\sigma$ is ID 's valid signature on m . Otherwise, this algorithm outputs *false*.

Correctness. Signatures generated by the algorithm $CB-Sign$ can pass through the verification in $CB-Verify$. That is,

$$CB-Verify(m, CB-Sign(m, CB-params, CB-mpk, ID, Cert_{ID}, CB-SK_{ID}, CB-PK_{ID}), CB-mpk, CB-params, ID, CB-PK_{ID}) = true.$$

Remark 4.1 *A certificate-based signature proves that one has a valid certificate (certifier's signature) of a public key and the associated private key. In this sense, certificate-based signatures are similar to aggregate signatures [BGLS03].*

4.3 Adversaries and Oracles

We now describe the oracles which will be used in the security model of certificate-based signatures in this chapter. We first give a brief description of adversaries in certificate-based signatures. Formal definitions of these adversaries will be given in Section 4.4.

The essential security of a certificate-based signature scheme requires that one can generate a valid signature under the public key $CB-PK_{ID}$ *if and only if* having the knowledge of both $Cert_{ID}$ and $CB-SK_{ID}$. In other words, one cannot generate a valid signature with only $Cert_{ID}$ or $CB-SK_{ID}$. As introduced in [LHM⁺07], adversaries in certificate-based signatures can be divided into two types: $CB-A_I$ and $CB-A_{II}$. Type I adversary $CB-A_I$ simulates the scenario where the adversary (anyone except the certifier) is allowed to replace public keys of any entities, but is not allowed to obtain the target user's certificate $Cert_{ID}$. Type II adversary $CB-A_{II}$ simulates a malicious certifier who is able to produce certificates but is assumed not to replace the target user's public key. We will use the following oracles to simulate potential attacking scenarios. In the remainder of this chapter, we write $\alpha \leftarrow \beta$ to denote the algorithmic action of assigning the value β to α .

1. $\mathcal{O}^{CB-UserCreate}$: This oracle receives an input $ID \in \Gamma$ and outputs the public key of user ID . It maintains two lists $L1^{PK}$ and $L2^{PK}$, which are initially empty and are used to record the information for each user ID . $L1^{PK} = \{(ID, CB-SK_{ID}, CB-PK_{ID})\}$ provides the information about user ID 's secret key and public

key when it is created. $L2^{PK} = \{(\text{ID}, \text{CB-}\overline{\text{PK}}_{\text{ID}})\}$ provides the information of ID's current public key, which is denoted as $\text{CB-}\overline{\text{PK}}_{\text{ID}}$.

- (a) For a fresh input ID, the oracle runs the algorithms CB-UserKeyGen to obtain the secret key CB-SK_{ID} and public key CB-PK_{ID} . It then adds $(\text{ID}, \text{CB-SK}_{\text{ID}}, \text{CB-PK}_{\text{ID}})$ to $L1^{PK}$ and $(\text{ID}, \text{CB-}\overline{\text{PK}}_{\text{ID}})$ to $L2^{PK}$ where $\text{CB-}\overline{\text{PK}}_{\text{ID}} \leftarrow \text{CB-PK}_{\text{ID}}$. After that, it outputs CB-PK_{ID} . In this case, ID is said to be *created*. Here we assume that other oracles (which will be defined later) only respond to the identity which has been created.
 - (b) Otherwise, ID has already been created. The oracle will search ID in $L1^{PK}$ and return CB-PK_{ID} as the output.
2. $\mathcal{O}^{\text{CB-PKReplace}}$: For a public key replacement query $(\text{ID}, \text{CB-PK}) \in \Gamma \times \mathcal{PK}^{\text{CB}}$, this oracle finds the user ID in the list $L2^{PK}$, sets $\text{CB-}\overline{\text{PK}}_{\text{ID}} \leftarrow \text{CB-PK}$ and updates the corresponding pair with $(\text{ID}, \text{CB-}\overline{\text{PK}}_{\text{ID}})$.
 3. $\mathcal{O}^{\text{CB-Corruption}}$: This oracle takes as input a query ID. It browses the list $L1^{PK}$ and outputs the secret key CB-SK_{ID} .
 4. $\mathcal{O}^{\text{CB-CertGen}}$: For a certificate request for $(\text{ID}, \text{CB-PK}) \in \Gamma \times \mathcal{PK}^{\text{CB}}$, this oracle runs the algorithm CB-CertGen and returns the certificate for $(\text{ID}, \text{CB-PK})$.
 5. $\mathcal{O}^{\text{CB-Sign}}$: Considering different levels of the signing power the challenger may have, this oracle can be further divided into following three types:
 - (a) $\mathcal{O}^{\text{CB-NormalSign}}$: This oracle takes as input a query (ID, m) , and outputs a signature $\text{CB-}\sigma$ such that $\text{true} = \text{CB-Verify}(m, \text{CB-}\sigma, \text{CB-params}, \text{ID}, \text{CB-PK}_{\text{ID}}, \text{CB-mpk})$. Here CB-PK_{ID} is ID's public key in the list $L1^{PK}$.
 - (b) $\mathcal{O}^{\text{CB-StrongSign}}$: This oracle takes as input a query $(\text{ID}, m, \text{coin})$, where m denotes the message to be signed, and $\text{coin} \in \{1, 2\}$. It acts differently according to the value of coin . If $\text{coin} = 1$, this oracle works the same as $\mathcal{O}^{\text{CB-NormalSign}}$. Otherwise $\text{coin} = 2$, this oracle first checks the list $L1^{PK}$ and $L2^{PK}$ to obtain ID's original public key CB-PK_{ID} and ID's current public key $\text{CB-}\overline{\text{PK}}_{\text{ID}}$. If $\text{CB-}\overline{\text{PK}}_{\text{ID}} = \text{CB-PK}_{\text{ID}}$, this oracle works as same as $\mathcal{O}^{\text{CB-NormalSign}}$. Otherwise, $\mathcal{O}^{\text{CB-StrongSign}}$ will ask the adversary to supply the secret key $\text{CB-}\overline{\text{SK}}_{\text{ID}}$ corresponding to $\text{CB-}\overline{\text{PK}}_{\text{ID}}$. After

Table 4.2: Sign Oracles

Public Keys	Normal Sign	Strong Sign	Super Sign
Original public keys	✓	✓	✓
Replaced public keys	×	✓ with conditions 4.2	✓

that, this oracle uses $CB-\overline{SK}_{ID}$ and the certificate for $(ID, CB-\overline{PK}_{ID})$ to generate the signature $CB-\sigma$, which will be returned as the answer.

- (c) $\mathcal{O}^{CB-SuperSign}$: For a query (ID, m) , this oracle first finds ID 's current public key $CB-\overline{PK}_{ID}$ in $L2^{PK}$. This oracle then outputs a signature σ such that $true = CB-Verify(m, \sigma, CB-params, ID, CB-\overline{PK}_{ID}, CB-mpk)$.

Remark 4.2 *The adversary must provide the corresponding secret key when issuing queries to $\mathcal{O}^{CB-StrongSign}$.*

Table 4.2 shows the differences among the three sign oracles. The power of adversaries can be easily classified by observing which sign oracle they can issue queries to. Clearly, adversaries who can request $\mathcal{O}^{CB-SuperSign}$ are the most powerful ones, while the weakest ones are those who can only request $\mathcal{O}^{CB-NormalSign}$.

Remark 4.3 *A Type II adversary $CB-A_{II}$, who simulates the malicious certifier, is not allowed to make any requests to $\mathcal{O}^{CB-CertGen}$.*

4.4 Security Models of Certificate-based Signatures

In this section, we will define security models of certificate-based signatures. Our models follow the standard methods: each security notion is defined by the game between the adversary and the challenger, which consists of several oracles defined in Section 4.3. In our definition, the notation $\{Q_1, Q_2, \dots, Q_n\} \nrightarrow \{\mathcal{O}^1, \mathcal{O}^2, \dots, \mathcal{O}^n\}$ denotes that “No query $Q \in \{Q_1, Q_2, \dots, Q_n\}$ can be submitted to any oracle $\mathcal{O} \in \{\mathcal{O}^1, \mathcal{O}^2, \dots, \mathcal{O}^n\}$ ”. A $(t, q_{UC}, q_{PKR}, q_C, q_{CG}, q_S)$ adversary refers to the adversary who runs in polynomial time t , makes at most q_{UC} queries to $\mathcal{O}^{CB-UserCreate}$, q_{PKR} queries to $\mathcal{O}^{CB-PKReplace}$, q_C queries to $\mathcal{O}^{CB-Corruption}$, q_{CG} queries to $\mathcal{O}^{CB-CertGen}$, q_S queries to $\mathcal{O}^{CB-Sign} \in \{\mathcal{O}^{CB-NormalSign}, \mathcal{O}^{CB-StrongSign}, \mathcal{O}^{CB-SuperSign}\}$.

The definition in this section is inspired by [HMS⁺07], which provides a new classification of potential adversaries against certificateless signatures. The security models in [HMS⁺07] not only include previous security definitions of certificateless signatures, but also introduce new types of adversaries. Following the definitions in [HMS⁺07], we classify the potential adversaries in certificate-based signatures according to their attack power. They are Normal Adversary, Strong Adversary and Super Adversary. Combined with the known type I adversary and type II adversary in certificate-based signatures, we now define the security of certificate-based signatures in different attack scenarios and relate them to prior definitions.

4.4.1 Security Against Normal Type I Adversary

We first define the Normal Type I adversary in certificate-based signatures, which is denoted as **Normal-CB- \mathcal{A}_I** . The essential attacking scenario of Normal-CB- \mathcal{A}_I is that the adversary can obtain some message/signature pairs $(m_i, CB-\sigma_i)$ which are generated by the target user using its own secret key and certificate. Our definition described below is inspired by and modified from the definition of Normal Type I adversary against certificateless signatures in [HMS⁺07].

Initial: The challenger runs the algorithm **CB-Setup**, returns $CB\text{-params}$ and $CB\text{-mpk}$ to \mathcal{A}_I .

Queries: In this phase, \mathcal{A}_I can adaptively make requests to $\mathcal{O}^{CB\text{-UserCreate}}$, $\mathcal{O}^{CB\text{-PKReplace}}$, $\mathcal{O}^{CB\text{-Corruption}}$, $\mathcal{O}^{CB\text{-CertGen}}$, $\mathcal{O}^{CB\text{-NormalSign}}$.

Output: After all queries, \mathcal{A}_I outputs a forgery $(m^*, CB-\sigma^*, ID^*)$. Let $CB\text{-}\overline{PK}_{ID^*}$ be the current public key of ID^* in $L2^{PK}$.

Restrictions: We say \mathcal{A}_I wins the game if the forgery satisfies the following requirements:

1. $true = CB\text{-Verify}(m^*, CB-\sigma^*, CB\text{-params}, ID^*, CB\text{-}\overline{PK}_{ID^*}, CB\text{-mpk})$;
2. $(ID^*, m^*) \not\rightarrow \mathcal{O}^{CB\text{-NormalSign}}$;
3. $(ID^*, CB\text{-}\overline{PK}_{ID^*}) \not\rightarrow \mathcal{O}^{CB\text{-CertGen}}$; and
4. $ID^* \not\rightarrow \mathcal{O}^{CB\text{-Corruption}}$.

The success probability that an adaptive chosen message and chosen identity adversary Normal-CB- \mathcal{A}_I wins the above game is denoted as $Succ_{\mathcal{A}_I, normal}^{cma, cida}$. We say

a certificate-based signature scheme is secure against a $(t, q_{UC}, q_{PKR}, q_C, q_{CG}, q_S)$ Normal-CB- \mathcal{A}_I if $Succ_{\mathcal{A}_I, normal}^{cma, cida}$ is negligible.

Remark 4.4 *Our definition is similar to that in [LHM⁺07], but with two improvements. Firstly, we allow the adversary to replace any user's public key, while the adversary in [LHM⁺07] can only replace the target user's public key. The other improvement is that the adversary in our model is allowed to obtain certificates of $(ID, CB-PK)$ s chosen by itself. This is different from the adversary in [LHM⁺07] who can only obtain certificates of original public keys generated by the challenger.*

4.4.2 Security Against Strong Type I Adversary

In this section, we boost the attack power of Normal Type I adversary and define the Strong Type I adversary: **Strong-CB- \mathcal{A}_I** . Strong-CB- \mathcal{A}_I is more powerful than Normal-CB- \mathcal{A}_I in the sense that Strong-CB- \mathcal{A}_I can access the oracle $\mathcal{O}^{CB-StrongSign}$. Apart from that, Strong-CB- \mathcal{A}_I is allowed to corrupt the target user ID^* 's original secret key. The attacking scenario is similar to those in certificateless signatures defined in [HWZD06, ZWXF06], and is formally defined as below.

The game between the challenger and a Strong-CB- \mathcal{A}_I is very similar to that defined in Section 4.4.1, but with two differences:

In the phase **Queries**, Strong-CB- \mathcal{A}_I have access to $\mathcal{O}^{CB-StrongSign}$ rather than $\mathcal{O}^{CB-NormalSign}$,

In **Restrictions**, $(ID^*, m^*) \rightarrow \mathcal{O}^{CB-StrongSign}$ and ID^* can appear as a query to $\mathcal{O}^{CB-Corruption}$.

The success probability that an adaptive chosen message and chosen identity adversary Strong-CB- \mathcal{A}_I wins the above game is denoted as $Succ_{\mathcal{A}_I, strong}^{cma, cida}$. We say a certificate-based signature scheme is secure against a $(t, q_{UC}, q_{PKR}, q_C, q_{CG}, q_S)$ Strong-CB- \mathcal{A}_I if $Succ_{\mathcal{A}_I, strong}^{cma, cida}$ is negligible.

4.4.3 Security Against Super Type I Adversary

In this section, we will define the Super Type I adversary, which is denoted as **Super-CB- \mathcal{A}_I** . Super-CB- \mathcal{A}_I is more powerful than Strong-CB- \mathcal{A}_I (and hence, more powerful than Normal-CB- \mathcal{A}_I) in the sense that Super-CB- \mathcal{A}_I has access to $\mathcal{O}^{CB-SuperSign}$. That is, Super-CB- \mathcal{A}_I is allowed to obtain a valid signature under the

public key chosen by itself without providing the corresponding secret key, which makes it the strongest Type I adversary. This is similar to the Super Type I adversary in certificateless signatures defined in [HMS⁺07].

The game between the challenger and a Super-CB- \mathcal{A}_I is very similar to that defined in Section 4.4.2, but with two differences:

In the phase **Queries**, Super-CB- \mathcal{A}_I is allowed to have access to $\mathcal{O}^{\text{CB-SuperSign}}$ rather than $\mathcal{O}^{\text{CB-StrongSign}}$,

In **Restrictions**, $(ID^*, m^*) \rightarrow \mathcal{O}^{\text{CB-SuperSign}}$.

The success probability of an adaptively chosen message and chosen identity adversary Super-CB- \mathcal{A}_I wins the above game is denoted as $Succ_{\mathcal{A}_I, super}^{cma, cida}$. We say a certificate-based signature scheme is secure against a $(t, q_{UC}, q_{PKR}, q_C, q_{CG}, q_S)$ Super-CB- \mathcal{A}_I if $Succ_{\mathcal{A}_I, super}^{cma, cida}$ is negligible.

4.4.4 Security Against Type II Adversary

In certificate-based signatures, a type II adversary **CB- \mathcal{A}_{II}** simulates the certifier who is equipped with the master secret key and might engage in adversarial activities like eavesdropping on signatures and making signing queries. Similar to the type I adversary, **CB- \mathcal{A}_{II}** could be also classified into **Normal-CB- \mathcal{A}_{II}** , **Strong-CB- \mathcal{A}_{II}** , **Super-CB- \mathcal{A}_{II}** , which has access to $\mathcal{O}^{\text{CB-NormalSign}}$, $\mathcal{O}^{\text{CB-StrongSign}}$, $\mathcal{O}^{\text{CB-SuperSign}}$, respectively. However, there is no need to particularly define **Strong-CB- \mathcal{A}_{II}** . $\mathcal{O}^{\text{CB-StrongSign}}$ can answer queries either by using $\mathcal{O}^{\text{CB-NormalSign}}$ (then $\mathcal{O}^{\text{CB-StrongSign}}$ is the same as $\mathcal{O}^{\text{CB-NormalSign}}$), or signing the message with the corresponding secret key provided by the adversary. Note that, **CB- \mathcal{A}_{II}** has the master secret key, and thus can calculate any user's certificate. If he has the secret key as well, **CB- \mathcal{A}_{II}** can generate the signature by himself and $\mathcal{O}^{\text{CB-StrongSign}}$ becomes useless. Therefore, for a type II adversary **CB- \mathcal{A}_{II}** , it is sufficient to define only two types of adversaries, namely **Normal-CB- \mathcal{A}_{II}** and **Super-CB- \mathcal{A}_{II}** . The definition of those two types of adversaries is described as follows.

Initial: The challenger runs the algorithm **CB-Setup** and returns the system parameters *CB-params*, master secret key *CB-msk* and master public key *CB-mpk* to \mathcal{A}_{II} .

Queries: \mathcal{A}_{II} can adaptively make requests to $\mathcal{O}^{\text{CB-UserCreate}}$, $\mathcal{O}^{\text{CB-PKReplace}}$, $\mathcal{O}^{\text{CB-Corruption}}$ and $\mathcal{O}^{\text{CB-Sign}}$, where $\mathcal{O}^{\text{CB-Sign}} \in \{\mathcal{O}^{\text{CB-NormalSign}}, \mathcal{O}^{\text{CB-SuperSign}}\}$.

Output: After all queries, \mathcal{A}_{II} outputs a forgery $(m^*, CB\text{-}\sigma^*, ID^*)$.

Restrictions: We say \mathcal{A}_{II} wins the game if the forgery satisfies the requirements as following:

1. $true \leftarrow \text{CB-Verify}(m, CB\text{-}\sigma^*, CB\text{-params}, ID^*, CB\text{-}PK_{ID^*}, CB\text{-mpk})$. Here $CB\text{-}PK_{ID^*}$ is the original public key in $L1^{PK}$;
2. $(ID^*, m^*) \not\rightarrow \mathcal{O}^{\text{CB-Sign}}$; and
3. $ID^* \not\rightarrow \mathcal{O}^{\text{CB-Corruption}}$.

The success probability that an adaptive chosen message and chosen identity adversary $\text{CB-}\mathcal{A}_{II}$ wins the above game is denoted as $\text{Succ}_{\mathcal{A}_{II}, type}^{cma, cida}$, where $type \in \{normal, super\}$. We say a certificate-based signature scheme is secure against a $(t, q_{UC}, q_{PKR}, q_C, q_S)$ $\text{CB-}\mathcal{A}_{II}$ if $\text{Succ}_{\mathcal{A}_{II}, type}^{cma, cida}$ is negligible. Here, $\mathcal{O}^{\text{CB-Sign}}$ will be $\mathcal{O}^{\text{CB-NormalSign}}$ if $type = normal$. Otherwise, $\mathcal{O}^{\text{CB-Sign}}$ is $\mathcal{O}^{\text{CB-SuperSign}}$.

4.4.5 Security Against Malicious-but-Passive Type II Adversary

We now define a more powerful type II adversary, who is allowed to generate the system parameter and the master secret/public key. This assumes that the third party certifier have already been malicious at the very beginning of the setup stage of the system, rather than being only given the parameter and the master secret/public key honestly generated by the challenger. Such attacks are first introduced to certificateless cryptosystems in [ACL⁺07]. In addition to this, even though we say that the certifier is malicious, we also assume (as in [ACL⁺07]) that the certifier is passive, in the sense that the certifier would not actively replace the user's public key or corrupt the user's secret key. It is shown in [ACL⁺07] that the malicious-but-passive third party KGC in certificateless cryptosystems like [ARP03] can have its master key pair specifically generated so that all the encrypted messages for the target victim can also be decrypted by the KGC. The security model of certificate-based encryption in [Gen03] also captures the essence of those attacks. The security of certificate-based signatures against a malicious-but-passive Type II adversary is defined by the following game:

Initial: The challenger executes \mathcal{A}_{II} on the security parameter 1^k . \mathcal{A}_{II} returns the system parameters $CB\text{-params}$ and master public key $CB\text{-mpk}$.

Queries: A malicious-but-passive \mathcal{A}_{II} can make queries to all oracles except $\mathcal{O}^{\text{CB-CertGen}}$. Since $CB\text{-params}$ and $CB\text{-mpk}$ are generated by the adversary \mathcal{A}_{II} , $\mathcal{O}^{\text{CB-UserCreate}}$ and $\mathcal{O}^{\text{CB-PKReplace}}$ have to be modified as following:

$\mathcal{O}^{\text{CB-UserCreate}}$: As defined in Section 4.3, this oracle receives an input $ID \in \Gamma$ and outputs the public key $CB\text{-PK}_{ID}$. After obtaining $CB\text{-PK}_{ID}$, a malicious-but-passive Type II adversary must provide ID 's certificate $Cert_{ID}$ for $(ID, CB\text{-PK}_{ID})$. This oracle then adds $(ID, CB\text{-SK}_{ID}, CB\text{-PK}_{ID}, Cert_{ID})$ to $L1^{PK}$, and $(ID, CB\text{-PK}_{ID}, Cert_{ID})$ to $L2^{PK}$.

$\mathcal{O}^{\text{CB-PKReplace}}$: For a malicious-but-passive Type II adversary, the input to this oracle should be $(ID, CB\text{-PK}, Cert)$ where $Cert$ is the corresponding certificate of CB-PK under the identity ID . This oracle searches $L2^{PK}$, finds a record related to ID and sets $CB\text{-}\overline{PK}_{ID} \leftarrow CB\text{-PK}$ and $\overline{Cert}_{ID} \leftarrow Cert$. It then updates the related tuple with $(ID, CB\text{-}\overline{PK}_{ID}, \overline{Cert}_{ID})$.

Output and Restrictions: As those defined in Sec. 4.4.4.

The success probability that an adaptive chosen message and chosen identity malicious-but-passive Type II adversary wins the above game is denoted as $Succ_{MP\text{-}\mathcal{A}_{II}, type}^{cma, cida}$, where $type \in \{normal, super\}$. We say a certificate-based signature scheme is secure against a $(t, q_{UC}, q_{PKR}, q_C, q_S)$ malicious-but-passive CB- \mathcal{A}_{II} if $Succ_{MP\text{-}\mathcal{A}_{II}, type}^{cma, cida}$ is negligible. Here, $\mathcal{O}^{\text{CB-Sign}}$ will be $\mathcal{O}^{\text{CB-NormalSign}}$ if $type = normal$. Otherwise, $\mathcal{O}^{\text{CB-Sign}}$ is $\mathcal{O}^{\text{CB-SuperSign}}$.

4.5 Generic Construction of Certificate-based Signatures

In this section, we will introduce a generic method to construct certificate-based signatures. Our construction is based on certificateless signatures whose description is as below.

4.5.1 Syntax of Certificateless Signatures

A certificateless signature (CLS) scheme is defined by six algorithms: **CL-Setup** (generates KGC's key pair $(CL\text{-msk}, CL\text{-mpk})$ and system's parameter), **CL-PPKExtract** (generates a user ID 's partial private key $CL\text{-PPK}_{ID}$), **CL-SSValue** (generates a

user ID 's secret value $CL-SV_{ID}$), $CL-SPKey$ (generates a user ID 's public key $CL-PK_{ID}$), $CL-Sign$ (generates a certificateless signature $CL-\sigma$ using $CL-PPK_{ID}$ and $CL-SV_{ID}$) and $CL-Verify$ (outputs *true* if a given signature is valid, or *false* otherwise). As one can see, to distinguish from the identity information in the certificate-based system (which is denoted as ID), we use the notion ID to denote the identity information in the certificateless system. For other notations, we put the prefix “ $CL-$ ” to indicate that they are in the certificateless cryptosystem. Please refer to [HMS⁺07] for the formal definition of each algorithm.

4.5.2 Generic Construction: CLS-2-CBS

In this section, we show how to convert a certificateless signature scheme into a certificate-based signature scheme. In our construction, we need a hash function $H : \Gamma \times \mathcal{PK}^{CB} \rightarrow \mathcal{ID}^{CL}$. Here, Γ is the identity information space in the certificate-based system, \mathcal{PK}^{CB} is the public key space in certificate-based system and \mathcal{ID}^{CL} denotes the space of identities in the certificateless cryptosystem².

Let CLS be the certificateless signature scheme described in Section 4.5.1. We now describe the generic construction $CLS-2-CBS$.

1. $CB-Setup(1^k) \rightarrow (CB-msk, CB-mpk, CB-params)$.
 - (a) Run algorithm $CL-Setup(1^k)$ of CLS to obtain $CL-params$, $CL-msk$ and $CL-mpk$. For the security parameter k , we assume that the public key size in a certificateless cryptosystem is at least 2^k ;
 - (b) Set $CB-params$ by extending $CL-params$ to include the description of Γ ;
 - (c) $(CB-msk, CB-mpk) \leftarrow (CL-msk, CL-mpk)$.
2. $CB-UserCreate(CB-mpk, CB-params, ID \in \Gamma) \rightarrow (CB-SK_{ID}, CB-PK_{ID})$.
 - (a) $CL-mpk \leftarrow CB-mpk$;
 - (b) Extract $CL-params$ from $CB-params$;

²Here, we use the hash function H to “connect” two identities in certificate-based signatures and certificateless signatures. This is different from the technique in the generic construction of certificate-based encryption proposed in [ARP05]. A recent work [KP05] pointed out a flaw of security proof in [ARP05]. That flaw does not exist in our construction, the details of which will be shown in the security proof later.

- (c) $CB-SK_{ID} \leftarrow \text{CL-SSValue}(CL-mpk, CL-params)$;
- (d) $CB-PK_{ID} \leftarrow \text{CL-SPKey}(CL-mpk, CL-params, CB-SK_{ID})$.
3. $\text{CB-CertGen}(CB-msk, CB-mpk, CB-params, ID \in \Gamma, CB-PK_{ID}) \rightarrow Cert_{ID}$.
- (a) $(CL-msk, CL-mpk) \leftarrow (CB-msk, CB-mpk)$;
- (b) Extract $CL-params$ from $CB-params$;
- (c) $H(ID, CB-PK_{ID}) \rightarrow ID \in \mathcal{ID}^{\mathcal{CL}}$;
- (d) $Cert_{ID} \leftarrow \text{CL-PPKExtract}(CL-msk, CL-mpk, CL-params, ID)$.
4. $\text{CB-Sign}(m, CB-params, CB-mpk, ID, Cert_{ID}, CB-SK_{ID}, CB-PK_{ID}) \rightarrow CB-\sigma$.
- (a) Extract $CL-params$ from $CB-params$;
- (b) $CL-mpk \leftarrow CB-mpk$;
- (c) $H(ID, CB-PK_{ID}) \rightarrow ID \in \mathcal{ID}^{\mathcal{CL}}$;
- (d) $(CL-SV_{ID}, CL-PK_{ID}) \leftarrow (CB-SK_{ID}, CB-PK_{ID})$ and $CL-PPK_{ID} \leftarrow Cert_{ID}$;
- (e) $CB-\sigma \leftarrow \text{CL-Sign}(m, CL-params, CL-mpk, ID, CL-SV_{ID}, CL-PK_{ID}, CL-PPK_{ID})$. One can see that the signature size of $CB-\sigma$ is the same as that in the underlying certificateless signature scheme.
5. $\text{CB-Verify}(CB-params, CB-mpk, ID, CB-PK_{ID}, (m, CB-\sigma)) \rightarrow \{true, false\}$.
- (a) Extract $CL-params$ from $CB-params$;
- (b) $CL-mpk \leftarrow CB-mpk$;
- (c) $H(ID, CB-PK_{ID}) \rightarrow ID \in \mathcal{ID}^{\mathcal{CL}}$;
- (d) $CL-PK_{ID} \leftarrow CB-PK_{ID}$;
- (e) $CL-\sigma \leftarrow CB-\sigma$.
- (f) Output $\text{CL-Verify}(CL-mpk, CL-params, ID, CL-PK_{ID}, (m, CL-\sigma))$.

Correctness. We show that any certificate-based signature produced by CB-Sign will pass through the verification in CB-Verify .

In our construction, a certificate-based signature is the output of the algorithm CL-Sign in the certificateless system, and algorithm CB-Verify also employs the verification algorithm CL-Verify in the certificateless system. To show the correctness of

our construction, it suffices to show that under the same $CL\text{-params}$ and $CL\text{-mpk}$, a certificateless signature produced by using the secret value $CL\text{-SV}_{ID}$ and the partial private key $CL\text{-PPK}_{ID}$ will pass through the check using the corresponding identity ID and its public key $CL\text{-PK}_{ID}$. This is ensured by the correctness of the underlying certificateless signature scheme, that is, for any signature $CL\text{-}\sigma$ produced by $\text{CL-Sign}(m, CL\text{-params}, CL\text{-mpk}, ID, CL\text{-SV}_{ID}, CL\text{-PK}_{ID}, CL\text{-PPK}_{ID})$, $\text{CL-Verify}(CL\text{-mpk}, CL\text{-params}, ID, CL\text{-PK}_{ID}, (m, CL\text{-}\sigma))$ will output *true*. Therefore, for any signature output by CB-Sign defined in our construction, the algorithm CB-Verify will always output *true*.

Security Analysis

Theorem 4.1 [*Security of CLS-2-CBS*] *CLS-2-CBS is secure (in the random oracle model) against adversaries defined in Section 4.4, assuming the underlying certificateless signature scheme CLS satisfying certain security requirements.*

The proof of Theorem 4.1 consists of several lemmas, which demonstrate the security relationship between our generic construction CLS-2-CBS and its underlying certificateless signature scheme CLS. Please refer to [HMS⁺07] for security definitions of CLS.

Lemma 4.2 (Security against Normal-CB- \mathcal{A}_I) *CLS-2-CBS is secure (in the random oracle model) against **Normal-CB- \mathcal{A}_I** defined in Section 4.4.1, if CLS is secure against **Normal-CL- \mathcal{A}_I** defined in [HMS⁺07].*

Proof: In the proof, we will regard hash function H as the random oracle and show that if there is a Normal-CB- \mathcal{A}_I who can forge a valid certificate-based signature of CLS-2-CBS with non-negligible probability, then there exists a Normal-CL- \mathcal{A}_I who can use Normal-CB- \mathcal{A}_I to forge a valid certificateless signature of CLS with almost the same probability.

In our proof, the challenger of a Normal-CB- \mathcal{A}_I is the Normal-CL- \mathcal{A}_I against the underlying CLS, who can make requests to its own challenger CL-Challenger . CL-Challenger is made up of several oracles as follows: $\mathcal{O}^{\text{CL-UserCreate}}$ (creates users in the certificateless cryptosystem), $\mathcal{O}^{\text{CL-Corruption}}$ (returns secret values of created users), $\mathcal{O}^{\text{CL-PPKExtract}}$ (returns partial private keys of created users), $\mathcal{O}^{\text{CL-PKReplace}}$ (replaces public keys of created users with the value provided by the adversary) and $\mathcal{O}^{\text{CL-NormalSign}}$ (returns certificateless signatures on messages chosen by the adversary). Please refer to [HMS⁺07] for the formal definition of each oracle. The description of our proof is as follows.

Initial: The CL-Challenger runs the algorithm CL-Setup of CLS and feeds the Normal-CL- \mathcal{A}_I with $CL\text{-mpk}$ and $CL\text{-params}$. Normal-CL- \mathcal{A}_I then returns $CB\text{-mpk}$ and $CB\text{-params}$ to Normal-CB- \mathcal{A}_I where $CB\text{-mpk}$ is defined to be $CL\text{-mpk}$ and $CB\text{-params}$ is defined by extending $CL\text{-params}$ to include the description of Γ . Before Normal-CB- \mathcal{A}_I submits any queries, Normal-CL- \mathcal{A}_I asks CL-Challenger to create q_{uc} users in the certificateless cryptosystem. Here q_{uc} is the number of queries Normal-CB- \mathcal{A}_I issues to $\mathcal{O}^{\text{CB-UserCreate}}$. Normal-CL- \mathcal{A}_I then records the information as $(ID_i, CL\text{-PK}_{ID_i})$, $i = 1, 2, \dots, q_{uc}$ in the list CL^{PK} . Here $ID_i \in \mathcal{ID}^{\mathcal{CL}}$, $CL\text{-PK}_{ID_i}$ is the original public key of ID_i in certificateless system.

Queries: As defined in Section 4.4.1, Normal-CB- \mathcal{A}_I can issue queries to following oracles. We now show how Normal-CL- \mathcal{A}_I can answer these queries.

- \mathcal{RO} : In the proof, the hash function H is viewed as the random oracle \mathcal{RO} . For a fresh input $(ID, \text{CB-PK}) \in \Gamma \times \mathcal{PK}^{\text{CB}}$, the output of \mathcal{RO} is a random element ID in $\mathcal{ID}^{\mathcal{CL}}$. Normal-CL- \mathcal{A}_I maintains a list $L^{\mathcal{RO}}$ consisting of $(ID, \text{CB-PK}, ID)$.
- $\mathcal{O}^{\text{CB-UserCreate}}$: At any time, Normal-CB- \mathcal{A}_I can request to create the user $ID_i \in \Gamma$ and expect to obtain ID_i 's public key $CB\text{-PK}_{ID_i}$. In response to such queries:
 1. For the i^{th} fresh query ID_i , Normal-CL- \mathcal{A}_I first checks the list CL^{PK} and finds the i^{th} pair $(ID_i, CL\text{-PK}_{ID_i})$. Normal-CL- \mathcal{A}_I then sets $CB\text{-PK}_{ID_i} \leftarrow CL\text{-PK}_{ID_i}$, $H(ID_i, CB\text{-PK}_{ID_i}) = ID_i$ and adds $(ID_i, CB\text{-PK}_{ID_i}, ID_i)$ into $L^{\mathcal{RO}}$. If $(ID_i, CB\text{-PK}_{ID_i})$ already appears in $L^{\mathcal{RO}}$, then the simulation fails and Normal-CL- \mathcal{A}_I aborts. This, however, happens only with negligible probability as $|\mathcal{PK}^{\mathcal{CL}}|$ is assumed to be greater than 2^k and k is the security parameter. Otherwise, it adds $(ID_i, \perp, CB\text{-PK}_{ID_i})$ into the list $L1^{PK}$. Meanwhile, it sets $CB\text{-}\overline{\text{PK}}_{ID_i} \leftarrow CB\text{-PK}_{ID_i}$ and adds $(ID_i, CB\text{-}\overline{\text{PK}}_{ID_i})$ into list $L2^{PK}$. Here, the notation \perp means that Normal-CL- \mathcal{A}_I does not know the corresponding secret key $CB\text{-SK}_{ID_i}$.
 2. In addition to maintain $L1^{PK}, L2^{PK}$, Normal-CL- \mathcal{A}_I will keep two additional lists $L1^{ID}$ and $L2^{ID}$ which will help it answer queries from Normal-CB- \mathcal{A}_I . $L1^{ID}$ consists of pairs with the form (ID_i, ID_i) where $ID_i \in \Gamma$ and

$ID_i \in \mathcal{ID}^{\mathcal{CL}}$. This list will help Normal-CL- \mathcal{A}_I to respond Normal-CB- \mathcal{A}_I 's corruption queries and NormalSign queries. $L2^{ID}$ consists of pairs with the form (ID_i, \overline{ID}_i) where $ID_i \in \Gamma$ and $\overline{ID}_i \in \mathcal{ID}^{\mathcal{CL}}$. In different phases, \overline{ID}_i could be the identity ID_i in the list $L1^{PK}$, or the identity $ID'_i \in \mathcal{ID}^{\mathcal{CL}}$ created at some time later.

For a user ID_i created in this oracle, Normal-CL- \mathcal{A}_I will add (ID_i, ID_i) into list $L1^{ID}$, where $ID_i \in \mathcal{ID}^{\mathcal{CL}}$ is ID_i 's corresponding identity in the list CL^{PK} . Meanwhile, Normal-CL- \mathcal{A}_I sets $\overline{ID}_i \leftarrow ID_i$ and adds (ID_i, \overline{ID}_i) into list $L2^{ID}$.

- $\mathcal{O}^{\text{CB-PKReplace}}$: At any time, Normal-CB- \mathcal{A}_I can replace a public key of a created user ID with the public key $CB-PK'_{ID}$ chosen by himself. In response, Normal-CL- \mathcal{A}_I first sets $CB-\overline{PK}_{ID} \stackrel{\$}{\leftarrow} CB-PK'_{ID}$, then
 1. Normal-CL- \mathcal{A}_I browses the list $L2^{PK}$ and rewrites the related pair as $(ID, CB-\overline{PK}_{ID})$. It then browses $L^{\mathcal{RO}}$.
 2. If $(ID, CB-\overline{PK}_{ID})$ appears in $L^{\mathcal{RO}}$ in the tuple $(ID, CB-\overline{PK}_{ID}, \overline{ID})$, Normal-CL- \mathcal{A}_I will make a user-create query \overline{ID} to CL-Challenger if \overline{ID} has not been created in the certificateless system. After that, Normal-CL- \mathcal{A}_I replaces \overline{ID} 's certificateless public key with $CB-\overline{PK}_{ID}$ and updates the corresponding pair in CL^{PK} with $(\overline{ID}, CB-\overline{PK}_{ID})$. Finally, Normal-CL- \mathcal{A}_I browses the list $L2^{ID}$ and updates the related pair with (ID, \overline{ID}) .
 3. Otherwise, Normal-CL- \mathcal{A}_I sets $H(ID, CB-\overline{PK}_{ID}) = \overline{ID}$, which is randomly chosen in $\mathcal{ID}^{\mathcal{CL}}$. It then adds $(ID, CB-\overline{PK}_{ID}, \overline{ID})$ into $L^{\mathcal{RO}}$. After that, Normal-CL- \mathcal{A}_I asks CL-Challenger to create the user \overline{ID} . After creating the identity \overline{ID} , Normal-CL- \mathcal{A}_I replaces \overline{ID} 's public key with $CB-\overline{PK}_{ID}$. Normal-CL- \mathcal{A}_I then updates CL^{PK} by adding $(\overline{ID}, CB-\overline{PK}_{ID})$. Finally, Normal-CL- \mathcal{A}_I browses the list $L2^{ID}$ and updates the related pair with (ID, \overline{ID}) .
- $\mathcal{O}^{\text{CB-Corruption}}$: At any time, Normal-CB- \mathcal{A}_I can request the secret key of a created user ID_i . In response, Normal-CL- \mathcal{A}_I checks the list $L1^{ID}$ and finds (ID_i, ID_i) . Then, it issues a corruption request ID_i to CL-Challenger who will return $CL-SV_{ID_i}$ to Normal-CL- \mathcal{A}_I , where $CL-SV_{ID_i}$ is the secret value of ID_i when it was created in the certificateless system. At last, Normal-CL- \mathcal{A}_I

sets $CB-SK_{ID_i} \leftarrow CL-SV_{ID_i}$, returns it to Normal-CB- \mathcal{A}_I and updates the information in the list $L1^{PK}$ as $(ID_i, CB-SK_{ID_i}, CB-PK_{ID_i})$.

Correctness: This oracle should return the user ID_i 's original secret key $CB-SK_{ID_i}$ when the user was created. Recall that $L1^{ID}$ contains pairs (ID_i, ID_i) $i = 1, 2, \dots, q_{uc}$, where $ID_i \in \mathcal{ID}^{CL}$ is the ID_i 's initial corresponding identity in certificateless system. ID_i is set as $H(ID_i, CB-PK_{ID_i})$ and $CL-PK_{ID_i} = CB-PK_{ID_i}$ which is the original public key of ID_i . Thus the secret value $CL-SV_{ID_i}$ of ID_i in certificateless system is the same as the secret key $CB-SK_{ID_i}$ of ID_i in certificate-based system.

- $\mathcal{O}^{CB-CertGen}$: At any time, Normal-CB- \mathcal{A}_I can request the certificate of $(ID, CB-PK)$ where CB-PK is chosen by the adversary itself. Normal-CL- \mathcal{A}_I will try to find an identity $\overline{ID} \in \mathcal{ID}^{CL}$, whose partial private key is ID's certificate under the public key CB-PK. To do that, Normal-CL- \mathcal{A}_I will check $L^{\mathcal{R}\mathcal{O}}$:
 1. If $(ID, CB-PK)$ appears in $L^{\mathcal{R}\mathcal{O}}$ in the tuple $(ID, CB-PK, \overline{ID})$, Normal-CL- \mathcal{A}_I will make a user-create query \overline{ID} to CL-Challenger if \overline{ID} has not been created in certificateless system.
 2. Otherwise, Normal-CL- \mathcal{A}_I sets $H(ID, CB-PK) = \overline{ID}$, which is randomly chosen in \mathcal{ID}^{CL} . It then adds $(ID, CB-PK, \overline{ID})$ into $L^{\mathcal{R}\mathcal{O}}$. After that, Normal-CL- \mathcal{A}_I asks CL-Challenger to create the user \overline{ID} .

For either case, Normal-CL- \mathcal{A}_I issues the partial private key query \overline{ID} to CL-Challenger who will return the partial private key $CL-PPK_{\overline{ID}}$. At last, Normal-CL- \mathcal{A}_I sets $Cert_{ID} \leftarrow CL-PPK_{\overline{ID}}$ and returns it to Normal-CB- \mathcal{A}_I .

- $\mathcal{O}^{CB-NormalSign}$: At any time, Normal-CB- \mathcal{A}_I can request the signature of (m_i, ID_i) . The Normal-CL- \mathcal{A}_I first finds the pair (ID_i, ID_i) in the list $L1^{ID}$. Then, Normal-CL- \mathcal{A}_I issues a certificateless signing query (ID_i, m_i) . As defined, Normal-CL- \mathcal{A}_I will obtain the signature $CL-\sigma_i$ such that $true = CL-Verify(CL-mpk, CL-params, ID_i, CL-PK_{ID_i}, (m_i, CL-\sigma_i))$. Normal-CB- \mathcal{A}_I will set $CB-\sigma_i \leftarrow CL-\sigma_i$, and return $CB-\sigma_i$ as the answer.

Correctness: Recall that for the pair (ID_i, ID_i) in $L1^{ID}$, ID_i is set as $H(ID_i, CB-PK_{ID_i})$ and $CB-PK_{ID_i} = CL-PK_{ID_i}$. Here, $CB-PK_{ID_i}$ is ID_i 's original public key in the list $L1^{PK}$. Therefore, $true = CB-Verify(CB-params,$

$CB\text{-mpk}, ID_i, CB\text{-PK}_{ID_i}, (m_i, CB\text{-}\sigma_i)$). That is, $CB\text{-}\sigma_i$ is ID_i 's valid signature for m_i under the original public key returned from $\mathcal{O}^{\text{CB-UserCreate}}$.

Output: After all queries, $\text{CB-}\mathcal{A}_I$ will output a forgery $(m^*, CB\text{-}\sigma^*, ID^*)$. If $\text{Normal-CB-}\mathcal{A}_I$ wins game, then:

1. $true = \text{CB-Verify}(m^*, CB\text{-}\sigma^*, CB\text{-params}, ID^*, CB\text{-}\overline{PK}_{ID^*}, CB\text{-mpk})$, where $(ID^*, CB\text{-}\overline{PK}_{ID^*})$ is in the list $L2^{PK}$. Here, $CB\text{-}\overline{PK}_{ID^*}$ is ID^* 's current public key.

That is, if $CL\text{-}\sigma^* \leftarrow CB\text{-}\sigma^*$, $true = \text{CL-Verify}(CL\text{-mpk}, CL\text{-params}, \overline{ID}^*, CL\text{-}\overline{PK}_{\overline{ID}^*}, (m^*, CL\text{-}\sigma^*))$. Here, $(ID^*, \overline{ID}^*) \in L2^{ID}$ which indicates that $CL\text{-}\overline{PK}_{\overline{ID}^*} = CB\text{-}\overline{PK}_{ID^*}$.

2. $(ID^*, m^*) \not\rightarrow \mathcal{O}^{\text{CB-NormalSign}}$.

That is, (\overline{ID}^*, m^*) has never been asked to $\mathcal{O}^{\text{CL-NormalSign}}$ of CLS.

3. $(ID^*, CB\text{-}\overline{PK}_{ID^*}) \not\rightarrow \mathcal{O}^{\text{CB-CertGen}}$.

That is, \overline{ID}^* has never been asked to $\mathcal{O}^{\text{CL-PPKExtract}}$ of CLS.

4. $ID^* \not\rightarrow \mathcal{O}^{\text{CB-Corruption}}$.

That is, \overline{ID}^* has never been asked to $\mathcal{O}^{\text{CL-Corruption}}$ of CLS.

If $\text{Normal-CL-}\mathcal{A}_I$ does not fail in the simulation, then it can output a valid forgery $(m^*, CL\text{-}\sigma^*, \overline{ID}^*)$ of the underlying certificateless signature scheme with the same success probability as $\text{Normal-CB-}\mathcal{A}_I$. Considering that $\text{Normal-CL-}\mathcal{A}_I$ could only fail in simulating H as the random oracle, which only happens with negligible probability $q_{UC}/2^k$ (q_{UC} is the number of user-create queries). Thus, $\text{Normal-CL-}\mathcal{A}_I$ wins the game with almost the same probability as $\text{Normal-CB-}\mathcal{A}_I$. This completes the proof of Lemma 4.10.

Lemma 4.3 (Security against Strong-CB- \mathcal{A}_I) *CLS-2-CBS is secure (in the random oracle model) against **Strong-CB- \mathcal{A}_I** defined in Section 4.4, if CLS is secure against **Strong-CL- \mathcal{A}_I** .*

Proof: In the proof, we will regard the hash function H as random oracle and show that if there is a **Strong-CB- \mathcal{A}_I** who can forge a valid certificate-based signature of CLS-2-CBS with non-negligible probability, then there exists a **Strong-CL- \mathcal{A}_I** who

can use Strong-CB- \mathcal{A}_I to forge a valid certificateless signature of CLS with the same probability.

In our proof, Strong-CL- \mathcal{A}_I will act as the challenger of Strong-CB- \mathcal{A}_I . Strong-CL- \mathcal{A}_I has its own challenger CL-Challenger, which can answer Strong-CL- \mathcal{A}_I 's queries.

Initial: Same as the proof of Lemma 4.10.

Queries: As defined in Section 4.4.2, Strong-CB- \mathcal{A}_I can issue queries to following oracles. We show how Strong-CL- \mathcal{A}_I can answer these queries.

- $\mathcal{O}^{\text{CB-UserCreate}}$, $\mathcal{O}^{\text{CB-PKReplace}}$, $\mathcal{O}^{\text{CB-Corruption}}$, $\mathcal{O}^{\text{CB-CertGen}}$: These oracles are simulated by Strong-CL- \mathcal{A}_I in the same way as described in the proof of Lemma 4.10.
- $\mathcal{O}^{\text{CB-StrongSign}}$: At any time, Strong-CB- \mathcal{A}_I can request the signature of $(m_i, \text{ID}_i, \text{coin})$ where $\text{coin} \in \{1, 2\}$.
 - If $\text{coin} = 1$, this oracle works as $\mathcal{O}^{\text{CB-NormalSign}}$, which is simulated in the same way as described in the proof of Lemma 4.10.
 - Otherwise $\text{coin} = 2$, this oracle first checks the list $L1^{PK}$ and $L2^{PK}$ to obtain ID_i 's original public key $CB-PK_{\text{ID}_i}$ and ID_i 's current public key $CB-\overline{PK}_{\text{ID}_i}$. If $CB-\overline{PK}_{\text{ID}_i} = CB-PK_{\text{ID}_i}$, this oracle works the same as $\mathcal{O}^{\text{CB-NormalSign}}$. Otherwise, $\mathcal{O}^{\text{CB-StrongSign}}$ asks Strong-CB- \mathcal{A}_I to supply the secret key $CB-\overline{SK}_{\text{ID}_i} \in \mathcal{SK}^{CB}$ corresponding to the current public key $CB-\overline{PK}_{\text{ID}_i}$. Then, Strong-CL- \mathcal{A}_I browses the list $L2^{ID}$ to find the pair $(\text{ID}_i, \overline{\text{ID}}_i)$ and makes a certificateless signing query $(\overline{\text{ID}}_i, m_i, CL-SV_{\overline{\text{ID}}_i})$ where $CL-SV_{\overline{\text{ID}}_i} \leftarrow CB-\overline{SK}_{\text{ID}_i}$. As defined, Strong-CL- \mathcal{A}_I will obtain a certificateless signature $CL-\sigma_i$ such that $\text{true} = \text{CL-Verify}(CL-mpk, CL-params, \overline{\text{ID}}_i, CL-\overline{PK}_{\overline{\text{ID}}_i}, (m_i, CL-\sigma_i))$, where $CL-\overline{PK}_{\overline{\text{ID}}_i}$ is $\overline{\text{ID}}_i$'s current public key. Strong-CB- \mathcal{A}_I will set $CB-\sigma_i \leftarrow CL-\sigma_i$, and return $CB-\sigma_i$ as the answer.

Output: After all queries, Strong-CB- \mathcal{A}_I will output a forgery $(m^*, CB-\sigma^*, \text{ID}^*)$.

If Strong-CB- \mathcal{A}_I wins game, then:

1. $\text{true} = \text{CB-Verify}(m^*, CB-\sigma^*, CB-params, \text{ID}^*, CB-\overline{PK}_{\text{ID}^*}, CB-mpk)$, where $(\text{ID}^*, CB-\overline{PK}_{\text{ID}^*})$ is in the list $L2^{PK}$. Here, $CB-\overline{PK}_{\text{ID}^*}$ is ID^* 's current public key.

That is, if $CL\text{-}\sigma^* \leftarrow CB\text{-}\sigma^*$, $true = \text{CL-Verify}(CL\text{-}mpk, CL\text{-}params, \overline{ID}^*, CL\text{-}\overline{PK}_{\overline{ID}^*}, (m^*, CL\text{-}\sigma^*))$, where $(ID^*, \overline{ID}^*) \in L2^{ID}$.

2. $(ID^*, m^*) \rightarrow \mathcal{O}^{\text{CB-StrongSign}}$.

That is, (\overline{ID}^*, m^*) has never been asked to $\mathcal{O}^{\text{CL-StrongSign}}$ of CLS.

3. $(ID^*, CB\text{-}\overline{PK}_{ID^*}) \rightarrow \mathcal{O}^{\text{CB-CertGen}}$.

That is, \overline{ID}^* has never been asked to $\mathcal{O}^{\text{CL-PPKExtract}}$ of CLS.

If Strong-CL- \mathcal{A}_I does not fail in the simulation, then it can output a valid forgery $(m^*, CL\text{-}\sigma^*, \overline{ID}^*)$ of the underlying certificateless signature scheme with the same success probability as Strong-CB- \mathcal{A}_I . As in the proof of Lemma 4.10, Strong-CL- \mathcal{A}_I could only fail in simulating H as \mathcal{RO} , which only happens with negligible probability $1/2^k$. Thus, Strong-CL- \mathcal{A}_I wins the game with almost the same probability as Strong-CB- \mathcal{A}_I . This completes the proof of Lemma 4.3.

Lemma 4.4 (Security against Super-CB- \mathcal{A}_I) *CLS-2-CBS is secure (in the random oracle model) against type I adversary **Super-CB- \mathcal{A}_I** defined in Section 4.4, if CLS is secure against **Super-CL- \mathcal{A}_I** .*

Proof: In the proof, we will regard the hash function H as random oracle and show that if there is a Super-CB- \mathcal{A}_I who can forge a valid certificate-based signature of CLS-2-CBS with non-negligible probability, then there exists a Super-CL- \mathcal{A}_I who can use Super-CB- \mathcal{A}_I to forge a valid certificateless signature of CLS with almost the same probability.

In our proof, Super-CL- \mathcal{A}_I will act as the challenger of Super-CB- \mathcal{A}_I . Super-CL- \mathcal{A}_I has his own challenger CL-Challenger, which can answer Super-CL- \mathcal{A}_I 's queries.

Initial: Same as the proof of Lemma 4.10.

Queries: As defined in Section 4.4.2, Super-CB- \mathcal{A}_I can issue queries to following oracles. We show how Super-CL- \mathcal{A}_I can answer these queries.

- $\mathcal{O}^{\text{CB-UserCreate}}$, $\mathcal{O}^{\text{CB-PKReplace}}$, $\mathcal{O}^{\text{CB-Corruption}}$, $\mathcal{O}^{\text{CB-CertGen}}$: These oracles are simulated by Super-CL- \mathcal{A}_I in the same way as described in the proof of Lemma 4.10.
- $\mathcal{O}^{\text{CB-SuperSign}}$: At any time, Super-CB- \mathcal{A}_I can request the signature of (ID_i, m_i) . Super-CL- \mathcal{A}_I first finds the pair (ID_i, \overline{ID}_i) in the list $L2^{ID}$. Then, Super-CL- \mathcal{A}_I issues a certificateless signing query (\overline{ID}_i, m_i) . As defined, Super-CL- \mathcal{A}_I

will obtain the signature $CL\text{-}\sigma_i$ such that $true = \text{CL-Verify}(CL\text{-}mpk, CL\text{-}params, \overline{ID}_i, CL\text{-}\overline{PK}_{\overline{ID}_i}, (m_i, CL\text{-}\sigma_i))$, where $CL\text{-}\overline{PK}_{\overline{ID}_i}$ is \overline{ID}_i 's current public key. Super-CB- \mathcal{A}_I will set $CB\text{-}\sigma_i \leftarrow CL\text{-}\sigma_i$, and return $CB\text{-}\sigma_i$ as the answer.

Correctness: Recall that for the pair (ID_i, \overline{ID}_i) in $L2^{ID}$, \overline{ID}_i is set as $H(ID_i, CB\text{-}\overline{PK}_{ID_i})$ where $CB\text{-}\overline{PK}_{ID_i}$ is ID_i 's current public key in the list $L2^{PK}$, and \overline{ID}_i 's current public key $CL\text{-}\overline{PK}_{\overline{ID}_i}$ has been set as $CB\text{-}\overline{PK}_{ID_i}$. Therefore, $true \leftarrow \text{CB-Verify}(CB\text{-}params, CB\text{-}mpk, ID_i, CB\text{-}\overline{PK}_{ID_i}, (m_i, CB\text{-}\sigma_i))$. That is, $CB\text{-}\sigma_i$ is ID_i 's valid signature for m_i under the current public key in the list $L2^{PK}$.

Output: After all queries, Super-CB- \mathcal{A}_I will output a forgery $(m^*, CB\text{-}\sigma^*, ID^*)$. If Super-CB- \mathcal{A}_I wins game, then:

1. $true = \text{CB-Verify}(m^*, CB\text{-}\sigma^*, CB\text{-}params, ID^*, CB\text{-}\overline{PK}_{ID^*}, CB\text{-}mpk)$, where $(ID^*, CB\text{-}\overline{PK}_{ID^*})$ is in the list $L2^{PK}$. Here, $CB\text{-}\overline{PK}_{ID^*}$ is ID^* 's current public key.

That is, if $CL\text{-}\sigma^* \leftarrow CB\text{-}\sigma^*$, $true = \text{CL-Verify}(CL\text{-}mpk, CL\text{-}params, \overline{ID}^*, CL\text{-}\overline{PK}_{\overline{ID}^*}, (m^*, CL\text{-}\sigma^*))$, where $(ID^*, \overline{ID}^*) \in L2^{ID}$.

2. $(ID^*, m^*) \not\rightarrow \mathcal{O}^{\text{CB-SuperSign}}$.

That is, (\overline{ID}^*, m^*) has never been asked to $\mathcal{O}^{\text{CL-SuperSign}}$ of CLS.

3. $(ID^*, CB\text{-}\overline{PK}_{ID^*}) \not\rightarrow \mathcal{O}^{\text{CB-CertGen}}$.

That is, \overline{ID}^* has never been asked to the $\mathcal{O}^{\text{CL-PPKExtract}}$ of CLS.

If Super-CL- \mathcal{A}_I does not fail in the simulation, then it can output a valid forgery $(m^*, CL\text{-}\sigma^*, \overline{ID}^*)$ of the underlying certificateless signature scheme with the same success probability as Super-CB- \mathcal{A}_I . As in the proof of Lemma 4.10, Super-CL- \mathcal{A}_I could only fail in simulating H as \mathcal{RO} , which only happens with negligible probability $1/2^k$. Thus, Super-CL- \mathcal{A}_I wins the game with almost the same probability as Super-CB- \mathcal{A}_I . This completes the proof of Lemma 4.4.

Lemma 4.5 (Security against CB- \mathcal{A}_{II}) *CLS-2-CBS is secure (in the random oracle model) against type II adversary CB- \mathcal{A}_{II} defined in Section 4.4.4, if CLS is secure against CL- \mathcal{A}_{II} .*

Proof: In the proof, we will regard the hash function H as the random oracle and show that if there is a $\text{CB-}\mathcal{A}_{II}$ (either $\text{Normal-CB-}\mathcal{A}_{II}$ or $\text{Super-CB-}\mathcal{A}_{II}$) who can forge a valid certificate-based signature of CLS-2-CBS with non-negligible probability, then there exists a $\text{CL-}\mathcal{A}_{II}$ (correspondingly, $\text{Normal-CL-}\mathcal{A}_{II}$, or $\text{Super-CL-}\mathcal{A}_{II}$) who can use $\text{CB-}\mathcal{A}_{II}$ to forge a valid certificateless signature of CLS with almost the same probability.

In our proof, the challenger of a $\text{CB-}\mathcal{A}_{II}$ is the $\text{CL-}\mathcal{A}_{II}$ against the underlying certificateless signature scheme, who can make requests to its own challenger CL-Challenger . The description of our proof is as follows.

Initial: The CL-Challenger runs the algorithm CL-Setup of CLS and feeds the $\text{CL-}\mathcal{A}_{II}$ with $\text{CL-}msk$, $\text{CL-}mpk$ and $\text{CL-}params$. $\text{CL-}\mathcal{A}_{II}$ then returns $(\text{CB-}msk, \text{CB-}mpk)$ and $\text{CB-}params$ to $\text{CB-}\mathcal{A}_{II}$ where $(\text{CB-}msk, \text{CB-}mpk)$ is defined to be $(\text{CL-}msk, \text{CL-}mpk)$ and $\text{CB-}params$ is defined by extending $\text{CL-}params$ to include the description of Γ . Before $\text{CB-}\mathcal{A}_{II}$ submits any queries, $\text{CL-}\mathcal{A}_{II}$ asks CL-Challenger to create q_{uc} users in the certificateless cryptosystem. Here q_{uc} is the number of queries $\text{CB-}\mathcal{A}_{II}$ issues to $\mathcal{O}^{\text{CB-UserCreate}}$. $\text{CL-}\mathcal{A}_{II}$ then records the information as $(ID_i, \text{CL-PK}_{ID_i})$, $i = 1, 2, \dots, q_{uc}$ in the list CL^{PK} . Here $ID_i \in \mathcal{ID}^{\text{CL}}$, CL-PK_{ID_i} is the original public key of ID_i in certificateless system.

Queries: As defined in Section 4.4.4, $\text{CB-}\mathcal{A}_{II}$ can issue queries to $\mathcal{O}^{\text{CB-UserCreate}}$, $\mathcal{O}^{\text{CB-PKReplace}}$, $\mathcal{O}^{\text{CB-Corruption}}$, $\mathcal{O}^{\text{CB-NormalSign}}$ (or, $\mathcal{O}^{\text{CB-SuperSign}}$). These oracles are simulated by $\text{CL-}\mathcal{A}_{II}$ in the same way as described in the proof of the security against Type I adversary in Lemma 4.10.

Output: After all queries, $\text{CB-}\mathcal{A}_{II}$ will output a forgery $(m^*, \text{CB-}\sigma^*, \text{ID}^*)$. If $\text{CB-}\mathcal{A}_{II}$ wins, then:

1. $\text{true} = \text{CB-Verify}(m^*, \text{CB-}\sigma^*, \text{CB-}params, \text{ID}^*, \text{CB-PK}_{\text{ID}^*}, \text{CB-}mpk)$,
where $(\text{ID}^*, \text{CB-PK}_{\text{ID}^*})$ is in the list $L1^{PK}$, that is, $\text{CB-PK}_{\text{ID}^*}$ is ID^* 's original public key.

That is, if $\text{CL-}\sigma^* \leftarrow \text{CB-}\sigma^*$, $\text{true} = \text{CL-Verify}(\text{CL-}mpk, \text{CL-}params, \text{ID}^*, \text{CL-PK}_{\text{ID}^*}, (m^*, \text{CL-}\sigma^*))$, where $(\text{ID}^*, \text{ID}^*) \in L1^{\text{ID}}$.

2. $(\text{ID}^*, m^*) \not\rightarrow \mathcal{O}^{\text{CB-Sign}}$.

That is, (ID^*, m^*) has never been asked to $\mathcal{O}^{\text{CL-Sign}}$ of CLS .

3. $\text{ID}^* \not\rightarrow \mathcal{O}^{\text{CB-Corruption}}$.

That is, ID^* has never been asked to $\mathcal{O}^{\text{CL-Corruption}}$ of CLS .

If $\text{CL-}\mathcal{A}_{II}$ does not fail in the simulation, then it can output a valid forgery $(m^*, \text{CL-}\sigma^*, \text{ID}^*)$ of the underlying certificateless signature scheme with the same success probability as $\text{CB-}\mathcal{A}_{II}$. As in the proof of Lemma 4.10, $\text{CL-}\mathcal{A}_{II}$ could only fail in simulating H as the random oracle, which only happens with negligible probability $q_{UC}/2^k$ (q_{UC} is the number of user-create queries). Thus, $\text{CL-}\mathcal{A}_{II}$ wins the game with almost the same probability as $\text{CB-}\mathcal{A}_{II}$. This completes the proof of Lemma 4.5.

Lemma 4.6 (Security against Malicious-CB- \mathcal{A}_{II}) *CLS-2-CBS is secure (in the random oracle model) against **Malicious-CB- \mathcal{A}_{II}** defined in Section 4.6, if CLS is secure against **Malicious-CL- \mathcal{A}_{II}** .*

Proof: In the proof, we will regard the hash function H as random oracle and show that if there is a Malicious-CB- \mathcal{A}_{II} who can forge a valid certificate-based signature of CLS-2-CBS with non-negligible probability, then there exists a Malicious-CL- \mathcal{A}_{II} who can use Malicious-CB- \mathcal{A}_{II} to forge a valid certificateless signature of CLS with almost the same probability.

In our proof, Malicious-CL- \mathcal{A}_{II} will act as the challenger of Malicious-CB- \mathcal{A}_{II} , at the meantime Malicious-CL- \mathcal{A}_{II} has his own challenger CL-Challenger (which consists of $\mathcal{O}^{\text{CL-UserCreate}}$, $\mathcal{O}^{\text{CL-SSValue}}$, $\mathcal{O}^{\text{CL-SPKey}}$, $\mathcal{O}^{\text{CL-PKReplace}}$, $\mathcal{O}^{\text{CL-Corruption}}$, $\mathcal{O}^{\text{CL-Sign}}$).

Initial: Given the security parameter 1^k , Malicious-CB- \mathcal{A}_{II} will sends its challenger, that is, Malicious-CL- \mathcal{A}_{II} , the system parameters CB-params and master public key CB-mpk . Malicious-CL- \mathcal{A}_{II} then extracts CL-params from CB-params , sets CL-mpk as CB-mpk and sends them to its own challenger CL-Challenger.

Queries: As defined in Section 4.4.1, Malicious-CB- \mathcal{A}_{II} can issue queries to following oracles. We show how Malicious-CL- \mathcal{A}_{II} can answer these queries:

- \mathcal{RO} : In the proof, the hash function H is viewed as the random oracle \mathcal{RO} . For a fresh input $(\text{ID}, \text{CB-PK}) \in \Gamma \times \mathcal{PK}^{\text{CB}}$, the output of \mathcal{RO} is a random element ID in \mathcal{ID}^{CL} . Malicious-CL- \mathcal{A}_{II} maintains a list $L^{\mathcal{RO}}$ consisting of $(\text{ID}, \text{CB-PK}, \text{ID})$.
- $\mathcal{O}^{\text{CB-UserCreate}}$: At any time Malicious-CB- \mathcal{A}_{II} can request to create the user $\text{ID}_i \in \Gamma$ and expect to obtain ID_i 's public key $\text{CB-PK}_{\text{ID}_i}$. In response to such queries:
 1. For the i^{th} fresh query ID_i , Malicious-CL- \mathcal{A}_{II} will choose a random identity $\text{ID}_i \in \mathcal{ID}^{\text{CL}}$ and ask CL-Challenger to create ID_i in the certificateless system. After obtaining ID_i 's certificateless public key $\text{CL-PK}_{\text{ID}_i}$,

Malicious-CL- \mathcal{A}_{II} will set $CB-PK_{ID_i} \leftarrow CL-PK_{ID_i}$, $H(ID_i, CB-PK_{ID_i}) = ID_i$ and adds $(ID_i, CB-PK_{ID_i}, ID_i)$ into $L^{\mathcal{R}\mathcal{O}}$. If $(ID_i, CB-PK_{ID_i})$ already appears in $L^{\mathcal{R}\mathcal{O}}$, then the simulation fails and Malicious-CL- \mathcal{A}_{II} aborts. This, however, happens only with negligible probability as $|\mathcal{PK}^{\mathcal{C}\mathcal{L}}|$ is assumed to be greater than 2^k and k is the security parameter. Malicious-CL- \mathcal{A}_{II} then sends $CB-PK_{ID_i}$ to Malicious-CB- \mathcal{A}_{II} .

2. After obtaining $CB-PK_{ID_i}$, Malicious-CB- \mathcal{A}_{II} must calculate the corresponding certificate $Cert_{ID_i}$ and return it to Malicious-CL- \mathcal{A}_{II} , who will forward $Cert_{ID_i}$ to CL-Challenger as the partial private key of ID_i in certificateless system. Malicious-CL- \mathcal{A}_{II} then adds $(ID_i, \perp, CB-PK_{ID_i}, Cert_{ID_i})$ into the list $L1^{PK}$. Here, the notation \perp means that Malicious-CL- \mathcal{A}_{II} does not know the corresponding secret key $CB-SK_{ID_i}$. Meanwhile, Malicious-CL- \mathcal{A}_{II} sets $CB-\overline{PK}_{ID_i} \leftarrow CB-PK_{ID_i}$ and adds $(ID_i, CB-\overline{PK}_{ID_i}, Cert_{ID_i})$ into list $L2^{PK}$.
3. In addition to maintain $L1^{PK}, L2^{PK}$, Malicious-CL- \mathcal{A}_{II} will keep two additional lists $L1^{ID}$ and $L2^{ID}$, which are the same as those in the proof of Lemma 4.10.

For a user ID_i created in this oracle, Malicious-CL- \mathcal{A}_{II} will add (ID_i, ID_i) into list $L1^{ID}$, where $ID_i \in \mathcal{ID}^{\mathcal{C}\mathcal{L}}$ is the ID_i 's corresponding identity in the list CL^{PK} . Meanwhile, Malicious-CL- \mathcal{A}_{II} sets $\overline{ID}_i \leftarrow ID_i$ and adds (ID_i, \overline{ID}_i) into list $L2^{ID}$.

- $\mathcal{O}^{CB-PK\text{Replace}}$: At any time Malicious-CB- \mathcal{A}_{II} can replace a public key of a created user ID with the public key $CB-PK'_{ID}$ chosen by himself. Meanwhile Malicious-CB- \mathcal{A}_{II} must provide the corresponding certificate $Cert'_{ID}$ of $CB-PK'_{ID}$. In response, Malicious-CL- \mathcal{A}_{II} first sets $CB-\overline{PK}_{ID} \leftarrow CB-PK'_{ID}$, then
 1. Malicious-CL- \mathcal{A}_{II} browses the list $L2^{PK}$ and rewrites the related tuple as $(ID, CB-\overline{PK}_{ID}, Cert'_{ID})$.
 2. If $(ID, CB-\overline{PK}_{ID})$ appears in $L^{\mathcal{R}\mathcal{O}}$ in the tuple $(ID, CB-\overline{PK}_{ID}, \overline{ID})$, Malicious-CL- \mathcal{A}_{II} will make a user-create query \overline{ID} to CL-Challenger if \overline{ID} has not been created in the certificateless system. After that, Malicious-CL- \mathcal{A}_{II} replaces \overline{ID} 's certificateless public key with $CB-\overline{PK}_{ID}$ and updates the related pair in $L2^{ID}$ with (ID, \overline{ID}) .

3. Otherwise, Malicious-CL- \mathcal{A}_{II} sets $H(\text{ID}, CB\text{-}\overline{PK}_{\text{ID}}) = \overline{ID}$, which is randomly chosen in $\mathcal{ID}^{\mathcal{CL}}$. It then adds $(\text{ID}, CB\text{-}\overline{PK}_{\text{ID}}, \overline{ID})$ into $L^{\mathcal{RO}}$ and asks CL-Challenger to create the user \overline{ID} . After creating \overline{ID} , Malicious-CL- \mathcal{A}_{II} replaces \overline{ID} 's public key with $CB\text{-}\overline{PK}_{\text{ID}}$ and updates the related pair in L^{2ID} with $(\text{ID}, \overline{ID})$.

- $\mathcal{O}^{\text{CB-Corruption}}$ and $\mathcal{O}^{\text{CB-NormalSign}}$: These two are simulated by Malicious-CL- \mathcal{A}_{II} as same as described in the proof of Lemma 4.10.
- $\mathcal{O}^{\text{CB-SuperSign}}$: This is simulated by Malicious-CL- \mathcal{A}_{II} as same as described in the proof of Lemma 4.4.

Output: After all queries, Malicious-CB- \mathcal{A}_{II} will output a forgery $(m^*, CB\text{-}\sigma^*, \text{ID}^*)$. If Malicious-CB- \mathcal{A}_{II} wins, then:

1. $\text{true} = \text{CB-Verify}(m^*, CB\text{-}\sigma^*, \text{CB-params}, \text{ID}^*, CB\text{-}PK_{\text{ID}^*}, \text{CB-mpk})$, where $(\text{ID}^*, CB\text{-}PK_{\text{ID}^*})$ is in the list L^{1PK} , that is, $CB\text{-}PK_{\text{ID}^*}$ is ID^* 's original public key.

That is, if $CL\text{-}\sigma^* \leftarrow CB\text{-}\sigma^*$, $\text{true} = \text{CL-Verify}(CL\text{-mpk}, CL\text{-params}, \text{ID}^*, CL\text{-}PK_{\text{ID}^*}, (m^*, CL\text{-}\sigma^*))$, where $(\text{ID}^*, \text{ID}^*) \in L^{1ID}$.

2. $(\text{ID}^*, m^*) \not\rightarrow \mathcal{O}^{\text{CB-Sign}}$.

That is, (ID^*, m^*) has never been asked to $\mathcal{O}^{\text{CL-Sign}}$ of CLS.

3. $\text{ID}^* \not\rightarrow \mathcal{O}^{\text{CB-Corruption}}$.

That is, ID^* has never been asked to $\mathcal{O}^{\text{CL-Corruption}}$ of CLS.

If Malicious-CL- \mathcal{A}_{II} does not fail in the simulation, then it can output a valid forgery $(m^*, CL\text{-}\sigma^*, \text{ID}^*)$ of the underlying certificateless signature scheme with the same success probability as Malicious-CB- \mathcal{A}_{II} . As in the proof of Lemma 4.10, Malicious-CL- \mathcal{A}_{II} could only fail in simulating H as \mathcal{RO} , which only happens with negligible probability $1/2^k$. Thus, Malicious-CL- \mathcal{A}_{II} wins the game with almost the same probability as Malicious-CB- \mathcal{A}_{II} . This completes the proof of Lemma 4.6.

The above lemmas complete the proof of Theorem 4.1.

4.6 Concrete Examples of CLS-2-CBS

By applying CLS-2-CBS to concrete certificateless signature schemes, we can obtain several new constructions of certificate-based signatures. This section will describe two of them, which are constructed from certificateless signature schemes proposed in [HMS⁺07]. Please refer to Section 2.1 for the bilinear groups and the complexity assumptions we use in the scheme.

4.6.1 Scheme I

The scheme described in this section is based on the certificateless signature scheme in Section 4.2 of [HMS⁺07]. It consists of following algorithms.

- **CB-Setup:** Let $(\mathbb{G}_1, \mathbb{G}_T)$ be bilinear groups as described in Section 2.1 where $|\mathbb{G}_1| = |\mathbb{G}_T| = p$, for some prime number $p \geq 2^k$, where k is the system security number. e denotes the bilinear mapping $\mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$. Let $H_0, H_1 : \{0, 1\}^* \rightarrow \mathbb{G}_1^*$ and $H_2 : \Gamma \times \mathbb{G}_1^* \rightarrow \mathcal{ID}$ be three secure cryptographic hash functions, where Γ is the set of identity information in the certificate-based system and \mathcal{ID} is the identity space defined in [HMS⁺07]. The certifier chooses a random number $s \in \mathbb{Z}_p^*$ and a generator g_1 of \mathbb{G}_1^* . The certifier then calculates system's master public key $CB\text{-}mpk = g_1^s$, where s is the master secret key $CB\text{-}msk$. The system's parameter $CB\text{-}params$ is $\{\mathbb{G}_1, \mathbb{G}_T, p, e, g_1, H_0, H_1, H_2, CB\text{-}mpk, \Gamma\}$.
- **CB-UserCreate:** The user ID chooses a random number $x_{ID} \in \mathbb{Z}_p^*$ and sets x_{ID} as the secret key. Here the valid secret key value space is $\mathcal{SK}^{CB} = \mathbb{Z}_p^*$. User ID can also calculate the public key $CB\text{-}PK_{ID} = g_1^{x_{ID}}$. Here the valid public key space is $\mathcal{PK}^{CB} = \mathbb{G}_1^*$.
- **CB-CertGen:** Given a user's identity information ID, the certifier first sets $\hat{ID} = H_2(ID \| CB\text{-}PK_{ID})$, then computes $Cert_{ID} = (H_0(\hat{ID}))^s$.
- **CB-Sign:** For a message m , the user ID sets $\hat{ID} = H_2(ID \| CB\text{-}PK_{ID})$ and computes the signature $CB\text{-}\sigma = Cert_{ID} + (H_1(m \| \hat{ID} \| CB\text{-}PK_{ID}))^{x_{ID}}$.
- **CB-Verify:** Given a pair (m, σ) and user ID's public key $CB\text{-}PK_{ID}$, after setting $\hat{ID} = H_2(ID \| CB\text{-}PK_{ID})$, anyone can check whether $e(CB\text{-}\sigma, g_1) \stackrel{?}{=} e(H_0(\hat{ID}), CB\text{-}mpk) e(H_1(m \| \hat{ID} \| CB\text{-}PK_{ID}), CB\text{-}PK_{ID})$. If the equality holds, this algorithm outputs *true*. Otherwise, this algorithm outputs *false*.

Theorem 4.7 (Security of Concrete Scheme I) *Scheme I is secure (in the random oracle model) against Normal-CB- \mathcal{A}_I and Super-CB- \mathcal{A}_{II} adaptive chosen message and chosen identity attacks, assuming that CDH problem is hard in \mathbb{G}_1 .*

Proof: The correctness of this theorem is due to Theorem 4.1 and the underlying certificateless signature scheme is provably secure (in the random oracle model) against Normal-CL- \mathcal{A}_I and Super-CL- \mathcal{A}_{II} if CDH problem is hard in \mathbb{G}_1 [HMS⁺07].

4.6.2 Scheme II

The scheme described in this section is based on the certificateless signature scheme in Section 4.3 of [HMS⁺07]. The first four algorithms are the same as those defined in Section 4.6.1, with the only difference that H_1 is defined as $\{0, 1\}^* \rightarrow \mathbb{Z}_p$. The CB-Sign and CB-Verify algorithms are described as follows:

- **CB-Sign:** For a message m , the user ID sets $\hat{\text{ID}} = H_2(\text{ID} \| \text{CB-PK}_{\text{ID}})$ and computes the signature $\sigma = (u, v, W)$ where
 - $u = H_1(m \| \hat{\text{ID}} \| \text{CB-PK}_{\text{ID}} \| g_1^{r_1} \| e(g_1, g_1)^{r_2})$ for random numbers $r_1, r_2 \in \mathbb{Z}_p$ chosen by user ID; and
 - $v = r_1 - ux_{\text{ID}} \pmod{p}$, $W = g_1^{r_2} - (\text{Cert}_{\text{ID}})^u$.
- **CB-Verify:** Given a message/signature pair $(m, \sigma = (u, v, W))$, ID's public key CB-PK_{ID} , by setting $\hat{\text{ID}} = H_2(\text{ID} \| \text{CB-PK}_{\text{ID}})$ anyone can check whether $u \stackrel{?}{=} H_1(m \| \hat{\text{ID}} \| \text{CB-PK}_{\text{ID}} \| g_1^v + (\text{CB-PK}_{\text{ID}})^u \| e(W, g_1) e(\text{CB-mpk}, H_0(\hat{\text{ID}}))^u)$. If the equality holds, this algorithm outputs *true*. Otherwise, this algorithm outputs *false*.

Theorem 4.8 (Security of Concrete Scheme II) *Scheme II is secure (in the random oracle model) against Super-CB- \mathcal{A}_I and Super-CB- \mathcal{A}_{II} adaptive chosen message and chosen identity attacks, assuming that CDH problem is hard in \mathbb{G}_1 .*

Proof: The correctness of this theorem is due to Theorem 4.1 and the underlying certificateless signature scheme is provably secure (in the random oracle model) against Super-CL- \mathcal{A}_I and Super-CL- \mathcal{A}_{II} if CDH problem is hard in \mathbb{G}_1 [HMS⁺07].

Remark 4.5 *Scheme II is the first certificate-based signature scheme which is provably secure against Super Type I and Type II adversary.*

Table 4.3: Efficiency Comparison

Scheme	Length	Signing Cost	Verification Cost
CBS in [LHM ⁺ 07]	$2 \mathbb{G}_1 $	$3E+2PA$	$4BM$
Scheme I	$ \mathbb{G}_1 $	$E+PA$	$3BM$
CBSa in [KPH04]	$3 \mathbb{G}_1 $	$3E$	$2E+3BM+2PA$
Scheme II	$ \mathbb{G}_1 + 2 \mathbb{Z}_p $	$4E+BM+PA$	$3E+2BM+PA$

Table 4.4: Security Level Comparison

Scheme	Security
CBS in [LHM ⁺ 07]	<i>Normal</i> \mathcal{A}_I and \mathcal{A}_{II}
Scheme I	<i>Normal</i> \mathcal{A}_I and <i>Super</i> \mathcal{A}_{II}
CBSa in [KPH04]	<i>Strong</i> \mathcal{A}_I and \mathcal{A}_{II}
Scheme II	<i>Super</i> \mathcal{A}_I and \mathcal{A}_{II}

4.6.3 Efficiency Comparison

We now make a comparison among existing certificate-based signature schemes, which are proposed in [LBSZ08, LHM⁺07, KPH04]³.

When compared to schemes (e.g. the first scheme in [LBSZ08]) without bilinear mapping, our Scheme I and Scheme II require more computational cost but they have a shorter signature length. When compared to schemes (e.g. the second scheme in [LBSZ08]) whose security is proven without random oracles, Scheme I and Scheme II have advantages of shorter system parameter, less computational cost and shorter signature length. The comparison among other schemes with similar constructions to ours is shown in Table 4.3 and Table 4.4. The notations in Table 4.3 are as follows: $|\mathbb{G}_1|$ and $|\mathbb{Z}_p|$ denote the bit length of an element in \mathbb{G}_1 and \mathbb{Z}_p , respectively; E denotes the exponentiation in \mathbb{G}_1 ; BM and PA denote the bilinear mapping operation and point addition in \mathbb{G}_1 , respectively.

As shown in Table 4.4, Scheme I and CBS in [LHM⁺07] have the similar security level (To be more precisely, our Scheme I is provably secure against Normal \mathcal{A}_I and Super \mathcal{A}_{II} , and CBS in [LHM⁺07] is provably secure against Normal \mathcal{A}_I and \mathcal{A}_{II}), while Scheme I has less computational operation and shorter signature length. The certificate-based signature scheme CBSa in [KPH04] is secure

³As the notion of certificate-based signatures is relatively new, those are the only known certificate-based signature schemes with formal security analysis.

against the adversary similar to the strong adversary defined in this chapter, while Scheme II is secure against the super adversary with comparable computational cost and signature length. In addition, the two pairing operations ($e(g_1, g_1)$ and $e(CB\text{-}mpk, H_0(\hat{ID}))$) in Scheme II can be computed in an off-line manner, which can further improve the efficiency of Scheme II. The comparison shows that by applying our generic construction to efficient certificateless signature schemes, one can obtain new certificate-based signature schemes with better performance than existing ones.

4.7 Certificate-based Encryption

This section reviews the definitions of certificate-based encryption and the oracles used in our security model of certificate-based encryption.

4.7.1 Syntax of Certificate-Based Encryption

A certificate-based encryption (CBE) scheme consists of the following algorithms:

1. $\text{CB-Setup}(1^k) \rightarrow (CB\text{-}masterkey, CB\text{-}params)$.

By taking as input a security parameter 1^k , the certifier runs the algorithm CB-Setup to generate the certifier's master key $CB\text{-}masterkey$ and the system parameter $CB\text{-}params$. $CB\text{-}params$ includes the description of a string space Γ , which can be any subset of $\{0, 1\}^*$. $CB\text{-}params$ are publicly accessible by all users in the system.

For the convenience of using the expression, in the remainder of this chapter, we still use $ID \in \Gamma$ to denote a user with the identity information ID . But actually, ID contains more information than the identity information.

2. $\text{CB-SetKeyPair}(CB\text{-}params) \rightarrow (CB\text{-}SK_{ID}, CB\text{-}PK_{ID})$.

The user with the identity ID runs the algorithm CB-SetKeyPair to generate its secret/public key pair $(CB\text{-}SK_{ID}, CB\text{-}PK_{ID}) \in \mathcal{SK}^{CB} \times \mathcal{PK}^{CB}$, by taking as input $CB\text{-}params$. Here, \mathcal{SK}^{CB} denotes the set of valid private key values and \mathcal{PK}^{CB} denotes the set of valid public key values. The descriptions of \mathcal{SK}^{CB} and \mathcal{PK}^{CB} are included in $CB\text{-}params$.

3. $\text{CB-CertGen}(CB\text{-}masterkey, CB\text{-}params, ID, CB\text{-}PK_{ID}) \rightarrow Cert_{ID}$.

The certifier runs the algorithm **CB-CertGen** to generate the certificate $Cert_{ID}$, by taking as input $CB-masterkey$, $CB-params$, ID and its public key $CB-PK_{ID}$.

4. $CB-Encrypt(m, CB-params, ID, CB-PK_{ID}) \rightarrow CB-cipher$.

The sender runs the algorithm **CB-Encrypt** to generate the ciphertext $CB-cipher$, by taking as input a message m to be encrypted, $CB-params$, the receiver's identity ID and its public key $CB-PK_{ID}$.

5. $CB-Decrypt(CB-params, CB-SK_{ID}, Cert_{ID}, CB-cipher) \rightarrow \{m, \perp\}$.

The receiver runs the algorithm **CB-Decrypt** to decrypt the ciphertext. By taking as input $CB-params$, a secret key $CB-SK_{ID}$, a certificate $Cert_{ID}$ and the ciphertext $CB-cipher$, this algorithm returns either a message m or the symbol \perp indicating a decryption failure.

Correctness. Ciphertexts generated by the algorithm **CB-Encrypt** can be successfully decrypted by **CB-Decrypt**. That is,

$$CB-Decrypt(CB-params, CB-SK_{ID}, Cert_{ID}, CB-Encrypt(m, CB-params, ID, CB-PK_{ID})) = m.$$

Remark 4.6 *Our definition is essentially the same as the one given in [ARP05] and does not include the Certificate Consolidate algorithm defined in [Gen03], which has the same output as CB-CertGen [ARP05].*

4.7.2 Adversaries and Oracles

In a certificate-based encryption scheme, the user ID can decrypt the ciphertext if and only if it has the knowledge of both certificate $Cert_{ID}$ and secret key $CB-SK_{ID}$. In other words, one cannot perform a correct decryption with only one secret, either $Cert_{ID}$ or $CB-SK_{ID}$. Thus, two types of adversaries have been considered in [Gen03]: Type I adversary $CB-A_I$ and Type II adversary $CB-A_{II}$. $CB-A_I$ can freely replace the public key of any user but is not allowed to query the target user's certificate. On the other hand, $CB-A_{II}$ simulates the certifier with the ability of producing certificates, but not allowed to replace the target user's public key.

The security of CBE is defined by the game between the adversary and the challenger, who maintains the following two lists.

$L1^{PK} = \{(\text{ID}, CB\text{-}SK_{\text{ID}}, CB\text{-}PK_{\text{ID}})\}$: List of original key pairs of created users.

$L2^{PK} = \{(\text{ID}, CB\text{-}\overline{PK}_{\text{ID}})\}$: List of current public keys of created users.

The following oracles are used in the security definition. We use $\alpha \leftarrow \beta$ to denote the algorithmic action of assigning the value β to α .

1. $\mathcal{O}^{\text{CB-UserCreate}}$: For a fresh input $\text{ID} \in \Gamma$, the oracle runs the algorithm CB-SetKeyPair to generate a secret/public key pair $(CB\text{-}SK_{\text{ID}}, CB\text{-}PK_{\text{ID}}) \in \mathcal{SK}^{CB} \times \mathcal{PK}^{CB}$. It returns $CB\text{-}PK_{\text{ID}}$ as the output. It sets $CB\text{-}\overline{PK}_{\text{ID}} = CB\text{-}PK_{\text{ID}}$, then adds $\{(\text{ID}, CB\text{-}SK_{\text{ID}}, CB\text{-}PK_{\text{ID}})\}$ and $\{(\text{ID}, CB\text{-}\overline{PK}_{\text{ID}})\}$ to $L1^{PK}$ and $L2^{PK}$, respectively. Otherwise, ID has already been created. The oracle browses ID in $L1^{PK}$ and returns $CB\text{-}PK_{\text{ID}}$ as the output. W.l.o.g., we assume that other oracles defined later only respond to the identity which has been created.
2. $\mathcal{O}^{\text{CB-PubKeyReplace}}$: For a query $(\text{ID}, \text{CB-PK}) \in \Gamma \times \mathcal{PK}^{CB}$, the challenger finds the user ID in the list $L2^{PK}$, sets $CB\text{-}\overline{PK}_{\text{ID}} \leftarrow \text{CB-PK}$ and updates the corresponding pair with $(\text{ID}, CB\text{-}\overline{PK}_{\text{ID}})$.
3. $\mathcal{O}^{\text{CB-CertGen}}$: For a query $(\text{ID}, \text{CB-PK})$, the challenger responds with an output of $\text{CB-CertGen}(CB\text{-}masterkey, CB\text{-}params, \text{ID}, \text{CB-PK})$.
4. $\mathcal{O}^{\text{Corruption}}$: For a query ID , the challenger browses the list $L1^{PK}$ and outputs the secret key $CB\text{-}SK_{\text{ID}}$. Note that the output of this oracle is always the original secret key of the user ID .
5. $\mathcal{O}^{\text{CB-Decrypt}}$:
 - (a) $\mathcal{O}^{\text{CB-NormalDecrypt}}$: This oracle performs the decryption using ID 's original secret key and the certificate of ID 's original public key.
For a query $(\text{ID}, CB\text{-}cipher)$, the challenger browses the list $L1^{PK}$ to obtain the corresponding secret/public key pair $(CB\text{-}SK_{\text{ID}}, CB\text{-}PK_{\text{ID}})$. Then, it runs CB-CertGen to obtain the certificate $Cert_{\text{ID}}$ of $CB\text{-}PK_{\text{ID}}$ and responds with the output of $\text{CB-Decrypt}(CB\text{-}SK_{\text{ID}}, Cert_{\text{ID}}, CB\text{-}cipher)$.
 - (b) $\mathcal{O}^{\text{CB-StrongDecrypt}}$: This oracle performs the decryption using the secret key (provided by the adversary) and the certificate of a public key.
For a query $(\text{ID}, CB\text{-}SK_{\text{ID}}, CB\text{-}cipher)$, the challenger first generates the public key $CB\text{-}PK_{\text{ID}}$ based on the secret key $CB\text{-}SK_{\text{ID}}$ provided by the

adversary. The challenger then runs the certificate generation algorithm CB-CertGen to obtain the certificate Cert_{ID} of CB-PK_{ID} and responds with the output of $\text{CB-Decrypt}(\text{CB-SK}_{\text{ID}}, \text{Cert}_{\text{ID}}, \text{CB-cipher})$.

- (c) $\mathcal{O}^{\text{CB-SuperDecrypt}}$: This oracle performs the decryption using the secret key and the certificate of a public key, where the public key is chosen by the adversary.

For a query $(\text{ID}, \text{CB-PK}, \text{CB-cipher})$, the challenger runs the certificate generation algorithm CB-CertGen to obtain the certificate Cert_{ID} of CB-PK and responds with the output of CB-Decrypt by taking $(\text{CB-SK}_{\text{ID}}, \text{Cert}_{\text{ID}}, \text{CB-cipher})$ as the input. Here, CB-SK_{ID} is the secret key extracted by the challenger from CB-PK .

In the random oracle model, adversaries are allowed to make requests to random oracles. We will use the following two notations in the security definition.

(t, q) -adversary: A probabilistic adversary who can make up to q queries in polynomial time t .

$Q \not\rightarrow \{\mathcal{O}^1, \mathcal{O}^2, \dots, \mathcal{O}^n\}$: The query Q cannot appear as a request to any oracle $\mathcal{O} \in \{\mathcal{O}^1, \mathcal{O}^2, \dots, \mathcal{O}^n\}$.

4.8 Security Models of Certificate-based Encryption

This section defines the potential adversaries on certificate-based encryption, which are classified into Normal Adversary, Strong Adversary and Super Adversary. The classification is inspired by the adversaries on certificate-based signature defined in [WMSH09].

4.8.1 Security against Normal Type I Adversary

The essential attacking scenario of Normal Type I adversary (Normal-CB- \mathcal{A}_I) is that the adversary can obtain the decryptions of ciphertexts which are generated using the target user's original private key. This is formally defined as follows.

Initial: The challenger runs the algorithm CB-Setup to generate CB-params and CB-masterkey . A normal type I adversary $\text{CB-}\mathcal{A}_I$ is given CB-params , and CB-masterkey is kept secret by the challenger.

Queries-I: $\text{CB-}\mathcal{A}_I$ can make queries to $\mathcal{O}^{\text{CB-UserCreate}}$, $\mathcal{O}^{\text{CB-PubKeyReplace}}$, $\mathcal{O}^{\text{CB-CertGen}}$, $\mathcal{O}^{\text{Corruption}}$ and $\mathcal{O}^{\text{CB-NormalDecrypt}}$.

Challenge: $\text{CB-}\mathcal{A}_I$ outputs an identity ID^* and two messages $m_0, m_1 \in \mathcal{M}$. Here, \mathcal{M} is the valid message space specified in CB-params . Let $\text{CB-}\overline{\text{PK}}_{\text{ID}^*}$ be the current public key of ID^* on the list $L2^{PK}$. $(\text{ID}^*, \text{CB-}\overline{\text{PK}}_{\text{ID}^*}, m_0, m_1)$ must satisfy the following restrictions:

1. $\text{ID}^* \not\rightarrow \mathcal{O}^{\text{CB-Corruption}}$;
2. $(\text{ID}^*, \text{CB-}\overline{\text{PK}}_{\text{ID}^*}) \not\rightarrow \mathcal{O}^{\text{CB-CertGen}}$; and
3. m_0 and m_1 are of equal length.

In response, the challenger picks a random bit $b \in \{0, 1\}$ and outputs a cipher text $\text{CB-cipher}^* = \text{CB-Encrypt}(m_b, \text{CB-params}, \text{ID}^*, \text{CB-}\overline{\text{PK}}_{\text{ID}^*})$.

Queries-II: $\text{CB-}\mathcal{A}_I$ continues making queries to the same oracles in **Queries-I**, with restrictions that:

1. $\text{ID}^* \not\rightarrow \mathcal{O}^{\text{CB-Corruption}}$;
2. $(\text{ID}^*, \text{CB-}\overline{\text{PK}}_{\text{ID}^*}) \not\rightarrow \mathcal{O}^{\text{CB-CertGen}}$; and
3. $(\text{ID}^*, \text{CB-cipher}^*) \not\rightarrow \mathcal{O}^{\text{CB-NormalDecrypt}}$ if $\text{CB-}\overline{\text{PK}}_{\text{ID}^*}$ is the original public key of ID^* (i.e., the public key generated by $\mathcal{O}^{\text{CB-CreateUser}}$).

Output: After all queries, the adversary $\text{CB-}\mathcal{A}_I$ outputs a guess $b' \in \{0, 1\}$ and wins the game if $b = b'$.

The success probability that a normal type I adaptive chosen ciphertext attacker $\text{Normal-CB-}\mathcal{A}_I$ has in the above game is $\Pr[b = b']$.

Definition 4.1 (Security against Normal-CB- \mathcal{A}_I) A certificate-based encryption scheme is (t, q, ϵ) -secure against a normal type I adversary if no (t, q) -adversary can have success probability at least $\epsilon + 1/2$ in the above game.

4.8.2 Security against Strong Type I Adversary

This subsection defines Strong-CB- \mathcal{A}_I who has the access to the oracle $\mathcal{O}^{\text{CB-StrongEncrypt}}$. At the same time, Strong-CB- \mathcal{A}_I is allowed to corrupt the target user's original secret key. Apparently, Strong-CB- \mathcal{A}_I is more powerful than Normal-CB- \mathcal{A}_I . The security of a certificate-based encryption scheme against Strong-CB- \mathcal{A}_I can be defined by almost the same game in Section 4.8.1, with the differences that:

1. During **Queries-I** and **Queries-II**, the decryption oracle is changed to $\mathcal{O}^{\text{CB-StrongDecrypt}}$.
2. The restrictions in **Challenge** are:
 - (a) $(\text{ID}^*, \text{CB-}\overline{\text{PK}}_{\text{ID}^*}) \rightarrow \mathcal{O}^{\text{CB-CertGen}}$, and
 - (b) m_0 and m_1 are of equal length.
3. The restrictions in **Queries-II** are:
 - (a) $(\text{ID}^*, \text{CB-}\overline{\text{PK}}_{\text{ID}^*}) \rightarrow \mathcal{O}^{\text{CB-CertGen}}$, and
 - (b) $(\text{ID}^*, \cdot, \text{CB-cipher}^*) \rightarrow \mathcal{O}^{\text{CB-StrongDecrypt}}$.

Similarly, the success probability that a strong type I adaptive chosen ciphertext attacker Strong-CB- \mathcal{A}_I has in the above game is $\Pr[b = b']$.

Definition 4.2 (Security against Strong-CB- \mathcal{A}_I) *A certificate-based encryption scheme is (t, q, ϵ) -secure against a strong type I adversary if no (t, q) -adversary can have success probability at least $\epsilon + 1/2$ in the above game.*

4.8.3 Security against Super Type I Adversary

In this subsection, we boost the attack power of the adversary again and define the Super Type I adversary Super-CB- \mathcal{A}_I , who has the access to the oracle $\mathcal{O}^{\text{CB-SuperEncrypt}}$. In other words, the adversary can issue decryption queries under the public key of the adversary's choice without providing the corresponding secret key. Obviously, Super-CB- \mathcal{A}_I is more powerful than Strong-CB- \mathcal{A}_I (and hence, more powerful than Normal-CB- \mathcal{A}_I). The security of a certificate-based encryption scheme against a super type I adversary can be defined by almost the same game in Section 4.8.1, with the differences that:

1. During **Queries-I** and **Queries-II**, the decryption oracle is changed to $\mathcal{O}^{\text{CB-SuperDecrypt}}$.
2. The restrictions in **Challenge** are:
 - (a) $(\text{ID}^*, \text{CB-}\overline{\text{PK}}_{\text{ID}^*}) \dashrightarrow \mathcal{O}^{\text{CB-CertGen}}$, and
 - (b) m_0 and m_1 are of equal length.
3. The restrictions in **Queries-II** are:
 - (a) $(\text{ID}^*, \text{CB-}\overline{\text{PK}}_{\text{ID}^*}) \dashrightarrow \mathcal{O}^{\text{CB-CertGen}}$, and
 - (b) $(\text{ID}^*, \text{CB-}\overline{\text{PK}}_{\text{ID}^*}, \text{CB-cipher}^*) \dashrightarrow \mathcal{O}^{\text{CB-SuperDecrypt}}$.

Similarly, the success probability that a super type I adaptive chosen ciphertext attacker $\text{Strong-CB-}\mathcal{A}_I$ has in the above game is $\Pr[b = b']$.

Definition 4.3 (Security against Super-CB- \mathcal{A}_I) *A certificate-based encryption scheme is (t, q, ϵ) -secure against a super type I adversary if no (t, q) -adversary can have success probability at least $\epsilon + 1/2$ in the above game.*

4.8.4 Security Against Type II Adversary

In certificate-based encryption, a type II adversary $\text{CB-}\mathcal{A}_{II}$ simulates the certifier with the master secret key, and thus can produce certificates. As the type I adversary, the type II adversary $\text{CB-}\mathcal{A}_{II}$ can be classified into $\text{Normal-CB-}\mathcal{A}_{II}$, $\text{Strong-CB-}\mathcal{A}_{II}$ and $\text{Super-CB-}\mathcal{A}_{II}$. Notice that $\text{CB-}\mathcal{A}_{II}$ can decrypt any ciphertexts if it has the user's secret key, and thus $\mathcal{O}^{\text{CB-StrongSign}}$ becomes meaningless in this case. Therefore, for a type II adversary $\text{CB-}\mathcal{A}_{II}$, it is sufficient to define two types of adversaries, namely $\text{Normal-CB-}\mathcal{A}_{II}$ and $\text{Super-CB-}\mathcal{A}_{II}$. Their definitions are described as follows.

Initial: The challenger runs the algorithm CB-Setup and sends $\text{CB-}\mathcal{A}_{II}$ the system parameters CB-params and the master secret key CB-masterkey .

Queries-I: $\text{CB-}\mathcal{A}_{II}$ can adaptively make requests to $\mathcal{O}^{\text{CB-UserCreate}}$, $\mathcal{O}^{\text{CB-PubKeyReplace}}$, $\mathcal{O}^{\text{CB-Corruption}}$ and $\mathcal{O}^{\text{CB-Decrypt}} \in \{\mathcal{O}^{\text{CB-NormalDecrypt}}, \mathcal{O}^{\text{CB-SuperDecrypt}}\}$.

Challenge: $\text{CB-}\mathcal{A}_{II}$ outputs an identity ID^* and two message $m_0, m_1 \in \mathcal{M}$ of equal length. Here, \mathcal{M} denotes the message space. Let $\text{CB-PK}_{\text{ID}^*}$ be the original public key of ID^* on the list $L1^{PK}$. The challenger chooses a random $b \in \{0, 1\}$ and responds with a cipher text $\text{CB-cipher}^* = \text{CB-Encrypt}(m_b, \text{CB-params}, \text{ID}^*, \text{CB-PK}_{\text{ID}^*})$. The restriction is that $\text{ID}^* \not\rightarrow \mathcal{O}^{\text{CB-Corruption}}$.

Queries-II: $\text{CB-}\mathcal{A}_{II}$ continues making queries to the same oracles in **Queries-I**, with restrictions that:

1. $\text{ID}^* \not\rightarrow \mathcal{O}^{\text{CB-Corruption}}$,
2. If $\mathcal{O}^{\text{CB-Decrypt}} = \mathcal{O}^{\text{CB-NormalDecrypt}}$, then $(\text{ID}^*, \text{CB-cipher}^*) \not\rightarrow \mathcal{O}^{\text{CB-Decrypt}}$,
and
3. If $\mathcal{O}^{\text{CB-Decrypt}} = \mathcal{O}^{\text{CB-SuperDecrypt}}$, then $(\text{ID}^*, \text{CB-PK}_{\text{ID}^*}, \text{CB-cipher}^*) \not\rightarrow \mathcal{O}^{\text{CB-Decrypt}}$, where $\text{CB-PK}_{\text{ID}^*}$ is ID^* 's original public key.

Output: After all queries, $\text{CB-}\mathcal{A}_{II}$ outputs a guess $b' \in \{0, 1\}$ and wins the game if $b = b'$.

The success probability that an adaptive chosen ciphertext attacker $\text{CB-}\mathcal{A}_{II}^{\text{type}}$ has in the above game is $\Pr[b = b']$, where $\text{type} \in \{\text{normal}, \text{super}\}$.

Definition 4.4 (Security against $\text{CB-}\mathcal{A}_{II}$) *A certificate-based encryption scheme is (t, q, ϵ) -secure against a type II adversary if no (t, q) -adversary can have success probability at least $\epsilon + 1/2$ in the above game.*

4.8.5 Security Against Malicious—but—Passive Type II Adversary

This section defines a more powerful type II adversary, known as malicious-but-passive Type II adversary, who is allowed to generate the system parameter and the master secret key. In other words, at the very beginning of the setup phase, the third party certifier has already been malicious by generating the master secret key, instead of being given the key pair by the challenger who actually generates the key pair honestly. Such attacks were first introduced to certificateless cryptosystems in [ACL⁺07]. However, as in [ACL⁺07], we assume that the certifier is passive, in the sense that the certifier would not actively replace the target user's public key or corrupt the target user's secret key, even though the certifier is malicious. The

security model of certificate-based encryption in [Gen03] also captures the essence of those attacks. The malicious-but-passive Type II adversary is defined as follows:

Initial: The challenger executes $\text{CB-}\mathcal{A}_{II}$ on the security parameter 1^k . \mathcal{A}_{II} returns the system parameters CB-params .

Queries-I: $\text{CB-}\mathcal{A}_{II}$ can adaptively make requests to $\mathcal{O}^{\text{CB-UserCreate}}$, $\mathcal{O}^{\text{CB-PubKeyReplace}}$, $\mathcal{O}^{\text{CB-Corruption}}$ and $\mathcal{O}^{\text{CB-Decrypt}} \in \{\mathcal{O}^{\text{CB-NormalDecrypt}}, \mathcal{O}^{\text{CB-SuperDecrypt}}\}$. Since CB-params and CB-mpk are generated by \mathcal{A}_{II} , we need to modify the following oracles:

- $\mathcal{O}^{\text{CB-UserCreate}}$: As defined in Section 4.8, this oracle receives an input $\text{ID} \in \Gamma$ and outputs the public key CB-PK_{ID} . After obtaining CB-PK_{ID} , a malicious-but-passive Type II adversary must provide ID 's certificate Cert_{ID} for $(\text{ID}, \text{CB-PK}_{\text{ID}})$. This oracle then adds $(\text{ID}, \text{CB-SK}_{\text{ID}}, \text{CB-PK}_{\text{ID}}, \text{Cert}_{\text{ID}})$ to $L1^{PK}$ and $(\text{ID}, \text{CB-PK}_{\text{ID}}, \text{Cert}_{\text{ID}})$ to $L2^{PK}$ respectively.
- $\mathcal{O}^{\text{CB-PKReplace}}$: For a malicious-but-passive Type II adversary, the input to this oracle should be $(\text{ID}, \text{CB-PK}, \text{Cert}_{\text{ID}})$ where Cert_{ID} is the corresponding certificate of CB-PK under the identity ID . This oracle searches $L2^{PK}$, finds a tuple containing ID and replaces that tuple with $(\text{ID}, \text{CB-PK}, \text{Cert}_{\text{ID}})$.
- $\mathcal{O}^{\text{CB-SuperDecrypt}}$: For a malicious-but-passive Type II adversary, the input to this oracle should be $(\text{ID}, \text{CB-PK}, \text{Cert}_{\text{ID}}, \text{CB-cipher})$ where Cert_{ID} is the corresponding certificate of CB-PK under the identity ID .

Challenge, Queries-II and Output: Same as those defined in Sec. 4.8.4.

The success probability that a malicious but passive $\text{CB-}\mathcal{A}_{II}^{\text{type}}$ has in the above game is $\Pr[b = b']$, where $\text{type} \in \{\text{normal}, \text{super}\}$.

Definition 4.5 (Security against Malicious but Passive $\text{CB-}\mathcal{A}_{II}$) *A certificate-based encryption scheme is (t, q, ϵ) -secure against a malicious but passive type II adversary if no (t, q) -adversary can have success probability at least $\epsilon + 1/2$ in the above game.*

4.9 Generic Construction of Certificate-based Encryption

In this section, we show how to convert a certificateless encryption scheme (CLE) into a certificate-based encryption scheme CBE. In our construction, we employ a hash function $H : \Gamma \times \mathcal{PK}^{CB} \rightarrow \mathcal{ID}^{CL}$, where Γ is the identity information space in the certificate-based system, \mathcal{PK}^{CB} is the public key space in certificate-based system and \mathcal{ID}^{CL} denotes the space of identities in the certificateless cryptosystem.

4.9.1 Syntax of Certificateless Encryption

First we will briefly review the algorithms used in the certificateless encryption. A certificateless encryption (CLE) scheme is defined by following algorithms. In order to distinguish from the identity information in the certificate-based system (which is denoted as ID), we use the notion ID to denote the identity information in the certificateless system. For other information, we put the prefix “CL-” to specify that this is in the certificateless system. The description of each algorithm is as follows, which are similar as those described in [Den08].

1. $\text{CL-Setup}(1^k) \rightarrow (CL\text{-masterkey}, CL\text{-params})$.

This algorithm takes as input a security parameter 1^k and returns the master key $CL\text{-masterkey}$ and a parameter $CL\text{-params}$. Here, $CL\text{-params}$ is shared in the system, and includes the identity information space \mathcal{ID}^{CL} , the set of valid secret values \mathcal{S}^{CL} and the set of valid public key values \mathcal{PK}^{CL} , etc.

2. $\text{CL-PartialPrivateKeyExtract}(CL\text{-masterkey}, CL\text{-params}, ID) \rightarrow CL\text{-PPK}_{ID}$.

This algorithm takes as input the master key $CL\text{-masterkey}$, the system parameter $CL\text{-params}$ and the user’s identity information $ID \in \mathcal{ID}^{CL}$. The output is the partial private key $CL\text{-PPK}_{ID}$, which will be secretly sent to the user.

3. $\text{CL-SetSecretValue}(CL\text{-params}) \rightarrow CL\text{-SV}_{ID}$.

This algorithm takes as input the system parameter $CL\text{-params}$. It outputs a secret value $CL\text{-SV}_{ID} \in \mathcal{S}^{CL}$.

4. $\text{CL-SetPrivateKey}(CL\text{-params}, CL\text{-PPK}_{ID}, CL\text{-SV}_{ID}) \rightarrow CL\text{-SK}_{ID}$.

This algorithm takes as input the system parameter $CL\text{-params}$, user's partial private key $CL\text{-PPK}$ and secret value $CL\text{-SV}_{ID}$. It outputs the private key $CL\text{-SK}_{ID}$ of the user.

5. $CL\text{-SetPublicKey}(CL\text{-params}, CL\text{-SV}_{ID}) \rightarrow CL\text{-PK}_{ID}$.

This algorithm takes as input the system parameter $CL\text{-params}$ and the user ID 's secret value $CL\text{-SV}_{ID} \in \mathcal{S}^{\mathcal{CL}}$. It outputs the public key $CL\text{-PK}_{ID} \in \mathcal{PK}^{\mathcal{CL}}$.

6. $CL\text{-Encrypt}(m, CL\text{-params}, ID, CL\text{-PK}_{ID}) \rightarrow CL\text{-cipher}$.

This algorithm takes as input the message m to be encrypted, system parameter $CL\text{-params}$, an identity ID and its public key $CL\text{-PK}_{ID} \in \mathcal{PK}^{\mathcal{CL}}$. It outputs a ciphertext $CL\text{-cipher}$.

7. $CL\text{-Decrypt}(CL\text{-params}, CL\text{-SK}_{ID}, CL\text{-cipher}) \rightarrow \{m, \perp\}$.

This algorithm takes as input the system parameter $CL\text{-params}$, a private key $CL\text{-SK}_{ID}$ and a ciphertext $CL\text{-cipher}$. It outputs a message m or a symbol \perp indicating a decryption failure.

Correctness. Ciphertexts generated by the algorithm $CL\text{-Encrypt}$ can be correctly decrypted using $CL\text{-Decrypt}$: For any $CL\text{-cipher} = CL\text{-Encrypt}(m, CL\text{-params}, ID, CL\text{-PK}_{ID})$, $\Pr[CL\text{-Decrypt}(CL\text{-params}, CL\text{-SK}_{ID}, CL\text{-cipher}) = m] = 1$.

Remark 4.7 *It is suggested in [ARP05] that a Set-User-Keys algorithm can be defined to replace both $CL\text{-SetSecretValue}$ and $CL\text{-SetPublicKey}$ algorithms together. Clearly a certificateless encryption scheme proposed in the original way can also be proposed in that new form.*

4.9.2 Generic Construction: CLE-2-CBE

Our generic construction can be described as follows:

1. $CB\text{-Setup}(1^k) \rightarrow (CB\text{-masterkey}, CB\text{-params})$.
 - (a) Run algorithm $CL\text{-Setup}(1^k)$ of CLE to obtain $CL\text{-params}$ and $CL\text{-masterkey}$;
 - (b) Set $CB\text{-params}$ by extending $CL\text{-params}$ to include the description of Γ ;

- (c) $CB\text{-masterkey} \leftarrow CL\text{-masterkey}$.
2. $CB\text{-SetKeyPair}(CB\text{-params}) \rightarrow (CB\text{-SK}_{ID}, CB\text{-PK}_{ID})$.
- (a) $CL\text{-masterkey} \leftarrow CB\text{-masterkey}$;
- (b) Extract $CL\text{-params}$ from $CB\text{-params}$;
- (c) $CB\text{-SK}_{ID} \leftarrow CL\text{-SetSecretValue}(CL\text{-params})$;
- (d) $CB\text{-PK}_{ID} \leftarrow CL\text{-SetPublicKey}(CL\text{-params}, CL\text{-SV}_{ID})$.
3. $CB\text{-CertGen}(CB\text{-masterkey}, CB\text{-params}, ID, CB\text{-PK}_{ID}) \rightarrow Cert_{ID}$.
- (a) $CL\text{-masterkey} \leftarrow CB\text{-masterkey}$;
- (b) Extract $CL\text{-params}$ from $CB\text{-params}$;
- (c) $H(ID, CB\text{-PK}_{ID}) \rightarrow ID \in \mathcal{ID}^{\mathcal{CL}}$;
- (d) $Cert_{ID} \leftarrow CL\text{-PartialPrivateKeyExtract}(CL\text{-masterkey}, CL\text{-params}, ID)$.
4. $CB\text{-Encrypt}(m, CB\text{-params}, ID, CB\text{-PK}_{ID}) \rightarrow CB\text{-cipher}$.
- (a) Extract $CL\text{-params}$ from $CB\text{-params}$;
- (b) $H(ID, CB\text{-PK}_{ID}) \rightarrow ID \in \mathcal{ID}^{\mathcal{CL}}$;
- (c) $CL\text{-PK}_{ID} \leftarrow CB\text{-PK}_{ID}$;
- (d) $CB\text{-cipher} \leftarrow CL\text{-Encrypt}(m, CL\text{-params}, ID, CL\text{-PK}_{ID})$.
5. $CB\text{-Decrypt}(CB\text{-params}, CB\text{-SK}_{ID}, Cert_{ID}, CB\text{-C}) \rightarrow \{m, \perp\}$.
- (a) Extract $CL\text{-params}$ from $CB\text{-params}$;
- (b) $CL\text{-SV}_{ID} \leftarrow CB\text{-SK}_{ID}$; $CL\text{-PPK}_{ID} \leftarrow Cert_{ID}$;
- (c) $CL\text{-SK}_{ID} \leftarrow CL\text{-SetPrivateKey}(CL\text{-params}, CL\text{-PPK}_{ID}, CL\text{-SV}_{ID})$;
- (d) $CL\text{-cipher} \leftarrow CB\text{-C}$;
- (e) Output $CL\text{-Decrypt}(CL\text{-params}, CL\text{-SK}_{ID}, CL\text{-cipher})$.

The correctness of the proposed scheme CLE-2-CBE is ensured by the underlying CLE.

Security Analysis

Theorem 4.9 [*Security of CLE-2-CBE*] *CLE-2-CBE is secure (in the random oracle model) against adversaries defined in Section 4.8, assuming the underlying certificateless encryption scheme CLS satisfying certain security requirements.*

The proof of Theorem 4.9 consists of several lemmas, which demonstrate the security relationship between our generic construction CLE-2-CBE and its underlying certificateless encryption scheme CLE. Please refer to [Den08] for security definitions of CLE.

Lemma 4.10 (Security against Normal-CB- \mathcal{A}_I) *CLE-2-CBE is secure (in the random oracle model) against **Normal-CB- \mathcal{A}_I** defined in Section 4.8.1, if CLE is secure against **Normal-CL- \mathcal{A}_I** defined in [Den08].*

Proof: In the proof, we will regard hash function H as the random oracle and show that if there is a Normal-CB- \mathcal{A}_I who can distinguish a ciphertext of CLE-2-CBE with non-negligible probability greater than $1/2$, then there exists a CL- \mathcal{A}_I who can use Normal-CB- \mathcal{A}_I to distinguish a ciphertext of CLE with almost the same probability in the game of Weak Type Ib security defined in [Den08].

During the simulation, CL- \mathcal{A}_I acts as the challenger of Normal-CB- \mathcal{A}_I . Let CL-Challenger be the challenger of CL- \mathcal{A}_I , who can adaptively make queries to several oracles defined in [Den08]. Please refer to [Den08] for the formal definition of those oracles.

- $\mathcal{O}^{\text{CL-UserCreate}}$: create users in the scenario of certificateless encryption.
- $\mathcal{O}^{\text{CL-PKReplace}}$: replace public keys of created users with the value chosen by the adversary.
- $\mathcal{O}^{\text{SV-Extract}}$: return secret values of created users.
- $\mathcal{O}^{\text{CL-PPKExtract}}$: return partial private keys of created users.
- $\mathcal{O}^{\text{CL-NormalDecrypt}}$: return the decryption of a ciphertext using the original private key of a created user.

Please refer to [Den08] for the formal definition of each oracle.

Initial: The CL-Challenger runs the algorithm CL-Setup of CLE and feeds the CL- \mathcal{A}_I with *CL-params*. CL- \mathcal{A}_I then returns *CB-params* to Normal-CB- \mathcal{A}_I where *CB-params* is defined by extending *CL-params* to include the description of Γ . Before

Normal-CB- \mathcal{A}_I makes any queries, CL- \mathcal{A}_I asks CL-Challenger to create q_{uc} users in the certificateless cryptosystem. Here q_{uc} is the number of queries Normal-CB- \mathcal{A}_I issues to $\mathcal{O}^{\text{CB-UserCreate}}$. CL- \mathcal{A}_I then records the information as $(ID_i, CL-PK_{ID_i})$, $i = 1, 2, \dots, q_{uc}$ in the list CL^{PK} . Here $ID_i \in \mathcal{ID}^{\mathcal{CL}}$ and $CL-PK_{ID_i}$ is the original public key of ID_i in certificateless encryption.

Queries-I: As defined in Section 4.8.1, Normal-CB- \mathcal{A}_I can issue queries to different oracles. Below we show how Normal-CL- \mathcal{A}_I can answer these queries.

- \mathcal{RO} : In the proof, the hash function H is viewed as the random oracle \mathcal{RO} . For a fresh input $(ID, \text{CB-PK}) \in \Gamma \times \mathcal{PK}^{\text{CB}}$, the output of \mathcal{RO} is a random element ID in $\mathcal{ID}^{\mathcal{CL}}$. Normal-CL- \mathcal{A}_I maintains a list $L^{\mathcal{RO}}$ consisting of $(ID, \text{CB-PK}, ID)$.
- $\mathcal{O}^{\text{CB-UserCreate}}$: At any time, Normal-CB- \mathcal{A}_I can request to create the user $ID_i \in \Gamma$ and expect to obtain ID_i 's public key CB-PK_{ID_i} . In response to such queries:
 1. For the i^{th} fresh query ID_i , Normal-CL- \mathcal{A}_I first checks the list CL^{PK} and finds the i^{th} pair $(ID_i, CL-PK_{ID_i})$. Normal-CL- \mathcal{A}_I then sets $\text{CB-PK}_{ID_i} \leftarrow CL-PK_{ID_i}$, $H(ID_i, \text{CB-PK}_{ID_i}) = ID_i$ and adds $(ID_i, \text{CB-PK}_{ID_i}, ID_i)$ into $L^{\mathcal{RO}}$. If $(ID_i, \text{CB-PK}_{ID_i})$ already appears in $L^{\mathcal{RO}}$, then the simulation fails and Normal-CL- \mathcal{A}_I aborts. This, however, occurs only with negligible probability as $|\mathcal{PK}^{\mathcal{CL}}|$ is assumed to be greater than 2^k and k is the security parameter. Otherwise, it adds $(ID_i, \perp, \text{CB-PK}_{ID_i})$ into the list $L1^{PK}$. Meanwhile, it sets $\text{CB-}\overline{\text{PK}}_{ID_i} \leftarrow \text{CB-PK}_{ID_i}$ and adds $(ID_i, \text{CB-}\overline{\text{PK}}_{ID_i})$ into list $L2^{PK}$. Here, the notation \perp means that Normal-CL- \mathcal{A}_I does not know the corresponding secret key CB-SK_{ID_i} .
 2. In addition to maintain $L1^{PK}, L2^{PK}$, Normal-CL- \mathcal{A}_I will keep two additional lists $L1^{ID}$ and $L2^{ID}$ which will help it answer queries from Normal-CB- \mathcal{A}_I . $L1^{ID}$ consists of pairs with the form (ID_i, ID_i) where $ID_i \in \Gamma$ and $ID_i \in \mathcal{ID}^{\mathcal{CL}}$. This list will help Normal-CL- \mathcal{A}_I answer Normal-CB- \mathcal{A}_I 's corruption queries and simulate $\mathcal{O}^{\text{CB-NormalDecrypt}}$. $L2^{ID}$ consists of pairs with the form (ID_i, \overline{ID}_i) where $ID_i \in \Gamma$ and $\overline{ID}_i \in \mathcal{ID}^{\mathcal{CL}}$. In different phases, \overline{ID}_i could be the identity ID_i in the list $L1^{PK}$, or the identity $ID'_i \in \mathcal{ID}^{\mathcal{CL}}$ created at some time later.

For a user ID_i created in this oracle, Normal-CL- \mathcal{A}_I will add (ID_i, ID_i) into list $L1^{ID}$, where $ID_i \in \mathcal{ID}^{\mathcal{CL}}$ is ID_i 's corresponding identity in the list

CL^{PK} . Meanwhile, $\text{Normal-CL-}\mathcal{A}_I$ sets $\overline{ID}_i \leftarrow ID_i$ and adds (ID_i, \overline{ID}_i) into list $L2^{ID}$.

- $\mathcal{O}^{\text{CB-PKReplace}}$: At any time, $\text{Normal-CB-}\mathcal{A}_I$ can replace a public key of a created user ID with the public key CB-PK chosen by himself. In response, $\text{Normal-CL-}\mathcal{A}_I$ first sets $CB\text{-}\overline{PK}_{ID} \stackrel{\S}{\leftarrow} \text{CB-PK}$, then
 1. $\text{Normal-CL-}\mathcal{A}_I$ browses the list $L2^{PK}$ and rewrites the related pair as $(ID, CB\text{-}\overline{PK}_{ID})$. It then browses $L^{\mathcal{R}\mathcal{O}}$.
 2. If $(ID, CB\text{-}\overline{PK}_{ID})$ appears in $L^{\mathcal{R}\mathcal{O}}$ in the tuple $(ID, CB\text{-}\overline{PK}_{ID}, \overline{ID})$, $\text{Normal-CL-}\mathcal{A}_I$ will make a user-create query \overline{ID} to CL-Challenger if \overline{ID} has not been created in the certificateless system. After that, $\text{Normal-CL-}\mathcal{A}_I$ replaces \overline{ID} 's certificateless public key with $CB\text{-}\overline{PK}_{ID}$ and updates the corresponding pair in CL^{PK} with $(\overline{ID}, CB\text{-}\overline{PK}_{ID})$. Finally, $\text{Normal-CL-}\mathcal{A}_I$ browses the list $L2^{ID}$ and updates the related pair with (ID, \overline{ID}) .
 3. Otherwise, $\text{Normal-CL-}\mathcal{A}_I$ sets $H(ID, CB\text{-}\overline{PK}_{ID}) = \overline{ID}$, which is randomly chosen in $\mathcal{ID}^{\mathcal{C}\mathcal{L}}$. It then adds $(ID, CB\text{-}\overline{PK}_{ID}, \overline{ID})$ into $L^{\mathcal{R}\mathcal{O}}$. After that, $\text{Normal-CL-}\mathcal{A}_I$ asks CL-Challenger to create the user \overline{ID} . After creating the identity \overline{ID} , $\text{Normal-CL-}\mathcal{A}_I$ replaces \overline{ID} 's public key with $CB\text{-}\overline{PK}_{ID}$. $\text{Normal-CL-}\mathcal{A}_I$ then updates CL^{PK} by adding $(\overline{ID}, CB\text{-}\overline{PK}_{ID})$. Finally, $\text{Normal-CL-}\mathcal{A}_I$ browses the list $L2^{ID}$ and updates the related pair with (ID, \overline{ID}) .
- $\mathcal{O}^{\text{CB-Corruption}}$: At any time, $\text{Normal-CB-}\mathcal{A}_I$ can request the secret key of a created user ID_i . $\text{Normal-CL-}\mathcal{A}_I$ responds as follows:
 1. Check the list $L1^{ID}$ and finds (ID_i, ID_i) ;
 2. Make a SV-Extract request ID_i to CL-Challenger who will return $CL\text{-}SV_{ID_i}$ to $\text{Normal-CL-}\mathcal{A}_I$. Here, $CL\text{-}SV_{ID_i}$ is the secret value of ID_i when it was created in the certificateless system.
 3. Set $CB\text{-}SK_{ID_i} \leftarrow CL\text{-}SV_{ID_i}$, return it to $\text{Normal-CB-}\mathcal{A}_I$ and update the information in the list $L1^{PK}$ as $(ID_i, CB\text{-}SK_{ID_i}, CB\text{-}PK_{ID_i})$.

Correctness: This oracle should return the user ID_i 's original secret key $CB\text{-}SK_{ID_i}$ when the user was created. Recall that $L1^{ID}$ contains pairs (ID_i, ID_i) $i = 1, 2, \dots, q_{uc}$, where $ID_i \in \mathcal{ID}^{\mathcal{C}\mathcal{L}}$ is the ID_i 's initial corresponding identity

in certificateless system. Meanwhile ID_i has been set as $H(ID_i, CB-PK_{ID_i})$ and $CL-PK_{ID_i} = CB-PK_{ID_i}$ which is the original public key of ID_i . Thus the secret value $CL-SV_{ID_i}$ of ID_i in certificateless system is the same as the secret key $CB-SK_{ID_i}$ of ID_i in certificate-based system.

- $\mathcal{O}^{CB-CertGen}$: At any time, $\text{Normal-CB-}\mathcal{A}_I$ can request the certificate of $(ID, CB-PK)$ where $CB-PK$ is chosen by the adversary itself. $\text{Normal-CL-}\mathcal{A}_I$ will try to find an identity $\overline{ID} \in \mathcal{ID}^{CL}$, whose partial private key is ID 's certificate under the public key $CB-PK$. To achieve that, $\text{Normal-CL-}\mathcal{A}_I$ will check $L^{\mathcal{R}O}$:
 1. If $(ID, CB-PK)$ appears in $L^{\mathcal{R}O}$ in the tuple $(ID, CB-PK, \overline{ID})$, $\text{Normal-CL-}\mathcal{A}_I$ will make a user-create query \overline{ID} to CL-Challenger if \overline{ID} has not been created in certificateless system.
 2. Otherwise, $\text{Normal-CL-}\mathcal{A}_I$ sets $H(ID, CB-PK) = \overline{ID}$, which is randomly chosen in \mathcal{ID}^{CL} . It then adds $(ID, CB-PK, \overline{ID})$ into $L^{\mathcal{R}O}$. After that, $\text{Normal-CL-}\mathcal{A}_I$ asks CL-Challenger to create the user \overline{ID} .

For either case, $\text{Normal-CL-}\mathcal{A}_I$ issues the partial private key query \overline{ID} to CL-Challenger who will return the partial private key $CL-PPK_{\overline{ID}}$. At last, $\text{Normal-CL-}\mathcal{A}_I$ sets $Cert_{ID} \leftarrow CL-PPK_{\overline{ID}}$ and returns it to $\text{Normal-CB-}\mathcal{A}_I$.

- $\mathcal{O}^{CB-NormalDecrypt}$: At any time, $\text{Normal-CB-}\mathcal{A}_I$ can request the decryption of $(ID_i, CB-cipher)$. The $\text{Normal-CL-}\mathcal{A}_I$ first finds the pair (ID_i, ID_i) in the list $L1^{ID}$ and sets $CL-cipher \leftarrow CB-cipher$. Then, $\text{Normal-CL-}\mathcal{A}_I$ issues a certificateless decryption query to CL-Challenger $(ID_i, CL-cipher)$. As defined, $\text{Normal-CL-}\mathcal{A}_I$ will obtain the message m_i such that $m_i = \text{CL-Decrypt}(CL-params, CL-SK_{ID_i}, CL-cipher)$. $\text{Normal-CB-}\mathcal{A}_I$ will return m_i as its answer.

Correctness: Note that for the pair (ID_i, ID_i) in $L1^{ID}$, ID_i is set as $H(ID_i, CB-PK_{ID_i})$ and $CB-PK_{ID_i} = CL-PK_{ID_i}$. Here, $CB-PK_{ID_i}$ is ID_i 's original public key in the list $L1^{PK}$, and the corresponding secret key is $CB-SK_{ID_i}$, which is the same as the secret value $CL-SV_{ID_i}$ in the certificateless system. In addition, $Cert_{ID_i}$ equals the partial private key $CL-PPK_{ID_i}$ in the certificateless system. Therefore, $m_i = \text{CB-Decrypt}(CB-params, CB-SK_{ID_i}, Cert_{ID_i}, CB-cipher)$.

Challenge: $\text{Normal-CB-}\mathcal{A}_I$ outputs an identity ID^* and two messages $m_0, m_1 \in \mathcal{M}$. Here, \mathcal{M} is the valid message space specified in $CB-params$. Let $CB-\overline{PK}_{ID^*}$ be the

current public key of ID^* on the list $L2^{PK}$. In response, $CL-\mathcal{A}_I$ finds $\overline{ID^*}$ on $L2^{ID}$ and sets $(\overline{ID^*}, m_0, m_1)$ as a challenge to **CL-Challenger**, who will return a ciphertext $CL\text{-cipher}^*$. $CL-\mathcal{A}_I$ sets $CB\text{-cipher}^* \leftarrow CL\text{-cipher}^*$ and sends it to **Normal-CB- \mathcal{A}_I** . **Queries-II**: $CB-\mathcal{A}_I$ continues making queries to the same oracles in **Queries-I**, with restrictions described in Section 4.8.1.

Output: After all queries, the adversary **Normal-CB- \mathcal{A}_I** outputs a guess $b' \in \{0, 1\}$ and wins the game if $b = b'$.

If **Normal-CL- \mathcal{A}_I** does not fail in the simulation, it can output b' as its own output and wins the game with the same success probability as **Normal-CB- \mathcal{A}_I** . Considering that **Normal-CL- \mathcal{A}_I** could only fail in simulating H as the random oracle, which only occurs with negligible probability $q_{UC}/2^k$ (q_{UC} is the number of user-create queries). Thus, **Normal-CL- \mathcal{A}_I** wins the game with almost the same probability as **Normal-CB- \mathcal{A}_I** . This completes the proof of Lemma 4.10.

Lemma 4.11 (Security against Strong-CB- \mathcal{A}_I) *CLE-2-CBE is secure (in the random oracle model) against Strong-CB- \mathcal{A}_I defined in Section 4.8.2, if CLE is secure against Strong-CL- \mathcal{A}_I defined in [Den08].*

Proof: In the proof, we will regard hash function H as the random oracle and show that if there is a Strong-CB- \mathcal{A}_I who can distinguish a ciphertext of CLE-2-CBE with non-negligible probability greater than $1/2$, then there exists a Strong-CL- \mathcal{A}_I who can use Strong-CB- \mathcal{A}_I to distinguish a ciphertext of CLE with almost the same probability.

In the following proof, Strong-CL- \mathcal{A}_I acts as the challenger of Strong-CB- \mathcal{A}_I , and can issue requests to its own challenger **CL-Challenger**.

Initial: Same as the proof of Lemma 4.10.

Queries-I: As defined in Section 4.8.2, Strong-CB- \mathcal{A}_I can issue queries to following oracles. We show how Strong-CL- \mathcal{A}_I can answer these queries.

- \mathcal{RO} , $\mathcal{O}^{CB\text{-UserCreate}}$, $\mathcal{O}^{CB\text{-PKReplace}}$, $\mathcal{O}^{CB\text{-Corruption}}$, $\mathcal{O}^{CB\text{-CertGen}}$: These oracles are simulated by Strong-CL- \mathcal{A}_I in the same way as described in the proof of Lemma 4.10.
- $\mathcal{O}^{CB\text{-StrongDecrypt}}$: At any time, Strong-CB- \mathcal{A}_I can request the decryption of $(ID_i, CB\text{-SK}_{ID_i}, CB\text{-cipher})$. Strong-CL- \mathcal{A}_I first generates $CB\text{-PK}_{ID_i}$ from $CB\text{-SK}_{ID_i}$, and lets $ID_i = H(ID_i, CB\text{-PK}_{ID_i})$. Strong-CL- \mathcal{A}_I then asks **CL-Challenger** to create the user ID_i in certificateless cryptosystem. After that,

Strong-CL- \mathcal{A}_I sets $CL-SV_{ID_i} \leftarrow CB-SK_{ID_i}$ and $CL-cipher \leftarrow CB-cipher$, and issues a certificateless decryption query $(ID_i, CL-SV_{ID_i}, CL-cipher)$ to CL-Challenger. Strong-CL- \mathcal{A}_I will return CL-Challenger's response to Strong-CB- \mathcal{A}_I .

Challenge: Strong-CB- \mathcal{A}_I outputs an identity ID^* and two messages $m_0, m_1 \in \mathcal{M}$. Here, \mathcal{M} is the valid message space specified in $CB-params$. Let $CB-\overline{PK}_{ID^*}$ be the current public key of ID^* on the list $L2^{PK}$. $(ID^*, CB-\overline{PK}_{ID^*}, m_0, m_1)$ satisfies the restrictions described in Section 4.8.2. Strong-CL- \mathcal{A}_I browses $L2^{ID}$ and finds \overline{ID}^* , then sets $(\overline{ID}^*, m_0, m_1)$ as a challenge to CL-Challenger, who picks $b \in \{0, 1\}$ and returns $CL-cipher^* = CL-Encrypt(ID^*, CL - \overline{PK}_{\overline{ID}^*})(M_b)$. At last, Strong-CL- \mathcal{A}_I sets $CB-cipher^* \leftarrow CL-Cipher^*$, and returns it to Strong-CB- \mathcal{A}_I .

Queries-II: CB- \mathcal{A}_I continues making queries to the same oracles in **Queries-I**, with restrictions described in Section 4.8.2.

Output: After all queries, the adversary Strong-CB- \mathcal{A}_I outputs a guess $b' \in \{0, 1\}$ and wins the game if $b = b'$.

If Strong-CL- \mathcal{A}_I does not fail in the simulation, it can output the b' as its own output and wins the game with the same success probability as Strong-CB- \mathcal{A}_I . Considering that Strong-CL- \mathcal{A}_I could only fail in simulating H as the random oracle, which only occurs with negligible probability $q_{UC}/2^k$ (q_{UC} is the number of user-create queries). Thus, Strong-CL- \mathcal{A}_I wins the game with almost the same probability as Strong-CB- \mathcal{A}_I . This completes the proof of Lemma 4.11.

Lemma 4.12 (Security against Super-CB- \mathcal{A}_I) *CLE-2-CBE is secure (in the random oracle model) against Super-CB- \mathcal{A}_I defined in Section 4.8.3, if CLE is secure against Super-CL- \mathcal{A}_I defined in [Den08].*

Proof: In the proof, we will regard hash function H as the random oracle and show that if there is a Super-CB- \mathcal{A}_I who can distinguish a ciphertext of CLE-2-CBE with non-negligible probability greater than $1/2$, then there exists a Super-CL- \mathcal{A}_I who can use Super-CB- \mathcal{A}_I to distinguish a ciphertext of CLE with almost the same probability.

In the following proof, Super-CL- \mathcal{A}_I acts as the challenger of Super-CB- \mathcal{A}_I , and can issue requests to its own challenger CL-Challenger.

Initial: Same as the proof of Lemma 4.10.

Queries-I: As defined in Section 4.8.3, Super-CB- \mathcal{A}_I can issue queries to following oracles. We show how Super-CL- \mathcal{A}_I can answer these queries.

- \mathcal{RO} , $\mathcal{O}^{\text{CB-UserCreate}}$, $\mathcal{O}^{\text{CB-PKReplace}}$, $\mathcal{O}^{\text{CB-Corruption}}$, $\mathcal{O}^{\text{CB-CertGen}}$: These oracles are simulated by Super-CL- \mathcal{A}_I in the same way as described in the proof of Lemma 4.10.
- $\mathcal{O}^{\text{CB-SuperDecrypt}}$: At any time, Super-CB- \mathcal{A}_I can request the signature of $(\text{ID}_i, \text{CB-PK}_{\text{ID}_i}, \text{CB-cipher})$. Super-CL- \mathcal{A}_I first lets $ID_i = H(\text{ID}_i, \text{CB-PK}_{\text{ID}_i})$. Super-CL- \mathcal{A}_I sets $CL\text{-PK}_{ID_i} \leftarrow \text{CB-PK}_{\text{ID}_i}$ then asks CL-Challenger to create the user ID_i in certificateless cryptosystem. After that, Super-CL- \mathcal{A}_I issues a certificateless decryption query $(ID_i, CL\text{-cipher})$ to CL-Challenger. Super-CL- \mathcal{A}_I will return CL-Challenger's response to Super-CB- \mathcal{A}_I .

Challenge: Super-CB- \mathcal{A}_I outputs an identity ID^* and two messages $m_0, m_1 \in \mathcal{M}$. Here, \mathcal{M} is the valid message space specified in CB-params . Let $\text{CB-}\overline{\text{PK}}_{\text{ID}^*}$ be the current public key of ID^* on the list $L2^{PK}$. $(\text{ID}^*, \text{CB-}\overline{\text{PK}}_{\text{ID}^*}, m_0, m_1)$ satisfies the restrictions described in Section 4.8.3.

Super-CL- \mathcal{A}_I browses $L2^{ID}$ and finds \overline{ID}^* , then sets $(\overline{ID}^*, m_0, m_1)$ as a challenge to CL-Challenger, who picks $b \in \{0, 1\}$ and returns $CL\text{-cipher}^*$. After that, Super-CL- \mathcal{A}_I sets $\text{CB-cipher}^* \leftarrow CL\text{-cipher}^*$, and returns it to Super-CB- \mathcal{A}_I .

Queries-II: Super-CB- \mathcal{A}_I continues making queries to the same oracles in **Queries-I**, with restrictions described in Section 4.8.3.

Output: After all queries, the adversary Super-CB- \mathcal{A}_I outputs a guess $b' \in \{0, 1\}$ and wins the game if $b = b'$.

If Super-CL- \mathcal{A}_I does not fail in the simulation, it can output the b' as its own output and wins the game with the same success probability as Super-CB- \mathcal{A}_I . Considering that Super-CL- \mathcal{A}_I could only fail in simulating H as the random oracle, which only occurs with negligible probability $q_{UC}/2^k$ (q_{UC} is the number of user-create queries). Thus, Super-CL- \mathcal{A}_I wins the game with almost the same probability as Super-CB- \mathcal{A}_I . This completes the proof of Lemma 4.12.

Lemma 4.13 (Security against CB- \mathcal{A}_{II}) CLE-2-CBE is secure (in the random oracle model) against type II adversary CB- \mathcal{A}_{II} defined in Section 4.8.4, if CLE is secure against CL- \mathcal{A}_{II} .

Proof: In the proof, we will regard hash function H as the random oracle and show that if there is a $\text{CB-}\mathcal{A}_{II}$ who can distinguish a ciphertext of CLE-2-CBE with non-negligible probability greater than $1/2$, then there exists a $\text{CL-}\mathcal{A}_{II}$ who can use $\text{CB-}\mathcal{A}_{II}$ to distinguish a ciphertext of CLE with almost the same probability.

In the following proof, $\text{CB-}\mathcal{A}_{II}$ against the underlying CLE acts as the challenger of a $\text{CB-}\mathcal{A}_{II}$, and can issue requests to its own challenger CL-Challenger .

Initial: The CL-Challenger runs the algorithm CL-Setup of CLE and feeds the $\text{CL-}\mathcal{A}_{II}$ with $\text{CL-}msk$ and $\text{CL-}params$. $\text{CL-}\mathcal{A}_{II}$ then return CB-masterkey and $\text{CB-}params$ to $\text{CB-}\mathcal{A}_{II}$ where CB-masterkey is defined to be $\text{CL-}msk$ and $\text{CB-}params$ is defined by extending $\text{CL-}params$ to include the description of Γ . Before $\text{CB-}\mathcal{A}_{II}$ submits any queries, $\text{CL-}\mathcal{A}_{II}$ asks CL-Challenger to create q_{uc} users in the certificateless cryptosystem. Here q_{uc} is the number of queries $\text{CB-}\mathcal{A}_{II}$ issues to $\mathcal{O}^{\text{CB-UserCreate}}$. $\text{CL-}\mathcal{A}_{II}$ then records the information as $(ID_i, \text{CL-PK}_{ID_i})$, $i = 1, 2, \dots, q_{uc}$ in the list CL^{PK} . Here $ID_i \in \mathcal{ID}^{\text{CL}}$, CL-PK_{ID_i} is the original public key of ID_i in certificateless system.

Queries-I: As defined in Section 4.8.4, $\text{CB-}\mathcal{A}_{II}$ can issue queries to \mathcal{RO} , $\mathcal{O}^{\text{CB-UserCreate}}$, $\mathcal{O}^{\text{CB-PKReplace}}$, $\mathcal{O}^{\text{CB-Corruption}}$, $\mathcal{O}^{\text{CB-NormalDecrypt}}$ (or, $\mathcal{O}^{\text{CB-SuperDecrypt}}$). These oracles are simulated by $\text{CL-}\mathcal{A}_{II}$ in the same way as described in the proof of the security against Type I adversary in Lemma 4.10.

Challenge: $\text{CB-}\mathcal{A}_{II}$ outputs an identity ID^* and two messages $m_0, m_1 \in \mathcal{M}$. Here, \mathcal{M} is the valid message space specified in $\text{CB-}params$. Let CB-PK_{ID^*} be the original public key of ID^* on the list $L1^{PK}$. $(ID^*, \text{CB-PK}_{ID^*}, m_0, m_1)$ satisfies the restrictions described in Section 4.8.4. $\text{CL-}\mathcal{A}_{II}$ browses $L1^{ID}$ and finds ID^* , then sets (ID^*, m_0, m_1) as a challenge to CL-Challenger , who picks $b \in \{0, 1\}$ and returns $\text{CL-cipher}^* = \text{CL-Encrypt}_{(ID^*, \text{CL-PK}_{ID^*})}(M_b)$. At last, $\text{CL-}\mathcal{A}_{II}$ sets $\text{CB-cipher}^* \leftarrow \text{CL-cipher}^*$, and returns it to $\text{CB-}\mathcal{A}_{II}$.

Queries-II: $\text{CB-}\mathcal{A}_{II}$ continues making queries to the same oracles in **Queries-I**, with restrictions described in Section 4.8.4.

Output: After all queries, the adversary $\text{CB-}\mathcal{A}_{II}$ outputs a guess $b' \in \{0, 1\}$ and wins the game if $b = b'$.

If $\text{CL-}\mathcal{A}_{II}$ does not fail in the simulation, it can output the b' as its own output and wins the game with the same success probability as $\text{CB-}\mathcal{A}_{II}$. Considering that $\text{CL-}\mathcal{A}_{II}$ could only fail in simulating H as the random oracle, which only happens

with negligible probability $q_{UC}/2^k$ (q_{UC} is the number of user-create queries). Thus, $\text{CL-}\mathcal{A}_{II}$ wins the game with almost the same probability as $\text{CB-}\mathcal{A}_{II}$. This completes the proof of Lemma 4.13.

Lemma 4.14 (Security against Malicious-but-passive $\text{CB-}\mathcal{A}_{II}$) *CLE-2-CBE is secure (in the random oracle model) against type II adversary $\text{CB-}\mathcal{A}_{II}$ defined in Section 4.8.5, if CLE is secure against $\text{CL-}\mathcal{A}_{II}$.*

Proof: In the proof, we will regard hash function H as the random oracle and show that if there is a Malicious-but-passive- $\text{CB-}\mathcal{A}_{II}$ who can distinguish a ciphertext of CLE-2-CBE with non-negligible probability, then there exists a Malicious-but-passive- $\text{CL-}\mathcal{A}_{II}$ who can use Malicious-but-passive- $\text{CB-}\mathcal{A}_{II}$ to distinguish a ciphertext of CLE with almost the same probability.

In the following proof, Malicious-but-passive- $\text{CL-}\mathcal{A}_{II}$ against the underlying CLE will act as the challenger of a Malicious-but-passive- $\text{CB-}\mathcal{A}_{II}$, and can issue requests to its own challenger CL-Challenger.

Initial: Malicious-but-passive- $\text{CL-}\mathcal{A}_{II}$ executes Malicious-but-passive- $\text{CB-}\mathcal{A}_{II}$ on the security parameter 1^k , who returns CB-params to Malicious-but-passive- $\text{CL-}\mathcal{A}_{II}$. Malicious-but-passive- $\text{CL-}\mathcal{A}_{II}$ then extracts CL-params from CB-params and sends it to its own challenger CL-Challenger.

Queries: As defined in Section 4.8.5, $\text{CB-}\mathcal{A}_{II}$ can issue queries to following oracles. We show how Malicious-but-passive- $\text{CL-}\mathcal{A}_{II}$ can answer these queries:

- \mathcal{RO} : In the proof, the hash function H is viewed as the random oracle \mathcal{RO} . For a fresh input $(\text{ID}, \text{CB-PK}) \in \Gamma \times \mathcal{PK}^{\text{CB}}$, the output of \mathcal{RO} is a random element ID in \mathcal{ID}^{CL} . Malicious-but-passive- $\text{CL-}\mathcal{A}_{II}$ maintains a list $L^{\mathcal{RO}}$ consisting of $(\text{ID}, \text{CB-PK}, ID)$.
- $\mathcal{O}^{\text{CB-UserCreate}}$: At any time Malicious-but-passive- $\text{CB-}\mathcal{A}_{II}$ can request to create the user $\text{ID}_i \in \Gamma$ and expect to obtain ID_i 's public key $\text{CB-PK}_{\text{ID}_i}$. To response:
 1. For the i^{th} fresh query ID_i , Malicious-but-passive- $\text{CL-}\mathcal{A}_{II}$ will choose a random identity $ID_i \in \mathcal{ID}^{\text{CL}}$ and ask CL-Challenger to create ID_i in the certificateless system. After obtaining ID_i 's certificateless public key $\text{CL-PK}_{\text{ID}_i}$, Malicious-but-passive- $\text{CL-}\mathcal{A}_{II}$ will set $\text{CB-PK}_{\text{ID}_i} \leftarrow \text{CL-PK}_{\text{ID}_i}$, $H(\text{ID}_i, \text{CB-PK}_{\text{ID}_i}) = ID_i$ and adds $(\text{ID}_i, \text{CB-PK}_{\text{ID}_i}, ID_i)$ into $L^{\mathcal{RO}}$. If $(\text{ID}_i, \text{CB-PK}_{\text{ID}_i})$ already appears in $L^{\mathcal{RO}}$, then the simulation fails and

Malicious-but-passive-CL- \mathcal{A}_{II} aborts. This, however, happens only with negligible probability as $|\mathcal{PK}^{\mathcal{CL}}|$ is assumed to be greater than 2^k . (Recall that k is system security parameter.) Malicious-but-passive-CL- \mathcal{A}_{II} then sends $CB-PK_{ID_i}$ to Malicious-but-passive-CB- \mathcal{A}_{II} .

2. After obtaining $CB-PK_{ID_i}$, Malicious-but-passive-CB- \mathcal{A}_{II} must calculate the corresponding certificate $Cert_{ID_i}$ and return it to Malicious-but-passive-CL- \mathcal{A}_{II} , who will forward $Cert_{ID_i}$ to CL-Challenger as the partial private key of ID_i in certificateless system. Malicious-but-passive-CL- \mathcal{A}_{II} then adds $(ID_i, \perp, CB-PK_{ID_i}, Cert_{ID_i})$ into the list $L1^{PK}$. Here, the notation \perp means that Malicious-but-passive-CL- \mathcal{A}_{II} does not know the corresponding secret key $CB-SK_{ID_i}$. Meanwhile, Malicious-but-passive-CL- \mathcal{A}_{II} sets $CB-\overline{PK}_{ID_i} \leftarrow CB-PK_{ID_i}$ and adds $(ID_i, CB-\overline{PK}_{ID_i}, Cert_{ID_i})$ into list $L2^{PK}$.
3. In addition to maintain $L1^{PK}, L2^{PK}$, Malicious-but-passive-CL- \mathcal{A}_{II} will keep two additional lists $L1^{ID}$ and $L2^{ID}$, which are the same as those in the proof of Lemma 4.10.

For a user ID_i created in this oracle, Malicious-but-passive-CL- \mathcal{A}_{II} will add (ID_i, ID_i) into list $L1^{ID}$, where $ID_i \in \mathcal{ID}^{\mathcal{CL}}$ is the ID_i 's corresponding identity in the list CL^{PK} . Meanwhile, Malicious-but-passive-CL- \mathcal{A}_{II} sets $\overline{ID}_i \leftarrow ID_i$ and adds (ID_i, \overline{ID}_i) into list $L2^{ID}$.

- $\mathcal{O}^{CB-PKReplace}$: At any time Malicious-but-passive-CB- \mathcal{A}_{II} can replace a public key of a created user ID with the public key $CB-PK'_{ID}$ chosen by himself. At the same time, Malicious-but-passive-CB- \mathcal{A}_{II} must provide the corresponding certificate $Cert'_{ID}$ of $CB-PK'_{ID}$. In response, Malicious-but-passive-CL- \mathcal{A}_{II} first sets $CB-\overline{PK}_{ID} \leftarrow CB-PK'_{ID}$, then

1. Malicious-but-passive-CL- \mathcal{A}_{II} browses the list $L2^{PK}$ and rewrites the related tuple as $(ID, CB-\overline{PK}_{ID}, Cert'_{ID})$.
2. If $(ID, CB-\overline{PK}_{ID})$ appears in $L^{\mathcal{RO}}$ in the tuple $(ID, CB-\overline{PK}_{ID}, \overline{ID})$, Malicious-but-passive-CL- \mathcal{A}_{II} will make a user-create query \overline{ID} to CL-Challenger if \overline{ID} has not been created in the certificateless system. After that, Malicious-but-passive-CL- \mathcal{A}_{II} replaces \overline{ID} 's certificateless public key with $CB-\overline{PK}_{ID}$ and updates the related pair in $L2^{ID}$ with (ID, \overline{ID}) .

3. Otherwise, Malicious-but-passive-CL- \mathcal{A}_{II} sets $H(\text{ID}, CB\text{-}\overline{PK}_{\text{ID}}) = \overline{ID}$, which is randomly chosen in $\mathcal{ID}^{\mathcal{CL}}$. It then adds $(\text{ID}, CB\text{-}\overline{PK}_{\text{ID}}, \overline{ID})$ into $L^{\mathcal{RO}}$ and asks CL-Challenger to create the user \overline{ID} . After creating \overline{ID} , Malicious-but-passive-CL- \mathcal{A}_{II} replaces \overline{ID} 's public key with $CB\text{-}\overline{PK}_{\text{ID}}$ and updates the related pair in L^{ID} with $(\text{ID}, \overline{ID})$.
- $\mathcal{O}^{\text{CB-Corruption}}$ and $\mathcal{O}^{\text{CB-NormalDecrypt}}$: These two are simulated by Malicious-but-passive-CL- \mathcal{A}_{II} as same as described in the proof of Lemma 4.10.
 - $\mathcal{O}^{\text{CB-SuperDecrypt}}$: This is simulated by Malicious-but-passive-CL- \mathcal{A}_{II} as same as described in the proof of Lemma 4.12.

Challenge, Queries-II, Output: Same as those described in the proof of Lemma 4.13.

If Malicious-but-passive-CL- \mathcal{A}_{II} does not fail in the simulation, it can output b' as its own output and wins the game with the same success probability as Malicious-but-passive-CB- \mathcal{A}_{II} . Considering that Malicious-but-passive-CL- \mathcal{A}_{II} could only fail in simulating H as the random oracle, which only occurs with negligible probability $q_{UC}/2^k$ (q_{UC} is the number of user-create queries). Thus, Malicious-but-passive-CL- \mathcal{A}_{II} wins the game with almost the same probability as Malicious-but-passive-CB- \mathcal{A}_{II} . This completes the proof of Lemma 4.14.

4.10 A Concrete Example of CLE-2-CBE

To demonstrate the application of the generic construction, this section describes a concrete certificate-based encryption scheme from a certificateless encryption scheme proposed in [DLP08]. Please refer to Section 2.1 for the bilinear groups and the complexity assumptions we use in the scheme.

4.10.1 A Concrete Scheme

The scheme described in this section is based on the certificateless encryption scheme in Section 4.2 of [DLP08]. It consists of following algorithms.

- **CB-Setup:** Let $(\mathbb{G}_1, \mathbb{G}_T)$ be bilinear groups where $|\mathbb{G}_1| = |\mathbb{G}_T| = p$, for some prime number $p \geq 2^k$, where k is the system security number. Let g be a

generator for \mathbb{G}_1 . Set $g_1 = g^\gamma$, for a random $\gamma \in \mathbb{Z}_p^*$, and pick a group element $g_2 \in \mathbb{G}_1$ and vectors (u', u_1, \dots, u_n) , (v', v_1, \dots, v_n) from \mathbb{G}_1^{n+1} . Define two functions:

$$F_u(\text{ID}) = u' \prod_{i=1}^n u_j^{i_j} \text{ and } F_v(w) = v' \prod_{i=1}^n v_j^{w_j},$$

where $\text{ID} = i_1 i_2 \dots i_n$ and $w = w_1 w_2 \dots w_n$. Let $H_0 : \Gamma \times \mathbb{G}_1 \times \mathbb{G} \rightarrow \mathcal{ID}$, where Γ is the set of identity information in the certificate-based system and $\mathcal{ID} = \{0, 1\}^*$ is the identity space in the certificateless cryptosystem. Select a collision-resistant hash function $H : \{0, 1\}^* \rightarrow \{0, 1\}^n$. The system parameter $CB\text{-params} = (\mathbb{G}_1, \mathbb{G}_T, g, e, p, g_1, g_2, u', u_1, \dots, u_n, v', v_1, \dots, v_n)$ and the master secret key $CB\text{-masterkey}$ is g_2^γ .

- **CB-SetKeyPair:** The user ID chooses a random number $x_{\text{ID}} \in \mathbb{Z}_p^*$, sets x_{ID} as the private key $CB\text{-SK}_{\text{ID}}$ and calculates the public key $CB\text{-PK}_{\text{ID}} = (X_{\text{ID}}, Y_{\text{ID}}) = (g^{x_{\text{ID}}}, g_1^{x_{\text{ID}}})$. Here the valid secret key and public key value spaces are $\mathcal{SK}^{CB} = \mathbb{Z}_p^*$ and $\mathcal{PK}^{CB} = \{(X, Y) \in \mathbb{G}_1 \times \mathbb{G}_1 : e(X, g_1) = e(g, Y)\}$, respectively.
- **CB-CertGen:** Given a user's identity information ID and the public key $CB\text{-PK}_{\text{ID}} = (X_{\text{ID}}, Y_{\text{ID}})$, the certifier sets $\hat{\text{ID}} = H_0(\text{ID}, X_{\text{ID}}, Y_{\text{ID}})$ and computes $Cert_{\text{ID}} = (cert_1, cert_2) = (g_2^\gamma \cdot F_u(\hat{\text{ID}}), g^r)$.
- **CB-Encrypt:** To encrypt a message $m \in \mathcal{M}$ for an entity ID and the public key $CB\text{-PK}_{\text{ID}} = (X_{\text{ID}}, Y_{\text{ID}})$, perform the following steps:
 - Check $e(X_{\text{ID}}, g_1) = e(g, Y_{\text{ID}})$ holds.
 - Set $\hat{\text{ID}} = H_0(\text{ID}, X_{\text{ID}}, Y_{\text{ID}})$.
 - Choose $s \in \mathbb{Z}_p^*$ and compute $CB\text{-cipher} = (c_0, c_1, c_2, c_3) = (m \cdot e(Y_{\text{ID}}, g_2)^s, g^s, F_u(\text{ID})^s, F_v(w)^s)$, where $w = H(c_0, c_1, c_2, \hat{\text{ID}}, CB\text{-PK}_{\text{ID}})$.
- **CB-Decrypt:** For a ciphertext $CB\text{-cipher} = (c_0, c_1, c_2, c_3) \in \mathcal{C}$ where \mathcal{C} is the ciphertext space, perform the following steps:
 - Set $\hat{\text{ID}} = H_0(\text{ID}, X_{\text{ID}}, Y_{\text{ID}})$.
 - Choose $r' \in \mathbb{Z}_p^*$ and compute $S_{\text{ID}} = (S_1, S_2) = (cert_1^{x_{\text{ID}}} \cdot F_u(\hat{\text{ID}})^{r'}, cert_2^{x_{\text{ID}}} \cdot g^{r'}) = (g_2^{\gamma x_{\text{ID}}} \cdot F_u(\hat{\text{ID}})^t, g^t)$, where $t = r x_{\text{ID}} + r'$.

- Check $e(c_1, F_u(\hat{\text{ID}}) \cdot F_v(w)) = e(g, c_2 \cdot c_3)$, where $w = H(c_0, c_1, c_2, \hat{\text{ID}}, \text{CB-PK}_{\text{ID}})$. Reject if the equation does not hold. Otherwise, return

$$m = c_0 \cdot \frac{e(c_2, S_2)}{e(c_1, S_1)}.$$

Theorem 4.15 (Security of the Concrete Scheme) *The above scheme is secure (in the random oracle model) against Super-CB- \mathcal{A}_I and Super-CB- \mathcal{A}_{II} adaptive chosen message and chosen identity attacks, assuming that 3-DDH problem is hard in \mathbb{G} .*

Proof: The correctness of this theorem is due to Theorem 4.9 and that the underlying certificateless encryption scheme is provably secure (in the standard model) against Super-CB- \mathcal{A}_I and Super-CB- \mathcal{A}_{II} if 3-DDH problem is hard in \mathbb{G}_1 [Gen03].

Remark 4.8 *Notice that the underlying CLE scheme [DLP08] is secure under the standard model, but after the conversion the CBE scheme is secure in the random oracle model. This is because the proposed generic construction is only proven secure in the random oracle model. Nevertheless, the resulting CBE scheme retains the merits of the underlying CLE scheme, namely secure against Super Type I and Type II adversaries.*

4.11 Conclusion

Certificateless public key cryptography and certificate-based public key cryptography are two newly proposed public key cryptosystems, both of which have a third party. This chapter described how to convert certificateless signature schemes (resp. certificateless encryption schemes) to certificate-based signature schemes (resp. certificate-based encryption schemes). We defined several new types of adversaries, and gave new security models of certificate-based signatures and certificate-based encryption. Our generic constructions of certificate-based signatures and certificate-based encryption are provably secure (in the random oracle model) under the security models defined in this chapter, if the underlying building blocks satisfy certain security requirements. We also showed that one can obtain efficient certificate-based signature/encryption schemes from certificateless signature/encryption schemes, by giving several concrete instances of our generic constructions. This would save the effort on the design of those schemes in certificate-based public key cryptography.

Chapter 5

Optimistic Fair Exchange with Strong Resolution-Ambiguity

Optimistic fair exchange makes use of a third party called arbitrator to ensure fairness in the exchange. The arbitrator does not need to be always online; instead, it only gets involved if something goes wrong (e.g., one party attempts to cheat or other faults occur). Recent research has shown that an optimistic fair exchange protocol secure in the single-user setting may be insecure in the multi-user setting. This chapter gives a sufficient condition for single-user secure optimistic fair exchange protocols remaining secure in the multi-user setting. The result of this chapter was presented at PKC 2010 [HMS⁺10].

5.1 Introduction

In a fair exchange protocol, two parties can exchange their items in a fair way so that no one can gain any advantage in the process. A simple way to realize fair exchange is to introduce an *online* trusted third party who acts as a mediator: each party sends the item to the trusted third party, who upon verifying the correctness of both items, forwards each item to the other party. A drawback of this approach is that the trusted third party is always involved in the exchange even if both parties are honest and no fault occurs. In practice, the trusted third party could become a bottleneck of the system and is vulnerable to the denial-of-service attack.

OFE (also known as off-line fair exchange) was introduced by Asokan, Schunter and Waidner in [ASW97]. An OFE protocol also needs a third party called “arbitrator”, who is not required to be online all the time. Instead, the arbitrator only gets invoked when something goes wrong (e.g., one party attempts to cheat or other faults occur). An OFE protocol involves three participants, namely the signer, the verifier and the arbitrator. The signer (say, Alice) first issues a verifiable “partial

signature” σ' to the verifier (say, Bob). Bob verifies the validity of σ' and fulfills his obligation if σ' is valid. After that, Alice sends Bob a “full signature” σ to complete the transaction. Thus, if no problem occurs, the arbitrator does not participate in the exchange. However, if Bob does not receive the full signature σ from Alice, Bob can send σ' (and the proof of fulfilling his obligation) to the arbitrator, who will convert σ' to σ for Bob.

An OFE protocol can be *setup-driven* or *setup-free* [ZB06b]. An OFE protocol is called setup-driven if an initial-key-setup procedure between a signer and the arbitrator is involved. On the other hand, an OFE protocol is called setup-free if the signer does not need to contact the arbitrator, except that the signer can obtain and verify the arbitrator’s public key certificate and vice versa. As shown in [DLY07], setup-free is more desirable for the realization of OFE in the multi-user setting. Another notion of OFE is *stand-alone* [ZB06b], which requires that the full signature be an ordinary signature.

5.1.1 Previous Work

As one of the fundamental problems in secure electronic transactions and digital rights management, fair exchange has been studied intensively since its introduction. It is known that optimistic fair exchange can be constructed (in a generic way) using “two signatures” construction [DR03], verifiably encrypted signature [ASW98, ASW00, BGLS03, CD00, LOS⁺06, ZM07, RS09], the sequential two-party multisignature (first introduced by Park, Chong and Siegel [PCS03], and then broken and repaired by Dodis and Reyzin [DR03]), the OR-proof [DLY07], and conventional signature and ring signature [HYWS08b]. In the following, we only review some results which are most relevant to this chapter.

OFE in the Single-user Setting

There are three parties involved in an OFE protocol, which are signer(s), verifier(s) and arbitrator(s). Most OFE schemes consider only the single-user setting; namely there is only one signer. The first formal security model of OFE was proposed in [ASW98, ASW00]. Dodis and Reyzin [DR03] defined a more generalized and unified model for non-interactive OFE, by introducing a new cryptographic primitive called *verifiably committed signature*. In [DR03], the security of a verifiably committed signature scheme (equivalently, an OFE protocol) in the single-user setting consists of three aspects: security against the signer, security against the verifier and

security against the arbitrator. While the arbitrator is not fully trusted, it is still assumed to be semi-trusted in the sense that the arbitrator will not collude with the signer or the verifier. *In the remainder of this chapter, an OFE protocol is single-user secure (or, secure in the single-user setting) means that it is secure in the single-user setting defined in [DR03].* Notice that their definition does not include all security notions of OFE (e.g., abuse-free [GJM99], non-repudiation [MK01, ZG96], timely-termination [ASW98, ASW00] and signer-ambiguity [HYWS08a]), but it does not affect the point we want to make in this chapter. Dodis and Reyzin [DR03] proposed a stand-alone but setup-driven verifiably committed signature scheme from Gap Diffie-Hellman problem. Constructions of stand-alone and setup-free verifiably committed signature were proposed in [ZB06a, ZB06b].

OFE in the Multi-user Setting

The security of non-interactive OFE in the multi-user setting was independently studied in [DLY07] and [ZSM07]. Optimistic fair exchange in the multi-user setting refers to the scenario where there are two or more signers in the system, but items are still exchanged between two parties. This is different from the multi-party exchange which considers the exchange among three or more parties.

In [DLY07], Dodis, Lee and Yum pointed out that the single-user security of OFE cannot guarantee the multi-user security. They presented a simple counterexample which is secure in the single-user setting but is insecure in a multi-user setting. (In the counterexample, a dishonest verifier in the multi-user setting can obtain a full signature without fulfilling the obligation.) Dodis, Lee and Yum defined the multi-user security model of OFE and provided a generic setup-free construction of optimistic fair exchange secure in the multi-user setting [DLY07]. The security of their construction relies on one-way functions in the random oracle model and trapdoor one-way permutations in the standard model. The analysis in [DLY07] shows that two well-known techniques of OFE (namely, constructions based on verifiably encrypted signatures and sequential two-party signatures) remain secure in the multi-user setting if the underlying primitives satisfy some security notions. Independently, Zhu, Susilo and Mu [ZSM07] also demonstrated a verifiably committed signature scheme which is secure in the model defined in [DR03] but is insecure in the multi-user setting. They defined the security notions of verifiably committed signature in the multi-user setting and proposed a concrete construction of multi-user secure stand-alone and setup-free verifiably committed signature [ZSM07]. The

non-interactive version of their scheme uses the Fiat-Shamir technique and requires a hash function, which is viewed as the random oracle in security analysis. Due to [DLY07], multi-user secure stand-alone and setup-free OFE protocols without random oracles can be constructed from verifiably encrypted signature schemes without random oracles [LOS⁺06, ZM07, RS09].

Certified-Key Model and Chosen-Key Model

Most OFE protocols are considered in the *certified-key model* where the user must prove the knowledge of the private key at the key registration phase. Therefore, the adversary is only allowed to make queries about certified public keys. Huang, Yang, Wong and Susilo [HYWS08b, HYWS08a] considered the multi-user security of OFE in the *chosen-key model*, where the adversary can make queries about public keys arbitrarily without requiring to show its knowledge of the corresponding private keys. Optimistic fair exchange protocols secure in the certified-key model may not be secure in the chosen-key model [HYWS08b].

Huang *et al.* [HYWS08b] proposed another generic construction for OFE. Their construction can lead to efficient setup-free OFE protocols that is secure in the standard model and the chosen-key model. Recently, the first efficient ambiguous OFE protocol was proposed in [HYWS08a]. The new protocol is proven secure in the multi-user setting and chosen-key model without relying on the random oracle assumption. Without any doubt, it is more desirable if cryptographic protocols can be proven secure in the chosen-key model. However, in this chapter, the security of OFE is considered in the certified-key model (as defined in [DLY07]), since certified-key model is reasonable and has been widely used in the research of public key cryptography. *In the remainder of this chapter, when we say an OFE protocol is multi-user secure (or, secure in the multi-user setting), it refers that the protocol is secure in the multi-user setting defined in [DLY07] (which is in the certified-key model).*

5.1.2 Motivation

The research on OFE has shown that:

- The single-user security of OFE does not guarantee the multi-user security [DLY07, ZSM07].

- Not all single-user secure OFE protocols are insecure in the multi-user setting [DLY07]. Several single-user secure protocols can be proven secure in the multi-user setting [DLY07].

However, it remains unknown *under which conditions single-user secure OFE protocols will be secure in the multi-user setting?* We believe the investigation of this question not only will provide a further understanding on the security of OFE in the multi-user setting, but also can introduce new constructions of multi-user secure optimistic fair exchange.

5.1.3 Our Contributions

In this chapter, we present a theoretical study of OFE and a new construction of OFE in the multi-user setting.

1. In Section 5.3, we introduce and define a new property: *Strong Resolution-Ambiguity*. Briefly speaking, an OFE protocol has the property of strong resolution-ambiguity if one can transform a partial signature σ' into a full signature σ using signer's private key or arbitrator's private key, and given such a pair (σ', σ) , it is infeasible to tell which key is used in the conversion. While there are some OFE protocols satisfying strong resolution-ambiguity, it is the first time this notion is addressed and formally defined.
2. *For an OFE protocol with strong resolution-ambiguity, we prove that its security in the single-user setting is preserved in the multi-user setting.* More precisely, we show that: (1) the security against the signer and the security against the verifier in the single-user setting are preserved in the multi-user setting for OFE protocols with strong resolution-ambiguity, and (2) the security against the arbitrator in the single-user setting is preserved in the multi-user setting (for OFE protocols either with or without strong resolution-ambiguity).

While there is no evidence showing that strong resolution-ambiguity is a necessary property for (multi-user secure) OFE protocols, our result provides a new approach for the security analysis of OFE protocols in the multi-user setting: One only needs to analyze the security in the single-user setting (rather than the more complex multi-user setting) for OFE protocols with strong resolution-ambiguity.

3. In Section 5.4, we provide a new construction of OFE with strong resolution-ambiguity. Our construction is a variant of the OFE protocol from the verifiably encrypted signature scheme proposed in [LOS⁺06]. The protocol presented in [LOS⁺06] has several desirable properties, e.g., setup-free, stand-alone and multi-user secure without random oracles under computational Diffie-Hellman assumption. Our protocol retains all these properties and is more efficient in generating, transmitting and verifying partial signatures. This however is achieved at the cost of larger key size.

5.2 Definitions of OFE in the Multi-user Setting

In this section, we describe the syntax and security definitions of OFE in the multi-user setting [DLY07].

5.2.1 Syntax of OFE

A setup-free non-interactive OFE protocol involves three parties: signer, verifier and arbitrator. It is defined by the following efficient algorithms. An algorithm is called efficient if it is a probabilistic polynomial-time Turing machine.

- **Setup^{TTP}**. The arbitrator setup algorithm takes as input a parameter **Param**, and gives as output a secret arbitration key **ASK** and a public partial verification key **APK**.
- **Setup^{User}**. The user setup algorithm takes as input **Param** and (optionally) **APK**, and gives as output a private signing key \mathbf{SK}_{U_i} and a public verification key \mathbf{PK}_{U_i} for the user U_i .
- **Sig** and **Ver**. These are similar to signing and verification algorithms in an ordinary digital signature scheme.
 - The signing algorithm **Sig**, run by a signer U_i , takes as input $(m, \mathbf{SK}_{U_i}, \mathbf{APK})$ and gives as output a signature σ_{U_i} on the message m . In fair exchange protocols, signatures generated by **Sig** are called as *full signatures*.
 - The verification algorithm **Ver**, run by a verifier, takes as input $(m, \sigma_{U_i}, \mathbf{PK}_{U_i}, \mathbf{APK})$ and returns **valid** or **invalid**. A signature σ_{U_i} is said to be a valid full signature of m under \mathbf{PK}_{U_i} if $\mathbf{Ver}(m, \sigma_{U_i}, \mathbf{PK}_{U_i}, \mathbf{APK}) = \mathbf{valid}$.

- **PSig** and **PVer**. These are partial signing and verification algorithms, where **PSig** together with **Res** (which will be defined soon) are functionally equivalent to **Sig**.
 - The partial signing algorithm **PSig**, run by a signer U_i , takes as input $(m, \text{SK}_{U_i}, \text{APK})$ and gives as output a signature σ'_{U_i} on m . To distinguish from those produced by **Sig**, signatures generated by **PSig** are called as *partial signatures*.
 - The partial verification algorithm **PVer**, run by a verifier, takes as input $(m, \sigma'_{U_i}, \text{PK}_{U_i}, \text{APK})$ and returns **valid** or **invalid**. A signature σ'_{U_i} is said to be a valid partial signature of m under PK_{U_i} if $\text{PVer}(m, \sigma'_{U_i}, \text{PK}_{U_i}, \text{APK}) = \text{valid}$.
- **Res**. The resolution algorithm **Res** takes as input a valid partial signature σ'_{U_i} of m under PK_{U_i} and the secret arbitration key **ASK**, and gives as output a signature σ_{U_i} . This algorithm is run by the arbitrator for a party U_j , who does not receive the full signature from U_i , but possesses a valid partial signature of U_i and a proof that he/she has fulfilled the obligation to U_i .

Correctness. If each signature is generated according to the protocol specification, then it should pass the corresponding verification algorithms. Namely,

1. $\text{Ver}(m, \text{Sig}(m, \text{SK}_{U_i}, \text{APK}), \text{PK}_{U_i}, \text{APK}) = \text{valid}$.
2. $\text{PVer}(m, \text{PSig}(m, \text{SK}_{U_i}, \text{APK}), \text{PK}_{U_i}, \text{APK}) = \text{valid}$.
3. $\text{Ver}(m, \text{Res}(m, \text{PSig}(m, \text{SK}_{U_i}, \text{APK}), \text{ASK}, \text{PK}_{U_i}), \text{PK}_{U_i}, \text{APK}) = \text{valid}$.

Resolution-Ambiguity [DLY07, DR03, HYWS08b, MK01, ZSM07]. Any “resolved signature” $\text{Res}(m, \text{PSig}(m, \text{SK}_{U_i}, \text{APK}), \text{ASK}, \text{PK}_{U_i})$ is (at least computationally) indistinguishable from the “actual signature” $\text{Sig}(m, \text{SK}_{U_i}, \text{APK})$.

Security of OFE. Intuitively, the fairness of an exchange requires that two parties exchange their items in a fair way so that either each party obtains the other’s item or neither party does. This requirement consists of the security against signer(s), the security against verifier(s) and the security against the arbitrator, which will be defined by the game between the adversary and the challenger. During the game, the challenger will maintain three initially empty lists: (1) *PK-List* contains the public

keys of created users; (2) *PartialSign-List* contains the partial signing queries made by the adversary; and (3) *Resolve-List* contains the resolution queries made by the adversary.

The definitions in the following sections are inspired by those in [DLY07], with modifications which we believe can demonstrate the difference between the single-user security and the multi-user security of OFE.

5.2.2 Security against Signer(s)

In an OFE protocol, the signer should not be able to generate a valid partial signature which cannot be converted into a valid full signature by the arbitrator. This property is defined by the following game.

- **Setup.** The challenger generates the parameter Param and the arbitrator's key pair (APK, ASK) by running $\text{Setup}^{\text{TTP}}$. The adversary \mathcal{A} is given Param and APK .

- **Queries.** Proceeding adaptively, \mathcal{A} can make following queries.

Creating-User-Queries. \mathcal{A} can create a user U_i by making a creating-user query (U_i, PK_{U_i}) . In order to convince the challenger to accept PK_{U_i} (i.e., add PK_{U_i} to the *PK-List*), \mathcal{A} must prove its knowledge of the legitimate private key SK_{U_i} . This can be realized by requiring the adversary to hand over the private key as suggested in [LOS⁺06], or generate a proof of knowledge [BG92] of the private key¹.

Resolution-Queries. For a resolution-query (m, σ', PK) satisfying $\text{PVer}(m, \sigma', \text{PK}, \text{APK}) = \text{valid}$, the challenger first browses *PK-List*. If $\text{PK} \notin \text{PK-List}$, an error symbol “ \top ” will be returned to the adversary. Otherwise, the challenger adds (m, PK) to the *Resolve-List* (if the pair (m, PK) is not there) and responds with an output of $\text{Res}(m, \sigma', \text{ASK}, \text{PK})$.

- **Output.** Eventually, \mathcal{A} outputs a triple $(m_f, \sigma'_f, \text{PK}^*)$ and wins the game if $\text{PK}^* \in \text{PK-List}$, $\text{PVer}(m_f, \sigma'_f, \text{PK}^*, \text{APK}) = \text{valid}$, and $\text{Ver}(m_f, \text{Res}(m_f, \sigma'_f, \text{ASK}, \text{PK}^*), \text{PK}^*, \text{APK}) = \text{invalid}$.

¹We will use the latter approach in the proof. Section 2.4 briefly reviews the definition of proof of knowledge.

Let $\text{Adv OFE}_{\mathcal{A}}$ be the probability that \mathcal{A} wins in the above game, taken over the coin tosses made by \mathcal{A} and the challenger. An adversary \mathcal{A} is said to $(t, q_{CU}, q_R, \epsilon)$ -break the security against signer(s) if in time t , \mathcal{A} makes at most q_{CU} *Creating-User-Queries*, q_R *Resolution-Queries* and $\text{Adv OFE}_{\mathcal{A}}$ is at least ϵ .

Definition 5.1 (Security against Signer(s)) *An OFE protocol is $(t, q_{CU}, q_R, \epsilon)$ -secure against signer(s) if no adversary $(t, q_{CU}, q_R, \epsilon)$ -breaks it.*

By setting $q_{CU} = 1$, we can define the security against the signer in the single-user setting, namely an OFE protocol is (t, q_R, ϵ) -secure against the signer in the single-user setting if no adversary $(t, 1, q_R, \epsilon)$ -breaks it.

5.2.3 Security against Verifier(s)

Briefly speaking, the security against verifier(s) requires that the verifier should not be able to generate a valid partial signature of a new message or generate a valid full signature without the assistance from the signer or the arbitrator.

The first requirement is ensured by the security against the arbitrator, namely even the arbitrator (knowing more than the verifier) cannot succeed in that attack. This will be defined shortly in Section 5.2.4. The second requirement is defined as below.

- **Setup.** The challenger generates the parameter **Param** and the arbitrator's key pair (APK, ASK) by running $\text{Setup}^{\text{TTP}}$. The challenger also generates a key pair $(\text{PK}^*, \text{SK}^*)$ by running $\text{Setup}^{\text{User}}$, and adds PK^* to *PK-List*. The adversary \mathcal{B} is given **Param**, **APK** and PK^* .
- **Queries.** Proceeding adaptively, \mathcal{B} can make all queries defined in Section 5.2.2 and *Partial-Signing-Queries* defined as follows.

Partial-Signing-Queries. For a partial-signing query (m, PK^*) , the challenger responds with an output of $\text{PSig}(m, \text{SK}^*, \text{APK})$. After that, (m, PK^*) is added to the *PartialSign-List*. (\mathcal{B} is allowed to make *Partial-Signing-Queries* only about PK^* as other public keys are created by \mathcal{B} .)

- **Output.** Eventually, \mathcal{B} outputs a pair (m_f, σ_f) and wins the game if $(m_f, \text{PK}^*) \notin \text{Resolve-List}$ and $\text{Ver}(m_f, \sigma_f, \text{PK}^*, \text{APK}) = \text{valid}$.

Let $\text{Adv OFE}_{\mathcal{B}}$ be the probability that \mathcal{B} wins in the above game, taken over the coin tosses made by \mathcal{B} and the challenger. An adversary \mathcal{B} is said to $(t, q_{CU}, q_{PS}, q_R, \epsilon)$ -break the security against verifier(s) if in time t , \mathcal{B} makes at most q_{CU} *Creating-User-Queries*, q_{PS} *Partial-Signing-Queries*, q_R *Resolution-Queries* and $\text{Adv OFE}_{\mathcal{B}}$ is at least ϵ .

Definition 5.2 (Security against Verifier(s)) *An OFE protocol is $(t, q_{CU}, q_{PS}, q_R, \epsilon)$ -secure against verifier(s) if no adversary $(t, q_{CU}, q_{PS}, q_R, \epsilon)$ -breaks it.*

Similarly, we can obtain the definition of the security against the verifier in the single-user setting, namely an OFE protocol is $(t, q_{PS}, q_R, \epsilon)$ -secure against the verifier in the single-user setting if no adversary $(t, 0, q_{PS}, q_R, \epsilon)$ -breaks it.

5.2.4 Security against the Arbitrator

In this section, we will define the security against the arbitrator and prove that the security against the arbitrator in the single-user setting is preserved in the multi-user setting.

The security against the arbitrator requires that the arbitrator, without the partial signature on a message m , should not be able to produce a valid full signature on m^2 . This notion is defined as follows.

- **Setup.** The challenger generates the parameter **Param**, which is given to the adversary \mathcal{C} .
- **Output-I.** \mathcal{C} generates the arbitrator's public key **APK** and sends it to the challenger. (\mathcal{C} is required to prove the knowledge of the legitimate private key **ASK**.) In response, the challenger generates a key pair $(\text{PK}^*, \text{SK}^*)$ by running $\text{Setup}^{\text{User}}$ and adds PK^* to *PK-List*. The adversary \mathcal{C} is given PK^* .
- **Queries.** Proceeding adaptively, \mathcal{C} can make *Creating-User-Queries* (defined in Section 5.2.2) and *Partial-Signing-Queries* (defined in Section 5.2.3).
- **Output-II.** Eventually, \mathcal{C} outputs a pair (m_f, σ_f) and wins the game if $(m_f, \text{PK}^*) \notin \text{PartialSign-List}$ and $\text{Ver}(m_f, \sigma_f, \text{PK}^*, \text{APK}) = \text{valid}$.

²As almost all previous work about OFE, we assume that signer-arbitrator collusion or verifier-arbitrator collusion will not occur. Please refer to [ASW00, DR03] for discussions of those attacks.

Let $\text{Adv OFE}_{\mathcal{C}}$ be the probability that \mathcal{C} wins in the above game, taken over the coin tosses made by \mathcal{C} and the challenger. An adversary \mathcal{C} is said to $(t, q_{CU}, q_{PS}, \epsilon)$ -break the security against the arbitrator if in time t , \mathcal{C} makes at most q_{CU} *Creating-User-Queries*, q_{PS} *Partial-Signing-Queries* and $\text{Adv OFE}_{\mathcal{C}}$ is at least ϵ .

Remark 5.1 *In the game, the adversary must first generate the arbitrator's public key APK before obtaining PK^* or making other queries. This reflects the definition of optimistic fair exchange as APK could be an input of algorithms $\text{Setup}^{\text{User}}$ and PSig . For concrete protocols where these algorithms do not require APK as the input, the adversary can obtain PK^* and/or make partial-signing-queries of PK^* before generating APK.*

Definition 5.3 (Security against the Arbitrator) *An optimistic fair exchange protocol is $(t, q_{CU}, q_{PS}, \epsilon)$ -secure against the arbitrator in the multi-user setting if no adversary $(t, q_{CU}, q_{PS}, \epsilon)$ -breaks it.*

We can obtain the definition of the security against the arbitrator in the single-user setting, namely an OFE protocol is (t, q_{PS}, ϵ) -secure against the arbitrator in the single-user setting if no adversary $(t, 0, q_{PS}, \epsilon)$ -breaks it. The following theorem shows that *the security against the arbitrator in the single-user setting is preserved in the multi-user setting.*

Theorem 5.1 *An OFE protocol is $(t, q_{CU}, q_{PS}, \epsilon)$ -secure against the arbitrator in the multi-user setting if it is $(t + t_1 q_{CU}, q_{PS}, \epsilon)$ -secure against the arbitrator in the single-user setting. Here, t_1 denotes the time unit to respond to one creating-user query.*

Proof. We denote by \mathcal{C}_S the adversary in the single-user setting and \mathcal{C}_M in the multi-user setting. We will show how to convert a successful \mathcal{C}_M to a successful \mathcal{C}_S . At the beginning, \mathcal{C}_S obtains Param from its challenger in the single-user setting.

- **Setup.** Param is given to \mathcal{C}_M .
- **Output-I.** Let APK be the arbitrator's public key created by \mathcal{C}_M in the multi-user setting. APK will be sent to \mathcal{C}_S 's challenger in the single-user setting. \mathcal{C}_S will make use of \mathcal{C}_M to generate a proof of knowledge, namely \mathcal{C}_S will act as a relay in the proof by forwarding all messages from its challenger to \mathcal{C}_M (or, from \mathcal{C}_M to its challenger). At the end of this phase, \mathcal{C}_S will be given a public

key PK^* , which will be forwarded to \mathcal{C}_M as its challenging public key in the multi-user setting.

- **Queries.** We show how \mathcal{C}_S can correctly answer \mathcal{C}_M 's queries.

Creating-User-Queries. For a creating-user query (U_i, PK_{U_i}) , \mathcal{C}_S will add PK_{U_i} to $PK\text{-List}$ if \mathcal{C}_M can generate a proof of knowledge of the legitimate private key.

Partial-Signing-Queries. For a partial-signing query (m, PK^*) , \mathcal{C}_S forwards it to its own challenger and sends the response to \mathcal{C}_M .

- **Output-II.** Eventually, \mathcal{C}_M will output a pair (m_f, σ_f) . \mathcal{C}_S will set (m_f, σ_f) as its own output in the single-user setting.

\mathcal{C}_S will win the game in the single-user setting if \mathcal{C}_M wins the game in the multi-user setting. It follows that the success probability of \mathcal{C}_S will be ϵ if \mathcal{C}_M can $(t, q_{CU}, q_{PS}, \epsilon)$ -break the security against the arbitrator in the multi-user setting.

It remains to show the time consumption in the proof. \mathcal{C}_S 's running time is the same as \mathcal{C}_M 's running time plus the time it takes to answer creating-user-queries, which we assume each query takes time at most t_1 . Therefore, the total time consumption is $t + t_1 q_{CU}$.

We have shown that for an OFE protocol, if there is an adversary $(t, q_{CU}, q_{PS}, \epsilon)$ -breaks the security against the arbitrator in the multi-user setting, then there is an adversary $(t + t_1 q_{CU}, q_{PS}, \epsilon)$ -breaks the security against the arbitrator in the single-user setting. This completes the proof of Theorem 5.1. \square

Section 5.3 will investigate the conditions under which the security against the signer and the security against the verifier in the single-user setting will remain in the multi-user setting.

5.3 Strong Resolution-Ambiguity

This section investigates a new property in optimistic fair exchange, which we call “Strong Resolution-Ambiguity”. We will give the definition of strong resolution-ambiguity and prove that for OFE protocols with that property, the security against the signer and the security against the verifier in the single-user setting are preserved in the multi-user setting. Before giving the formal definition, we first review a generic construction of OFE [DR03].

OFE from Sequential Two-Party Multisignatures

A multisignature scheme allows any subgroup of users to jointly sign a document such that a verifier is convinced that each user of the subgroup participated in the signing. To construct an OFE protocol, one can use a simple type of multisignature, which is called sequential two-party multisignature. In this construction, the signer first generates two key pairs (pk, sk) and (APK, ASK) , where (pk, APK, ASK) are sent to the arbitrator through a secured channel. The signer's private key SK is the pair (sk, ASK) and the arbitrator's private key is ASK . The partial signature σ' of a message m is an ordinary signature generated using sk , and the full signature σ is the multisignature generated using σ' and ASK . Given a valid partial signature, both the arbitrator and the signer can convert it to a full signature using ASK . (Recall that ASK is the arbitrator's private key and part of the signer's private key.) It is thus virtually infeasible to distinguish who (the signer or the arbitrator) converts the partial signature to the full signature. This is the essential requirement of OFE with strong resolution-ambiguity, which is formally defined as follows.

5.3.1 Definition of Strong Resolution-Ambiguity

We first introduce a probabilistic polynomial-time algorithm Convert which allows the signer to convert a partial signature to a full one. The definition of Convert is given as below.

- **Convert.** This algorithm takes as input the signer's private key SK_{U_i} , (optionally) arbitrator's public key APK , a message m and its valid partial signature σ' . The output is the signer's full signature σ on m .

In a trivial case, each OFE protocol has an algorithm $\text{Convert} = \text{Sig}$. (In this case the full signature generated by Convert could be totally independent of the partial signature.) Our interest here is to investigate non-trivial Convert and compare it with the resolution algorithm Res . Recall that, with the knowledge of ASK , one can also convert a partial signature to a full one using Res . This makes the following question interesting: *Given a valid partial signature σ' , what are the differences between full signatures produced by Convert and those produced by Res ?* The answer to this question inspires the definition of strong resolution-ambiguity.

To formally define the strong resolution-ambiguity, we assume the arbitrator's key pair satisfies an NP-relation R_{TRP} , and users' key pairs satisfy another NP-relation R_U . An NP-relation R is a subset of $\{0, 1\}^* \times \{0, 1\}^*$ for which there

exists a polynomial f such that $|y| \leq f(|x|)$ for all $(x, y) \in R$, and there exists a polynomial-time algorithm for deciding membership in R .

In an OFE protocol defined in Section 5.2, let (APK, ASK) be any pair in R_{TP} , and let $(\text{PK}_{U_i}, \text{SK}_{U_i})$ be any pair in R_U . For any pair (m, σ') satisfying $\text{PVer}(m, \sigma', \text{PK}_{U_i}, \text{APK}) = \text{valid}$, we define

- $\mathbb{D}_{\text{Convert}}^{(m, \sigma')}$: probability distribution of full signatures produced by $\text{Convert}(m, \sigma', \text{SK}_{U_i}, \text{APK})$.
- $\mathbb{D}_{\text{Res}}^{(m, \sigma')}$: probability distribution of full signatures produced by $\text{Res}(m, \sigma', \text{PK}_{U_i}, \text{ASK})$.

Definition 5.4 (Strong Resolution-Ambiguity) *An OFE protocol is said to satisfy strong resolution-ambiguity if there exists an algorithm Convert as defined above such that $\mathbb{D}_{\text{Convert}}^{(m, \sigma')}$ is identical to $\mathbb{D}_{\text{Res}}^{(m, \sigma')}$ ³.*

Strong Resolution-Ambiguity and Resolution-Ambiguity: A Brief Comparison

An OFE protocol with strong resolution-ambiguity will satisfy resolution-ambiguity if Sig is defined as $(\text{PSig} + \text{Convert})$, namely the signer first generates a partial signature and then converts it to a full one using Convert . In this case, actual signatures (generated by Sig) are indistinguishable from resolved signatures (generated by Res). However, resolution-ambiguity cannot ensure strong resolution-ambiguity which requires that one can use the signer’s private key to convert a partial signature to a full one and the conversion is indistinguishable from that using the arbitrator’s private key.

5.3.2 Strong Resolution-Ambiguity in Concrete OFE Protocols

It is evident that the generic construction of OFE from sequential two-party multisignature [DR03] (reviewed at the beginning of Section 5.3) has the strong resolution-ambiguity property by defining $\text{Convert} = \text{Res}$. Below are some other concrete examples of OFE with/without strong resolution-ambiguity.

³As we shall show shortly, strong resolution-ambiguity will make adversaries in the single-user setting have almost the same information as the adversaries in the multi-user setting, but “identical distribution” is a very strong requirement.

OFE from Verifiably Encrypted Signatures

Let OFE-VES be OFE protocols constructed from verifiably encrypted signatures. If the algorithm **Sig** is deterministic (e.g., the verifiably encrypted signature scheme in [BGLS03]), then OFE-VES will have the strong resolution-ambiguity property. For any valid partial signature of m , there is only one output of the algorithm **Res**, namely the unique full signature of m . By defining $\text{Convert} = \text{Sig}$, $\mathbb{D}_{\text{Convert}}^{(m, \sigma')}$ and $\mathbb{D}_{\text{Res}}^{(m, \sigma')}$ will be identical and the protocols satisfy strong resolution-ambiguity. OFE-VES with probabilistic **Sig** algorithms could also have the strong resolution-ambiguity property. One example is the OFE protocol from the verifiably encrypted signature scheme proposed in [LOS⁺06]. In [LOS⁺06], the **Sig** algorithm is the signing algorithm in Waters signature [Wat05], and the partial signature σ' is the encryption of the full signature σ using **APK**. After extracting σ from σ' , the arbitrator will randomize σ such that the output of **Res** is a full signature uniformly distributed in the full signature space. This makes the distribution of full signatures produced by **Res** the same as that of full signatures generated by $\text{Convert} = \text{Sig}$.

A Concrete Instance of the Generic Construction in [HYWS08b]

The generic construction of OFE in [HYWS08b] is based on a conventional signature scheme and a ring signature scheme, both of which can be constructed efficiently without random oracles. In the protocol, the signer and the arbitrator first generate their own key pairs. The full signature of a message m is a pair (s_1, s_2) , where s_1 is the signer's conventional signature on the message m , and s_2 is a ring-signature on m and s_1 . Either the signer or the arbitrator is able to generate s_2 . This construction will satisfy strong resolution-ambiguity if the distribution of ring signatures generated by the signer is the same as that of ring signatures generated by the arbitrator (e.g., 2-User ring signature scheme without random oracles [BKM06]).

A Concrete Protocol without Strong Resolution-Ambiguity

One example of OFE protocols *without* strong resolution-ambiguity is the single-user secure but multi-user *insecure* OFE protocol proposed in [DLY07]. In this protocol, the full signature of a message m is $\sigma = (r, \delta)$, where δ is the signer's conventional signature on " $m||y$ ", $y = f(r)$, and f is a trapdoor one-way permutation. The partial signature is defined as $\sigma' = (y, \delta)$. To convert (y, δ) to a full signature, the arbitrator uses his/her private key f^{-1} to compute $r = f^{-1}(y)$ and obtain the full signature (r, δ) . Given a message m and its full signature (r, δ) , it is hard to tell if (r, δ) is produced by **Sig** directly, or first generated by **PSig** and then by **Res**. Thus,

as shown in [DLY07], the property “resolution-ambiguity” is satisfied. On the other hand, this protocol does not have strong resolution-ambiguity as f is a trapdoor one-way permutation. Suppose, otherwise, there is an algorithm **Convert** such that for a partial signature σ' , the outputs of $\text{Convert}(m, \sigma', \text{SK}_{U_i}, f)$ have the same probability distribution as those of $\text{Res}(m, \sigma', \text{PK}_{U_i}, f^{-1})$. Note that for $\sigma' = (y, \delta)$, **Res** will output a pair (r, δ) such that $y = f(r)$. It follows that $\text{Convert}(m, \sigma', \text{SK}_{U_i}, f)$ must also output (r, δ) satisfying $y = f(r)$ if the protocol has strong resolution-ambiguity. This breaks the one-wayness of f , namely given y , there is an efficient algorithm **Convert** which can find r such that $f(r) = y$ without the trapdoor f^{-1} .

Notice that given a partial signature σ' , the signer can generate a full signature σ such that σ is indistinguishable from the one converted by the arbitrator. To do that, the signer needs to maintain a list $\{(r, y) : y = f(r)\}$ when he/she produces the partial signature $\sigma' = (y, \delta)$. Later on, for a partial signature (y, δ) , the signer can search the list and find the matching pair (r, y) . In this case, the signer can generate a full signature (r, δ) which is indistinguishable from the one converted by the resolution algorithm **Res**. However, this approach does not satisfy the definition of **Convert** since it requires an additional input r . (Recall that the inputs of **Convert** are only SK_{U_i} , (m, σ') and **APK**.)

5.3.3 Security of OFE with Strong Resolution-Ambiguity

Theorem 5.1 has shown that the security against the arbitrator in the single-user setting is preserved in the multi-user setting. This section considers the other two security notions, and we will prove that:

1. For OFE protocols with strong resolution-ambiguity, the security against the signer in the single-user setting remains in the multi-user setting (Theorem 5.2).
2. For OFE protocols with strong resolution-ambiguity, the security against the verifier in the single-user setting remains in the multi-user setting (Theorem 5.3).

Theorem 5.2 *An OFE protocol with strong resolution-ambiguity is $(t, q_{CU}, q_R, \epsilon)$ -secure against signers in the multi-user setting, if it is $(t + t_1 q_{CU} + t_2 q_R, q_R, \epsilon/q_{CU})$ -secure against the singer in the single-user setting. Here, t_1 is the time unit depends*

on the validity of the proof of knowledge and t_2 is the time unit depends on the algorithm `Convert` in the protocol.

Proof. We denote by \mathcal{A}_S the adversary in the single-user setting and \mathcal{A}_M in the multi-user setting. In the proof, we use the standard method by showing that for an OFE protocol with strong resolution-ambiguity, a successful \mathcal{A}_M can be converted into a successful \mathcal{A}_S . We first give a high-level description of the proof.

\mathcal{A}_S will act as the challenger of \mathcal{A}_M in the proof and answer all queries from the latter. \mathcal{A}_S will set the challenging public key PK^* of \mathcal{A}_M as its own challenging public key, and set \mathcal{A}_M 's output as its own output. The most difficult part in the proof is how \mathcal{A}_S can correctly answer resolution queries from \mathcal{A}_M . For resolution queries related to PK^* , \mathcal{A}_S can use its own challenger to generate correct responses. However, this is not feasible for resolution queries about other public keys (since \mathcal{A}_S 's challenger only responds to queries about PK^*). Fortunately, such queries can be correctly answered by \mathcal{A}_S if the OFE protocol has strong resolution-ambiguity. For a resolution query $(m, \sigma', \text{PK}_{U_i})$, \mathcal{A}_S can convert σ' to a full signature σ using the algorithm `Convert` and the private key SK_{U_i} . Due to Def. 5.4, this perfectly simulates the real game between \mathcal{A}_M and the challenger in the multi-user setting. The private key SK_{U_i} can be extracted by \mathcal{A}_S due to the validity of the proof of knowledge required in the creating-user phase. The details of the proof are given as follows.

At the beginning, \mathcal{A}_S obtains the `Param` and `APK` from its challenger.

- **Setup.** In this phase, \mathcal{A}_S gives `Param` and `APK` to \mathcal{A}_M .
- **Queries.** We show how \mathcal{A}_S can correctly answer \mathcal{A}_M 's queries.

Creating-User-Queries. For a creating-user query (U_i, PK_{U_i}) , \mathcal{A}_S will ask \mathcal{A}_M to generate a proof of knowledge of the legitimate private key. If \mathcal{A}_M can successfully convince \mathcal{A}_S with probability at least ϱ , then \mathcal{A}_S can extract the private key SK_{U_i} (i.e., $(\text{PK}_{U_i}, \text{SK}_{U_i}) \in R_U$) in time $O(\frac{1}{\varrho - \kappa})$, where κ is the knowledge error associated with the proof of knowledge. After that, \mathcal{A}_S will add PK_{U_i} to the *PK-List* and $(\text{PK}_{U_i}, \text{SK}_{U_i})$ to the *SK-List*, respectively. *SK-List* is an initially empty list and consists of key pairs of created users.

During this phase, \mathcal{A}_S randomly picks a public key $\text{PK}_{U_j} \in \text{PK-List}$ and sends (U_j, PK_j) to its own challenger in the single-user setting as a creating-user query. With SK_{U_j} , \mathcal{A}_S is able to generate a proof of knowledge and

convince the challenger to accept PK_{U_j} .

Resolution-Queries. For a resolution query (m, σ', PK) where $\text{PK} \in \text{PK-List}$,

1. If $\text{PK} = \text{PK}_{U_j}$, \mathcal{A}_S forwards (m, σ', PK) to its own challenger and sends the response to \mathcal{A}_M .
2. Otherwise, \mathcal{A}_S generates the full signature by running the algorithm **Convert** with the private key SK (which is extracted during the creating-user-queries phase). This perfectly simulates the real attacking scenario (i.e., full signatures generated by the arbitrator) if the protocol has strong resolution-ambiguity.

- **Output.** \mathcal{A}_M outputs a triple $(m_f, \sigma'_f, \text{PK}^*)$, which will be set as \mathcal{A}_S 's output.

\mathcal{A}_S will win the game in the single-user setting if \mathcal{A}_M wins the game in the multi-user setting and $\text{PK}^* = \text{PK}_{U_j}$. Therefore, the success probability of \mathcal{A}_S will be ϵ/q_{CU} if $\mathcal{A}_M(t, q_{CU}, q_R, \epsilon)$ -breaks the security against signers in the multi-user setting.

It remains to show the time consumption in the proof. \mathcal{A}_S 's running time is the same as \mathcal{A}_M 's running time plus the time it takes to answer creating-user-queries and resolution-queries. We assume each creating-user-query takes time t_1 (which depends on the validity of the proof of knowledge), and each resolution-query takes time t_2 (which is the time unit to generate a full signature using the algorithm **Convert**). Therefore, the total time consumption is $t + t_1 q_{CU} + t_2 q_R$.

We have shown that for an OFE protocol with strong resolution-ambiguity, if there is an adversary $(t, q_{CU}, q_R, \epsilon)$ -breaks its security against signers in the multi-user setting, then there is an adversary $(t + t_1 q_{CU} + t_2 q_R, q_R, \epsilon/q_{CU})$ -breaks its security against the signer in the single-user setting. This completes the proof of Theorem 5.2.

□

Theorem 5.3 *An OFE protocol with strong resolution-ambiguity is $(t, q_{CU}, q_{PS}, q_R, \epsilon)$ -secure against verifiers in the multi-user setting, if it is $(t + t_1 q_{CU} + t_2 q_R, q_{PS}, q_R, \epsilon)$ -secure against the verifier in the single-user setting. Here, t_1 is the time unit depends on the validity of the proof of knowledge and t_2 is the time unit depends on the algorithm **Convert** in the protocol.*

Proof. We denote by \mathcal{B}_S the adversary in the single-user setting and \mathcal{B}_M in the multi-user setting. At the beginning, \mathcal{B}_S obtains **Param**, **APK** and PK^* from its challenger.

- **Setup.** In this phase, \mathcal{B}_S adds PK^* to the *PK-List* and sends $(\text{Param}, \text{APK}, \text{PK}^*)$ to \mathcal{B}_M .
- **Queries.** We show how \mathcal{B}_S can correctly answer \mathcal{B}_M 's queries.

Creating-User-Queries. For a creating-user query (U_i, PK_{U_i}) , \mathcal{B}_S will ask \mathcal{B}_M to generate a proof of knowledge of the legitimate private key. If \mathcal{B}_M can successfully convince \mathcal{B}_S with probability at least ϱ , then \mathcal{B}_S can extract the private key SK_{U_i} (i.e., $(\text{PK}_{U_i}, \text{SK}_{U_i}) \in R_U$) in time $O(\frac{1}{\varrho - \kappa})$, where κ is the knowledge error associated with the proof of knowledge. After that, \mathcal{B}_S will add PK_{U_i} to the *PK-List* and $(\text{PK}_{U_i}, \text{SK}_{U_i})$ to the *SK-List*, respectively.

Partial-Signing-Queries. For a partial-signing query (m, PK^*) , \mathcal{B}_S forwards it to its own challenger and sends the response to \mathcal{B}_M .

Resolution-Queries. For a resolution query (m, σ', PK) where $\text{PK} \in \text{PK-List}$,

1. If $\text{PK} = \text{PK}^*$, \mathcal{B}_S forwards $(m, \sigma', \text{PK}^*)$ to its own challenger and sends the response to \mathcal{B}_M .
2. Otherwise, \mathcal{B}_S generates the response by running the algorithm `Convert` with the corresponding private key SK on the *SK-List*. This perfectly simulates the real attacking scenario (i.e., full signatures generated by the arbitrator) if the protocol has strong resolution-ambiguity.

- **Output.** \mathcal{B}_M outputs a pair (m_f, σ_f) , which will be set as the output of \mathcal{B}_S in the single-user setting.

\mathcal{B}_S will win the game in the single-user setting if \mathcal{B}_M wins the game in the multi-user setting. Therefore, the success probability of \mathcal{B}_S will be ϵ if \mathcal{B}_M can $(t, q_{CU}, q_{PS}, q_R, \epsilon)$ -break the security against verifiers in the multi-user setting.

It remains to show the time consumption in the proof. \mathcal{B}_S 's running time is the same as \mathcal{B}_M 's running time plus the time it takes to answer creating-user-queries and q_R resolution-queries. We assume each creating-user query takes time t_1 (which depends on the validity of the proof of knowledge), and each resolution-query takes time t_2 (which depends on the algorithm `Convert` in the protocol). Therefore, the total time consumption is $t + t_1 q_{CU} + t_2 q_R$.

We have shown that for an OFE protocol with strong resolution-ambiguity, if there is an adversary $(t, q_{CU}, q_{PS}, q_R, \epsilon)$ -breaks its security against verifiers in the

multi-user setting, then there is an adversary $(t + t_1q_{CU} + t_2q_R, q_{PS}, q_R, \epsilon)$ -breaks its security against the verifier in the single-user setting. This completes the proof of Theorem 5.3. \square

Remark 5.2 *Our analysis only shows that strong resolution-ambiguity is a sufficient condition for single-user secure OFE protocols remaining secure in the multi-user setting. There is no evidence showing that it is also a necessary property for multi-user secure OFE protocols.*

5.4 New OFE Protocol with Strong Resolution-Ambiguity

This section describes a new OFE protocol with strong resolution-ambiguity. The protocol is based on Waters signature [Wat05].

5.4.1 The Proposed Protocol

Let $(\mathbb{G}_1, \mathbb{G}_T)$ be bilinear groups of prime order p and let g be a generator of \mathbb{G}_1 . e denotes the bilinear map $\mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$ defined in Section 2.1. Let n be the bit-string length of the message to be signed. For an element m in $\{0, 1\}^n$, let $\mathcal{M} \subseteq \{1, 2, \dots, n\}$ be the set of all i for which the i^{th} bit m_i is 1. The parameter Param is $(\mathbb{G}_1, \mathbb{G}_T, p, g, e, n)$.

- **Setup^{TTP}**. Given Param, the arbitrator chooses a random number $w \in \mathbb{Z}_p$ and calculates $W = g^w$. The arbitrator's public key APK is W , and the private key ASK is w .
- **Setup^{User}**. Given Param, this algorithm outputs a private signing key $\text{SK}_{U_i} = (x_{U_i}, y_{U_i})$ and a public verification key $\text{PK}_{U_i} = (X_{U_i}, Y_{U_i}, \vec{v}_{U_i})$, where
 1. x_{U_i} and y_{U_i} are randomly chosen in \mathbb{Z}_p .
 2. $X_{U_i} = e(g, g)^{x_{U_i}}$ and $Y_{U_i} = g^{y_{U_i}}$.
 3. \vec{v}_{U_i} is a vector consisting of $n + 1$ elements $V_0, V_1, V_2, \dots, V_n$. All these elements are randomly selected in \mathbb{G}_1 .

- **Sig.** Given a message m , the signer U_i uses the private key x_{U_i} to generate a Waters signature $\sigma = (\sigma_1, \sigma_2)$, where $\sigma_1 = g^{x_{U_i}} \cdot (V_0 \prod_{i \in \mathcal{M}} V_i)^r$, $\sigma_2 = g^r$ and r is a random number in \mathbb{Z}_p .
- **Ver.** Given a message-signature pair (m, σ) and U_i 's public key $\text{PK}_{U_i} = (X_{U_i}, Y_{U_i}, \vec{v}_{U_i})$, this algorithm outputs **valid** if $e(\sigma_1, g) = X_{U_i} \cdot e(V_0 \prod_{i \in \mathcal{M}} V_i, \sigma_2)$. Otherwise, this algorithm outputs **invalid**.
- **PSig.** Given a message m and the arbitrator's public key W , the signer U_i first runs **Sig** to obtain a full signature (σ_1, σ_2) . After that, U_i calculates $\sigma'_1 = \sigma_1 \cdot W^{y_{U_i}}$ and $\sigma'_2 = \sigma_2$. The partial signature σ' is (σ'_1, σ'_2) .
- **PVer.** Given a pair (m, σ') , U_i 's public key PK_{U_i} and arbitrator's public key APK (which is W), one parses σ' as (σ'_1, σ'_2) . This algorithm outputs **valid** if $e(\sigma'_1, g) = X_{U_i} \cdot e(Y_{U_i}, W) \cdot e(V_0 \prod_{i \in \mathcal{M}} V_i, \sigma'_2)$. Otherwise, it outputs **invalid**.
- **Res.** Given a valid partial signature σ' of the message m under a public key $\text{PK}_{U_i} = (X_{U_i}, Y_{U_i}, \vec{v}_{U_i})$, the arbitrator first parses σ' as (σ'_1, σ'_2) . After that, the arbitrator uses his/her private key w to calculate $\sigma_1 = \sigma'_1 \cdot (Y_{U_i})^{-w}$ and $\sigma_2 = \sigma'_2$. The arbitrator then chooses a random number $r' \in \mathbb{Z}_p$ and calculates $\sigma_1^R = \sigma_1 \cdot (V_0 \prod_{i \in \mathcal{M}} V_i)^{r'}$ and $\sigma_2^R = \sigma_2 \cdot g^{r'}$. The output of the algorithm **Res** is (σ_1^R, σ_2^R) .

5.4.2 Scheme Analysis

It is evident that our protocol is setup-free (no interaction between the user and the arbitrator at key generation phase) and stand-alone (the full signature is Waters signature). We now show that it also satisfies resolution-ambiguity and strong resolution-ambiguity.

Resolution-Ambiguity: Due to the random number r' , the resolved signature produced by **Res** is uniformly distributed in the valid signature space of Waters signature. Thus, resolved signatures are indistinguishable from actual signatures produced by **Sig** and the protocol satisfies resolution-ambiguity.

Strong Resolution-Ambiguity: One can find an algorithm **Convert**, which is the same as **Sig**, such that given any partial signature σ' , the outputs of **Convert** are indistinguishable from those produced by **Res**, both of which are uniformly distributed

in the valid signature space of Waters signature. Thus, the proposed protocol also satisfies strong resolution-ambiguity.

Multi-user Security. We now show that the proposed protocol is secure in the multi-user setting.

Theorem 5.4 *The proposed protocol is multi-user secure under computational Diffie-Hellman assumption.*

As the protocol has strong resolution-ambiguity, we only need to prove it is secure in the single-user setting (due to the analysis in Section 5.3.3). We will show that the security of the proposed protocol is based on the existential unforgeability of Waters signature. The proof consists of the following Lemmas.

Lemma 5.5 (Security against the Signer) *The proposed protocol is secure against the signer.*

Proof. We show that any valid partial signature can be converted to a valid full signature by the arbitrator.

If $\sigma' = (\sigma'_1, \sigma'_2)$ is a valid partial signature of a message m under the public key $\text{PK}_{U_i} = (X_{U_i}, Y_{U_i}, \vec{v}_{U_i})$, then $e(\sigma'_1, g) = X_{U_i} \cdot e(Y_{U_i}, W) \cdot e(V_0 \prod_{i \in \mathcal{M}} V_i, \sigma'_2)$. Note that $X_{U_i} = e(g, g)^{x_{U_i}}$ (for some $x_{U_i} \in \mathbb{Z}_p$), $W = g^w$ and $\sigma'_2 = g^r$ (for some $r \in \mathbb{Z}_p$), we have

$$e(\sigma'_1, g) = e(g^{x_{U_i}}, g) \cdot e(Y_{U_i}, g^w) \cdot e(V_0 \prod_{i \in \mathcal{M}} V_i, g^r).$$

Therefore, $\sigma'_1 = g^{x_{U_i}} \cdot Y_{U_i}^w \cdot (V_0 \prod_{i \in \mathcal{M}} V_i)^r$, for some $x_{U_i}, r \in \mathbb{Z}_p$.

In the algorithm **Res**, σ_1 is computed as $\sigma'_1 \cdot Y_{U_i}^{-w}$, and σ_2 is set as σ'_2 . Thus, $\sigma_1 = g^{x_{U_i}} \cdot (V_0 \prod_{i \in \mathcal{M}} V_i)^r$. Note that $X_{U_i} = e(g, g)^{x_{U_i}}$ and $\sigma_2 = g^r$. Therefore, $e(\sigma_1, g) = X_{U_i} \cdot e(V_0 \prod_{i \in \mathcal{M}} V_i, \sigma_2)$, which means (σ_1, σ_2) is a valid full signature in our protocol. This also holds true for (σ_1^R, σ_2^R) , where $\sigma_1^R = \sigma_1 \cdot (V_0 \prod_{i \in \mathcal{M}} V_i)^{r'}$, $\sigma_2^R = \sigma_2 \cdot g^{r'}$ and r' is a random number in \mathbb{Z}_p . Therefore, the output of **Res** will be a valid full signature if the partial signature is valid.

This completes the analysis of the security against the signer. \square

Lemma 5.6 (Security against the Verifier) *Our protocol is $(t, q_{PS}, q_R, \epsilon)$ -secure against the verifier in the single-user setting if Waters signature is $(t + t_1 \cdot q_{PS}, q_R, \epsilon)$ -existentially unforgeable, where t_1 is the time cost to generate one partial signature in the proposed protocol.*

Proof. We will prove that if there is an adversary \mathcal{B} can $(t, 0, q_{PS}, q_R, \epsilon)$ -break the security against the verifier, then there is an algorithm \mathcal{F} can $(t+t_1 \cdot q_{PS}, q_R, \epsilon)$ -break the existential unforgeability of Waters signature.

At the beginning, \mathcal{F} is given the parameter $(\mathbb{G}_1, \mathbb{G}_T, p, g, e, n)$ and a public key (g_1, g_2, \vec{v}) of Waters signature, where $\vec{v} = (V_0, V_1, \dots, V_n)$.

- **Setup.** \mathcal{F} first chooses two random elements $s, t \in \mathbb{Z}_p$, and computes $X^* = e(g_1, g_2), Y^* = (g_1)^{-1} \cdot g^s$ and $W = g_2 \cdot g^t$. Then, \mathcal{F} sets $\text{Param} = (\mathbb{G}_1, \mathbb{G}_T, p, g, e, n)$, the challenge public key $\text{PK}^* = (X^*, Y^*, \vec{v})$, and the arbitrator's public key $\text{APK} = W$. \mathcal{F} also calculates $(g_1)^{-t} \cdot (g_2)^s \cdot g^{st}$ for future use. \mathcal{B} is given $(\text{Param}, \text{APK}, \text{PK}^*)$.
- *Partial-Signing-Queries.* For a partial signing query (m, PK^*) , \mathcal{F} first chooses a random number $r \in \mathbb{Z}_p$, and then computes $\sigma'_1 = (g_1)^{-t} \cdot (g_2)^s \cdot g^{st} \cdot (V_0 \cdot \prod_{i \in \mathcal{M}} V_i)^r$ and $\sigma'_2 = g^r$. (σ'_1, σ'_2) is a valid partial signature on m as

$$\begin{aligned}
& e(\sigma'_1, g) \\
&= e((g_1)^{-t} \cdot (g_2)^s \cdot g^{st} \cdot (V_0 \cdot \prod_{i \in \mathcal{M}} V_i)^r, g) \\
&= e((g_1)^{-t} \cdot (g_2)^s \cdot g^{st}, g) \cdot e((V_0 \cdot \prod_{i \in \mathcal{M}} V_i)^r, g) \\
&= X^* \cdot e(Y^*, W) \cdot e(V_0 \cdot \prod_{i \in \mathcal{M}} V_i, \sigma'_2).
\end{aligned}$$

- *Resolution-Queries.* For a resolution query $(m, \sigma', \text{PK}^*)$ satisfying $\text{PVer}(m, \sigma', \text{PK}^*, \text{APK}) = \text{valid}$, \mathcal{F} sends m^* to its challenger as a signing query of Waters signature. Let the response be (σ_1, σ_2) , which will be sent to \mathcal{B} as the answer.
- **Output.** Eventually, \mathcal{B} will output a pair (m_f, σ_f) . \mathcal{F} will set (m_f, σ_f) as its own forgery of Waters signature.

If \mathcal{B} wins the game, then $(m_f, \text{PK}^*) \notin \text{Resolve-List}$ and $\text{Ver}(m_f, \sigma_f, \text{PK}^*, \text{APK}) = \text{valid}$. In this case, \mathcal{F} has obtained a valid Waters signature σ_f of a new message m_f . (Notice that m_f is not one of signing queries of Waters signature if $(m_f, \text{PK}^*) \notin \text{Resolve-List}$.) It remains to show the time consumption in the proof. \mathcal{F} 's running time is the same as \mathcal{B} 's running time plus the time it takes to answer partial-signing queries, where each query takes the time cost to generate one partial signature in our protocol and is denoted by t_1 . Thus, the total time consumption is $t + t_1 q_{PS}$.

We have shown that there is an algorithm \mathcal{F} can $(t + t_1 q_{PS}, q_R, \epsilon)$ -break the existential unforgeability of Waters signature, if there is an adversary \mathcal{B} can $(t, 0, q_{PS}, q_R, \epsilon)$ -break the security against the verifier of the proposed scheme. This completes the proof of Lemma 5.6. \square

Due to Lemma 5.6 and Theorem 5.3, our protocol is secure against verifiers in the multi-user setting.

Lemma 5.7 (Security against the Arbitrator) *Our protocol is (t, q_{PS}, ϵ) -secure against the arbitrator if Waters signature is $(t + t_1 \cdot q_{PS}, q_{PS}, \epsilon)$ -existentially unforgeable, where t_1 is the time cost to generate one partial signature in the proposed protocol.*

Proof. We will prove that if there is an adversary \mathcal{C} can $(t, 0, q_{PS}, \epsilon)$ -break the security against the arbitrator, then there is an algorithm \mathcal{F} can $(t + t_1 \cdot q_{PS}, q_{PS}, \epsilon)$ -break the existential unforgeability of Waters signature.

At the beginning, \mathcal{F} is given the parameter $(\mathbb{G}_1, \mathbb{G}_T, p, g, e, n)$ and a public key (g_1, g_2, \vec{v}) of Waters signature, where $\vec{v} = (V_0, V_1, \dots, V_n)$.

- **Setup.** \mathcal{F} first sends the parameter $\text{Param} = (\mathbb{G}_1, \mathbb{G}_T, p, g, e, n)$ to \mathcal{C} . As the algorithm $\text{Setup}^{\text{User}}$ in our protocol does not require the arbitrator's public key as input, \mathcal{F} also needs to send the target public key PK^* to \mathcal{C} . To generate PK^* , \mathcal{F} chooses a random element $s \in \mathbb{Z}_p$, and computes $X^* = e(g_1, g_2)$ and $Y^* = g^s$. \mathcal{C} is given the challenge public key $\text{PK}^* = (X^*, Y^*, \vec{v})$.
- **Output-I.** After obtaining Param and PK^* , \mathcal{C} will generate the arbitrator's public key APK and a proof of knowledge of the corresponding secret key ASK .
- **Partial-Signing-Queries.** For a partial-signing query (m, PK^*) , \mathcal{F} sends m^* to its challenger as a signing query of Waters signature. Let the response be (σ_1, σ_2) . After that, \mathcal{F} calculates the partial signature $\sigma'_1 = \sigma_1 \cdot (\text{APK})^s$ and $\sigma'_2 = \sigma_2$. Notice that the generation of (σ'_1, σ'_2) is the same as that defined in PSig . \mathcal{C} is given (σ'_1, σ'_2) .
- **Output-II.** Eventually, \mathcal{C} will output a pair (m_f, σ_f) . \mathcal{F} will set (m_f, σ_f) as its own forgery of Waters signature.

\mathcal{C} wins the game if $(m_f, \text{PK}^*) \notin \text{PartialSign-List}$ and $\text{Ver}(m_f, \sigma_f, \text{PK}^*, \text{APK}) = \text{valid}$. In this case, \mathcal{F} has obtained a valid Waters signature σ_f of a new message m_f . (Notice that m_f is not one of signing queries of Waters signature if $(m_f, \text{PK}^*) \notin \text{PartialSign-List}$.) It remains to show the time consumption in the proof. \mathcal{F} 's running time is the same as \mathcal{C} 's running time plus the time it takes to answer partial-signing queries, where each query takes (at most) the time cost to generate one partial signature in our protocol and is denoted by t_1 . Thus, the total time consumption is $t + t_1 q_{PS}$.

We have shown that there is an algorithm \mathcal{F} can $(t + t_1 q_{PS}, q_{PS}, \epsilon)$ -break the existential unforgeability of Waters signature, if there is an adversary \mathcal{C} can $(t, 0, q_{PS}, \epsilon)$ -break the security against the arbitrator of the proposed protocol. This completes the proof of Lemma 5.7. \square

Due to Lemma 5.7 and Theorem 5.1, our protocol is secure against the arbitrator in the multi-user setting.

5.4.3 Comparison to Previous Protocols

Table. 5.1 compares the known OFE protocols which have the same properties as the newly proposed one (namely, non-interactive, setup-free, stand-alone and multi-user secure without random oracles). The comparison is made from the following aspects: (1) underlying complexity assumption, (2) partial signature size and full signature size, and (3) the computational cost of signing and verifying partial signatures and full signatures. We consider the cost of signing and verifying partial signatures since the signer must generate a partial signature in each exchange, which will be verified by the verifier and could also be checked again by the arbitrator. Therefore, the efficiency of signing and verifying partial signatures is at least as important as that of full signatures.

We use the following notations in the comparison.

- CDH: Computational Diffie-Hellman assumption.
- CT-CDH: Chosen-target computational Diffie-Hellman assumption [Bol03].
- SDH: Strong Diffie-Hellman assumption.
- $|\mathbb{G}_1|$: bit length of an element in \mathbb{G}_1 .
- $|\mathbb{Z}_p|$: bit length of an element in \mathbb{Z}_p .

Table 5.1: Multi-user Secure Stand-Alone and Setup-Free OFE Protocols without Random Oracles

	Our Protocol	[LOS ⁺ 06]	[ZM07]	[RS09]
Assumption	CDH	CDH	CT-CDH	SDH
Full Signature	[Wat05]	[Wat05]	[Wat05]	[BB04]
Signature Size ^{PSig}	$2 \mathbb{G}_1 $	$3 \mathbb{G}_1 $	$2 \mathbb{G}_1 $	$2 \mathbb{G}_1 + \mathbb{Z}_p $
Signing Cost ^{PSig}	$C_W + 1\text{Exp}_{\mathbb{G}_1}$	$C_W + 2\text{Exp}_{\mathbb{G}_1}$	C_W	$C_{BB} + 2\text{Exp}_{\mathbb{G}_1}$
Verification Cost ^{PVer}	$2BM + 1BM$	$3BM$	$2BM + 1BM$	$2BM + 4BM$

- C_W : Computational cost of generating one Waters signature [Wat05].
- C_{BB} : Computational cost of generating one BB signature [BB04].
- $\text{Exp}_{\mathbb{G}_1}$: Exponentiation in \mathbb{G}_1 .
- $\text{Exp}_{\mathbb{G}_1}$: Pre-computable exponentiation in \mathbb{G}_1 .
- BM : Bilinear mapping.
- BM : Pre-computable bilinear mapping.

In Table. 5.1, the most efficient one is the protocol constructed from the verifiably encrypted signature scheme in [RS09], whose security assumption is strong Diffie-Hellman assumption (SDH). The other three protocols are all based on Waters signature, but the security of the protocol in [ZM07] can only be reduced to a stronger assumption: chosen-target computational Diffie-Hellman assumption (CT-CDH). Our protocol and the one proposed in [LOS⁺06] are designed in a similar manner. When compared with [LOS⁺06], our protocol has a shorter partial signature size and is more efficient in signing and verifying partial signatures. This is achieved at the cost of larger key size (one more pair (y_{U_i}, Y_{U_i}) in $\mathbb{Z}_p \times \mathbb{G}_1$).

5.5 Conclusion

This chapter shows several new results about OFE in the multi-user setting. We formally defined the *Strong Resolution-Ambiguity* in OFE and demonstrated several concrete OFE protocols with that property. In the certified-key model, we prove that for OFE protocols with strong resolution-ambiguity, the security in the single-user

setting can guarantee the security in the multi-user setting. In addition to theoretical investigations, a new construction of OFE with strong resolution-ambiguity was proposed. The new protocol is setup-free, stand-alone, and provably secure in the multi-user setting without random oracles.

Chapter 6

Conclusions

Cryptographic protocols introduce extra computational cost to computer systems, which is a significant burden for power-constrained devices. A promising solution is to employ a powerful server to carry out the expensive computations. A server-aided verification signature scheme $\text{SAV-}\Sigma$ consists of a digital signature scheme and a server-aided verification protocol. The purpose of $\text{SAV-}\Sigma$ is to enable signature verifier to perform signature verification with less computational cost, by executing the server-aided verification protocol with the server.

In Chapter 3, we defined the existential unforgeability of server-aided verification signatures, and proved that it includes the existential unforgeability of signature schemes and the soundness of server-aided verification protocols under the same assumption in [GL05]. We also provided the first constructions of existentially unforgeable SAV-BLS and SAV-Waters . The existential unforgeability of SAV-BLS can be reduced to the hardness of BDH problem in the random oracle model. SAV-Waters inherits the desirable property of Waters signature, which can be proven to be existentially unforgeable without random oracles under GBDH assumption. After that, we investigated the collusion between the signer and the server, and defined the security models to capture the collusion attack and its stronger version on server-aided verification signatures. Concrete server-aided verification signature schemes secure against collusion attacks are also provided.

Certificate-based public key cryptography and certificateless public key cryptography are proposed to ease the certificate management problem in traditional public key infrastructure. Although they are two different notions, CB-PKC and CL-PKC are closely related and possess several common features. The encryption (or, signature verification) does not require public key authenticity verification, and the decryption (or, message signing) in both notions requires two pieces of information: one is generated by the third party, and the other one is generated by the public

key owner.

We provided the generic construction of certificate-based signatures from certificateless signatures, and the generic construction of certificate-based encryption from certificateless encryption. We refined the security definitions of those two notions and proved the security of the generic constructions in the random oracle model. Several concrete instances are also given to demonstrate the application of our generic constructions.

Fair exchange allows two parties to exchange their items in a fair way so that no one can gain any advantage in the process. A trivial way to realize fair exchange is to introduce an *online* trusted third party who acts as a mediator. A more practical method is optimistic fair exchange (OFE). An OFE protocol involves three participants: signer, verifier and a third party called arbitrator (who does not need to be fully trusted). Most OFE protocols are designed only in the single-user setting, namely there is only one signer. However, OFE protocols are most likely used in the multi-user setting where there are two or more signers in the system (but items are still exchanged between two parties). Recent research has shown that the single-user security of OFE cannot guarantee the multi-user security [DLY07, ZSM07].

We demonstrated several new results on optimistic fair exchange in the multi-user setting. We first introduced and defined a new property of OFE called Strong Resolution-Ambiguity. For an OFE protocol with strong resolution-ambiguity, we proved that its security in the single-user setting is preserved in the multi-user setting, which provides a further understanding on the security of OFE in the multi-user setting. Additionally, we provided a new construction of OFE with strong resolution-ambiguity, which possesses several desirable properties and is multi-user secure without the random oracle assumption.

Bibliography

- [ACL⁺07] Man Ho Au, Jing Chen, Joseph K. Liu, Yi Mu, Duncan S. Wong, and Guomin Yang. Malicious kgc attacks in certificateless cryptography. In *ACM conference, ASIACCS*, pages 277–290, 2007.
- [ALSY07] Man Ho Au, Joseph K. Liu, Willy Susilo, and Tsz Hon Yuen. Certificate based (linkable) ring signature. In Dawson and Wong [DW07], pages 79–92.
- [And92] Ross J. Anderson. An attack on server-assisted authentication protocols. *Electronic Letters*, 28(15):1473, 1992.
- [ARP03] Sattam S. Al-Riyami and Kenneth G. Paterson. Certificateless public key cryptography. In Chi-Sung Laih, editor, *ASIACRYPT*, volume 2894 of *Lecture Notes in Computer Science*, pages 452–473. Springer, 2003.
- [ARP05] Sattam S. Al-Riyami and Kenneth G. Paterson. Cbe from cl-pke: A generic construction and efficient schemes. In Serge Vaudenay, editor, *Public Key Cryptography*, volume 3386 of *Lecture Notes in Computer Science*, pages 398–415. Springer, 2005.
- [ASW97] N. Asokan, Matthias Schunter, and Michael Waidner. Optimistic protocols for fair exchange. In *ACM Conference on Computer and Communications Security*, pages 7–17, 1997.
- [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*, Springer, pages 591–606, 1998.

- [ASW00] N. Asokan, Victor Shoup, and Michael Waidner. Optimistic fair exchange of digital signatures. *IEEE Journal on Selected Areas in Communication*, 18(4):593–610, Apr 2000.
- [BB04] Dan Boneh and Xavier Boyen. Short signatures without random oracles. In Cachin and Camenisch [CC04], pages 56–73.
- [BF01] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the weil pairing. In Joe Kilian, editor, *CRYPTO*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229. Springer, 2001.
- [BF03] Dan Boneh and Matthew K. Franklin. *SIAM J. of Computing*, 32(3):586–615, 2003.
- [BG92] Mihir Bellare and Oded Goldreich. On defining proofs of knowledge. In Ernest F. Brickell, editor, *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 Biham [Bih03], pages 416–432.
- [BGMW92] Ernest F. Brickell, Daniel M. Gordon, Kevin S. McCurley, and David Bruce Wilson. Fast exponentiation with precomputation (extended abstract). In *EUROCRYPT*, *volume 658 of Lecture Notes in Computer Science*, pages 200–207, 1992.
- [Bih03] Eli Biham, editor. *Advances in Cryptology - EUROCRYPT 2003, International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4-8, 2003, Proceedings*, volume 2656 of *Lecture Notes in Computer Science*. Springer, 2003.
- [BKM06] Adam Bender, Jonathan Katz, and Ruggero Morselli. Ring signatures: Stronger definitions, and constructions without random oracles. In Shai Halevi and Tal Rabin, editors, *TCC*, volume 3876 of *Lecture Notes in Computer Science*, pages 60–79. Springer, 2006.
- [BLS01] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pairing. In Colin Boyd, editor, *ASIACRYPT*, volume 2248 of *Lecture Notes in Computer Science*, pages 514–532. Springer, 2001.

- [BM94] John Burns and Chris J. Mitchell. Parameter selection for server-aided RSA computation schemes. *IEEE Transaction on Computers*, 42(2):163–147, 1994.
- [BNN04] Mihir Bellare, Chanathip Namprempre, and Gregory Neven. Security proofs for identity-based identification and signature schemes. In Cachin and Camenisch [CC04], pages 268–286.
- [Bol03] Alexandra Boldyreva. Threshold signatures, multisignatures and blind signatures based on the gap-diffie-hellman-group signature scheme. In Yvo Desmedt, editor, *Public Key Cryptography*, volume 2567 of *Lecture Notes in Computer Science*, pages 31–46. Springer, 2003.
- [BQ95] Philippe Béguin and Jean-Jacques Quisquater. Fast server-aided RSA signatures secure against active attacks. In Coppersmith [Cop95], pages 57–69.
- [BSN06] Lynn Margaret Batten and Reihaneh Safavi-Naini, editors. *Information Security and Privacy, 11th Australasian Conference, ACISP 2006, Melbourne, Australia, July 3-5, 2006, Proceedings*, volume 4058 of *Lecture Notes in Computer Science*. Springer, 2006.
- [CC04] Christian Cachin and Jan Camenisch, editors. *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*, volume 3027 of *Lecture Notes in Computer Science*. Springer, 2004.
- [CD00] Jan Camenisch and Ivan Damgård. Verifiable encryption, group encryption, and their applications to separable group signatures and signature sharing schemes. In Tatsuaki Okamoto, editor, *ASIACRYPT*, volume 1976 of *Lecture Notes in Computer Science*, pages 331–345. Springer, 2000.
- [Cop95] Don Coppersmith, editor. *Advances in Cryptology - CRYPTO '95, 15th Annual International Cryptology Conference, Santa Barbara, California, USA, August 27-31, 1995, Proceedings*, volume 963 of *Lecture Notes in Computer Science*. Springer, 1995.

- [CS98] Ronald Cramer and Victor Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In Hugo Krawczyk, editor, *CRYPTO*, volume 1462 of *Lecture Notes in Computer Science*, pages 13–25. Springer, 1998.
- [Den08] Alexander W. Dent. A survey of certificateless encryption schemes and security models. *Int. J. Inf. Sec.*, 7(5):349–377, 2008.
- [DH76] Witfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Trans. Info. Th.*, 22(6):644–654, 1976.
- [DLP08] Alexander W. Dent, Benoît Libert, and Kenneth G. Paterson. Certificateless encryption schemes strongly secure in the standard model. In Ronald Cramer, editor, *Public Key Cryptography*, volume 4939 of *Lecture Notes in Computer Science*, pages 344–359. Springer, 2008.
- [DLY07] Yevgeniy Dodis, Pil Joong Lee, and Dae Hyun Yum. Optimistic fair exchange in a multi-user setting. In Okamoto and Wang [OW07], pages 118–133.
- [DR03] Yevgeniy Dodis and Leonid Reyzin. Breaking and repairing optimistic fair exchange from podc 2003. In *Digital Rights Management Workshop*, pages 47–54, 2003.
- [DW07] Ed Dawson and Duncan S. Wong, editors. *Information Security Practice and Experience, Third International Conference, ISPEC 2007, Hong Kong, China, May 7-9, 2007, Proceedings*, volume 4464 of *Lecture Notes in Computer Science*. Springer, 2007.
- [FMR99] Gerhard Frey, Michael Müller, and Hans-Georg Rück. The Tate pairing and the discrete logarithm applied to elliptic curve cryptosystems. *IEEE Trans. Info. Th.*, 45(5):1717–1719, 1999.
- [Gen03] Craig Gentry. Certificate-based encryption and the certificate revocation problem. In Biham [Bih03], pages 272–293.
- [GHK06] David Galindo, Javier Herranz, and Eike Kiltz. On the generic construction of identity-based signatures with additional properties. In Xuejia Lai and Kefei Chen, editors, *ASIACRYPT*, volume 4284 of *Lecture Notes in Computer Science*, pages 178–193. Springer, 2006.

- [Gir91] Marc Girault. Self-certified public keys. In *EUROCRYPT volume 547 of Lecture Notes in Computer Science, Springer*, pages 490–497, 1991.
- [GJM99] Juan A. Garay, Markus Jakobsson, and Philip D. MacKenzie. Abuse-free optimistic contract signing. In Michael J. Wiener, editor, *CRYPTO*, volume 1666 of *Lecture Notes in Computer Science*, pages 449–466. Springer, 1999.
- [GL05] Marc Girault and David Lefranc. Server-aided verification: Theory and practice. In Bimal K. Roy, editor, *ASIACRYPT*, volume 3788 of *Lecture Notes in Computer Science*, pages 605–623. Springer, 2005.
- [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.
- [GP03] Marc Girault and Jean Claude Paillès. On-line/off-line rsa-like, 2003.
- [GPS08] Steven D. Galbraith, Kenneth G. Paterson, and Nigel P. Smart. Pairings for cryptographers. *Discrete Applied Mathematics*, 156(16):3113–3121, 2008.
- [GQ02] Marc Girault and Jean-Jacques Quisquater. GQ + GPS = new ideas + new protocols, 2002.
- [GRK00] Rosario Gennaro, Tal Rabin, and Hugo Krawczyk. Rsa-based undeniable signatures. *J. Cryptology*, 13(4):397–416, 2000.
- [GS05] M. Choudary Gorantla and Ashutosh Saxena. An efficient certificateless signature scheme. In *CIS (2)*, pages 110–116, 2005.
- [HMS⁺07] Xinyi Huang, Yi Mu, Willy Susilo, Duncan S. Wong, and Wei Wu. Certificateless signature revisited. In Josef Pieprzyk, Hossein Ghodosi, and Ed Dawson, editors, *ACISP*, volume 4586 of *Lecture Notes in Computer Science*, pages 308–322. Springer, 2007.
- [HMS⁺10] Xinyi Huang, Yi Mu, Willy Susilo, Wei Wu, and Yang Xiang. Further observations on optimistic fair exchange protocols in the multi-user setting. In Phong Q. Nguyen and David Pointcheval, editors, *Public*

- Key Cryptography*, volume 6056 of *Lecture Notes in Computer Science*, pages 124–141. Springer, 2010.
- [HSMZ05] Xinyi Huang, Willy Susilo, Yi Mu, and Futai Zhang. On the security of certificateless signature schemes from asiacrypt 2003. In Yvo Desmedt, Huaxiong Wang, Yi Mu, and Yongqing Li, editors, *CANS*, volume 3810 of *Lecture Notes in Computer Science*, pages 13–25. Springer, 2005.
- [HWZD06] Bessie C. Hu, Duncan S. Wong, Zhenfeng Zhang, and Xiaotie Deng. Key replacement attack against a generic construction of certificateless signature. In Batten and Safavi-Naini [BSN06], pages 235–246.
- [HYWS08a] Qiong Huang, Guomin Yang, Duncan S. Wong, and Willy Susilo. Ambiguous optimistic fair exchange. In Josef Pieprzyk, editor, *ASIACRYPT*, volume 5350 of *Lecture Notes in Computer Science*, pages 74–89. Springer, 2008.
- [HYWS08b] Qiong Huang, Guomin Yang, Duncan S. Wong, and Willy Susilo. Efficient optimistic fair exchange secure in the multi-user setting and chosen-key model without random oracles. In Tal Malkin, editor, *CT-RSA*, volume 4964 of *Lecture Notes in Computer Science*, pages 106–120. Springer, 2008.
- [iKS93] Shin ichi Kawamura and Ashushi Shimbo. Fast server-aided secret computation protocols for modular exponentiation. *IEEE Journal on Selected Areas in Communications*, 11(5):778–784, 1993.
- [Jou04] Antoine Joux. A one round protocol for tripartite diffie-hellman. *J. of Cryptology*, 17(4):263–276, 2004.
- [KP05] Bo Gyeong Kang and Je Hong Park. Is it possible to have CBE from CL-PKE? Cryptology ePrint Archive, Report 2005/431, 2005. <http://eprint.iacr.org/>.
- [KPH04] Bo Gyeong Kang, Je Hong Park, and Sang Geun Hahn. A certificate-based signature scheme. In Tatsuaki Okamoto, editor, *CT-RSA*, volume 2964 of *Lecture Notes in Computer Science*, pages 99–111. Springer, 2004.

- [Lan73] Serge Lang. *Elliptic Functions*. Addison Wesley, Reading, Mass, 1973.
- [LBSZ08] Joseph K. Liu, Joonsang Baek, Willy Susilo, and Jianying Zhou. Certificate-based signature schemes without pairings or random oracles. In Tzong-Chen Wu, Chin-Laung Lei, Vincent Rijmen, and Der-Tsai Lee, editors, *ISC*, volume 5222 of *Lecture Notes in Computer Science*, pages 285–297. Springer, 2008.
- [LHM⁺07] Jiguo Li, Xinyi Huang, Yi Mu, Willy Susilo, and Qianhong Wu. Certificate-based signature: Security model and efficient construction. In Javier Lopez, Pierangela Samarati, and Josep L. Ferrer, editors, *EuroPKI*, volume 4582 of *Lecture Notes in Computer Science*, pages 110–125. Springer, 2007.
- [LL95] Chae Hoon Lim and Pil Joong Lee. Security and performance of server-aided RSA computation protocols. In Coppersmith [Cop95], pages 70–83.
- [LOS⁺06] Steve Lu, Rafail Ostrovsky, Amit Sahai, Hovav Shacham, and Brent Waters. Sequential aggregate signatures and multisignatures without random oracles. In Serge Vaudenay, editor, *EUROCRYPT*, volume 4004 of *Lecture Notes in Computer Science*, pages 465–485. Springer, 2006.
- [LZ08] Joseph K. Liu and Jianying Zhou. Efficient certificate-based encryption in the standard model. In Rafail Ostrovsky, Roberto De Prisco, and Ivan Visconti, editors, *SCN*, volume 5229 of *Lecture Notes in Computer Science*, pages 144–155. Springer, 2008.
- [MILY92] Tsutomu Matsumoto, Hideki Imai, Chi-Sung Lai, and Sung-Ming Yen. On verifiable implicit asking protocols for RSA computation. In Jennifer Seberry and Yuliang Zheng, editors, *AUSCRYPT*, volume 718 of *Lecture Notes in Computer Science*, pages 296–307. Springer, 1992.
- [MK01] Olivier Markowitch and Steve Kremer. An optimistic non-repudiation protocol with transparent trusted third party. In George I. Davida and Yair Frankel, editors, *ISC*, volume 2200 of *Lecture Notes in Computer Science*, pages 363–378. Springer, 2001.

- [MKI88] Tsutomu Matsumoto, Koki Kato, and Hideki Imai. Speeding up secret computations with insecure auxiliary devices. In Shafi Goldwasser, editor, *CRYPTO*, volume 403 of *Lecture Notes in Computer Science*, pages 497–506. Springer, 1988.
- [MOV93] Alfred Menezes, Tatsuaki Okamoto, and Scott Vanstone. Reducing elliptic curve logarithms to logarithms in a finite field. *IEEE Trans. Info. Th.*, 39(5):1639–1646, 1993.
- [MR06] Paz Morillo and Carla R’afols. Certificate-based encryption without random oracles. Cryptology ePrint Archive, Report 2006/012, 2006. <http://eprint.iacr.org/>.
- [MW96] Ueli M. Maurer and Stefan Wolf. Diffie-hellman oracles. In Neal Koblitz, editor, *CRYPTO*, volume 1109 of *Lecture Notes in Computer Science*, pages 268–282. Springer, 1996.
- [NS98] Phong Q. Nguyen and Jacques Stern. The béguin-quisquater server-aided RSA protocol from crypto ’95 is not secure. In Kazuo Ohta and Dingyi Pei, editors, *ASIACRYPT*, volume 1514 of *Lecture Notes in Computer Science*, pages 372–379. Springer, 1998.
- [OW07] Tatsuaki Okamoto and Xiaoyun Wang, editors. *Public Key Cryptography - PKC 2007, 10th International Conference on Practice and Theory in Public-Key Cryptography, Beijing, China, April 16-20, 2007, Proceedings*, volume 4450 of *Lecture Notes in Computer Science*. Springer, 2007.
- [Par06] Je Hong Park. An attack on the certificateless signature scheme from euc workshops 2006. Cryptology ePrint Archive, Report 2006/442, 2006. <http://eprint.iacr.org/>.
- [PCS03] Jung Min Park, Edwin K.P. Chong, and Howard Jay Siegel. Constructing fair-exchange protocols for E-commerce via distributed computation of RSA signatures. In *Proceedings of the twenty-second annual symposium on Principles of distributed computing*, pages 172–181, New York, 2003. ACM.

- [PW92] Birgit Pfitzmann and Michael Waidner. Attacks on protocols for server-aided RSA computation. In *EUROCRYPT, volume 658 of Lecture Notes in Computer Science, Springer*, pages 153–162, 1992.
- [QS89] Jean-Jacques Quisquater and Marijke De Soete. Speeding up smart card RSA computation with insecure coprocessors. In *Smart Cards 2000*, pages 191–197, 1989.
- [RS09] Markus Rückert and Dominique Schröder. Security of verifiably encrypted signatures and a construction without random oracles. In Hovav Shacham and Brent Waters, editors, *Pairing*, volume 5671 of *Lecture Notes in Computer Science*, pages 17–34. Springer, 2009.
- [Sha84] Adi Shamir. Identity-based cryptosystems and signature schemes. In G. R. Blakley and David Chaum, editors, *CRYPTO*, volume 196 of *Lecture Notes in Computer Science*, pages 47–53. Springer, 1984.
- [Wat05] Brent Waters. Efficient identity-based encryption without random oracles. In Ronald Cramer, editor, *EUROCRYPT*, volume 3494 of *Lecture Notes in Computer Science*, pages 114–127. Springer, 2005.
- [WMSH08] Wei Wu, Yi Mu, Willy Susilo, and Xinyi Huang. Server-aided verification signatures: Definitions and new constructions. In Joonsang Baek, Feng Bao, Kefei Chen, and Xuejia Lai, editors, *ProvSec*, volume 5324 of *Lecture Notes in Computer Science*, pages 141–155. Springer, 2008.
- [WMSH09] Wei Wu, Yi Mu, Willy Susilo, and Xinyi Huang. Certificate-based signatures revisited. *Journal of Universal Computer Science*, 15(8):1659–1684, 2009.
- [WMSH11] Wei Wu, Yi Mu, Willy Susilo, and Xinyi Huang. Provably secure server-aided verification signatures,. *Journal of Computers and Mathematics with Applications*, accepted, 27 Jan. 2011.
- [WWYH10] Zhiwei Wang, Licheng Wang, Yixian Yang, and Zhengming Hu. Comment on Wu et al.’s server-aided verification signature schemes. *International Journal of Network Security*, 10(2):158–160, 2010.

- [YL92] Sung-Ming Yen and Chi-Sung Laih. More about the active attack on the server-aided secret computation protocol. *Electronic Letters*, 28(24):2250, 1992.
- [YL93] Sung-Ming Yen and Chi-Sung Laih. Server-aided honest computation for cryptographic applications. *Electronic Letters*, 26(12):61–64, 1993.
- [YL04] Dae Hyun Yum and Pil Joong Lee. Generic construction of certificate-less signature. In Huaxiong Wang, Josef Pieprzyk, and Vijay Varadharajan, editors, *ACISP*, volume 3108 of *Lecture Notes in Computer Science*, pages 200–211. Springer, 2004.
- [ZB06a] Huafei Zhu and Feng Bao. More on stand-alone and setup-free verifiably committed signatures. In Batten and Safavi-Naini [BSN06], pages 148–158.
- [ZB06b] Huafei Zhu and Feng Bao. Stand-alone and setup-free verifiably committed signatures. In David Pointcheval, editor, *CT-RSA*, volume 3860 of *Lecture Notes in Computer Science*, pages 159–173. Springer, 2006.
- [ZF06] Zhenfeng Zhang and Dengguo Feng. Key replacement attack on a certificateless signature scheme. Cryptology ePrint Archive, Report 2006/453, 2006. <http://eprint.iacr.org/>.
- [ZG96] Jianying Zhou and Dieter Gollmann. A fair non-repudiation protocol. In *IEEE Symposium on Security and Privacy*, pages 55–61. IEEE Computer Society, 1996.
- [ZM07] Jianhong Zhang and Jian Mao. A novel verifiably encrypted signature scheme without random oracle. In Dawson and Wong [DW07], pages 65–78.
- [ZSM07] Huafei Zhu, Willy Susilo, and Yi Mu. Multi-party stand-alone and setup-free verifiably committed signatures. In Okamoto and Wang [OW07], pages 134–149.
- [ZSNS04] Fangguo Zhang, Reihaneh Safavi-Naini, and Willy Susilo. An efficient signature scheme from bilinear pairings and its applications. In Feng

Bao, Robert H. Deng, and Jianying Zhou, editors, *Public Key Cryptography*, volume 2947 of *Lecture Notes in Computer Science*, pages 277–290. Springer, 2004.

- [ZWXF06] Zhenfeng Zhang, Duncan S. Wong, Jing Xu, and Dengguo Feng. Certificateless public-key signature: Security model and efficient construction. In Jianying Zhou, Moti Yung, and Feng Bao, editors, *ACNS*, volume 3989 of *Lecture Notes in Computer Science*, pages 293–308, 2006.