2014

Contributions on fair exchange of digital signatures

Yang Wang
University of Wollongong

Contributions on Fair Exchange of Digital Signatures

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

Doctor of Philosophy

from

UNIVERSITY OF WOLLONGONG

by

Yang Wang

School of Computer Science and Software Engineering
February 2014

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.

Yang Wang
February 13, 2014

Abstract

Fair exchange protocols aim to allow two parties to exchange digital items in a fair manner. Optimistic fair exchange (OFE) is a kind of protocols that solves the fair exchange problem with the help of a trusted third party (TTP), usually referred to as an 'arbitrator'. The participation of the arbitrator is only required when there is a dispute between the exchanging parties. In the literature, the highest level of security of optimistic fair exchange is the multi-user security in the chosen-key model, proposed by Huang, Yang, Wong and Susilo in CT-RSA 2008. They showed that an efficient optimistic fair exchange scheme, secure in this sense, can be constructed generically from a conventional signature and a ring signature. In particular, the underlying ring signature is required to be unforgeable under an adaptive attack, against a static adversary in the 2-user setting.

Concurrent signatures, introduced by Chen, Kudla and Paterson in Eurocrypt 2004, allow two parties to produce two ambiguous signatures until an extra piece of information, the keystone, is released by one of the parties. Upon the release of the keystone, the ambiguity will be revoked and the signatures bind to their true signers concurrently. Concurrent signatures, however, are known to fall just short of fully solving the long standing fair exchange of signatures problem. The price for not requiring any TTP is that the party who holds the keystone always has an advantage over the other party in controlling when and whether the protocol completes or not.

In this thesis, the fair exchange of digital signatures problem is studied. In the first part of this thesis, the OFE security is strengthened and a more practical model which is called *enhanced chosen-key model* is proposed. Unlike the existing multi-user setting and chosen-key model, an adversary in the enhanced chosen-key model is further provided with the signing oracle that outputs full signatures generated by the signer, and may even be allowed to have access to the arbitrator's secret key. The necessity for the new model is demonstrated. It is shown that two existing

methodologies for constructing optimistic fair exchange schemes remain valid in the enhanced chosen-key model.

Next, the efficiency of OFE constructions is improved. A model for 2-user ring signatures called 'unforgeability against restricted adaptive attacks' is proposed and it is shown that the new ring model is strictly weaker than the model of unforgeability under an adaptive attack, against a static adversary in the 2-user setting. Based on the finding that 2-user ring signatures secure in this weaker model will suffice to guarantee the security of the resulting OFE schemes in the enhanced chosen-key model, the most efficient OFE scheme whose security does not rely on random oracles is proposed.

New situations in OFE are also investigated, and the applications of OFE are extended. The scenarios where the digital items to be exchanged are threshold signatures are considered, and the notion of threshold-oriented OFE (TOFE) together with a security model and an efficient concrete scheme is proposed. It is identified that in some scenarios, it is required that prior to the completion of the protocol, no information about the actual exchange should be revealed to an outsider. To achieve this purpose, the notion of perfect ambiguous optimistic fair exchange (PAOFE) is introduced, the security model for PAOFE is proposed and a generic construction based on existing primitives is offered. OFE in the attribute-based setting is considered and the notion of attribute-based optimistic fair exchange (ABOFE) is proposed. The security model and a construction for ABOFE are presented as well.

Finally, concurrent signatures is revisited. It is showed that theoretically the party who controls the keystone in a concurrent signature scheme has more advantages than previously thought. In particular, the notion is examined and the advantages are classified into three levels.It is demonstrated that concurrent signatures may be abused in practical applications if one fails to take these advantages into account.

Acknowledgement

First of all, I would like to express my sincere gratitude to my supervisor, Professor Willy Susilo, for giving me the chance of studying at the University of Wollongong and for his patient guidance and ongoing support in my research. Professor Susilo's enormous encouragement has always given me the will and strength to progress.

I would like to thank my co-supervisor Professor Mu Yi, for his guidance and immense knowledge. I still remember the invaluable suggestions he gave me the first time we met. I am grateful to my co-supervisor, Dr. Man Ho Au. His immense expertise in applied cryptography improved my research skills. His meticulous scholarship affected my attitude about research in a positive way. Dr. Man Ho Au has always been kind to me. Every time I come to him with a question, he is willing to have a discussion with me, even after work or on the weekend. My progress would have been impossible without his critical and detailed comments.

I appreciate the inspiring cooperation with Dr. Duncan Wong, Dr. Guilin Wang and Associate Professor Mark Manulis. I would like to thank them for their generous helps and insightful comments. The discussions with them are always productive.

Many thanks also goes to the members of my research groups, including Dr. Guomin Yang, Dr. Yong Yu, Dr. Xinyi Huang, Dr. Wei Wu, Dr. Fuchun Guo, Dr. Jinguang Han, Miao Zhou, Nan Li, Zhenfei Zhang, Shams Qzai and Yafu Ji for their kind help. The working environment with them is so positive. I would like to thank in particular Fuchun Guo, Jinguang Han and Nan Li for their generous help and companionship.

I am lucky to have Lingqiao Liu as my roommate. The food he prepares sometimes is really delicious. I thank Ming Li and Xing Zhao for being by my side during the PhD life.

Deep in my heart I am grateful to the postgraduate research scholarship jointly offered by the China Scholarship Council (CSC) and the University of Wollongong (UOW), which was crucial for my study abroad. Thanks for my supervisor for my

Master's Degree, Professor Chao Li's encouragement and support about doing my PhD overseas.

Finally I would like to thank my father Changjun Wang, my mother Qiaoling Yang and my brother Long Wang for their love.

Publications

The thesis is based on the following publications or manuscripts, which were finished before the submission of this thesis.

1. Yang Wang, Man Ho Au, and Willy Susilo. Perfect Ambiguous Optimistic Fair Exchange (Extended version available from http://eprint.iacr.org/2012/46). In Tat Wing Chim, Tsz Hon Yuen, editors, Information and Communications Security, 14th International Conference, ICICS 2012, Proceedings, Lecture Notes in Computer Science 7618, pages 142-153, Springer, 2012.

2. Yang Wang, Man Ho Au, Joseph K. Liu, Tsz Hon Yuen and Willy Susilo, Threshold-Oriented Optimistic Fair Exchange. In Javier Lopez, Xinyi Huang, Ravi Sandhu, editors, Network and System Security, 7th International Conference, NSS 2013, Proceedings, Lecture Notes in Computer Science 7873, pages 424-438, Springer, 2013.

3. Willy Susilo, Man Ho Au, Yang Wang and Duncan S. Wong, Fairness in Concurrent Signatures Revisited. In Colin Boyd, Leonie Simpson, editors, Information Security and Privacy, 18th Australasian Conference, ACISP 2013, Proceedings, Lecture Notes in Computer Science 7959, pages 318-329, Springer, 2013.

4. Yang Wang, Man Ho Au, Willy Susilo and Guilin Wang. Collusion-Resistance in Optimistic Fair Exchange. In submission to IEEE Transactions on Information Forensics and Security.

5. Yang Wang, Man Ho Au, and Willy Susilo. Optimistic Fair Exchange based on Ring Signatures. In submission to IEEE Transactions on Information Forensics and Security.

6. Yang Wang, Man Ho Au, and Willy Susilo. Attribute-based Optimistic Fair Exchange: How to Restrict Brokers with Policies. Accepted by Theoretical Computer Science on 28 January, 2014.

7. Yang Wang, Man Ho Au, and Willy Susilo. Optimistic Fair Exchange in the Enhanced Chosen-key Model. In submission to Theoretical Computer Science.

Below are the papers that have been published but are beyond the scope of this thesis.

1. Yang Wang, Mark Manulis, Man Ho Au, and Willy Susilo, Relations among Privacy notions for Signcryption and Key Invisible Sign-then-Encrypt (Extended version available from http://eprint.iacr.org/2013/230). In Colin Boyd, Leonie Simpson, editors, Information Security and Privacy, 18th Australasian Conference, ACISP 2013, Proceedings, Lecture Notes in Computer Science 7959, pages 187-202, Springer, 2013.

Contents

Abstract **v**

Acknowledgement **vii**

Publications **ix**

1 Introduction **1**

1.1 Summary of this Thesis 3

2 Background **6**

2.1 Preliminaries 6

2.1.1 Notations 6

2.1.2 Bilinear Pairing 7

2.1.3 Complexity Notion and Assumptions 7

2.2 Cryptographic Tools 10

2.2.1 Secret Sharing 10

2.2.2 Hash Function 11

2.2.3 General Forking Lemma 12

2.2.4 Zero-Knowledge Proof-of-Knowledge 13

2.2.5 Digital Signature 14

2.2.6 Ring Signatures 15

2.2.7 Public-key Encryption 17

2.2.8 Ciphertext-Policy Attribute-based Encryption 19

2.3 Optimistic Fair Exchange 21

2.3.1 Definition 22

2.3.2 Security in Multi-User setting and Chosen-key Model 24

2.4 Ambiguous Optimistic Fair Exchange 26

2.5 Concurrent Signatures . . . 30
2.5.1 Definition of Concurrent Signatures . . . 31
2.5.2 Security Model . . . 32

I OFE with Stronger Security 36

3 Optimistic Fair Exchange in the Enhanced Chosen-key Model 37
3.1 Introduction . . . 37
3.1.1 Motivation . . . 39
3.1.2 The Contributions . . . 41
3.2 The Enhanced Chosen-key Model . . . 42
3.3 Separation of the Proposed Model and the Existing Model . . . 43
3.3.1 Security Analysis. . . . 47
3.3.2 An attack in the Enhanced Model . . . 53
3.4 Identical Distribution not Sufficient . . . 54
3.4.1 Security Analysis. . . . 56
3.4.2 An attack in the Enhanced Model . . . 60
3.5 Chapter Summary . . . 61

4 Collusion-Resistance in Optimistic Fair Exchange 62
4.1 Introduction . . . 62
4.1.1 The Contributions . . . 63
4.1.2 Organization . . . 64
4.2 The Enhanced Chosen-key Model . . . 64
4.2.1 Implications and Limitations of The Enhanced Model . . . 65
4.3 Separation of the Proposed Model and the Existing Model . . . 65
4.3.1 High Level Description . . . 66
4.3.2 Construction of A*-OFE . . . 67
4.3.3 Security Analysis. . . . 70
4.3.4 A Concrete Attack against A*-OFE in the Enhanced Model . . . 79
4.4 Previous Paradigms Revisited . . . 82
4.4.1 Verifiably Encrypted Signature Paradigm . . . 82
4.4.2 Conventional Signature and Ring signature Paradigm . . . 86
4.5 Chapter Summary . . . 89

II OFE with Better Efficency **91**

5 Optimistic Fair Exchange based on Ring Signatures **92**

5.1 Introduction . . . 92
5.1.1 The Contributions . . . 93
5.1.2 Chapter Organization . . . 94
5.2 A New Model for 2-User Ring Signatures . . . 94
5.2.1 Separating Adaptive Model from Restricted Adaptive Model . 95
5.3 Security of the Generic Construction . . . 100
5.4 A new efficient OFE scheme . . . 102
5.4.1 Comparison . . . 104
5.5 Chapter Summary . . . 105

III OFE with New Features **107**

6 Threshold-Oriented Optimistic Fair Exchange **108**

6.1 Introduction . . . 108
6.1.1 Contribution. . . . 110
6.2 Definition of TOFE . . . 110
6.2.1 Syntax . . . 110
6.2.2 A Typical Usage of the TOFE Algorithms . . . 112
6.2.3 Security Model . . . 113
6.3 Construction . . . 116
6.4 Chapter Summary . . . 122

7 Perfect Ambiguous Optimistic Fair Exchange **123**

7.1 Introduction . . . 123
7.1.1 The Contributions . . . 125
7.1.2 Chapter Organization . . . 126
7.2 A new Encryption Notion . . . 126
7.3 Perfect Ambiguous Optimistic Fair Exchange . . . 127
7.3.1 PAOFE Models . . . 128
7.4 Generic Construction . . . 132
7.4.1 Security Analysis . . . 133
7.5 Comparison with Designated Verifier Signature . . . 138

7.6 Chapter Summary . . . 138

8 Attribute-based Optimistic Fair Exchange: How to Restrict Brokers with Policies 140

8.1 Introduction . . . 140
8.1.1 The Contributions . . . 141
8.1.2 Chapter Organization . . . 142
8.2 Attribute-based Optimistic Fair Exchange . . . 142
8.2.1 A Typical Usage of ABOFE . . . 144
8.2.2 Security Model . . . 145
8.3 Construction . . . 148
8.3.1 Security Analysis . . . 150
8.4 Instantiation . . . 153
8.4.1 Efficency Analysis . . . 159
8.5 Chapter Summary . . . 159

IV Fairness of Concurrent Signatures 162

9 Fairness in Concurrent Signatures Revisited 163

9.1 Introduction . . . 163
9.1.1 Fairness in Practice . . . 164
9.1.2 The Contributions . . . 165
9.1.3 Related Work . . . 165
9.1.4 Chapter Organization . . . 166
9.2 Abusing Fairness in Concurrent Signatures . . . 167
9.2.1 Advantage Level 0 . . . 167
9.2.2 Advantage Level 1 . . . 167
9.2.3 Advantage Level 2 . . . 168
9.3 Abusing Fairness in Asymmetric Concurrent Signatures . . . 169
9.3.1 Review of Nguyen's asymmetric concurrent signatures . . . 170
9.3.2 A Concrete Level-1 Abuse on Nguyen's Scheme . . . 171
9.4 Comparison of Concurrent Signatures and OFE . . . 171
9.5 Chapter Summary . . . 172

V Conclusion and Future Work **173**

10 Conclusion and Future Work **174**

10.1 Summary of the Thesis 174

10.1.1 OFE with Stronger Security 174

10.1.2 OFE with Better Efficiency 174

10.1.3 OFE with New Features 175

10.1.4 Concurrent Signatures Revisited 175

10.2 Future Work 175

Bibliography **176**

List of Tables

5.1 Performance Comparison 105

6.1 Digital items that are exchanged in OFE 110

8.1 Costs of algorithms in the instantiation of ABOFE. 160

List of Figures

2.1 **A typical execution of OFE without disputes.** 22
2.2 **An execution of OFE when disputes occur.** 23
2.3 **An overview of Concurrent Signatures.** 31

Chapter 1

Introduction

With the wide use of open networks such as Internet, electronic commerce (e-commerce), the exchange of digital items, becomes more and more important. A fundamental requirement for electronic commerce is the establishment of mechanisms allowing exchanges of digital items between two parties in a fair way.

In traditional commerce, fairness is easy to achieve as the physical items can be exchanged simultaneously over a counter. Such a simultaneous exchange is not feasible over networks as no physical counter exists. One party involved in an electronic commercial exchange may refuse to offer its item after having received the other party's.

One kind of solution was provided by gradual exchange protocols [Gol84, EGL85, BCDvdG87, Cle89, Dam94, BN00, GP03] where two parties exchange their digital items in turns "little-by-little" over many rounds. The basic idea of these kinds of protocols is that if each party alternately releases a small portion of the digital item, one bit or several bits for example, then both parties have comparable knowledge whenever the protocol is terminated prematurely, and thus neither party has a considerable advantage over the other. However, these solutions may not provide true fairness, because one party owing a considerably stronger computational power may abort at some stage and compute the remaining parts of the other party's item while the other party fails to do so. Even if both parties have equal computational power, such protocols are normally too interactive and cumbersome.

Another solution is to resort to a trusted third party (TTP) to ensure fairness. A straightforward way to achieve this is to have an on-line trusted third party involved as mediator. In such a case, each party sends its digital item and expectation to the trusted third party, who first checks the validity of the items to be exchanged and the expectations of the parties, and then forwards each item to the other party. Some variations are discussed in [CTS95, DGLW96, FR97]. However, in practical

applications, the on-line trusted third party is a single point of failure. Once it is down, no transaction can be conducted. Another drawback is that the trusted third party becomes a bottleneck due to the involvement in every single exchange. Also, this type of trusted third party is required to be fully trusted by both parties, which, in practice, is a very strong assumption.

Optimistic fair exchange (OFE), introduced by Asokan, Schunter and Waidner [ASW97], is another kind of protocol that uses a third party as arbitrator, but only in some limited circumstances. The third party only gets involved when one party cheats or some network failure occurs. In the vast majority of transactions where the parties follow the protocol honestly, the third party will not be involved at all. An optimistic fair exchange protocol comprises signers, verifiers, and a trusted third party, called the 'arbitrator'. In such a protocol, Alice the signer first delivers a partial signature to Bob the verifier. A valid partial signature serves as a partial commitment from Alice about the exchange to take place and assures Bob that he will receive Alice's full signature at the end of the protocol. The assurance follows from the fact that the arbitrator is capable of converting the partial signature into a full one. After verifying the validity of the partial signature, Bob fulfills his obligation by delivering his digital item to Alice, after which Alice should send her full signature to complete the exchange. In the case where Alice refuses to send her full signature or there is a network failure, Bob asks the arbitrator to make a resolution, in which the arbitrator converts the partial signature into a full one and sends it back to Bob.

Due to the optimistic usage of the trusted third party, OFE has attracted much research [GJM99, Mic03, Wan05, ES05, OMO08, HV11, HWS11a] since its introduction. The highest level of security of optimistic fair exchange in the literature is the multi-user security in the chosen-key model, proposed by Huang et al. in CT-RSA 2008. The security comprises three aspects: security against signers, security against verifiers and security against the arbitrator. Scenarios cover instances when any of the three parties is dishonest. There exist two generic constructions of optimistic fair exchange schemes which are secure in this model, one based on verifiably encrypted signatures, and the other based on conventional signatures and ring signatures [HYWS08b].

Most existing optimistic fair exchange schemes share the property that Bob could use Alice's partial signature as a means to convince others that Alice has agreed to commit to something, but this may not be desirable in some applications such as

fair negotiation. To address this problem, Boyd and Foo [BF98] and Chen [Che98] independently proposed OFE schemes in which a partial signature is not publicly verifiable and an interactive protocol between the signer and the verifier must be executed to convince the verifier of the validity of a partial signature. In [GJM99], Garay, Jakobsson and MacKenzie proposed a new notion called 'abuse-free', which requires that no party can ever prove to a third party that he is capable of choosing whether to validate or invalidate a contract. They introduced a new technique named 'private contract signature', from which an abuse-free contract signing protocol was constructed. In 2008, Huang, Yang, Wong and Susilo [HYWS08a] studied a variant of optimistic fair exchange, in which a new security notion called 'signer ambiguity' was introduced, which requires that Bob be able to forge a partial signature that is indistinguishable from a real one generated by Alice. With this property, Bob will not be able to convince any outsiders that a partial signature was indeed generated by Alice. They named OFE with this new security property 'ambiguous optimistic fair exchange' (AOFE). In addition, they proposed a formal security model for AOFE.

In 2004, Chen et al. [CKP04] introduced a new cryptographic primitive called 'concurrent signatures', which is reviewed as a new and different approach to solve the fair exchange problem. Such schemes allow two parties to produce two ambiguous signatures in the sense that both signatures do not bind to their true signers. This is due to the fact that either party alone is able to forge two ambiguous signatures. Upon the release of an extra piece of information called the keystone, however, both signatures will bind to their signers concurrently. Concurrent signatures enable fair exchange protocols to be built without the involvement of a trusted third party, but it is at the sacrifice that the initial signer (one of the two parties who control the keystone) always has the extra power of deciding when or even whether the keystone is released. The initial signer might privately show the other party's signature together with the unreleased keystone to outsiders.

1.1 Summary of this Thesis

In this thesis, the issues of fair exchange of digital signatures are investigated. The attention is mainly focused on optimistic fair exchange, as it is the most-widely used approach for solving the fair exchange problem. In this chapter, the significance and overview of fair exchange of digital items have been introduced. In particular, the

three existing approaches for solving the fair exchange problem was reviewed.

In Chapter 2, the notations is introduced and background knowledge required to understand this thesis is reviewed. An introduction to bilinear pairing and some complexity assumptions are firstly given. Then definitions of several cryptographic tools are recalled such as secret sharing, hash functions, zero-knowledge proof-of-knowledge, digital signature and encryption schemes. Finally the important primitives for fair exchange, including optimistic fair exchange (OFE), ambiguous optimistic fair exchange (AOFE) and concurrent signature are reviewed.

Next, the enhanced chosen-key model is proposed to strengthen the existing OFE security in Chapter 3 and 4. In Chapter 3, it is identified how existing OFE models fail to capture the scenario that an adversary may have access to the full signatures generated by the signer. It is claimed that in the security against verifiers (the arbitrator, resp.), the dishonest verifier (arbitrator, resp.) must be explicitly provided the signing oracle which outputs the signer's full signature. The necessity of doing this is demonstrated through two counterexamples. In Chapter 4, the security against signers is further strengthened by allowing the dishonest signer to have the arbitrator's secret key, which captures a possible collusion between the dishonest signer and the potentially dishonest arbitrator. This new model is distinguished from existing models. At the end of Chapter 4, an affirmative answer is provided to the basic and natural question of whether or not there exists schemes in the enhanced chosen-key model.

In Chapter 5, the efficiency of existing OFE constructions is improved. Specifically, the popular paradigm of constructing OFE schemes from conventional signatures and ring signatures are revisited. A model for 2-user ring signatures called unforgeability against restricted adaptive attacks is proposed and then it is shown that 2-user ring signatures secure in this model will suffice to guarantee the security of the resulting OFE schemes, rather than the previous understanding that the ring signature scheme should satisfy a stronger model unforgeable under an adaptive attack, against a static adversary in the 2-user setting. By adopting a 2-user ring signatures secure only in the weaker model, the most efficient OFE instantiation in the standard model is gained.

Furthermore, new situations about OFE are investigated. In Chapter 6, optimistic fair exchange of threshold signatures is considered. Specifically, the signatures are created by a subset of legitimate signers of a party instead of a single signer. In Chapter 7, the notion of perfect ambiguous optimistic fair exchange (PAOFE) is

proposed, which in essence is AOFE but with a new property, 'perfect ambiguity', which intuitively guarantees that no outsider, not even the arbitrator, is able to infer any useful information (the identities of the two involving parties, for example) from the partial signature. In Chapter 8, OFE in the attribute-based setting is considered, which for the first time takes into account the cases that in an exchange the user's attributes such as nationality and age may be involved.

Finally, in Chapter 9, the fairness in concurrent signatures is revisited. It is shown that theoretically the initial signer in a concurrent signature scheme can have two more advantages except the commonly known one. Thus in applications where concurrent signatures are potentially to be adopted, the users have to pay special attention to the advantages the initial signer owns and make sure these advantages are acceptable.

The thesis is concluded in Chapter 10.

Chapter 2

Background

In this chapter, the notations, definitions and complexity assumptions that will be used throughout this thesis are intorduced. Cryptographical tools that will be adopted in this thesis and fair exchange primitives including optimistic fair exchange (OFE), ambiguous optimistic fair exchange (AOFE) and concurrent signatures are briefly reviewed. For more information about cryptography theory, refer to the books [MVO96, Mao03].

2.1 Preliminaries

2.1.1 Notations

Throughout the thesis, the following notations are used. If n is a positive integer, $[n]$ is used to denote the set $\{1, \ldots, n\}$. If p is a prime, $\mathbb{Z}_p$ and $\mathbb{Z}_p^*$ is used to denote the sets $\{0, \ldots, p-1\}$ and $\{a | a \in \mathbb{Z}_p \wedge \gcd(a, p) = 1\}$, respectively. $k \in \mathbb{N}$ denotes a security parameter. For a finite set S, $|S|$ denotes its cardinality and $s \leftarrow S$ denotes that an element s is chosen uniformly at random from S. For a bit string m, $|m|$ denotes the bit length of m. For any bit strings m_1 and m_2, $m_1||m_2$ denotes the concatenation of m_1 and m_2. By $x := y$, it means variable x is assigned with the value of y. $y \leftarrow A^O(x)$ means that the algorithm A, on input x and having access to oracle O, outputs y. $[A_1(\mathsf{in}_1) \rightarrow \mathsf{out}_1] \overset{\mathbf{P}}{\Longleftrightarrow} [A_2(\mathsf{in}_2) \rightarrow \mathsf{out}_2]$ is used to denote that two PPT algorithms A_1 and A_2 outputs out_1 and out_2 respectively upon the completion of the protocol $\mathbf{P}$ in which A_1 takes as input in_1 and A_2 takes as input in_2.

2.1.2 Bilinear Pairing

Let G, G_T be two cyclic groups of the same order p for some large prime p. It is said that e is a bilinear map [BF01] if $e : G \times G \to G_T$ satisfies the following properties.

- (Bilinearity) For all elements of $g, h \in G$, $a, b \in \mathbb{Z}_p$, it holds that

$$e(g^a, h^b) = e(g, h)^{ab}.$$

- (Non-degeneracy) There exists $g, h \in G$ such that $e(g, h)$ is not the identity element of G_T.

- (Efficiency) The group operation in G and the map e can be computed efficiently.

2.1.3 Complexity Notion and Assumptions

A basic notion in complexity theory is negligible function [Bel02]. Specifically, it is required that the probability of an adversary successfully attacking a cryptography system be negligible in terms of a security parameter k.

Definition 2.1 (Negligible function). *A function $\mu : \mathbb{N} \to \mathbb{R}$ is negligible if for any positive integer c there exists an integer N_c such that for all $k > N_c$,*

$$\mu(k) < \frac{1}{k^c}.$$

Another fundamental notion in complexity theory is the computational indistinguishability [Gol90] of two distributions.

Definition 2.2 (Computational Indistinguishability). *Two distribution families $\{X_k\}_{k\in\mathbb{N}}$ and $\{Y_k\}_{k\in\mathbb{N}}$ are computationally indistinguishable if for every probabilistic polynomial time algorithm $\mathcal{A}$, the following quantity*

$$\mu(k) = \Big| \Pr_{x\in X_k}[\mathcal{A}(x) = 1] - \Pr_{x\in Y_k}[\mathcal{A}[x] = 1] \Big|$$

is a negligible function in k.

The security of many cryptosystems relies on well-established assumptions. An assumption normally assumes some problem is intractable and no algorithm can solve it in polynomial time. More specifically, there is no proof that no such efficient

algorithm exists, but it is widely believed that this is the case. The complexity assumptions that has been proposed in the literature and will be used in this thesis will be reviewed.

The discrete logarithm (DL) assumption [Odl85] is one of the fundamental assumptions in public-key cryptography.

Definition 2.3 (Discrete Logarithm Assumption). *Let G be a cyclic group with generator g of prime order $p \geq 2^k$, where k is a security parameter. Given (g, u) where $u = g^a$ is a random element of G, the discrete logarithm assumption holds for G if for every probabilistic polynomial time adversary $\mathcal{A}$, the advantage of $\mathcal{A}$ in computing the discrete logarithm of u to the base g*

$$\mathsf{Adv}_{\mathcal{A}}(k) = \Pr[\mathcal{A}(g, u) = a]$$

is negligible in k.

Diffie-Hellman proposed the computational Diffie-Hellman (CDH) Assumption [DH76] in their seminal paper "New Directions in Cryptography".

Definition 2.4 (Computational Diffie-Hellman Assumption). *Let G be a cyclic group with generator g of prime order $p \geq 2^k$, where k is a security parameter. Given (g, g^a, g^b) where a, b are uniformly at random chosen from $\mathbb{Z}_p$, the computational Diffie-Hellman assumption holds for G if for every probabilistic polynomial time adversary $\mathcal{A}$, the advantage of $\mathcal{A}$ in computing g^{ab}*

$$\mathsf{Adv}_{\mathcal{A}}(k) = \Pr[\mathcal{A}(g, g^a, g^b) = g^{ab}]$$

is negligible in k.

It is clear that if there is an algorithm which can breaks the DL assumption, then this algorithm can also be used to break the CDH assumption.

The decisional Diffie-Hellman (DDH) assumption is the decisional version of the CDH assumption. It was firstly introduced in [Bra93] and further surveyed in [Bon98].

Definition 2.5 (Decisional Diffie-Hellman Assumption). *Let G be a cyclic group with generator g of prime order $p \geq 2^k$, where k is a security parameter. Let a, b, c be uniformly and independently chosen from $\mathbb{Z}_p$. The decisional Diffie-Hellman*

assumption holds for G if for every probabilistic polynomial time adversary $\mathcal{A}$, the advantage of $\mathcal{A}$ in distinguishing (g, g^a, g^b, g^{ab}) from (g, g^a, g^b, g^c)

$$\mathsf{Adv}_{\mathcal{A}}(k) = \left| \Pr[\mathcal{A}(g, g^a, g^b, g^{ab}) = 1] - \Pr[\mathcal{A}(g, g^a, g^b, g^c) = 1] \right|$$

is negligible in k.

Note that the DDH assumption will not hold in the bilinear pairing setting, however it is believed that the CDH assumption still holds in the pairing setting.

Let G be of prime order $p \geq 2^k$, where k is a security parameter. Let g be a generator of G, and u be a random element of G. Let further x be a random element in $\mathbb{Z}_p$ and $\mathbf{Q} =< g, g^x, u, \{(g^{1/(x+s_i)}, g^{s_i}, u^{s_i})\}_{i=1}^{q} >$. Below three dynamic computational assumptions are reviewed.

Definition 2.6 (q-SDH Assumption [BB08]). *The q-Strong Diffie-Hellman (q-SDH) assumption holds for G if for every probabilistic polynomial time adversary $\mathcal{A}$ with input $(g, g^x, g^{x^2}, \cdots, g^{x^q})$, the advantage of $\mathcal{A}$ in computing a tuple $(s, g^{\frac{1}{x+s}})$*

$$\mathsf{Adv}_{\mathcal{A}}(k) = \Pr[\mathcal{A}(g, g^x, g^{x^2}, \cdots, g^{x^q}) = (s, g^{\frac{1}{x+s}})]$$

is negligible in k.

Definition 2.7 (q-HSDH Assumption [BW07]). *The q-Hidden Strong Diffie-Hellman (q-HSDH) assumption holds for G if for every probabilistic polynomial time adversary $\mathcal{A}$ with input $\boldsymbol{Q}$, the advantage of $\mathcal{A}$ in computing another triple $(g^{1/(x+s)}, g^s, u^s)$*

$$\mathsf{Adv}_{\mathcal{A}}(k) = \Pr[\mathcal{A}(\boldsymbol{Q}) = (g^{1/(x+s)}, g^s, u^s)]$$

is negligible in k, where $s \in \mathbb{Z}_p$ and $s \notin \{s_1, \cdots, s_q\}$.

Definition 2.8 (q-DHSDH Assumption [HW11]). *The q-Decisional Hidden Strong Diffie-Hellman (q-DHSDH) assumption holds for G if for every probabilistic polynomial time adversary $\mathcal{A}$, the advantage of $\mathcal{A}$ in distinguishing $(\boldsymbol{Q}, u^s, g^{1/(x+s)})$ from $(\boldsymbol{Q}, u^s, Z)$*

$$\mathsf{Adv}_{\mathcal{A}}(k) = \left| \Pr[\mathcal{A}(\boldsymbol{Q}, u^s, g^{1/(x+s)}) = 1] - \Pr[\mathcal{A}(\boldsymbol{Q}, u^s, Z) = 1] \right|$$

is negligible in k, where s is a random element in $\mathbb{Z}_p$ and Z is a random element of G.

The above three assumptions are dynamic in the sense that the value q can change for different polynomial time adversaries. They are variants of the CDH assumption and being used to construct cryptosystems for various functionalities without random oracles. The security analysis of them are proposed by Cheon [Che06]. Note that the three assumptions are stronger than the CDH assumption in the sense that if the CDH assumption does not hold, then the three assumptions will be broken.

2.2 Cryptographic Tools

2.2.1 Secret Sharing

The principle of the well-known Shamir secret sharing scheme [Sha79] is reviewed here. Roughly speaking, a secret sharing scheme allows a user to divide a secret into n pieces, called shares, so that any t share holders together can recover the secret. The major idea is that it takes t points to define a polynomial, say, $f(x)$ of degree $t-1$. One could generate f in such a way that $f(0)$ is the secret to be shared. Each share is then a point $(i, f(i))$. Now with t points, one could recover the polynomial and thus the value $f(0)$. On the other hand, with only $t-1$ points, nothing about $f(0)$ would be revealed since there are exponentially many curves that pass through those $t-1$ points.

Preparation Let x be the secret to be shared. Randomly pick a polynomial f of degree $t-1$ such that $f(0) = x$. Each share is defined as $(i, f(i))$ for $i = 1$ to n.

Reconstruction One could make use of Lagrange interpolation to recover the value $f(0)$ when t points are given.

- Let $\mathcal{I}$ be a set such that $|\mathcal{I}| = t$ and that for all $i \in \mathcal{I}$, $f(i)$ is known.
- The Lagrange polynomial interpolation technique states that

$$f(x) := \sum_{i \in \mathcal{I}} f(i)\lambda_i(x),$$

where $\lambda_i(x)$, called the Lagrange basis polynomials, is defined as

$$\lambda_i(x) := \prod_{j \in \mathcal{I} \setminus \{i\}} \frac{x-j}{i-j}.$$

Since people are interested in $f(0)$ in the secret sharing scheme, λ_i is used to denote the value of $\lambda_i(0)$ and referred to as the Lagrange coefficient.

- Thus, to recover the secret, one first computes the Lagrange coefficient λ_i as

$$\lambda_i := \prod_{j \in \mathcal{I} \setminus \{i\}} \frac{-j}{i-j}.$$

- Then, $f(0)$ can be recovered as

$$f(0) := \sum_{i \in \mathcal{I}} f(i)\lambda_i.$$

2.2.2 Hash Function

A hash function, $H : \{0,1\}^* \rightarrow \{0,1\}^n$, is an efficient algorithm that maps a bit string of variable length to a bit string of fixed length n [CW79]. The outputs of a hash function are called hash values. Hash functions are primarily used to generate a shortened reference of an original bit string. In this thesis, it is required that the hash function should be *collision resistant* [Dam88]. That is, it is computationally infeasible to find two bit strings m_0 and m_1 with $m_0 \neq m_1$ such that $H(m_0) = H(m_1)$.

The random oracle model (ROM), first introduced by Fiat and Shamir [FS87] and formalized by Bellare and Rogaway [BR93], may be adopted in the security proof of many cryptosystems in which hash functions are involved. In the random oracle model, a hash function is treated as a random oracle that responds to each query with a truly random value chosen uniformly from the hash value space, except that it always responds the same value for the same query. That is, to gain a hash value of an input, one has to query the random oracle, and through this means a strong randomness of the hash function's output is guaranteed.

Since the output of real hash functions, for example SHA and MD5, does not follow the uniform distribution over the hash value space, some believe that a proof in the random oracle model is heuristic and such a proof may not guarantee the security when the random oracle is replaced by a concrete hash function. For instance, the research by Canetti, Goldreich and Halevi [CGH98, CGH04] demonstrated that cryptographic schemes secure in the random oracle model may not be secure when the random oracle is replaced by a real hash function. However, others argue that the counterexamples are totally artificial and practical system will not be designed that

way. In a word, the random oracle model is still widely accepted in cryptography security proof as it may lead to better reduction, even though designing schemes whose security does not rely on the random oracle is, without any doubt, more desirable.

2.2.3 General Forking Lemma

The General Forking Lemma [BN06], proposed by Bellare and Neven, is a general version of the Forking Lemma in [PS96] and has been used to prove the security of a variety of digital signature schemes and other random-oracle based cryptographic constructions. It is recalled here to better understand the thesis.

Lemma 2.1 (General Forking Lemma). *Fix an integer $Q \geq 1$ and a set H of size $|H| \geq 2$. Let $\mathcal{A}$ be a randomized algorithm that on input $x, h_1, \cdots, h_Q$ returns a pair (J, σ) where $J \in \{0, \cdots, Q\}$ and σ is referred as side output. Let IG be a randomized algorithm called the input generator. Let $acc_{\mathcal{A}} = \Pr[J \geq 1 : x \leftarrow IG; h_1, \cdots, h_Q \leftarrow H; (J, \sigma) \leftarrow \mathcal{A}(x, h_1, \cdots, h_Q)]$ be the accepting probability of $\mathcal{A}$. The forking algorithm $F_{\mathcal{A}}$ associated to $\mathcal{A}$ is the randomized algorithm that takes as input x and proceeds as follows:*

Algorithm $F_{\mathcal{A}}(x)$
- *pick coins* ρ *for* $\mathcal{A}$ *at random*
- $h_1, \cdots, h_Q \leftarrow H$
- $(J, \sigma) \leftarrow \mathcal{A}(x, h_1, \cdots, h_Q; \rho)$
- *If* $J = 0$ *then return* $(0, \perp, \perp)$
- $h'_J, \cdots, h'_Q \leftarrow H$
- $(J', \sigma') \leftarrow \mathcal{A}(x, h_1, \cdots, h_{J-1}, h'_J, \cdots, H'_Q; \rho)$
- *If* $(J = J'$ *and* $h_J \neq h'_J)$ *then return* $(1, \sigma, \sigma')$
- *Else return*$(0, \perp, \perp)$.

Let $frk = \Pr[b = 1 : x \leftarrow IG; (b, \sigma, \sigma') \leftarrow F_{\mathcal{A}}(x)]$. *Then* $frk \geq acc_{\mathcal{A}}(\frac{acc_{\mathcal{A}}}{Q} - \frac{1}{|H|})$.

Roughly speaking the lemma says that if an algorithm $\mathcal{A}$ runs two times in related executions and the process forks at a certain point, then with non-negligible probability (with overwhelming probability if $|H|$ is large) $\mathcal{A}$ can generate two different outputs with the same property. In this thesis the General Forking Lemma is

used to prove the security of the concrete OFE schemes in Section 3.3 and Section 4.3.

2.2.4 Zero-Knowledge Proof-of-Knowledge

In cryptography, a zero-knowledge proof, introduced by Goldwasser, Micali and Rackoff in [GMR85], is a protocol by which a prover convinces a verifier of the truth of a given statement, without leaking any additional information except the fact that the statement is true. A proof of knowledge [BG93], is a protocol by which the prover convinces a verifier that it knows some secret. Specifically, the prover proves it knows some value w such that $(x, w) \in R$ where R is a relation and x is publicly known to both the prover and the verifier. Putting the two notions together, a zero-knowledge proof-of-knowledge is a protocol by which a prover convinces a verifier that it knows some secret while not revealing any information about the secret.

Let R be an efficiently computable binary relation. For a pair $(x, w) \in R$ x is called the statement and w the witness. Let L be the language consisting of statements in R. Let $negl(\cdot)$ be a negligible function in the security parameter k.

A non-interactive proof system Π for a language L (a relation R) consists of a key generation algorithm K, a prover algorithm P and a verifier algorithm V. The key generation algorithm takes 1^k as input and outputs a common reference string σ. The prover algorithm takes as input (σ, x, w) and outputs a proof π. The verifier algorithm takes as input (σ, x, π) and outputs $\top$ if the proof is acceptable and $\perp$ otherwise.

Definition 2.9 *Π a non-interactive zero-knowledge proof system for a language L (a relation R) if it satisfies the following three properties.*

- Completeness. *For any adversary $\mathcal{A}$,*

$$\Pr\left[\begin{array}{ll} \sigma \leftarrow K(1^k); & \\ (x, w) \leftarrow \mathcal{A}(\sigma); & : V(\sigma, x, \pi) = \perp \text{ if } x \in L \\ \pi \leftarrow P(\sigma, x, w) & \end{array}\right] \leq negl(k).$$

- Soundness. *For any adversary $\mathcal{A}$,*

$$\Pr\left[\begin{array}{ll} \sigma \leftarrow K(1^k); & \\ (x, \pi) \leftarrow \mathcal{A}(\sigma) & : V(\sigma, x, \pi) = \top \text{ if } x \notin L \end{array}\right] \leq negl(k).$$

- Zero-knowledge. *There exists a PPT simulator* $S = (S_1, S_2)$ *such that for any adversary* $\mathcal{A}$*,*

$$\left|\Pr\left[\sigma \leftarrow K(1^k) : \mathcal{A}^{P(\sigma,\cdot,\cdot)}(\sigma) = 1\right] - \Pr\left[(\sigma,\tau) \leftarrow S_1(1^k) : \mathcal{A}^{S(\sigma,\tau,\cdot,\cdot)}(\sigma) = 1\right]\right|$$

$\leq negl(k)$*, where* $S(\sigma, \tau, x, w) = S_2(\sigma, \tau, x)$ *for* $(x, w) \in R$*.*

Definition 2.10 Π *a proof of knowledge system for a language* L *(a relation* R*) if it satisfies the following two properties.*

- Completeness. *This is the same as in Definition 2.9.*
- Knowledge Extraction. *There exists a knowledge extractor* $E = (E_1, E_2)$ *such that for any adversary* $\mathcal{A}$*,*

$$\left|\Pr\left[\sigma \leftarrow K(1^k) : \mathcal{A}(\sigma) = 1\right] - \Pr\left[(\sigma,\xi) \leftarrow E_1(1^k) : \mathcal{A}(\sigma) = 1\right]\right| \leq negl(k),$$

and for any adversary $\mathcal{A}$ *we have*

$$\Pr\left[\begin{array}{ll} (\sigma,\xi) \leftarrow E_1(1^k); & \\ (x,\pi) \leftarrow \mathcal{A}(\sigma); & : V(\sigma,x,\pi) = \top \text{ but } (x,w) \notin R \\ w \leftarrow E_2(\sigma,\xi,x,\pi) & \end{array}\right] \leq negl(k).$$

Note that the knowledge extraction property implies soundness property, because if one can extract a witness from a proof, then the corresponding statement obviously belongs to the language. A proof of knowledge system is zero-knowledge if it further satisfies the zero-knowledge property defined above.

2.2.5 Digital Signature

Digital signature, first proposed by Diffie and Hellman [DH76], is the electronic counterpart of the traditional handwriting signature. A valid digital signature can authenticate the identity of the sender of a message or the signer of a document. It can also ensure that the message or document is not altered in transfer.

SYNTAX. The syntax of a digital signature scheme is formalized in [GMR88] and consists of the following three algorithms:

$\mathsf{KGen}(1^k) \rightarrow (sk, vk)$. The key generation algorithm takes as input 1^k where k is a security parameter, and outputs a signing key sk and a verification key vk.

$\mathsf{Sig}(sk, m) \rightarrow \sigma$. The signing algorithm algorithm takes as input a signing key sk and a message m from the associated message space $\mathcal{M}$, and outputs a signature σ.

$\mathsf{Ver}(m, \sigma, vk) \rightarrow \top/\bot$. The verification algorithm Ver takes a message m, a signature σ and a verification key pk and outputs either a valid symbol $\top$ or an invalid symbol $\bot$.

Correctness of a signature scheme requires that $\mathsf{Ver}(m, \mathsf{Sig}(sk, m), vk) = \top$, for any $m \in \mathcal{M}$ and any key pair (sk, vk) outputted by the key generation algorithm $\mathsf{KGen}(1^k)$.

SECURITY. The standard security notion for signatures is *existential unforgeability under adaptive chosen message attacks* [GMR88], denoted by $\mathsf{UF\text{-}CMA}$. Intuitively, this notion requires that an adversary is not able to generate a signature on a new message on behalf of a target signer. Formally, it is defined through the following experiment conducted between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$.

Setup : The challenger $\mathcal{C}$ runs $\mathsf{KGen}(1^k)$ to generate a signing/verification key pair (sk, vk), gives the verification key vk to the adversary $\mathcal{A}$, and keeps sk as private.

Queries : $\mathcal{A}$ can adaptively and sequentially make q signing queries to $\mathcal{C}$ where q is bounded by a polynomial in the security parameter k. For the i-th query, $\mathcal{A}$ submits a message m_i to $\mathcal{C}$, who returns a signature generated by $\mathsf{Sig}(sk, m)$.

Output : $\mathcal{A}$ outputs a message signature pair (m^*, σ^*). $\mathcal{A}$ wins the experiment if m^* has not been previously queried by $\mathcal{A}$ and $\mathsf{Ver}(m^*, \sigma^*, vk) = \top$.

Definition 2.11 *A digital signature scheme is existentially unforgeable against adaptive chosen message attacks (*$\mathsf{UF\text{-}CMA}$*-secure) if no PPT adversary wins the above experiment with non-negligible probability.*

In this thesis, the terminology used in [HYWS08b] is sometimes followed and the signature schemes reviewed above are refered to as conventional signature schemes to distinguish them from ring signature schemes that will be reviewed below. In addition, the signatures generated by a conventional signature scheme are sometimes referred as conventional signatures.

2.2.6 Ring Signatures

Ring signature, introduced by Rivest, Shamir and Tauman in Asiacrypt 2001 [RST01], is a kind of digital signature that can be generated by any member of a number of

users that form a ring. A ring signature convinces people that the signer is in the ring, but the exact identity of the signer will be hidden. Due to its importance in the applications where user's privacy is of concern, ring signatures have been widely researched.

SYNTAX. A ring signature scheme $\mathsf{RS} = (\mathsf{KGen}, \mathsf{Sig}, \mathsf{Ver})$ is formalized by the following three efficient algorithms:

$\mathsf{KGen}(1^k) \rightarrow (rsk, rpk)$. The key generation algorithm takes as input 1^k and outputs a secret key rsk and a public key rpk.

$\mathsf{Sig}(rsk, m, R) \rightarrow \sigma$. The signing algorithm takes as input a secret key rsk, a message m from the associated message space $\mathcal{M}$ and a set of public keys $R := \{rpk_i\}_{i=1}^{l}$ such that $rpk \in R$ where (rsk, rpk) is a key pair outputted by $\mathsf{KGen}(1^k)$, and outputs a ring signature σ.

$\mathsf{Ver}(m, \sigma, R) \rightarrow \top/\bot$. The verification algorithm takes as input a message m, a ring signature σ and a list of public keys $R := \{rpk_i\}_{i=1}^{l}$, and outputs $\top$ or $\bot$ for accept or reject.

Correctness of a ring signature scheme requires that $\mathsf{Ver}(m, \mathsf{Sig}(rsk, m, R), R) = \top$, for any $m \in \mathcal{M}$ and any key pair (rsk, rpk) outputted by the key generation algorithm $\mathsf{KGen}(1^k)$ such that $rpk \in R$.

A 2-user ring signature scheme is a specific case of the above that only supports rings R for which $|R| = 2$.

SECURITY. Two security properties that a ring signature scheme should satisfy are *anonymity* and *unforgeability*. Intuitively, anonymity requires that no one should be able to determine which number of a ring has actually generated the signature, and unforgeability requires that no one should be able to generate a ring signature if none of the ring members' private keys is known.

A fundamental requirement for anonymity will be reviewed, named *basic anonymity* considered in [BKM06]. It is defined through the following experiment conducted between a challenger $\mathcal{C}$ and an the adversary $\mathcal{A}$.

Setup : The challenger $\mathcal{C}$ runs $\mathsf{KGen}(1^k)$ to generate a number of key pairs $\{rsk_i, rpk_i\}_{i=1}^{n(k)}$ and gives the set of public keys $R := \{rpk_i\}_{i=1}^{n(k)}$ to the adversary $\mathcal{A}$.

Queries : $\mathcal{A}$ chooses an index s, a message m and a set S such that $rpk_s \in S$ and $S \subset R$, and submits (s, m, S) to $\mathcal{C}$ for a signing query, who returns the output of $\mathsf{Sig}(rsk_s, m, S)$ to $\mathcal{A}$. $\mathcal{A}$ can adaptively make a polynomial number of such queries.

Challenge : $\mathcal{A}$ submits (m^*, i_0, i_1, R^*) where $R^* \subset R$, $rpk_{i_0}, rpk_{i_1} \in R^*$. $\mathcal{C}$ randomly flips a coin $b \in \{0,1\}$ and gives $\mathcal{A}$ the challenge ring signature $\sigma^* \leftarrow \mathsf{Sig}(rsk_{i_b}, m^*, R^*)$.

Guess : $\mathcal{A}$ outputs a guess $b' \in \{0,1\}$. $\mathcal{A}$ wins the experiment if $b' = b$.

Definition 2.12 *A ring signature scheme satisfies basic anonymity if no PPT adversary wins the above experiment with probability non-negligibly greater than* $1/2$.

For the security of unforgeability, the model *unforgeability under an adaptive attack, against a static adversary in the 2-user setting* [Boy07] is considered. It is defined through the following experiment conducted between a challenger $\mathcal{C}$ and an the adversary $\mathcal{A}$.

Setup : The challenger $\mathcal{C}$ runs $\mathsf{KGen}(1^k)$ twice to generate two key pairs (rsk_0, rpk_0) and (rsk_1, rpk_1), respectively, and gives the two public keys $R := \{rpk_0, rpk_1\}$ to the adversary $\mathcal{A}$.

Queries : $\mathcal{A}$ chooses an index $i \in \{0,1\}$, a message m and a set S where $|S| = 2$, $S \cap R \neq \emptyset$ and $rpk_i \in S$, and submits (i, m, S) to $\mathcal{C}$ for a signing query. $\mathcal{C}$ returns the output of $\mathsf{Sig}(rsk_i, m, S)$ to $\mathcal{A}$. $\mathcal{A}$ can adaptively make a polynomial number of such queries.

Output : $\mathcal{A}$ outputs a message signature pair (m^*, σ^*). $\mathcal{A}$ wins the experiment if $\mathsf{Ver}(m^*, \sigma^*, R) = \top$ and $(\cdot, m^*, R)$ has not been previously queried by $\mathcal{A}$.

Definition 2.13 *A ring signature scheme is (existentially) unforgeable under an adaptive attack, against a static adversary in the 2-user setting if there is no adversary wins the experiment with non-negligible probability.*

2.2.7 Public-key Encryption

In traditional private-key (symmetric) encryption scheme, the encryption and decryption processes uses the same secret key. Thus two involving parties must somehow share the secret key before communication. In 1976, Diffie and Hellman [DH76] introduced the concept of public-key (asymmetric) encryption, in which data can be encrypted under a public key and only the user with the corresponding secret key can decrypt the ciphertext. Compared with private-key encryption, one prominent advantage of public-key encryption is that two communicating parties do not require a initial procedure to share a key.

SYNTAX. The syntax of a public-key encryption scheme is formalized in [GM84] and consists of the following three algorithms:

$\mathsf{KGen}(1^k) \rightarrow (dk, ek)$. The key generation algorithm takes as input 1^k, and outputs a decryption key dk and an encryption key ek.

$\mathsf{Enc}(ek, m) \rightarrow c$. The encryption algorithm takes as input an encryption key ek and a message m from the associated message space $\mathcal{M}$, and outputs a ciphertext c.

$\mathsf{Dec}(dk, c) \rightarrow m$. The decryption algorithm takes as input a decryption key dk and a ciphertext c and returns a corresponding message m.

Correctness of a public-key encryption scheme requires that $\mathsf{Dec}(dk, \mathsf{Enc}(ek, m)) = m$, for any $m \in \mathcal{M}$ and any key pair (dk, ek) outputted by the key generation algorithm $\mathsf{KGen}(1^k)$.

SECURITY. The primary goal of a public-key encryption scheme is to guarantee the privacy of data. The standard security notion for public-key encryption schemes is *indistinguishability against adaptive chosen ciphertext attacks* [RS92], denoted by IND-CCA2. Intuitively, IND-CCA2 means that given a properly generated encryption key, no adversary $\mathcal{A}$ can distinguish encryptions of any two-equal length messages m_0, m_1 under this key. IND-CCA2 security captures strong message (data)-privacy property and guarantees that, given a challenge ciphertext, no valid information about the underlying message (plaintext, or data) will be leaked. Formally, it is defined through the following experiment conducted between a challenger $\mathcal{C}$ and an the adversary $\mathcal{A}$.

Setup : The challenger $\mathcal{C}$ runs $\mathsf{KGen}(1^k)$ to generate a decryption/encryption key pair (dk, ek), gives the encryption key ek to the adversary $\mathcal{A}$, and keeps dk as private.

Phase 1 Queries : $\mathcal{A}$ can adaptively make a polynomial number of decryption queries to $\mathcal{C}$. For each query, $\mathcal{A}$ submits a ciphertext c to $\mathcal{A}$, who returns a message m outputted by $\mathsf{Dec}(dk, c)$.

Challenge phase : $\mathcal{A}$ submits two messages m_0 and m_1 with equal bit length. $\mathcal{C}$ randomly flips a coin $b \in \{0, 1\}$ and computes $c_b = \mathsf{Enc}(ek, m_b)$. $\mathcal{C}$ returns c_b to $\mathcal{A}$.

Phase 2 Queries : $\mathcal{A}$ can further adaptively make a polynomial number of decryption queries to $\mathcal{C}$ with the only limitation that the challenge ciphertext c_b should not be queried. For each query, $\mathcal{A}$ submits a ciphertext c to $\mathcal{A}$, who returns a message m outputted by $\mathsf{Dec}(dk, c)$.

Guess : $\mathcal{A}$ outputs a guess $b' \in \{0,1\}$. $\mathcal{A}$ wins the game if $b' = b$.

Definition 2.14 *A public-key encryption scheme is indistinguishable against adaptive chosen ciphertext attacks (*IND-CCA2*-secure) if no PPT adversary wins the above experiment with non-negligible probability.*

Another security notion for public-key encryption is named *indistinguishability of keys against adaptive chosen ciphertext attacks* [BBDP01], denoted by IK-CCA2. IK-CCA2 captures strong key-privacy property. It means that given two randomly selected encryption keys ek_0 and ek_1, no adversary $\mathcal{A}$ can distinguish encryptions of the same message m under the two different keys. Given a challenge ciphertext, no valid information about the underlying key will be leaked in an IK-CCA2-secure encryption scheme. Formally, this security notion is defined through the following experiment conducted between a challenger $\mathcal{C}$ and an the adversary $\mathcal{A}$.

Setup : The challenger $\mathcal{C}$ runs $\mathsf{KGen}(1^k)$ to generate two decryption/encryption key pair (dk_0, ek_0) and (dk_1, ek_1), separately. $\mathcal{C}$ gives the encryption keys ek_0 and ek_1 to the adversary $\mathcal{A}$, and keeps dk_0 and dk_1 as private.

Phase 1 Queries : $\mathcal{A}$ can adaptively make a polynomial number of decryption queries to $\mathcal{C}$ with respect to the two keys ek_0 and ek_1. When $\mathcal{A}$ submits a ciphertext c to $\mathcal{C}$ to ask for a decryption with respect to ek_0, $\mathcal{C}$ returns a message m outputted by $\mathsf{Dec}(dk_0, c)$. When $\mathcal{A}$ submits a ciphertext c to $\mathcal{C}$ to ask for a decryption with respect to ek_1, $\mathcal{C}$ returns a message m outputted by $\mathsf{Dec}(dk_1, c)$.

Challenge phase : $\mathcal{A}$ submits a challenge message m^* to $\mathcal{C}$. $\mathcal{C}$ randomly flips a coin $b \in \{0,1\}$ and computes $c_b = \mathsf{Enc}(ek_b, m^*)$. $\mathcal{C}$ returns c_b to $\mathcal{A}$.

Phase 2 Queries : $\mathcal{A}$ can further adaptively make a polynomial number of decryption queries to $\mathcal{C}$ with the only limitation that the challenge ciphertext c_b should not be queried. For each query, $\mathcal{C}$ answer as in Phase 1 queries.

Guess : $\mathcal{A}$ outputs a guess $b' \in \{0,1\}$. $\mathcal{A}$ wins the game if $b' = b$.

Definition 2.15 *A public-key encryption scheme is indistinguishable of keys against adaptive chosen ciphertext attacks (*IK-CCA2*-secure) if no PPT adversary wins the above experiment with non-negligible probability.*

2.2.8 Ciphertext-Policy Attribute-based Encryption

Before reviewing the notion of ciphertext-policy attribute-based encryption (CP-ABE), the definition of an access structure is reviewed.

Definition 2.16 (Access structure [Bei96]) *Let $\{P_1, \cdots, P_n\}$ be a set of parties. A collection $\mathbb{A} \subseteq 2^{\{P_1,\cdots,P_n\}}$ is monotone if for any B, C: if $B \in \mathbb{A}$ and $B \subseteq C$ then $C \subseteq \mathbb{A}$. An access structure (respectively, monotonic access structure) is a collection (respectively, monotone collection) $\mathbb{A}$ of non-empty subsets of $\{P_1, \cdots, P_n\}$, i.e., $\mathbb{A} \subseteq 2^{\{P_1,\cdots,P_n\}} \setminus \{\emptyset\}$. The sets in $\mathbb{A}$ are called the authorized sets, and the sets not in $\mathbb{A}$ are called unauthorized sets.*

The notion and security model of ciphertext-policy attribute-based encryption introduced in [LW12] is reviewed here. A ciphertext-policy attribute-based encryption scheme $\mathcal{E}$ comprises four efficient algorithms: $\mathcal{E} := (\mathsf{Setup}, \mathsf{Encrypt}, \mathsf{KeyGen}, \mathsf{Decrypt})$.

$\mathsf{Setup}(1^k, U)$. The setup algorithm takes as input 1^k and attribute universe description U, and outputs the public parameters PM and a master key MK.

$\mathsf{Encrypt}(\mathrm{PM}, M, \mathbb{A})$. The encryption algorithm takes as input the public parameters PM, a message M, and an access structure $\mathbb{A}$ over the universe of attributes, and outputs a ciphertext CT such that only a user that possesses a set of attributes that satisfies the access structure $\mathbb{A}$ will be able to decrypt the ciphertext. It is assumed that the ciphertext implicitly contains $\mathbb{A}$.

$\mathsf{KeyGen}(\mathrm{MK}, S)$. The key generation algorithm takes as input the master key MK and a set of attributes S, and outputs a private key SK.

$\mathsf{Decrypt}(\mathrm{PM}, \mathrm{CT}, \mathrm{SK})$. The decryption algorithm takes as input the public parameters PM, a ciphertext CT, which contains an access policy $\mathbb{A}$, and a private key SK, which is a private key for a set of attributes S. If the set of attributes S satisfies the access structure $\mathbb{A}$ then the algorithm will decrypt the ciphertext and returns a message M.

Correctness for a CP-ABE scheme $\mathcal{E}$ states that $\mathsf{Decrypt}(\mathrm{PM}, \mathsf{Encrypt}(\mathrm{PM}, M, \mathbb{A}), \mathsf{KeyGen}(\mathrm{MK}, S)) = M$, for any set of attributes S that satisfies the access structure $\mathbb{A}$.

Security Model.

The full security for CP-ABE [LW12] is described by a security game between a challenger and an adversary. The game proceeds as follows.

Setup : The challenger runs Setup algorithm and sends the public parameters PM to the adversary.

Phase 1 : The adversary adaptively queries the challenger for private keys corresponding to sets of attributes $S_1, \cdots, S_{q_1}$. For the query with respect to S_k , the challenger responds with a secret key outputted by $\mathsf{KeyGen}(\mathrm{MK}, S_k)$.

Challenge : The adversary submits two equal length messages M_0 and M_1, and an access structure $\mathbb{A}$ such that none of the previous sets $S_1, \cdots, S_{q_1}$ satisfies the access structure. The challenger flips a random coin $b \in \{0, 1\}$, and encrypts M_b under $\mathbb{A}$, producing CT. It sends CT to the attacker.

Phase 2 : The adversary adaptively queries the challenger for private keys corresponding to sets of attributes $S_{q_1+1}, \cdots, S_q$, with the restriction that none of these satisfies $\mathbb{A}$. For the query with respect to S_k, the challenger responds with a secret key outputted by $\mathsf{KeyGen}(\mathrm{MK}, S_k)$.

Guess : The adversary outputs a guess $b' \in \{0, 1\}$.

The advantage of an adversary in this game is defined as $\Pr[b' = b] - \frac{1}{2}$.

Definition 2.17 *A ciphertext-policy attribute-based encryption scheme is fully secure if all polynomial time adversaries have at most a negligible advantage in the above security game.*

2.3 Optimistic Fair Exchange

The notion of Optimistic Fair Exchange (OFE) was introduced by Asokan, Schunter and Waidner [ASW97] in 1997 to solve the fair exchange problem over open networks. An optimistic fair exchange of digital signatures protocol comprises three kinds of participants: a signer, a verifier and a semi-trusted third party called an "arbitrator". Typically such a protocol is conducted in three message flows. First, Alice the signer initiates the protocol by delivering a partial signature to Bob the verifier. A valid partial signature not only serves as evidence to Bob that Alice has

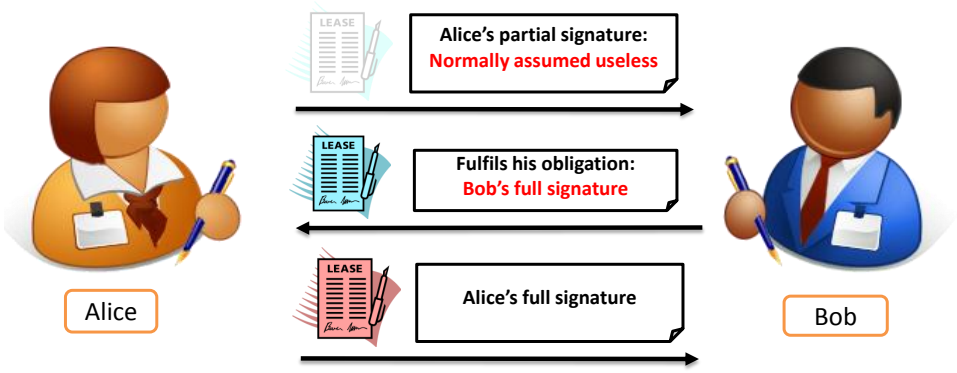

Figure 2.1: **A typical execution of OFE without disputes.**

committed to endorse a certain message, but also assures Bob that he will receive Alice's full signature at the end of the protocol. The assurance follows the fact that the arbitrator is capable of converting the partial signature into a full one. In the second step, Bob delivers his full signature to Alice. Later, if Alice is honest, she will send her full signature to Bob in the third step, and this completes the exchange process (see Figure 2.1). In case Alice refuses to do so or there is a network failure, Bob can ask the arbitrator to make a resolution, who will convert Alice's partial signature into a full one and send it back to Bob (see Figure 2.2). Note that under the normal situation, participation of the arbitrator is not required and thus, the term "optimistic".

Below the definition for non-interactive optimistic fair exchange (OFE) in the multi-user setting and chosen-key model [HYWS08b] is reviewed.

2.3.1 Definition

Definition 2.18 (Optimistic Fair Exchange) *A non-interactive optimistic fair exchange scheme involves the users (signers and verifiers) and the arbitrator, and consists of the following (probabilistic) polynomial-time algorithms:*

- $\mathsf{Setup}^{\mathsf{TTP}}$*: On input* 1^k*, this algorithm generates a secret key* ASK*, and a public key* APK *of the arbitrator.*
- $\mathsf{Setup}^{\mathsf{User}}$*: On input* 1^k *and (optionally)* APK*, this algorithm outputs a secret/public key pair* $(\mathsf{SK}, \mathsf{PK})$*. For a user* U_i*,* $(\mathsf{SK}_i, \mathsf{PK}_i)$ *is used to denote the user's key pair.*

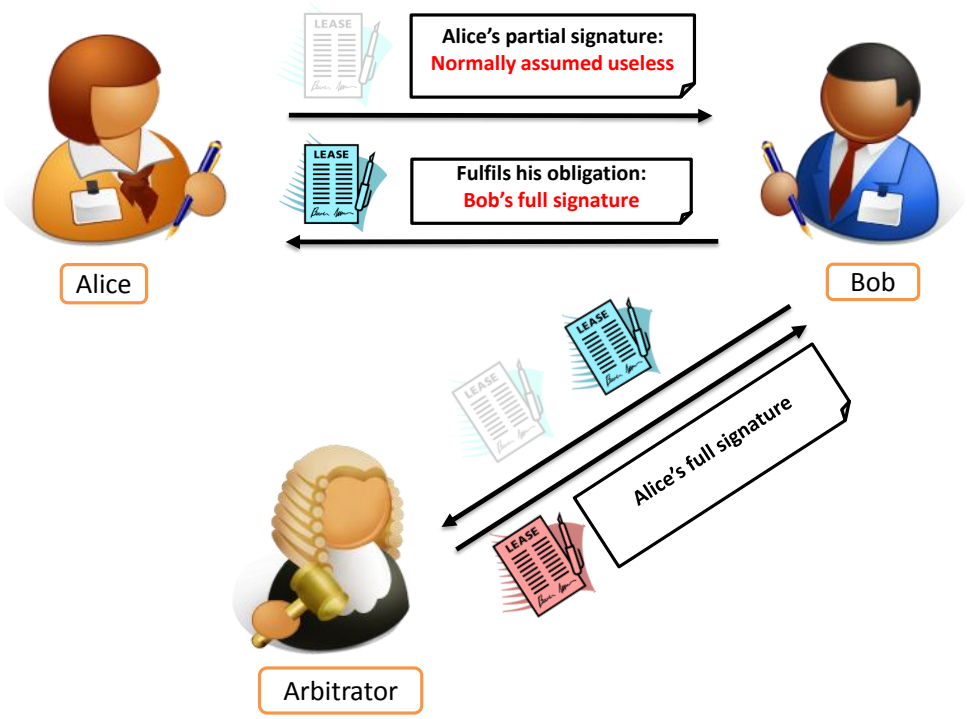

Figure 2.2: **An execution of OFE when disputes occur.**

- Sig *and* Ver*: These are similar to conventional signing and verification algorithms of a conventional signature scheme.* $\mathsf{Sig}(m, \mathsf{SK}_i, \mathsf{APK})$*, outputs a (full) signature* σ *of* U_i*'s on message* m*, while* $\mathsf{Ver}(m, \sigma, \mathsf{PK}_i, \mathsf{APK})$ *outputs* $\top$ *or* $\perp$*, indicating* σ *is a valid full signature of* U_i*'s on* m *or not.*

- PSig *and* PVer*: These are partial signing and verification algorithms respectively.* $\mathsf{PSig}(m, \mathsf{SK}_i, \mathsf{APK})$ *outputs a partial signature* σ_P*, while* $\mathsf{PVer}(m, \sigma_P, \mathsf{PK}_i, \mathsf{APK})$ *outputs* $\top$ *for acceptance or* $\perp$ *for rejection.*

- Res*: This is the resolution algorithm run by the arbitrator.* $\mathsf{Res}(m, \sigma_P, \mathsf{ASK}, \mathsf{PK}_i)$ *outputs a full signature* σ*, or* $\perp$ *indicating the failure of resolving a partial signature.*

Correctness property means the output is what one wants if everybody follows the algorithms. More specifically, it states that

- $\mathsf{Ver}(m, \mathsf{Sig}(m, \mathsf{SK}_i, \mathsf{APK}), \mathsf{PK}_i, \mathsf{APK}) = \top$,
  $\mathsf{PVer}(m, \mathsf{PSig}(m, \mathsf{SK}_i, \mathsf{APK}), \mathsf{PK}_i, \mathsf{APK}) = \top$, *and*
  $\mathsf{Ver}(m, \mathsf{Res}(m, \mathsf{PSig}(m, \mathsf{SK}_i, \mathsf{APK}), \mathsf{ASK}, \mathsf{PK}_i), \mathsf{PK}_i, \mathsf{APK}) = \top$.

Resolution ambiguity property states that any "resolved signature" $\mathsf{Res}(m, \mathsf{PSig}(m, \mathsf{SK}_i, \mathsf{APK}), \mathsf{ASK}, \mathsf{PK}_i)$ *is computationally indistinguishable from the "actual signature"* $\mathsf{Sig}(m, \mathsf{SK}_i, \mathsf{APK})$.

In a concrete OFE protocol, the signer U_i first runs PSig to generates a partial signature σ_P and sends it to the verifier. The verifier runs PVer to check the validity of the partial signature and fulfills its obligation if the output is $\top$. After that, the signer U_i runs Sig to generate a full signature σ and sends it to the verifier. In this normal case, the arbitrator does not need to be involved. However, in the cases there is a network failure or the signer U_i refuses to send its full signature σ, the verifier can approach the arbitrator with the partial signature σ_P and a proof that it has fulfilled its obligation. If the partial signature σ_P and the proof are both valid, the arbitrator runs Res to convert the partial signature σ_P into a full one and sends it to the verifier. As in the previous definitions such as [DR03, DLY07], the definition does not address the subtle issue of how to prove to the arbitrator that one has fulfilled its obligation in an specific exchange.

It should be emphasised that in the existing definition of resolution ambiguity, the distinguisher does not have any adaptive adversarial capability. Specifically, the distinguisher is neither given any oracle access nor the arbitrator's secret key.

2.3.2 Security in Multi-User setting and Chosen-key Model

The security of an optimistic fair exchange scheme consists of three aspects: security against signers, security against verifiers, and security against the arbitrator. The security in the multi-user setting and chosen-key model [HYWS08b] guarantees the highest level of security for optimistic fair exchange in literatures. This security model is reviewed below.

SECURITY AGAINST SIGNERS. Intuitively, it is required that no signer should be able to produce a partial signature that is accepted by the verifier but cannot be converted into a full signature by the honest arbitrator. This ensures fairness for verifiers, that is, after the verifier fulfills its obligation, it can always get the signer's full signature. Formally, consider the following experiment.

$$\begin{aligned}
\mathsf{Setup}^{\mathsf{TTP}}(1^k) &\rightarrow (\mathsf{ASK}, \mathsf{APK}) \\
(m, \sigma_P, \mathsf{PK}^*) &\leftarrow \mathcal{A}^{O_{\mathsf{Res}}}(\mathsf{APK}) \\
\sigma &\leftarrow \mathsf{Res}(m, \sigma_P, \mathsf{ASK}, \mathsf{PK}^*) \\
\text{success of } \mathcal{A} &:= \begin{bmatrix} \mathsf{PVer}(m, \sigma_P, \mathsf{PK}^*, \mathsf{APK}) = \top \\ \mathsf{Ver}(m, \sigma, \mathsf{PK}^*, \mathsf{APK}) = \bot \end{bmatrix}
\end{aligned}$$

where the resolution oracle O_{Res} takes as input a valid partial signature σ_P on message m under the public key PK_i (i.e. $(m, \sigma_P, \mathsf{PK}_i)$ such that $\mathsf{PVer}(m, \sigma_P, \mathsf{PK}_i, \mathsf{APK}) = \top$), and outputs a full signature σ on m under PK_i. In the resolution oracle queries, the adversary can make queries with respect to adversarially chosen public keys, without requiring to know the corresponding secret keys. The advantage of $\mathcal{A}$ in this experiment is defined as the probability of success of $\mathcal{A}$.

SECURITY AGAINST VERIFIERS. This aspect of security guarantees that no verifier should be able to generate a full signature or convert any partial signature σ_P into a full one by himself. This is to ensure fairness for the signer, that is, to gain the signer's full signature, the verifier has to fulfill its obligation first. Formally, consider the following experiment.

$$\begin{aligned}
\mathsf{Setup}^{\mathsf{TTP}}(1^k) &\rightarrow (\mathsf{ASK}, \mathsf{APK}) \\
\mathsf{Setup}^{\mathsf{User}}(1^k) &\rightarrow (\mathsf{SK}, \mathsf{PK}) \\
(m, \sigma) &\leftarrow \mathcal{A}^{O_{\mathsf{PSig}}, O_{\mathsf{Res}}}(\mathsf{PK}, \mathsf{APK}) \\
\text{success of } \mathcal{A} &:= \begin{bmatrix} \mathsf{Ver}(m, \sigma, \mathsf{PK}, \mathsf{APK}) = \top \\ (m, \cdot) \notin Query(\mathcal{A}, O_{\mathsf{Res}}) \end{bmatrix}
\end{aligned}$$

where oracle O_{Res} is described in the experiment of security against signers, $Query(\mathcal{A}, O_{\mathsf{Res}})$ is the set of queries made by $\mathcal{A}$ to oracle O_{Res}, and oracle O_{PSig} takes as input a message m and outputs a partial signature σ_P on m under PK. It is widely believed[1] that there is no need to provide $\mathcal{A}$ with access to the signing oracle O_{Sig}, which takes as input a message m and outputs a full signature σ on m under the challenge signer's public PK, as O_{Sig} could be simulated by O_{PSig} and O_{Res}. The advantage of $\mathcal{A}$ in this experiment is defined as the probability of success of $\mathcal{A}$.

SECURITY AGAINST THE ARBITRATOR. This aspect of security guarantees that the arbitrator should not be able to produce a full signature on behalf of

[1] In Chapter 3 it will be shown that the signing oracle is in fact needed and it will be one contribution of this thesis.

the signer on any message, unless the arbitrator has seen a partial signature on that message. This is to ensure fairness for the signer, that is, the arbitrator can only generate signatures on messages that the signer herself will potentially sign. Formally, consider the following experiment.

$$\begin{aligned} \mathsf{Setup}^{\mathsf{User}}(1^k) &\rightarrow (\mathsf{SK}, \mathsf{PK}) \\ (\mathsf{ASK}^*, \mathsf{APK}) &\leftarrow \mathcal{A}(\mathsf{PK}) \\ (m, \sigma) &\leftarrow \mathcal{A}^{O_{\mathsf{PSig}}}(\mathsf{ASK}^*, \mathsf{APK}, \mathsf{PK}) \\ \text{success of } \mathcal{A} &:= \left[\begin{array}{c} \mathsf{Ver}(m, \sigma, \mathsf{PK}, \mathsf{APK}) = \top \\ m \notin Query(\mathcal{A}, O_{\mathsf{PSig}}) \end{array} \right] \end{aligned}$$

where ASK^* is $\mathcal{A}$'s state information, which might not be the corresponding private key of APK[2], oracle O_{PSig} is described in the previous experiment, and $Query(\mathcal{A}, O_{\mathsf{PSig}})$ is the set of queries made by $\mathcal{A}$ to oracle O_{PSig}. As in the experiment of security against verifiers, the adversary is not given the signing oracle access, as it is believed that the adversary can gain a full signature by firstly receiving a partial signature from the partial signing oracle and then converting it into a full one by using the secret arbitrator key. The advantage of $\mathcal{A}$ in this experiment is defined as the probability of success of $\mathcal{A}$.

Definition 2.19 *A non-interactive optimistic fair exchange scheme is said to be secure in the multi-user setting and chosen-key model if no PPT adversary wins any of the above experiments with non-negligible advantage.*

2.4 Ambiguous Optimistic Fair Exchange

Most existing optimistic fair exchange schemes share the property that Bob could use Alice's partial signature as a mean to convince others that Alice has agreed to commit in something, which may not be desirable in some applications such as fair negotiation. In 2008, Huang, Yang, Wong and Susilo [HYWS08a] studied a variant of optimistic fair exchange, in which a new security notion called *signer ambiguity* was introduced, which requires that Bob is able to forge a partial signature that is indistinguishable from a real partial signature generated by Alice. With this

[2]The adversary is allowed to adversarially choose the arbitrator's public key and may even not know the corresponding secret key. In the process, the adversary is allowed to keep the useful information as its state information.

property, Bob will not be able to convince any outsiders that a partial signature was indeed generated by Alice. They named OFE with this new security property *Ambiguous Optimistic Fair Exchange* (AOFE). Besides, they proposed a formal security model for AOFE in the multi-user setting and chosen key model.

Recall that in OFE, given a partial signature, outsiders can be convinced that it is generated by the signer Alice. However, in AOFE, outsiders can only be convinced that a partial signature is generated by either the signer Alice or the verifier Bob. The main difference between OFE and AOFE is that in AOFE the verifier Bob cannot convince outsiders of the authorship of the partial signature generated by the signer Alice. To achieve the signer ambiguity in AOFE, it requires to include both Alice and Bob's public keys into the (partial) signing and verification algorithms. That is to say, the syntax for OFE will not be suitable for AOFE. The notion and security model of the ambiguous optimistic fair exchange protocol introduced in [HYWS08a] is reviewed here.

Definition 2.20 *An ambiguous optimistic fair exchange scheme involves the users (signers and verifiers) and the arbitrator, and consists of the following (probabilistic) polynomial-time algorithms:*

- PMGen*: On input* 1^κ *where* κ *is a security parameter, it outputs a system parameter* PM*.*
- $\mathsf{Setup}^{\mathsf{TTP}}$*: On input* PM*, the algorithm generates a secret key* ASK*, and a public key* APK *of the arbitrator.*
- $\mathsf{Setup}^{\mathsf{User}}$*: On input* PM *and (optionally)* APK*, it outputs a secret/public key pair* (SK, PK)*. For a user* U_i*,* (SK_i, PK_i) *is used to denote the user's key pair.*
- Sig *and* Ver*:* $\mathsf{Sig}(M, SK_i, PK_i, PK_j, APK)$*, outputs a (full) signature* σ *on* M *of user* U_i *with the designated verifier* U_j*, while* $\mathsf{Ver}(M, \sigma, PK_i, PK_j, APK)$ *outputs* $\top$ *or* $\perp$*, indicating* σ *is* U_i*'s valid full signature on* M *with the designated verifier* U_j *or not.*
- PSig *and* PVer*: These are partial signing and verification algorithms respectively.* $\mathsf{PSig}(M, SK_i, PK_i, PK_j, APK)$ *outputs a partial signature* σ_P*, while* $\mathsf{PVer}(M, \sigma_P, \mathbf{PK}, APK)$ *outputs* $\top$ *or* $\perp$*, where* $\mathbf{PK} = \{PK_i, PK_j\}$*.*

- Res*: This is the resolution algorithm.* $\mathsf{Res}(M, \sigma_P, ASK, \mathbf{PK})$*, where* $\mathbf{PK} = \{PK_i, PK_j\}$*, outputs a full signature* σ*, or* $\perp$ *indicating the failure of resolving a partial signature.*

Resolution ambiguity property states that

- *any "resolved signature"* $\mathsf{Res}(M,\ \mathsf{PSig}(M,\ SK_i,\ PK_i,\ PK_j,\ APK),\ ASK,\ \{PK_i,\ PK_j\})$ *is computationally indistinguishable from the "actual signature"* $\mathsf{Sig}(M,\ SK_i,\ PK_i,\ PK_j,\ APK)$.

The security of an ambiguous optimistic fair exchange scheme consists of four aspects: signer ambiguity, security against signers, security against verifiers, and security against the arbitrator. The security models of them in the multi-user setting and chosen-key model are reviewed below.

SIGNER AMBIGUITY. Intuitively, this security notion requires that, given a partial signature σ_P from a signer A, a verifier B should not be able to convince others that A was indeed the signer of σ_P, as B can generate partial signatures that look indistinguishable from those generated by A. Formally, consider the following experiment.

$$
\begin{aligned}
PM &\leftarrow \mathsf{PMGen}(1^k) \\
(ASK, APK) &\leftarrow \mathsf{Setup}^{\mathsf{TTP}}(PM) \\
(M, (SK_0, PK_0), (SK_1, PK_1), \delta) &\leftarrow D^{O_{\mathsf{Res}}}(APK) \\
b &\leftarrow \{0,1\} \\
\sigma_P &\leftarrow \mathsf{PSig}(M, SK_b, PK_b, PK_{1-b}, APK) \\
b' &\leftarrow D^{O_{\mathsf{Res}}}(\sigma_P, \delta) \\
\text{success of } A &:= \begin{bmatrix} b' = b \\ (M, \sigma_P, \{PK_0, PK_1\}) \notin Query(D, O_{\mathsf{Res}}) \end{bmatrix}
\end{aligned}
$$

where δ is D's state information, oracle O_{Res} takes as input a valid partial signature σ_P of user U_i on message M with respect to verifier U_j (i.e. $(M,\ \sigma_P,\ PK_i,\ PK_j)$ such that $\mathsf{PVer}(M,\ \sigma_P,\ \{PK_i,\ PK_j\},\ APK) = \top$), and outputs a full signature σ on M under PK_i, PK_j, and $Query(D, O_{\mathsf{Res}})$ is the set of valid queries D issued to the resolution oracle. The advantage of $\mathcal{A}$ in this experiment is defined as the probability of success of $\mathcal{A}$.

SECURITY AGAINST SIGNERS. This security notion requires no signer should be able to produce a partial signature that looks good to a verifier but cannot be resolved to a full signature by the honest arbitrator. Formally, consider the following experiment.

$$\begin{aligned}
PM &\leftarrow \mathsf{PMGen}(1^k)\\
(ASK, APK) &\leftarrow \mathsf{Setup}^{\mathsf{TTP}}(PM)\\
(SK_B, PK_B) &\leftarrow \mathsf{Setup}^{\mathsf{User}}(PM, APK)\\
(M, \sigma_P, PK_A) &\leftarrow \mathcal{A}^{O^B_{\mathsf{PSig}}, O_{\mathsf{Res}}}(APK, PK_B)\\
\sigma &\leftarrow \mathsf{Res}(M, \sigma_P, ASK, \{PK_A, PK_B\})\\
&\qquad \mathsf{PVer}(M, \sigma_P, \{PK_A, PK_B\}, APK) = \top\\
\text{success of } \mathcal{A} &:= \left[\begin{array}{c} \mathsf{Ver}(M, \sigma, PK_A, PK_B, APK) = \bot \\ (M, PK_A) \notin Query(\mathcal{A}, O^B_{\mathsf{PSig}}) \end{array}\right]
\end{aligned}$$

where oracle O_{Res} is described in the previous experiment, oracle O^B_{PSig} takes as input (M, PK_i) and outputs a signature on M with respect to PK_i and PK_B generated using SK_B, and $Query(\mathcal{A}, O^B_{\mathsf{PSig}})$ is the set of queries made by $\mathcal{A}$ to oracle O^B_{PSig}. The advantage of $\mathcal{A}$ in this experiment is defined as the probability of success of $\mathcal{A}$.

SECURITY AGAINST VERIFIERS. This security notion requires no verifier should be able to complete any partial signature σ_P into a full signature, without explicitly asking the arbitrator to do so. Formally, consider the following experiment.

$$\begin{aligned}
PM &\leftarrow \mathsf{PMGen}(1^{\mathsf{k}})\\
(ASK, APK) &\leftarrow \mathsf{Setup}^{\mathsf{TTP}}(PM)\\
(SK_A, PK_A) &\leftarrow \mathsf{Setup}^{\mathsf{User}}(PM, APK)\\
(M, \sigma, PK_B) &\leftarrow \mathcal{A}^{O_{\mathsf{PSig}}, O_{\mathsf{Res}}}(APK, PK_A)\\
\text{success of } \mathcal{A} &:= \left[\begin{array}{c} \mathsf{Ver}(M, \sigma, PK_A, PK_B, APK) = \top \\ (M, \cdot, \{PK_A, PK_B\}) \notin Query(\mathcal{A}, O_{\mathsf{Res}}) \end{array}\right]
\end{aligned}$$

where oracle O_{Res} is described in the experiment of signer ambiguity, $Query(\mathcal{A}, O_{\mathsf{Res}})$ is the set of queries made by $\mathcal{A}$ to oracle O_{Res}, and oracle O_{PSig} takes as input (M, PK_j) and outputs a signature on M with respect to PK_A and PK_j generated using SK_A. The advantage of $\mathcal{A}$ in this experiment is defined as the probability of success of $\mathcal{A}$.

SECURITY AGAINST THE ARBITRATOR. This security notion requires that the arbitrator should not be able to produce a full signature without explicitly asking

the signer to generate a partial one. Formally, consider the following experiment.

$$\begin{aligned} PM &\leftarrow \mathsf{PMGen}(1^k) \\ (APK, ASK^*) &\leftarrow \mathcal{A}(PM) \\ (SK_A, PK_A) &\leftarrow \mathsf{Setup}^{\mathsf{User}}(PM, APK) \\ (M, \sigma, PK_B) &\leftarrow \mathcal{A}^{O_{\mathsf{PSig}}}(ASK^*, APK, PK_A) \\ \text{success of } \mathcal{A} &:= \begin{bmatrix} \mathsf{Ver}(M, \sigma, PK_A, PK_B, APK) = \top \\ (M, PK_B) \notin Query(\mathcal{A}, O_{\mathsf{PSig}}) \end{bmatrix} \end{aligned}$$

where ASK^* is $\mathcal{A}$'s state information, which might not be the corresponding private key of APK, oracle O_{PSig} is described in the previous experiment, and $Query(\mathcal{A}, O_{\mathsf{PSig}})$ is the set of queries made by $\mathcal{A}$ to oracle O_{PSig}. The advantage of $\mathcal{A}$ in this experiment is defined as the probability of success of $\mathcal{A}$.

Definition 2.21 *An ambiguous optimistic fair exchange scheme is said to be secure in the multi-user setting and chosen-key model if no PPT adversary wins any of the above experiments with non-negligible advantage.*

2.5 Concurrent Signatures

Concurrent Signatures, introduced by Chen, Kudla and Paterson in [CKP04], is viewed as a new primitive for solving fair exchange of digital signatures problem without the help of a third trusted party. In a concurrent signature scheme, two parties exchange their signatures in such a way that the signatures will concurrently bind to their respective signers upon the release of some important information named *keystone*. Before the release of the keystone, the signatures are ambiguous and no outsiders can distinguish the real signers, as either party may have generated both signatures. An overview of concurrent signatures is illustrated in Figure 2.3.

Compared with optimistic fair exchange, concurrent signature schemes do not require any trusted third party (TTP). However, it is at the sacrifice that the party who controls the keystone always has an advantage over the other party that it can freely choose when to release the keystone or whether to release the keystone or not. Concurrent signatures are believed to be useful in the applications where the above concern is not a problem for the involving parties.

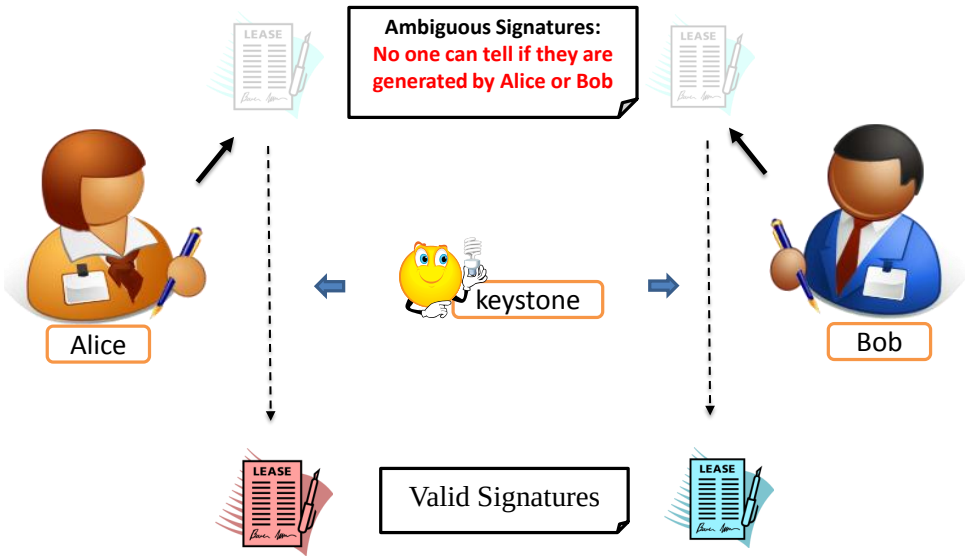

Figure 2.3: **An overview of Concurrent Signatures.**

2.5.1 Definition of Concurrent Signatures

The definition of concurrent signatures from [CKP04] is reviewed. A concurrent signature comprises four algorithms (SETUP, ASIGN, AVERIFY, VERIFY). Their formal definitions are given below.

SETUP. On input security parameter 1^λ, this probabilistic algorithm outputs the description of the set of participants $\mathcal{U}$, the message space $\mathcal{M}$, the signature space $\mathcal{S}$, the keystone space $\mathcal{K}$, the keystone fix space $\mathcal{F}$, and a function KGEN : $\mathcal{K} \to \mathcal{F}$. It is also assumed the public keys $\{X_i\}$ and their respective secret keys $\{x_i\}$ are also generated by this algorithm. π is used to denote additional system parameters. It is assumed that $(\pi, \mathcal{U}, \mathcal{M}, \mathcal{S}, \mathcal{K}, \mathcal{F}, \mathsf{KGEN}, \{X_i\})$ are available to all participants while each user retains his/her own secret key x_i.

ASIGN. On input (X_i, X_j, x_i, h_2, M) where $X_i, X_j \in \{X_i\}$ such that $X_i \neq X_j$, x_i being the secret key corresponds to the public key X_i, $h_2 \in \mathcal{F}$, $M \in \mathcal{M}$, this algorithm outputs an ambiguous signature $\sigma = (s, h_1, h_2)$ on message M, where $s \in \mathcal{S}$, $h_1, h_2 \in \mathcal{F}$.

AVERIFY. On input (σ, X_i, X_j, M), where $\sigma = (s, h_1, h_2)$, $s \in \mathcal{S}$, $h_1, h_2 \in \mathcal{F}$, X_i, X_j are distinct public keys and $M \in \mathcal{M}$, outputs 0/1. It is required that AVERIFY$((s, h_1, h_2), X_i, X_j, M)$ = AVERIFY$((s, h_2, h_1), X_j, X_i, M)$.

VERIFY. On input $(k, (\sigma, X_i, X_j, M))$ such that $k \in \mathcal{K}$ and $\sigma = (s, h_1, h_2)$, it outputs 0 if $h_2 \neq \mathsf{KGEN}(k)$. Otherwise, it outputs $\mathsf{AVERIFY}(\sigma, X_i, X_j, M)$.

Below is a recap of the interactive protocol in which the above algorithms are used in the exchange of signatures in a concurrent manner amongst two participants.

1. Suppose SETUP has been executed and all participants have their own key pair already. Below how Alice with key pair (X_A, x_A) exchanges signatures with Bob with key pair (X_B, x_B) is described.

2. Alice picks a random $k \in \mathcal{K}$, computes $f = \mathsf{KGEN}(k)$ and obtains $\sigma_A := (s_A, h_A, f)$ from $\mathsf{ASIGN}(X_A, X_B, x_A, f, M_A)$. Alice sends σ_A, M_A to Bob.

3. Bob verifies Alice's ambiguous signature by invoking $\mathsf{AVERIFY}(\sigma_A, X_A, X_B, M_A)$. If σ_A is valid, Bob obtains $\sigma_B := (s_B, h_B, f)$ from $\mathsf{ASIGN}(X_B, X_A, x_B, f, M_B)$. Bob sends (σ_B, M_B) to Alice.

4. Alice verifies Bob's ambiguous signature by invoking $\mathsf{AVERIFY}(\sigma_B, X_B, X_A, M_B)$. If σ_B is valid, Alice releases the keystone k. Parse S_A as $(k, \sigma_A, X_A, X_B, M_A)$ and S_B as $(k, \sigma_B, X_B, X_A, M_B)$.

5. Everybody can now verify both signatures S_A and S_B using VERIFY.

The correctness is defined in the usual manner. Specifically, if $\sigma = \mathsf{ASIGN}(X_i, X_j, x_i, f, M)$ and $S = (k, \sigma, X_i, X_j, M)$, then $\mathsf{AVERIFY}(\sigma, X_i, X_j, M) = 1$. In addition, if $f = \mathsf{KGEN}(k)$ for some $k \in \mathcal{K}$, then $\mathsf{VERIFY}(S) = 1$.

2.5.2 Security Model

As discussed in [CKP04], a concurrent signature should satisfy three security requirements, namely, *unforgeability*, *ambiguity* and *fairness*. For completeness, these security requirements we reviewed as follows.

Unforgeability

The following game is used to capture the existential unforgeability of a concurrent signature.

Definition 2.22 (Unforgeability) *A concurrent signature is unforgeable if no PPT adversary $\mathcal{A}$ can win the following game with a challenger $\mathcal{C}$.*

Setup. $\mathcal{C}$ *invokes* $\mathsf{SETUP}(1^\lambda)$ *for a security parameter* 1^λ *and obtains a set of parameters* $(\pi, \mathcal{U}, \mathcal{M}, \mathcal{S}, \mathcal{K}, \mathcal{F}, \mathsf{KGEN}, \{X_i\})$ *and the corresponding set of secret keys* $\{x_i\}$. $\mathcal{C}$ *gives the set of parameters to* $\mathcal{A}$ *while retaining the set of secret keys* $\{x_i\}$.

Queries. $\mathcal{A}$ *is allowed to issue to following queries in an adaptive manner.*

1. KGEN*:* $\mathcal{C}$ *randomly generates* $k \in_R \mathcal{K}$ *and returns* $f = \mathsf{KGEN}(k)$ *to* $\mathcal{A}$.
2. *Keystone Reveal: On input* f *such that* f *is an output of the* KGEN *query,* $\mathcal{C}$ *returns* k *such that* $f = \mathsf{KGEN}(k)$. *Otherwise* $\mathcal{C}$ *returns* $\perp$.
3. ASIGN*: On input* (X_i, X_j, h_2, M) *such that* $X_i, X_j \in \{X_i\}$, $h_2 \in \mathcal{F}$, $M \in \mathcal{M}$, $\mathcal{C}$ *replies with* $\mathsf{ASIGN}(X_i, X_j, x_i, h_2, M)$.
4. *Secret Key Reveal: On input* $X_i \in \{X_i\}$, $\mathcal{C}$ *returns* x_i.

Output. Finally $\mathcal{A}$ *outputs a signature* $\sigma^* = (s^*, h^*, f^*)$, *a message* M^* *and two public keys* X_c^*, X_d^*. $\mathcal{A}$ *wins the game if* $\mathsf{AVERIFY}(\sigma^*, X_c^*, X_d^*, M^* = 1)$ *and either one of the following is true:*

- (X_c^*, X_d^*, f^*, M^*) *is not an input to the* ASIGN *query and* X_c^*, X_d^* *is not an input to the secret key reveal query.*
- $\mathcal{A}$ *has not made any* ASIGN *query of the form* (X_c^*, X, f^*, M^*) *for all* $X \in \{X_i\} \setminus \{X_c^*\}$, *no secret key reveal query was made with input* X_c^* *and* f^* *is the output of* KGEN *query or* $\mathcal{A}$ *also outputs* k^* *such that* $f^* = \mathsf{KGEN}(k^*)$.

Ambiguity

The following game is used to capture ambiguity of a concurrent signature.

Definition 2.23 (Ambiguity) *A concurrent signature is ambiguous if no PPT adversary* $\mathcal{A}$ *can win the following game with a challenger* $\mathcal{C}$.

Setup. Same as Setup in Definition 2.22.

Phase 1. $\mathcal{A}$ *is allowed to made a sequence of* KGEN*, Keystone Reveal,* ASIGN *and Secret Key Reveal query, which are answered as in Definition 2.22.*

Challenge. $\mathcal{A}$ *outputs two public keys* X_i, X_j *and a message* M *as challenge.* $\mathcal{C}$ *randomly picks* $k \in_R \mathcal{K}$ *and computes* $f = \mathsf{KGEN}(k)$. $\mathcal{C}$ *then flips a fair coin* $b \in_R \{0,1\}$. *If* $b = 0$, $\mathcal{C}$ *computes* $\sigma_0 = \mathsf{ASIGN}(X_i, X_j, x_i, f, M)$. *Otherwise,* $\mathcal{C}$ *computes* $(s, h, f) = \mathsf{ASIGN}(X_j, X_i, x_j, f, M)$ *and parse* σ_1 *as* (s, f, h). $\mathcal{C}$ *returns* σ_b *to* $\mathcal{A}$.

Phase 2. $\mathcal{A}$ *can make another sequence of queries as in phase 1.*

Output. Finally $\mathcal{A}$ *outputs a guess bit* b'. $\mathcal{A}$ *wins the game if* $b = b'$ *and* $\mathcal{A}$ *did not make any Keystone Reveal query on input* f *or* h.

Fairness

The following game is used to capture fairness of a concurrent signature.

Definition 2.24 (Fairness) *A concurrent signature is fair if no PPT adversary* $\mathcal{A}$ *can win the following game with a challenger* $\mathcal{C}$.

Setup. Same as Setup in Definition 2.22.

Queries. $\mathcal{A}$ *is allowed to made a sequence of* KGEN, *Keystone Reveal,* ASIGN *and Secret Key Reveal query, which are answered as in Definition 2.22.*

Challenge. $\mathcal{A}$ *outputs two public keys* X_i, X_j *and two messages* M_i, M_j, *together with* $\sigma_i = (s_i, h_1, h_2)$ *such that* $\mathsf{AVERIFY}((s_i,\ h_1,\ h_2),\ X_i,\ X_j,\ M_i) = 1$. $\mathcal{C}$ *returns* $\sigma_j = (s_j, h_3, h_2) = \mathsf{ASIGN}(X_j, X_i, x_j, h_2, M_j)$.

Output. Finally $\mathcal{A}$ *either outputs a value* k. $\mathcal{A}$ *wins the game if* $f = \mathsf{KGEN}(k)$ *such that* f *was a previous output from* KGEN *query and no Keystone Reveal query on* f *was made.*

Part I

OFE with Stronger Security

Chapter 3

Optimistic Fair Exchange in the Enhanced Chosen-key Model

The security for OFE in the multi-user setting and chosen-key model consists of three aspects: security against signers, security against verifiers and security against the arbitrator. In this chapter, the security against verifiers and the security against the arbitrator are strengthened by allowing the adversary to have an extra oracle access.

3.1 Introduction

Due to its fundamental role in electronic commerce, optimistic fair exchange has been extensively studied [MS01, Mic03, DR03, Wan05, ES05, LOS+06, ZM07, OMO08, RS09, HV11, HWS11a]. It has been shown that optimistic fair exchange schemes can be constructed from *verifiably encrypted signatures* [ASW98, CD00, BGLS03, LOS+06, ZM07, RS09] and *sequential two-party multisignatures* [DR03], respectively. It is widely accepted that optimistic fair exchange schemes should have the property 'resolution ambiguity' [DR03, DLY07, HYWS08b], namely the actual signatures generated by the signer should be at least computationally indistinguishable from the resolved signatures generated by the arbitrator. As the intervention of an arbitrator could be due to a network failure, rather than the signer cheats, an optimistic fair exchange scheme with resolution ambiguity property can avoid bad publicity for the signer.

One main goal of modern cryptography is to define security models capturing possible practical attacks for cryptographic schemes. Defining a suitable security model for OFE protocols turns out to be an iterative process.

Early optimistic fair exchange protocols was studied in the single-user setting and the security model assumed only one signer and one verifier. The first formal

security model was proposed in [ASW97, ASW98] but fails to consider the case that the arbitrator itself may be dishonest. A more generalized model in the single-user setting was suggested by Dodis and Reyzin [DR03] to take into account a dishonest arbitrator.

Since many users may share the same arbitrator in the real world, the security model in the single-user setting does not capture the possible attacks by colluding dishonest users. In 2007, Dodis, Lee and Yum [DLY07] considered the multi-user security of optimistic fair exchange which allows dishonest users collude to cheat another user. They separated the security of optimistic fair exchange between single-user setting and multi-user setting by showing that an optimistic fair exchange instance provably secure in the single-user setting is not necessarily secure in the multi-user setting. Independently, this was also studied by Zhu, Susilo and Mu in 2007 [ZSM07].

Since then, the security of optimistic fair exchange intuitively covers the following three aspects.

- Security against signers: the signer should not be able to generate a partial signature that can not be converted into a full one by the honest arbitrator.
- Security against verifiers: the verifier should not be able to generate a full signature of the signer's by himself.
- Security against the arbitrator: the arbitrator should not be able to generate a full signature on behalf of the signer while not seeing a corresponding partial one.

In an orthogonal dimension, most optimistic fair exchange protocols are studied in the *certified-key* model (also known as the *registered-key* model [BCNP04]). In this model, it is assumed that the authenticity of public keys are verifiable and each user in the system should show its knowledge of the corresponding secret key in the public key registration stage to resist key substitution attacks. That is to say, in this model, the dishonest signer and the dishonest arbitrator has to show their knowledge of their corresponding secret keys, and the dishonest verifier can only make resolution queries with respect to the target signer and other public keys whose secret keys are known.

However, in the public key infrastructure, when a certification authority issues a certificate of a user's public key, the user is not required to show its knowledge

of the secret key. Thus the certified-key model is not practical enough, as it relies on a stronger assumption than the normal authenticity assumption placed on the certification authority.

In 2008, Huang, Yang, Wong and Susilo [HYWS08b] studied the security of optimistic fair exchange in the *chosen-key model*, in which the adversary can adversarially choose public keys without knowing the corresponding secret key. This model provides more realistic power to the adversary in attacking the honest users. On the one hand, the adversary can choose its own public key without knowing the corresponding secret key. Specifically, in the security against the arbitrator, the dishonest arbitrator is even allowed to set its own public key after having seen the target signer's public key. In the certified-key model, however, the dishonest arbitrator has to set its key pair before seeing the target signer's public key. On the other hand, for a dishonest signer or verifier, the adversary is allowed to make resolution queries with respect to arbitrary public keys, without knowing the corresponding private keys. They demonstrated, through an example, that a provably secure fair exchange in the certified-key model may not be secure in the chosen-key model. Furthermore, a generic optimistic fair exchange construction secure in the chosen-key model based on conventional signatures and ring signatures was proposed.

3.1.1 Motivation

The most fundamental work of OFE is to define security models, that capture realistic attacks. Based on these models, secure schemes will be designed. These security models are very essential to ensure that they capture practical situation, which will ensure that the protocols can be adopted in practice. In this chapter it is observed that the existing OFE models in fact do not capture realistic situation, where the adversary (i.e. the dishonest verifiers or the dishonest arbitrator) can actually observe the full signatures generated by the signer before launching the actual attack. This realistic situation implies that in the security model, the adversary should have been provided with the signing oracle, which will produce full signatures generated by the signer.

In all the proposed optimistic fair exchange models, in the security against the arbitrator, the dishonest arbitrator is only given a partial signing oracle, which takes as input a message and outputs the target signer's partial signature on this message. Furthermore, in the security against the verifier, the dishonest verifier is given the

partial signing oracle and a resolution oracle, which takes as input a partial signature and outputs a resolved signature. It should be noted that the two aspects of the OFE security have the common feature that the adversary (the dishonest verifiers or the dishonest arbitrator) is not provided the access to the actual signatures. This situation in fact separates the security models of OFE schemes from the realistic situation, which may result in the insecurity of the scheme in practice, which will hinder the adoption of OFE in practice.

To date, it is commonly believed that it is not required to provide the dishonest arbitrator with the signing oracle, as the dishonest arbitrator can generate a full signature by first gaining a partial signature from the partial signing oracle and then converting it into a full one using its own secret key. Similarly, in the models of security against the verifier, the dishonest verifier is given the partial signing oracle and a resolution oracle, which takes as input a partial signature and outputs a full signature, as it is commonly acknowledged that the signer can gain a full signature by first requesting a partial signature from the partial signing oracle and then asking the resolution oracle to convert it into a full one. Unfortunately, by carefully analyzing the existing models, it should be noticed that the above models at most allow the adversary (the dishonest arbitrator or the dishonest verifier) to gain the partial signatures and the resolved signatures generated by the arbitrator. For a malicious arbitrator who generates its public key based on a target signer and has no knowledge of its own secret key, it may not even be able to get a resolved signature. Hence, the real scenario that an adversary may also have access to the actual signatures generated by the signer is not captured. The crux of the issue in the security models is the lack of signing oracle access, which has been believed to be unnecessary since the partial signing oracle and the resolution oracle have been provided.

Furthermore, it is also identified that the notion of resolution ambiguity does not imply signing oracle is not of help to the adversary. The distinguisher in the definition of resolution ambiguity is not equipped with any power, i.e. it is not offered with any oracle access, not to mention the arbitrator's secret key, the resolution ambiguity property only guarantees a preliminary level of computational indistinguishability between the actual signature and the resolved signature. There is no guarantee that the actual signature is still computationally indistinguishable from the resolved signature in the view of the dishonest verifier, who can have partial

signing oracle access and resolution oracle access, let alone the dishonest arbitrator, who can have partial signing oracle access and choose the arbitration key pair himself.

3.1.2 The Contributions

In this chapter the security of OFE when an adversary is given an signing oracle is studied. The contribution comprises three aspects.

First, the enhanced chosen-key model for optimistic fair exchange that explicitly captures the real scenario that an adversary may have access to the actual signatures is proposed in Section 3.2. That is, besides the partial signing oracle and resolution oracle, the adversary is also provided with the signing oracle.

Next, to show the necessity of the enhanced model, in Section 3.3, it is shown that, even for a honest but curious arbitrator (the arbitrator correctly generates its key pair and knows its own secret arbitration key), the security in the multi-user setting and chosen-key model does not imply that in the enhanced model with a concrete optimistic fair exchange scheme that serves as a counterexample. In other works, the current definition of resolution ambiguity (the actual signatures generated by the signer should be computationally indistinguishable from the resolved signatures generated by the arbitrator) fails to guarantee that depriving the signing oracle is reasonable.

Since the current definition of resolution ambiguity fails to justify the absence of the signing oracle, a natural and straightforward question would be "if the definition of resolution ambiguity is further strengthened, would it be possible to fill the gap between the enhanced model and existing model?" Unfortunately, the answer is no. Specifically, in Section 3.4, it is demonstrated that, for a malicious arbitrator (the arbitrator adversarially chooses its arbitration public key and without knowing the corresponding secret key), a provably secure fair exchange in the existing model may not be secure in the enhanced model, even if the definition of resolution ambiguity is strengthened to the highest level, i.e., the actual signatures generated by the signer and the resolved signatures generated by the arbitrator have the identical distribution. This answer about the question firmly certifies the significance of the proposal of the enhanced chosen-key model.

3.2 The Enhanced Chosen-key Model

As discussed in the introduction of this chapter, the actual signature generated by the signer may be different from the resolved signature generated by the arbitrator based on a corresponding partial one. Thus it seems natural and straightforward that a security model should capture the real scenario that an adversary may see the actual signatures generated by the signer. To explicitly take this case into account, the following enhanced chosen-key model is proposed, i.e., in the experiments of security against verifiers and security against the arbitrator, the adversary we offered with the signing oracle access.

The syntax for non-interactive optimistic fair exchange (OFE) follows the one in the multi-user setting and chosen-key model proposed in [HYWS08b] and reviewed in Chapter 2. The security against signers is the same as that in the existing model, as the dishonest signer itself does not need to be provided a signing oracle. In the enhanced chosen-key model, the security against verifiers and the security against the arbitrator are defined as follows.

SECURITY AGAINST VERIFIERS. It is required that the probability that any PPT adversary $\mathcal{A}$ succeeds in the following experiment is negligible in k.

$$\begin{aligned} \mathsf{Setup}^{\mathsf{TTP}}(1^k) &\rightarrow (\mathsf{ASK}, \mathsf{APK}) \\ \mathsf{Setup}^{\mathsf{User}}(1^k) &\rightarrow (\mathsf{SK}, \mathsf{PK}) \\ (m, \sigma) &\leftarrow \mathcal{A}^{O_{\mathsf{PSig}}, O_{\mathsf{Sig}}, O_{\mathsf{Res}}}(\mathsf{PK}, \mathsf{APK}) \\ \text{success of } \mathcal{A} &:= \begin{bmatrix} \mathsf{Ver}(m, \sigma, \mathsf{PK}, \mathsf{APK}) = \top \\ (m, \cdot) \notin Query(\mathcal{A}, O_{\mathsf{Res}}) \\ m \notin Query(\mathcal{A}, O_{\mathsf{Sig}}) \end{bmatrix} \end{aligned}$$

where O_{PSig}, O_{Res} and $Query(\mathcal{A}, O_{\mathsf{Res}})$ are the same as reviewed in the existing model reviewed in Section 2.3.2, oracle O_{Sig} takes as input a message m and outputs a full signature σ on m generated using the challenge signer's secret key SK, and $Query(\mathcal{A}, O_{\mathsf{Sig}})$ is the set of queries made by $\mathcal{A}$ to oracle O_{Sig}.

SECURITY AGAINST THE ARBITRATOR. It is required that the probability

that any PPT adversary $\mathcal{A}$ succeeds in the following experiment is negligible in k.

$$\begin{aligned}
\mathsf{Setup}^{\mathsf{User}}(1^k) &\rightarrow (\mathsf{SK}, \mathsf{PK}) \\
(\mathsf{ASK}^*, \mathsf{APK}) &\leftarrow \mathcal{A}(\mathsf{PK}) \\
(m, \sigma) &\leftarrow \mathcal{A}^{O_{\mathsf{PSig}}, O_{\mathsf{Sig}}}(\mathsf{ASK}^*, \mathsf{APK}, \mathsf{PK}) \\
\text{success of } \mathcal{A} &:= \begin{bmatrix} \mathsf{Ver}(m, \sigma, \mathsf{PK}, \mathsf{APK}) = \top \\ m \notin Query(\mathcal{A}, O_{\mathsf{PSig}}) \\ m \notin Query(\mathcal{A}, O_{\mathsf{Sig}}) \end{bmatrix}
\end{aligned}$$

where ASK^*, O_{PSig} and $Query(\mathcal{A}, O_{\mathsf{PSig}})$ are the same as reviewed in the existing model reviewed in Section 2.3.2, and O_{Sig} and $Query(\mathcal{A},\ O_{\mathsf{Sig}})$ are the same as described in the above experiment.

Note that the models here are defined for OFE, rather than AOFE reviewed in Section 2.4. Therefore the PMGen algorithm in AOFE will not be involved here. Besides, in AOFE, the verifier's public key will be needed when a signer generates a partial or full signature. Thus the adversary in the security against verifiers in AOFE will need to output its own public key, but the adversary in the security against verifiers in OFE does not need to do so.

3.3 Separation of the Proposed Model and the Existing Model

In this section, a concrete optimistic fair exchange scheme that is secure in the multi-user setting and chosen key model reviewed in Section 2.3.2 but not secure against the arbitrator in the enhanced chosen-key model is presented. This demonstrates that the previous understanding that there is no need to provide the adversary with the signing oracle for the security against the arbitrator is wrong. That is to say, this counterexample clearly shows that the existing models fail to capture the the real scenario that an adversary may have access to the signer's actual signatures, and the enhanced chosen-key model is more complete and practical as it clearly captures this scenario. For the security against verifiers, a counterexample is not given to show that it makes a difference whether the signing oracle is given or not. However, since offering the signing oracle can capture the the real scenario that the verifier can have access to the signer's actual signatures, it is better to use the enhanced

model. Besides, even for the security against verifiers, the enhanced model is at least as strong as the existing one.

Prior to proposing the concrete optimistic fair exchange scheme that will serve as a counterexample, the high level description will be firstly provided. The concrete optimistic fair exchange scheme is constructed in the bilinear pairing setting. let $e : G \times G \to G_T$ be a bilinear pairing where g is a generator of G. The signer's public key is $X = g^x$ and a random $u \in G$. The arbitrator's public key is $Y = g^y$. The signer and the arbitrator keep their secret keys x and y as private, respectively. Let further $H_0 : \{0,1\}^* \to G$ and $H_1 : \{0,1\}^* \to \mathbb{Z}_p$ be two hash functions that will be viewed as random functions. The partial signature on a message m is a conventional signature $(H_0(m)^{1/(x+s)}, g^s, u^s) \in G^3$, which can be seen as a variation of the signatures from [HK12, HWS11a] in which they employ a programmable hash function [HK12] to guarantee the security without random oracles.

To fully sign a message, the signer does as follow.

1. The signer firstly generates a partial signature $(H_0(m)^{1/(x+s)}, g^s, u^s)$ on message m.

2. Let $\bar{m}$ be the complementary message of m, i.e. $\bar{m}$ and m are of the same bit-length but each i-th bits of them are different.

3. The signer then generates a designated confirmer signature [Cha94, HWS11a] $(\delta', v', \theta') = (H_0(\bar{m})^{1/(x+s')}, Y^{s'}, u^{s'})$ with the arbitrator being the confirmer on message $\bar{m}$ where s' is a random. This confirmer signature can be seen as a variation of the confirmer signature from [HWS11a] in which the hash function H_0 is replaced with a programmable hash function.

4. Note that $e(\delta', u)^x = e(H_0(\bar{m}), u)/e(\delta', \theta') := W$. The signer finally makes a non-interactive zero-knowledge proof of knowledge with respect to the message m and the signer's public key (X, u) that it knows the value x such that $g^x = X$ and $e(\delta', u)^x = W$ or it knows the arbitrator's secret key y. The signer can always make such a proof by using its secret key x.

5. The full signature comprises the partial signature, the designated confirmer signature and the non-interactive zero-knowledge proof of knowledge.

To convert a partial signature into a full one, the arbitrator uniformly samples a designated confirmer signature $(\delta', v', \theta') = (Z, Y^{s'}, u^{s'})$ from the signer's designated

signature space where $Z \in G$ and an exponent s' are randomly chosen. Let $W = e(H_0(\bar{m}), u)/e(\delta', \theta')$. The arbitrator then makes a non-interactive zero-knowledge proof of knowledge with respect to the message m and the signer's public key (X, u) that it knows the value x such that $g^x = X$ and $e(\delta', u)^x = W$ or it knows the arbitrator's secret key y. The arbitrator can always make the non-interactive zero-knowledge proof of knowledge using its secret key y.

Note that due to the invisibility property of a confirmer signature, the designated confirmer signature sampled by the arbitrator is indistinguishable from the one that is generated by the signer. Besides, the non-interactive zero-knowledge proof of knowledge guarantees the proof made by the signer is indistinguishable from the proof made by the arbitrator. Thus the resolution ambiguity property of the concrete optimistic fair exchange scheme holds.

Let G be a multiplicative cyclic group of prime order p, g be a generator of G, and $e : G \times G \to G_T$ be a bilinear pairing where G_T is a multiplicative group of order p. Let $H_0 : \{0,1\}^* \to G$ and $H_1 : \{0,1\}^* \to \mathbb{Z}_p$ be two hash functions. It is assumed that the public parameter (G, p, e, H_1, H_2) is shared by all users. The concrete OFE scheme is as follows.

- $\mathsf{Setup}^{\mathsf{TTP}}(1^k)$: The arbitrator chooses at random $y \in \mathbb{Z}_p$ and computes $Y = g^y$. The public key is set as $\mathsf{APK} = Y$, and the arbitrator keeps $\mathsf{ASK} = y$ as private.

- $\mathsf{Setup}^{\mathsf{User}}(1^k)$: Each user U_i chooses at random two secret values $x_i \in \mathbb{Z}_p$ and $u_i \in G$, and calculates $X_i = g^{x_i}$. The user sets $(\mathsf{SK}_i, \mathsf{PK}_i) = (x_i, (X_i, u_i))$.

- $\mathsf{PSig}(m, \mathsf{SK}_i, \mathsf{APK})$: To partially sign a message m, the user U_i chooses a random $s \in \mathbb{Z}_p$ and returns $\sigma_P := (H_0(m)^{1/(x_i+s)}, g^s, u_i^s)$.

- $\mathsf{PVer}(m, \sigma_P, \mathsf{PK}_i, \mathsf{APK})$: Given a partial signature σ_P from user U_i, the verifier parses σ_P as (δ, v, θ), and returns $\top$ if both $e(v, u_i) = e(g, \theta)$ and $e(\delta, X_i v) = e(H_0(m), g)$ hold. Otherwise, $\perp$ is returned.

- $\mathsf{Sig}(m, \mathsf{SK}_i, \mathsf{APK})$: To fully sign a message m, the user U_i

  1. chooses a random $s \in \mathbb{Z}_p$ and computes $\sigma_P := (H_0(m)^{1/(x_i+s)}, g^s, u_i^s)$.
  2. chooses a random $s' \in \mathbb{Z}_p$ and computes $(\delta', \gamma', \theta') := (H_0(\bar{m})^{1/(x_i+s')}, Y^{s'}, u_i^{s'})$ where $\bar{m}$ is the complementary message of m.

3. Denote $W := e(H_0(\bar{m}), u_i)/e(\delta', \theta')$. Note that $W = e(\delta', u_i)^{x_i}$. The user U_i chooses uniformly at random $r, c_1, t_1 \in \mathbb{Z}_p$, and computes $c = H_1(m||\mathsf{PK}_i||g^r||e(\delta', u_i)^r||g^{t_1}Y^{c_1})$, $c_0 = c - c_1 \pmod{p}$ and $t_0 = r - c_0 x_i \pmod{p}$.
4. The full signature is set as $\sigma := (\sigma_P, \delta', \gamma', \theta', c_0, t_0, c_1, t_1)$.

- $\mathsf{Ver}(m, \sigma, \mathsf{PK}_i, \mathsf{APK})$: Given a full signature σ from user U_i, a verifier does as follows.

1. Check whether $c_0, t_0, c_1, t_1 \in \mathbb{Z}_p$ and $\mathsf{PVer}(m, \sigma_P, \mathsf{PK}_i, \mathsf{APK}) = \top$.
2. Check whether $e(\gamma', u_i) = e(Y, \theta')$.
3. Compute $W = e(H_0(\bar{m}), u_i)/e(\delta', \theta')$ and
$$c = H_1(m||\mathsf{PK}_i||g^{t_0}(X_i)^{c_0}||e(\delta', u_i)^{t_0}W^{c_0}||g^{t_1}Y^{c_1}).$$
4. Verify whether $c_0 + c_1 = c \pmod{p}$.
5. If all the above hold, $\top$ is returned and otherwise, $\perp$ is returned.

- $\mathsf{Res}(m, \sigma_P, \mathsf{ASK}, \mathsf{PK}_i)$: For the user U_i's partial signature σ_P on message m, the arbitrator

1. first checks whether $\mathsf{PVer}(m, \sigma_P, \mathsf{PK}_i) = \top$. If so, continues; otherwise, returns $\perp$.
2. chooses at random $s' \in \mathbb{Z}_p$, $Z \in G$ and computes $(\delta', \gamma', \theta') := (Z, Y^{s'}, u_i^{s'})$.
3. computes $W = e(H_0(\bar{m}), u_i)/e(\delta', \theta')$ where $\bar{m}$ is the complementary message of m.
4. chooses uniformly at random $r, c_0, t_0 \in \mathbb{Z}_p$, and computes
$$c = H_1(m||\mathsf{PK}_i||g^{t_0}(X_i)^{c_0}||e(\delta', u_i)^{t_0}W^{c_0}||g^r),$$
$c_1 = c - c_0 \pmod{p}$ and $t_1 = r - c_1 y \pmod{p}$.
5. The full signature is set as $\sigma := (\sigma_P, \delta', \gamma', \theta', c_0, t_0, c_1, t_1)$.

3.3.1 Security Analysis.

It is not hard to verify that, in the above construction, any (partial) signature created by Sig (PSig) will be valid under Ver (PVer), and that any signature created by the arbitrator using Res based on a partial signature generated by PSig will be valid under Ver. Thus the correctness property of the above construction holds.

Based on the q-DHSDH Assumption, the resolution ambiguity property also holds.

Next the specific construction is secure in the multi-user setting and chosen-key model reviewed in Section 2.3.2 will be shown. Intuitively, the security against signers holds unconditionally as the arbitrator is always able to convert a signer's partial signature into a full one. Note that the full signature consists of two, essentially independent, parts. The first part is exactly the partial signature and the second part can be generated by the arbitrator independent of the partial signature. Therefore if the partial signature is correct then the second part can always be generated and the full signature will be correct. The security against verifiers holds due to the fact that one cannot make the proof of knowledge without knowing the signer's secret key or the arbitrator's secret key. The security against the arbitrator holds due to the unforgeability of the conventional signature scheme that generates the partial signature.

Theorem 3.1 *The above concrete optimistic fair exchange scheme is unconditionally secure against signers in the multi-user setting and chosen-key model.*

Proof. For any message m and any valid signature σ_P on m under the verification key PK_i, the arbitrator can always convert it into a full signature by using its own secret key ASK. Therefore, the security against signers always hold. □

Theorem 3.2 *The above concrete optimistic fair exchange scheme is secure against verifiers in the multi-user setting and chosen-key model if the q-HSDH Assumption holds.*

Proof. Suppose an adversary $\mathcal{A}$ makes q queries to the partial signing oracle and breaks the security against verifiers with non-negligible probability. It will be shown how to construct an algorithm $\mathcal{R}$ that breaks the q-HSDH Assumption.

Note that algorithm $\mathcal{R}$ is given $\mathbf{Q} =< g, g^x, u, \{(g^{1/(x+s_i)}, g^{s_i}, u^{s_i})\}_{i=1}^{q} >$. Its goal is to output another distinct triple $(g^{1/(x+s)}, g^s, u^s)$. Algorithm $\mathcal{R}$ simulates the

challenger and interacts with adversary $\mathcal{A}$. Algorithm $\mathcal{R}$ chooses a random integer $x' \in \mathbb{Z}_p$, sets $X' = g^{x'}$, flips a coin and gets a random bit b. According to the random bit b, $\mathcal{R}$ performs one of the following two games.

- Game 0 ($b = 0$): $\mathcal{R}$ forwards $\mathsf{PK} := (g^x, u)$ and $\mathsf{APK} = Y := X'$ to $\mathcal{A}$ and simulates the oracles for $\mathcal{A}$.

  H_0 Queries. At any time adversary $\mathcal{A}$ can query the random oracle H_0. To respond to these queries, $\mathcal{R}$ maintains a list of tuples $\langle m_i, a_i \rangle$ as explained below. This list is referred to as H_0-list. The list is initially empty. When $\mathcal{A}$ queries the oracle H_0 on a message $m_i \in \{0,1\}^*$, algorithm $\mathcal{R}$ responds as follows:

  1. If the query m_i already appears on the H_0-list in some tuple $\langle m_i, a_i \rangle$, then algorithm $\mathcal{R}$ responds with $H(m_i) = g^{a_i}$.
  2. Otherwise, $\mathcal{R}$ generates a random $a_i \in \mathbb{Z}_p$, adds the tuple $\langle m_i, a_i \rangle$ to the H_0-list and responds to $\mathcal{A}$ as $H(m_i) = g^{a_i}$.

  H_1 Queries. $\mathcal{A}$ can also query the random oracle H_1. To respond to these queries, $\mathcal{R}$ maintains a list of tuples $\langle string^{(i)}, c^{(i)} \rangle$ as explained below. This list is referred to as H_1-list. The list is initially empty. When $\mathcal{A}$ queries the oracle H_1 at a point $string \in \{0,1\}^*$, algorithm $\mathcal{R}$ responds as follows:

  1. If the query $string$ already appears on the H-list in some tuple $\langle string, c \rangle$, then algorithm $\mathcal{R}$ responds with $H(string) = c \in \mathbb{Z}_p$.
  2. Otherwise, $\mathcal{R}$ generates a random $c \in \mathbb{Z}_p$, adds the tuple $\langle string, c \rangle$ to the H_1-list and responds to $\mathcal{A}$ as $H(string) = c \in \mathbb{Z}_p$.

  PSig Queries. For the i-th partial signing query on message m_i, $\mathcal{R}$ checks whether there is a tuple $\langle m_i, a_i \rangle$ in the H_0-list. If not, $\mathcal{R}$ generates a random $a_i \in \mathbb{Z}_p$, adds the tuple $\langle m_i, a_i \rangle$ to the H_0-list. $\mathcal{R}$ returns $((g^{1/(x+s_i)})^{a_i}, g^{s_i}, u^{s_i})$ to $\mathcal{A}$ as the reply for the partial signing query.

  Res Queries. Given a resolution query $(m, \sigma_P, \mathsf{PK}_i)$ where $\mathsf{PK}_i = (X_i, u_i)$ is the signer's public key, algorithm $\mathcal{R}$ responds to this query as follows:

  1. checks whether $\mathsf{PVer}(\sigma_P, m, \mathsf{PK}_i) = \top$. If so, continues; otherwise, $\mathcal{R}$ responds to $\mathcal{A}$ with a special symbol $\perp$.

2. chooses at random $s' \in \mathbb{Z}_p$, $Z \in G$ and computes $(\delta', \gamma', \theta') := (Z, Y^{s'}, u_i^{s'})$ where $\bar{m}$ is the complementary message of m.
3. computes $W = e(H_1(\bar{m}), u_i)/e(\delta', \theta')$.
4. chooses uniformly at random $c_0, t_0, c_1, t_1 \in \mathbb{Z}_p$, and adds

$$< m||\mathsf{PK}_i||g^{t_0}(X_i)^{c_0}||e(\delta', u_i)^{t_0}W^{c_0}||g^{t_1}Y^{c_1}, c_0 + c_1 >$$

to the H_1-list. Since c_0, t_0, c_1, t_1 are all randomly chosen, the probability that $\mathcal{A}$ has previously asked a H_1 query on the string

$$m||\mathsf{PK}_i||g^{t_0}(X_i)^{c_0}||e(\delta', u_i)^{t_0}W^{c_0}||g^{t_1}Y^{c_1}$$

is negligible.

5. The full signature is set as $\sigma := (\sigma_P, \delta', \gamma', \theta', c_0, t_0, c_1, t_1)$ and returned to $\mathcal{A}$ as the reply of the resolution query.

- Game 1 ($b = 1$): $\mathcal{R}$ forwards $\mathsf{PK} := (X', u)$ and $\mathsf{APK} = Y := g^x$ to $\mathcal{A}$ and simulates the oracles for $\mathcal{A}$.

  The H_0, H_1 and resolution queries are simulated the same as in Game 0. For the i-th partial signing query on message m_i, $\mathcal{R}$ checks whether there is a tuple $\langle m_i, a_i \rangle$ in the H_0-list. If not, $\mathcal{R}$ generates a random $a_i \in \mathbb{Z}_p$, adds the tuple $\langle m_i, a_i \rangle$ to the H_0-list. $\mathcal{R}$ chooses at random $s_i' \in \mathbb{Z}_p$ and returns $(g^{a_i/(x'+s_i')}, g^{s_i'}, u^{s_i'})$ to $\mathcal{A}$ as the reply for the partial signing query.

It is easy to see that the above Hash queries, partial signing queries and resolution queries are indistinguishably simulated. Finally, $\mathcal{A}$ halts. It either admits failure, in which case so does $\mathcal{R}$, or it returns a full signature σ^* on message m^* without asking the resolution oracle with respect to the message m^* and the challenge signer's public key PK.

By the General Forking Lemma [BN06] (a standard rewinding technique in random oracle model), with non-negligible probability algorithm $\mathcal{R}$ is able to extract the value x or x'. The algorithm $\mathcal{R}$ wins if it extracts the value x. Note that the adversary $\mathcal{A}$ can not distinguish between Game 0 and Game 1, because the distributions of the simulation in the two games are the same. Therefore, if $\mathcal{A}$ succeeds with a non-negligible probability ϵ, the algorithm $\mathcal{R}$ wins with a non-negligible probability. □

Theorem 3.3 *The above concrete optimistic fair exchange scheme is secure against the arbitrator in the multi-user setting and chosen-key model if the q-SDH Assumption and q-HSDH assumption hold.*

Proof. Suppose $\mathcal{A}$ makes q partial signing queries. Let m_i be the i-th query submitted by $\mathcal{A}$, $\sigma_P^{(i)} = (\delta_i, v_i, \theta_i)$ be the reply of the partial signing oracle for the i-th query, and s_i be the exponent such that $v_i = g^{s_i}$. Finally the adversary $\mathcal{A}$ outputs a full signature σ^* on message m^* without asking the partial signing query on message m^*. Let $\sigma_P^* = (\delta^*, v^*, \theta^*) = (\delta^*, g^{s^*}, u^{s^*})$ be the partial signature contained in the full signature σ^*. Below two situations are distinguished.

Type I: There exists $1 \leq i \leq q$ such that $v^* = v_i$, which implies that $s^* = s_i$.

Type II: For all $1 \leq i \leq q$, $v^* \neq v_i$, which implies that $s^* \notin \{s_1, \cdots, s_q\}$.

Type I adversary

Suppose the Type I adversary $\mathcal{A}$ breaks the security against the arbitrator in the multi-user setting and chosen-key model. It will be shown how to construct an algorithm $\mathcal{R}$ to break the q-SDH Assumption. Recall that $\mathcal{R}$ is given $(g, g^x, g^{x^2}, \cdots, g^{x^q})$ as input, its goal is to output $(g^{1/(x+s)}, s)$. $\mathcal{R}$ selects uniformly at random $s_i \in \mathbb{Z}_p$ used for answering the partial signing queries. Let $S = \{s_1, \cdots, s_q\}$ be the set of all s_i, and let $S^i = S \setminus \{s_i\}$. $\mathcal{R}$ also uniformly chooses $i^* \in [q]$. Let $S^* = S \setminus \{s_{i^*}\}$. Define

$$p^*(\eta) = \prod_{t \in S^*} (\eta + t), \quad \text{and} \quad p(\eta) = \prod_{t \in S} (\eta + t).$$

Note that $\deg(p^*) = q - 1$ and $\deg(p) = q$. Define

$$g' = g^{p^*(x)}, h = g^{p(x)}, X = g^x.$$

$\mathcal{R}$ randomly chooses $u \in G$ and sets $\mathsf{PK} := (X, u)$. $\mathcal{R}$ simulates the challenger and answers the queries from $\mathcal{A}$.

H_0 Queries. At any time adversary $\mathcal{A}$ can query the random oracle H_0. Suppose $\mathcal{A}$ makes in total q_0 distinct queries to the random oracle H_0. To respond to these queries, $\mathcal{R}$ independently and uniformly chooses $j^* \in [q_0]$ and maintains a list of tuples $\langle m_i', a_i' \rangle$ as explained below. This list is referred to as H_0-list. The list is initially empty. When $\mathcal{A}$ queries the oracle H_0 for the j-th distinct message $m_j' \in \{0,1\}^*$, algorithm $\mathcal{R}$ responds as follows:

1. If the tuple $\langle m_j', a_j' \rangle$ is on the H_0-list, algorithm $\mathcal{R}$ responds with $H(m_j') = h^{a_j'}$ for $j \neq j^*$ and $H(m_{j'}) = g' h^{a_j'}$ for $j = j^*$.

2. Otherwise, if $j \neq j^*$, $\mathcal{R}$ generates a random $a'_j \in \mathbb{Z}_p$, adds the tuple $\langle m'_j, a'_j \rangle$ to the H_0-list and responds to $\mathcal{A}$ as $H(m_j) = h^{a'_j}$. If $i = j^*$, $\mathcal{R}$ generates a random $a'_{j^*} \in \mathbb{Z}_p$, adds the tuple $\langle m'_{j^*}, a'_{j^*} \rangle$ to the H_0-list and responds to $\mathcal{A}$ as $H(m_{j^*}) = g'h^{a'_{j^*}}$.

$\mathsf{H_1}$ Queries. $\mathcal{A}$ can also query the random oracle H_1. To respond to these queries, $\mathcal{R}$ maintains a list of tuples $\langle string^{(i)}, c^{(i)} \rangle$ as explained below. This list is referred to as H_1-list. The list is initially empty. When $\mathcal{A}$ queries the oracle H_1 at a point $string \in \{0,1\}^*$, algorithm $\mathcal{R}$ responds as follows:

1. If the query $string$ already appears on the H-list in some tuple $\langle string, c \rangle$, then algorithm $\mathcal{R}$ responds with $H(string) = c \in \mathbb{Z}_p$.

2. Otherwise, $\mathcal{R}$ generates a random $c \in \mathbb{Z}_p$, adds the tuple $\langle string, c \rangle$ to the H_1-list and responds to $\mathcal{A}$ as $H(string) = c \in \mathbb{Z}_p$.

PSig Queries. Without loss of generality, assume that for each partial signing query on a message m_i, the adversary has previously asked a H_0 query on m_i. Then for the i-th partial signing query, $\mathcal{R}$ finds the tuple $\langle m'_j, a'_j \rangle$ in the H_0-list such that $m_i = m'_j$. If $m_i = m'_{j^*}$, $\mathcal{R}$ aborts and returns failure. Otherwise $\mathcal{R}$ computes $v_i = g^{s_i}$, $\theta_i = u^{s_i}$ and

$$\delta_i = h^{a'_j/(x+s_i)} = g^{a'_j \prod_{t \in S^i}(x+t)}. \tag{3.1}$$

Since $g^x, g^{x^2}, \cdots, g^{x^q}$ are known, $\mathcal{R}$ can generate the partial signature $\sigma_P^{(i)} = (\delta_i, v_i, \theta_i)$ without explicitly knowing the secret key x, but instead using the right-hand side of (3.1) for computing δ_i.

Finally, the adversary $\mathcal{A}$ outputs a full signature σ^* on message m^* without asking the partial signing query on message m^*. From the full signature query, $\mathcal{R}$ can gain a valid partial signature $\sigma_P^* = (\delta^*, v^*, \theta^*) = (\delta^*, g^{s^*}, u^{s^*})$ on message m^*. If $s^* \neq s_{i^*}$ or $m^* \neq m'_{j^*}$, $\mathcal{R}$ returns failure. Otherwise, $\mathcal{R}$ can computes a tuple $(g^{1/(x+s^*)}, s^*)$.

Note that

$$\delta^* = (g'h^{a'_{j^*}})^{1/(x+s_{i^*})} = g^{p^*(x)/(x+s_{i^*})} g^{a'_{j^*} p^*(x)}.$$

From δ^* and the knowledge of $g^{a'_{j^*} p^*(x)}$, $\mathcal{R}$ can derive

$$\delta' = (\delta^* / g^{a'_{j^*} p^*(x)}) = g^{p^*(x)/(x+s_{i^*})}.$$

Let $p^*(\eta)/(\eta + s_{i^*}) = p'(\eta) + \gamma_{-1}/(\eta + s_{i^*})$ for some polynomial $p'(\eta)$ of $q-2$ and some $\gamma_{-1} \in \mathbb{Z}_p^*$. Thus $\mathcal{R}$ can further derive

$$\delta'' = (\delta'/g^{p'(x)})^{1/\gamma_{-1}} = (g^{\gamma_{-1}/(x+s_{i^*})})^{1/\gamma_{-1}} = g^{1/(x+s^*)}.$$

Note that i^* and j^* are uniformly chosen and independent of the adversary $\mathcal{A}$'s view, thus the case $s^* = s_{i^*}$ or $m^* = m'_{j^*}$ happens with a non-negligible probability at least $1/qq_0$, which means $\mathcal{R}$ can breaks the q-SDH assumption.

Type II adversary

Suppose the Type I adversary $\mathcal{A}$ breaks the security against the arbitrator in the multi-user setting and chosen-key model. It will be shown how to construct an algorithm $\mathcal{R}$ to break the q-HSDH Assumption.

Note that algorithm $\mathcal{R}$ is given $\mathbf{Q} =< g, g^x, u, \{(g^{1/(x+s_i)}, g^{s_i}, u^{s_i})\}_{i=1}^q >$. Its goal is to output another distinct triple $(g^{1/(x+s)}, g^s, u^s)$. $\mathcal{R}$ forwards $\mathsf{PK} := (g^x, u)$ to $\mathcal{A}$ and simulates the challenger and interacts with adversary $\mathcal{A}$ as follows.

$\mathsf{H_0}$ Queries. At any time adversary $\mathcal{A}$ can query the random oracle H_0. To respond to these queries, $\mathcal{R}$ maintains a list of tuples $\langle m_i, a_i \rangle$ as explained below. This list is referred to as H_0-list. The list is initially empty. When $\mathcal{A}$ queries the oracle H_0 on a message $m_i \in \{0,1\}^*$, algorithm $\mathcal{R}$ responds as follows:

1. If the query m_i already appears on the H_0-list in some tuple $\langle m_i, a_i \rangle$, then algorithm $\mathcal{R}$ responds with $H_0(m_i) = g^{a_i} \in \mathbb{Z}_p$.

2. Otherwise, $\mathcal{R}$ generates a random $a_i \in \mathbb{Z}_p$, adds the tuple $\langle m_i, a_i \rangle$ to the H_0-list and responds to $\mathcal{A}$ as $H_0(m_i) = g^{a_i} \in \mathbb{Z}_p$.

$\mathsf{H_1}$ Queries. $\mathcal{A}$ can also query the random oracle H_1. To respond to these queries, $\mathcal{R}$ maintains a list of tuples $\langle string^{(i)}, c^{(i)} \rangle$ as explained below. This list is referred to as H_1-list. The list is initially empty. When $\mathcal{A}$ queries the oracle H_1 at a point $string \in \{0,1\}^*$, algorithm $\mathcal{R}$ responds as follows:

1. If the query $string$ already appears on the H-list in some tuple $\langle string, c \rangle$, then algorithm $\mathcal{R}$ responds with $H(string) = c \in \mathbb{Z}_p$.

2. Otherwise, $\mathcal{R}$ generates a random $c \in \mathbb{Z}_p$, adds the tuple $\langle string, c \rangle$ to the H_1-list and responds to $\mathcal{A}$ as $H(string) = c \in \mathbb{Z}_p$.

PSig Queries. Without loss of generality, it is assumed that for each partial signing query on a message m_i, the adversary has previously asked a H_0 query on m_i. For the i-th partial signing query on message m_i, $\mathcal{R}$ seek out the tuple $\langle m'_j, a'_j \rangle$ in the H_0-list such that $m'_j = m_i$. $\mathcal{R}$ returns $((g^{1/(x+s_i)})^{a'_j}, g^{s_i}, u^{s_i})$ to $\mathcal{A}$ as the reply for the partial signing query.

It is easy to see that the Hash queries and partial signing queries are perfectly simulated. Finally, $\mathcal{A}$ halts. It either admits failure, in which case so does $\mathcal{R}$, or it returns a full signature σ^* on message m^* without asking the partial signing query on message m^*. From the full signature query, $\mathcal{R}$ can gain a valid partial signature $\sigma_P^* = (\delta^*, v^*, \theta^*) = (\delta^*, g^{s^*}, u^{s^*})$ on message m^*. Due to the randomness of the outputs of H_0 oracle, with overwhelming probability $\mathcal{A}$ have submitted the message m^* to the H_0 oracle. Let $H_0(m^*) = g^a$ where $\langle m^*, a \rangle$ is stored in the H_0-list. Since $s^* \notin \{s_1, \cdots, s_q\}$, $\mathcal{R}$ can simply outputs $((\delta^*)^{1/a}, v^*, \theta^*)$ and break the q-HSDH assumption.

The theorem follows from the two cases discussed above. □

3.3.2 An attack in the Enhanced Model

It will be shown that the above concrete construction is insecure in the enhanced model. More specifically, it will be shown that if a dishonest arbitrator can have access to the signer's signing oracle, he is able to generate a full signature on a new message without explicitly observing a corresponding partial signature.

Recall that a full signature on message m comprises a partial signature on message m, a designated confirmer signature on message $\bar{m}$ with the arbitrator being the designated confirmer where $\bar{m}$ is the complementary message of m, and a zero-knowledge proof of knowledge. If the arbitrator sees a full signature generated by the signer on message m, then the arbitrator has a designated confirmer signature on $\bar{m}$. Being the designated confirmer, the arbitrator is able to convert the designated confirmer signature into a conventional signature, which is exactly a partial signature of the optimistic fair exchange scheme on message $\bar{m}$. Thus the arbitrator will be able to generate a full signature on message $\bar{m}$ without asking the signer to generate a corresponding partial one.

Indeed, the arbitrator can ask the signer to generate a full signature on a message m. Denote the full signature on m is $(\sigma_P, \delta', \gamma', \theta', c_0, t_0, c_1, t_1)$, which is generated by the signer using algorithm Sig. Then the arbitrator can generate a full signature

on message $\bar{m}$ where $\bar{m}$ is the complementary message of m as follows.

1. Compute $\sigma_P = (\delta', (\gamma')^{1/y}, \theta')$.
2. Choose at random $s \in \mathbb{Z}_p$, $Z \in G$ and computes $(\delta, \gamma, \theta) := (Z, Y^s, u_i^s)$.
3. Compute $W = e(H_0(m), u_i)/e(\delta, \theta)$.
4. Choose uniformly at random $r, c'_0, t'_0 \in \mathbb{Z}_p$, and computes

$$c' = H_1(\bar{m}||\mathsf{PK}_i||g^{t'_0}(X_i)^{c'_0}||e(\delta, u_i)^{t'_0}W^{c'_0}||g^r),$$

   $c'_1 = c' - c'_0 \pmod{p}$ and $t'_1 = r - c'_1 y \pmod{p}$.
5. The full signature is set as $\sigma := (\sigma_P, \delta, \gamma, \theta, c'_0, t'_0, c'_1, t'_1)$.

It is not hard to check that σ is a valid signature on message $\bar{m}$ under the target signer's public key $\mathsf{PK}_i = (X_i, u_i)$.

3.4 Identical Distribution not Sufficient

In this section it will be shown that even the resolved signature generated by the arbitrator has the same distribution with the actual signatures generated by the signer, it still makes a difference whether an malicious arbitrator is allowed to have access to the signing oracle or not. This is showed by a concrete counterexample.

The intuitions of the concrete optimistic fair exchange scheme are as follows. The partial signature is the BLS signature [BLS04], and the full signature will be the partial signature plus the modified Bender-Katz-Morselli 2-user ring signature between the signer and the arbitrator [BKM06] (also can be viewed as Waters's signature [Wat05]). Note that in Waters's signature scheme, each user's public key includes g_1, g_2 and a set of Waters hash generators $u', u_1, \cdots, u_n$. In the OFE instantiation, all the users share the same Waters hash generators. Besides, it is required that all users use the arbitrator's public key as g_2 so that each user can use a single key to generate both the BLS signature and the modified Bender-Katz-Morselli 2-user ring signature.

Let G be a multiplicative cyclic group of prime order p, g be a generator of G, and $e : G \times G \to G_T$ be a bilinear pairing where G_T is a multiplicative group of order p. Let $H_0 : \{0,1\}^* \to G$ and $H_1 : \{0,1\}^* \to \{0,1\}^n$ be two collision-resistant

hash functions. It is assumed that the public parameter (G, p, e, H_1, H_2) is shared by all users. The concrete OFE scheme is as follows.

$\mathsf{Setup}^{\mathsf{TTP}}$: The arbitrator chooses uniformly at random Waters hash generators $u', u_1, \cdots, u_n \leftarrow G$ and exponents $y \leftarrow \mathbb{Z}_p$ and sets $Y = g^y$. APK is set as $(Y, u', u_1, \cdots, u_n)$. ASK is set as y.

$\mathsf{Setup}^{\mathsf{User}}$: User U_i chooses uniformly at random an exponent $x_i \leftarrow \mathbb{Z}_p$ and sets $\mathsf{SK}_i = x_i$ and $\mathsf{PK}_i = g^{x_i}$.

$\mathsf{PSig}(m, \mathsf{SK}_i, \mathsf{APK})$: To partial sign a message, the user U_i computes and sets the partial signature as $\sigma_P = H_0(m)^{x_i}$.

$\mathsf{PVer}(m, \sigma_P, \mathsf{PK}_i, \mathsf{APK})$: Given a partial signature σ_P from user U_i, a verifier verifies whether $e(\sigma_P, g) = e(H_0(m), \mathsf{PK}_i)$ holds. If so, it returns $\top$; otherwise it returns $\perp$.

$\mathsf{Sig}(m, \mathsf{SK}_i, \mathsf{APK})$: To partial sign a message, the user U_i computes $\sigma_P \leftarrow \mathsf{PSig}(m, \mathsf{SK}_i, \mathsf{APK})$ and $(m_1, \cdots, m_n) \leftarrow H_1(m||\mathsf{PK}_i)$. It then chooses $r \leftarrow \mathbb{Z}_p$ and compute

$$S_1 = Y^{x_i} \cdot (u' \prod_{j=1}^{n} u_j^{m_j})^r, \quad \text{and} \quad S_2 = g^r.$$

The full signature is set as $\sigma = (\sigma_P, S_1, S_2)$.

$\mathsf{Ver}(m, \sigma_P, \mathsf{PK}_i, APK)$: Given a full signature σ from user U_i, a verifier verifiers whether $\mathsf{PVer}(m, \sigma_P, \mathsf{PK}_i, \mathsf{APK}) = \top$ and whether

$$e(Y, \mathsf{PK}_i) = e(S_1, g) \cdot e(S_2^{-1}, u' \prod_{j=1}^{n} u_j^{m_j}).$$

If both hold, it returns $\top$; otherwise, it returns $\perp$.

$\mathsf{Res}(m, \sigma_P, \mathsf{ASK}, \mathsf{PK}_i)$: It first verifies whether σ_P is a valid partial signature by running $\mathsf{PVer}(m, \sigma_P, \mathsf{PK}_i, \mathsf{APK})$. If σ_P is invalid, it returns $\perp$. Otherwise, it computes $(m_1, \cdots, m_n) \leftarrow H_1(m||\mathsf{PK}_i)$, chooses $r \leftarrow \mathbb{Z}_p$, computes

$$S_1 = \mathsf{PK}_i^y \cdot (u' \prod_{j=1}^{n} u_j^{m_j})^r, \quad \text{and} \quad S_2 = g^r,$$

and returns $\sigma = (\sigma_P, S_1, S_2)$.

It is not hard to verify that, in the above construction, any (partial) signature created by Sig (PSig) will be valid under Ver (PVer), and that any signature created by the arbitrator using Res based on a partial signature generated by PSig will be valid under Ver. Thus the correctness property of the above construction holds.

It is easy to see the actual signatures generated by the signer and the resolved signatures generated by the arbitrator have the identical distribution, thus the resolution ambiguity also holds.

3.4.1 Security Analysis.

It will be shown that the specific construction is secure in the multi-user setting and chosen-key model reviewed in Section 2.3.2. Intuitively, the security against signers holds unconditionally as the arbitrator is always able to convert a signer's partial signature into a full one. The security against verifiers holds due to unforgeability of the BLS signature and the modified Bender-Katz-Morselli 2-user ring signature. The security against the arbitrator holds due to the unforgeability of the BLS signature.

Theorem 3.4 *The concrete optimistic fair exchange scheme above is unconditionally secure against signers in the multi-user setting and chosen-key model.*

Proof. Obviously, for any message m and any valid signature σ_P on m under the verification key PK_i, the arbitrator can always convert it into a full signature by using its own secret key ASK. Therefore, the security against signers always hold. □

Theorem 3.5 *The concrete optimistic fair exchange scheme above is secure against verifiers in the multi-user setting and chosen-key model if CDH assumption holds in G.*

Proof. Suppose an adversary $\mathcal{A}$ breaks the security against verifiers with non-negligible probability. it will be shown how to construct an algorithm $\mathcal{R}$ that breaks the CDH assumption in G.

Note that algorithm $\mathcal{R}$ is given an CDH instance $(G, p, g, A = g^a, B = g^b)$. Its goal is to output g^{ab}. Algorithm $\mathcal{R}$ simulates the challenger and interacts with adversary $\mathcal{A}$. Let q be the number of different messages contained in the queries $\mathcal{A}$ has made to the resolution oracle.

At the start, the algorithm $\mathcal{R}$ sets an integer, $l = 4q$, and chooses uniformly at random an integer, k^* between 0 and n. It then chooses an n-length vector, $\vec{x} = (x_i)$, where the elements of $\vec{x}$ are chosen uniformly at random between 0 and $l - 1$ and a value, x', chosen uniformly at random between 0 and $l - 1$. Besides, algorithm $\mathcal{R}$ chooses a random $y' \in \mathbb{Z}_p$ and an n-length vector, $\vec{y} = (y_i)$, where the

elements of $\vec{y}$ are chosen at random in $\mathbb{Z}_p$. Algorithm $\mathcal{R}$ keeps these values private. let $e : G \times G \rightarrow G_T$ be a bilinear pairing where G_T is a multiplicative group of order p and $H_1 : \{0,1\}^* \rightarrow \{0,1\}^n$ be a collision-resistant hash function. Algorithm $\mathcal{R}$ sets $\mathsf{PK} := A$ and $\mathsf{APK} = Y := B$ and assigns the Waters hash generators parameters $u' = B^{p-k^*m+x'}g^{y'}$ and $u_i = B^{x_i}g^{y_i}$. Algorithm $\mathcal{R}$ forwards PK, APK and the public parameters $(G, p, e, H_0, H_1, u', u_1, \cdots, u_n)$ to the adversary. From the view of the adversary, the distribution of the simulated public parameters is identical to the real construction. The hash function H_0 is viewed as a random oracle.

For ease of analysis the following functions are defined where $\tilde{m}$ is the set of indices i such that $m_i = 1$ while $H_1(m) = (m_1, \cdots, m_n)$:

$$F(m) = (p - mk^*) + x' + \sum_{i \in \tilde{m}} x_i, \quad \text{and} \quad J(m) = y' + \sum_{i \in \tilde{m}} y_i.$$

A binary function $K(m)$ is defined as

$$K(m) = \begin{cases} 0, & \text{if } \; x' + \sum_{i \in \tilde{m}} x_i \equiv 0 \pmod{l} \\ 1, & \text{otherwise.} \end{cases}$$

$\mathsf{H_0}$ Queries. At any time adversary $\mathcal{A}$ can query the random oracle H_0. To respond to these queries, $\mathcal{R}$ maintains a list of tuples $\langle M_i, a_i \rangle$ as explained below. This list is referred to as H_0-list. The list is initially empty. When $\mathcal{A}$ queries the oracle H_0 on a message $M_i \in \{0,1\}^*$, algorithm $\mathcal{R}$ responds as follows:

1. If the query M_i already appears on the H_0-list in some tuple $\langle M_i, a_i \rangle$, then algorithm $\mathcal{R}$ responds with $H(M_i) = g^{a_i}$.

2. Otherwise, $\mathcal{R}$ generates a random $a_i \in \mathbb{Z}_p$, adds the tuple $\langle M_i, a_i \rangle$ to the H_0-list and responds to $\mathcal{A}$ as $H(M_i) = g^{a_i}$.

PSig Queries. For the i-th partial signing query on message M_i, $\mathcal{R}$ checks whether there is a tuple $\langle M_i, a_i \rangle$ in the H_0-list. If not, $\mathcal{R}$ generates a random $a_i \in \mathbb{Z}_p$, adds the tuple $\langle M_i, a_i \rangle$ to the H_0-list. $\mathcal{R}$ returns A^{a_i} to $\mathcal{A}$ as the reply for the partial signing query.

Res Queries. Given a resolution query $(m, \sigma_P, \mathsf{PK}_i)$ where PK_i could be PK or could be adversarially-generated by the adversary, algorithm $\mathcal{R}$ responds to this query as follows:

1. checks whether $\mathsf{PVer}(\sigma_P, m, \mathsf{PK}_i) = \top$. If so, continues; otherwise, $\mathcal{R}$ responds to $\mathcal{A}$ with a special symbol $\perp$.

2. chooses a random $r \in \mathbb{Z}_p$, and computes the signature as

$$S_1 = g_1^{\frac{-J(m||\mathsf{PK}_i)}{F(m||\mathsf{PK}_i)}}(u' \prod_{i \in m||\tilde{\mathsf{PK}}_i} u_i)^r, \quad S_2 = g_1^{\frac{-1}{F(m||\mathsf{PK}_i)}} g^r.$$

3. The full signature is set as $\sigma := (\sigma_P, S_1, S_2)$ and returned to $\mathcal{A}$ as the reply of the resolution query.

Let $\tilde{r} = r - \frac{\alpha}{F(m||\mathsf{PK}_i)}$ and $\mathsf{PK}_i = g^\alpha$. Then

$$\begin{aligned}
S_1 &= \mathsf{PK}_i^{\frac{-J(m||\mathsf{PK}_i)}{F(m||\mathsf{PK}_i)}}(u' \textstyle\prod_{i \in m||\tilde{\mathsf{PK}}_i} u_i)^r \\
&= \mathsf{PK}_i^{\frac{-J(m||\mathsf{PK}_i)}{F(m||\mathsf{PK}_i)}}(B^{F(m||\mathsf{PK}_i)} g^{J(m||\mathsf{PK}_i)})^r \\
&= B^\alpha (B^{F(m||\mathsf{PK}_i)} g^{J(m||\mathsf{PK}_i)})^{-\frac{\alpha}{F(m||\mathsf{PK}_i)}} (B^{F(m||\mathsf{PK}_i)} g^{J(m||\mathsf{PK}_i)})^r \\
&= B^\alpha (u' \textstyle\prod_{i \in m||\tilde{\mathsf{PK}}_i} u_i)^{r - \frac{\alpha}{F(m||\mathsf{PK}_i)}} \\
&= B^\alpha (u' \textstyle\prod_{i \in m||\tilde{\mathsf{PK}}_i} u_i)^{\tilde{r}}
\end{aligned}$$

Additionally,

$$S_2 = g_1^{\frac{-1}{F(m||\mathsf{PK}_i)}} g^r = g^{r - \frac{\alpha}{F(m||\mathsf{PK}_i)}} = g^{\tilde{r}}.$$

Algorithm $\mathcal{R}$ will be able to perform this computation if and only if $F(m||\mathsf{PK}_i) \neq 0 \mod p$.

Finally the adversary $\mathcal{A}$ halts. It either admits failure, in which case so does $\mathcal{R}$, or it returns a full signature $\sigma^* = (\sigma_P^*, S_1^*, S_2^*)$ on message m^* without asking the resolution oracle with respect to the message m^* and the challenge signer's public key PK. Due to the collision-resistant property of the hash function, $H_1(m^*||\mathsf{PK}) \neq H(m||\mathsf{PK}_i)$ for any message m and PK_i that had been submitted to the resolution query before. Let $m^*\tilde{||}\mathsf{PK}$ be the set of indices i such that $m_i^* = 1$ where $H(m^*||\mathsf{PK}) = (m_1^*, \cdots, m_n^*)$. If $x' + \sum_{i \in m^*\tilde{||}\mathsf{PK}} x_i = k^* m$, then

$$e(\frac{S_1^*}{(S_2^*)^{y + \sum_{i \in m^*\tilde{||}\mathsf{PK}} y_i}}, g) = \frac{e(A,B)e(S_2^*, u' \prod_{j=1}^n u_j^{m_j^*})}{e((S_2^*)^{y + \sum_{i \in m^*\tilde{||}\mathsf{PK}} y_i}, g)} = e(A,B).$$

Therefore algorithm $\mathcal{R}$ can solve the CDH problem by computing g^{ab} as

$$g^{ab} = \frac{S_1^*}{(S_2^*)^{y + \sum_{i \in m^*\tilde{||}\mathsf{PK}} y_i}}.$$

Similar to [Wat05], the probability that the simulator does not abort during simulating the signing oracle and the equation $x' + \sum_{i \in m^* || \mathsf{PK}} x_i = k^* m$ holds is at least $\lambda = \frac{1}{8(n+1)q}$, which is non-negligible. This completes the proof. □

Theorem 3.6 *The concrete optimistic fair exchange scheme above is secure against the arbitrator in the multi-user setting and chosen-key model if the CDH assumption holds in G.*

Proof. Suppose an adversary breaks the security against verifiers with non-negligible probability. It will be shown how to construct an algorithm $\mathcal{R}$ that breaks the CDH assumption in G.

Note that algorithm $\mathcal{R}$ is given an CDH instance $(G, p, g, A = g^a, B = g^b)$. Its goal is to output g^{ab}. The algorithm $\mathcal{R}$ chooses uniformly at random $u', u_1, \cdots, u_n \in G$. let $e : G \times G \rightarrow G_T$ be a bilinear pairing where G_T is a multiplicative group of order p and $H_1 : \{0,1\}^* \rightarrow \{0,1\}^n$ be a collision-resistant hash function. Algorithm $\mathcal{R}$ sets $\mathsf{PK} := A$ and forwards PK and the public parameters $(G, p, e, H_0, H_1, u', u_1, \cdots, u_n)$ to the adversary $\mathcal{A}$. From the view of the adversary, the distribution of the simulated public parameters is identical to the real construction. The hash function H_0 is viewed as a random oracle. Suppose the adversary $\mathcal{A}$ makes in total q different queries to the random oracle. At the start, the algorithm $\mathcal{R}$ chooses uniformly at random an integer k^* such that $1 \leq k^* \leq q$.

$\mathsf{H_0}$ Queries. At any time adversary $\mathcal{A}$ can query the random oracle H_0. To respond to these queries, $\mathcal{R}$ maintains a list of tuples $\langle M_i, a_i \rangle$ as explained below. This list is referred to as H_0-list. The list is initially empty. When $\mathcal{A}$ makes the i different queries the oracle H_0 on a message $M_i \in \{0,1\}^*$, algorithm $\mathcal{R}$ responds as follows:

1. $\mathcal{R}$ checks whether $i = k^*$. If so, $\mathcal{R}$ returns B to $\mathcal{A}$ as the reply and adds the tuple $\langle M_i, B \rangle$ to the H_0-list.

2. Otherwise, $\mathcal{R}$ generates a random $a_i \in \mathbb{Z}_p$, adds the tuple $\langle M_i, a_i \rangle$ to the H_0-list and responds to $\mathcal{A}$ as $H(M_i) = g^{a_i}$.

PSig Queries. Without loss of generality, assume that for each partial signing query, the adversary $\mathcal{A}$ has previously submitted the corresponding message to the H_0 query. For a partial signing query on message M, $\mathcal{R}$ checks the H_0-list. Suppose $M = M_i$ where M_i is i-th different query $\mathcal{A}$ has made to the H_0 oracle. If $i = k^*$,

$\mathcal{R}$ aborts and returns failure. Otherwise, $\mathcal{R}$ returns A^{a_i} to $\mathcal{A}$ as the reply for the partial signing query.

Finally $\mathcal{A}$ outputs (m^*, σ^*) such that m^* is not queried to the partial signing oracle. Since k^* is chosen uniformly at random and independent of the adversary $\mathcal{A}$'s behavior, thus with non-negligible probability at least $1/q$ it should be $m^* = M_{k^*}$, in which case $\sigma^* = g^{ab}$. Thus algorithm $\mathcal{R}$ can outputs σ^* and breaks the CDH assumption with non-negligible probability. □

3.4.2 An attack in the Enhanced Model

The malicious arbitrator can always sign on behalf the signer as follows when it is provided the signing oracle. The reason that an adversary can succeed in the enhanced model is that the signing oracle can provide useful information to the adversary. Specifically, the adversary can forge a new signature as follows.

1. The arbitrator computes $Y = H_0(m^*)$ where m^* is the message the arbitrator tries to forge a signature on. Besides, the arbitrator chooses uniformly at random $x', x_1, \cdots, x_n \in \mathbb{Z}_p$ and sets $u' = g^{x'}, u_1 = g^{x_1}, \cdots, u_n = g^{x_n}$. The public key APK is set as $(Y, u', u_1, \cdots, u_n)$. Note that APK is malicious generated in the sense that the arbitrator does not know the corresponding secret key of Y.

2. Let the challenger's public key is $X = g^x$. The arbitrator asks a signing query on a message $m \neq m^*$, and gains a ring signature

$$S_1 = Y^x \cdot (u' \prod_{j=1}^{n} u_j^{m_j})^r, \quad \text{and} \quad S_2 = g^r,$$

where $(m_1, \cdots, m_n) \leftarrow H_1(m||X)$.

3. With the knowledge of $x', x_1, \cdots, x_n$, the arbitrator computes

$$A = (S_2)^{x'} \prod_{j=1}^{n} (S_2)^{x_j m'_j}$$

and therefore $Y^x = S_1/A$.

4. The arbitrator sets $\sigma_P := Y^x$, chooses at random $r' \in \mathbb{Z}_p$, and computes

$$S'_1 = Y^x \cdot (u' \prod_{j=1}^{n} u_j^{m'_j})^{r'}, \quad \text{and} \quad S'_2 = g^{r'},$$

where $(m'_1, \cdots, m'_n) \leftarrow H_1(m^*||X)$.

5. The full signature is set as $\sigma := (\sigma_P, S'_1, S'_2)$.

It is not hard to check that σ is a valid signature on message m^* under the target signer's public key X and the arbitrator has not made a partial signing query on the message m^*.

3.5 Chapter Summary

Defining proper security models capturing possible realistic powers of an adversary is of significant importance in the study of optimistic fair exchange. In this chapter it is identified that existing models failed to capture the reality that an adversary can have access to full signatures generated by the signer. An observation was made that the previous perception that a signing oracle can be simulated by the partial signing oracle and resolution oracle was wrong. That is to say, existing models are not complete as they deprived dishonest users of the chance of observing the challenge signer's actual signatures. To make existing models more practical and complete, an enhanced model for optimistic fair exchange that explicitly provides the adversary with the signing oracle was proposed. Separations between existing chosen-key model and the enhanced model are demonstrated.

Chapter 4

Collusion-Resistance in Optimistic Fair Exchange

In this chapter, the OFE security model about security against signers is strengthened by allowing the dishonest signer having access to the arbitrator's secret key. To some extent, this captures the scenario that the signer and the arbitrator may collude to disadvantage a verifier.

4.1 Introduction

An optimistic fair exchange protocol comprises three kinds of participants, namely a signer, a verifier, and a semi-trusted third party named an "arbitrator". Typically such a protocol is conducted in three message flows. Firstly, Alice the signer initiates the protocol by delivering a partial signature to Bob the verifier. A valid partial signature not only serves as an evidence to Bob that Alice has committed to endorse a certain message, but also assures Bob that he will receive Alice's full signature at the end of the protocol. The assurance follows the fact that the arbitrator is capable of converting the partial signature into a full one. In the second step, Bob delivers his full signature to Alice. Later, if Alice is honest, she will send her full signature to Bob in the third step, and this completes the exchange process. Note that under the normal situation, participation of the arbitrator is not required and thus, the term 'optimistic'. Meanwhile, the arbitrator is trusted in two senses. Alice trusts the arbitrator would not convert her partial signature into a full signature unless Bob submits his full signature. At the same time, Bob trusts the arbitrator that if he submits his full signature, the arbitrator would convert Alice's partial signature into a full one, should Alice fails to do so.

Note that the arbitrator in OFE should correctly make a resolution. If the arbitrator converts the signer's partial signature into a full one without a verifier

having submitted his own full signature, the signer will be disadvantaged. On the other hand, if the arbitrator refuses to convert the signer's partial signature into a full one even if a verifier has submitted his own full signature, the verifier will be disadvantaged. Without this basic assumption that the arbitrator correctly make a resolution, the fairness of OFE will not be guaranteed.

However, since the third party could be a random member of the network, it is realistic that it may potentially help one party like its business partner. Thus the collusion of the arbitrator with the signer or the verifier should be taken into account, as long as the arbitrator correctly makes the required resolution request whenever needed. In this chapter the scenario is considered that the arbitrator may implicitly collude with the signer in the sense of sharing its secret arbitrator key with the signer. If the fairness for the verifier can still be achieved when this collusion is considered, the trust for the arbitrator will be reduced for the verifier side. This is meaningful in practice, as it is easier for the signer to choose an arbitrator unilaterally than both mutually distrusted parties to choose a common trusted third party as an arbitrator. Furthermore, allowing the signer having access to the arbitrator's secret key also captures the real scenario of leakage of the arbitrator's secret key.

4.1.1 The Contributions

In this chapter, the following contributions are made.

1. The definition of security against signers in the enhanced chosen-key model proposed in Chapter 3 is updated, such that it now captures the previously overlooked case in optimistic fair exchange schemes, in which the potentially dishonest arbitrator may implicitly collude with a signer by offering its secret arbitration key. The enhanced chosen-key model now assures collusion-resistance in OFE. It is also showed that the security against signers in the enhanced model guarantees stronger security than that in existing multi-user setting and chosen-key model with a concrete example that shows the separation.

2. After updating the enhanced chosen-key model, the security of the existing schemes in the enhanced model is investigated. Two well known methodologies for constructing optimistic fair exchange schemes are revisited, namely one based on verifiably encrypted signatures, and the other one based on the

combination of conventional signatures and ring signatures. The result shows that these paradigms will remain secure in the enhanced model.

4.1.2 Organization

In the next section, the current definition for security against signers is modified to allow a dishonest signer having access to the arbitrator's secret key, and explanations about the updated model is made. In Section 4.3, the multi-user setting and chosen-key model and the enhanced chosen-key model are separated by offering a concrete OFE scheme that serves as a counterexample. In Section 4.4, two popular generic constructions of OFE protocols are revisited, namely, schemes based on verifiably encrypted signature and those based on conventional signature plus ring signature and it is shown that they yield schemes which remain secure in the enhanced model with slight modifications.

4.2 The Enhanced Chosen-key Model

It is easy to see that the existing models do not capture the possible collusion amongst the arbitrator and signer. That is, when a dishonest signer has access to the arbitrator's secret key, security of the schemes proven in this model will be unclear. To capture this case and make the security model more practical, the following model about security against signers is proposed.

SECURITY AGAINST SIGNERS. It is required that the probability that any PPT adversary $\mathcal{A}$ succeeds in the following experiment is negligible in k.

$$\begin{aligned} \mathsf{Setup}^{\mathsf{TTP}}(1^k) &\rightarrow (\mathsf{ASK}, \mathsf{APK}) \\ (m, \sigma_P, \mathsf{PK}^*) &\leftarrow \mathcal{A}(\mathsf{ASK}, \mathsf{APK}) \\ \sigma &\leftarrow \mathsf{Res}(m, \sigma_P, \mathsf{ASK}, \mathsf{PK}^*) \\ \text{success of } \mathcal{A} &:= \begin{bmatrix} \mathsf{PVer}(m, \sigma_P, \mathsf{PK}^*, \mathsf{APK}) = \top \\ \wedge \\ \mathsf{Ver}(m, \sigma, \mathsf{PK}^*, \mathsf{APK}) = \bot \end{bmatrix} \end{aligned}$$

In other words, no signer, even with the knowledge of the arbitrator's secret key, should be able to produce a partial signature that looks good to a verifier but cannot be resolved to a full signature. The case that the arbitrator and a dishonest signer may implicitly collude is captured by allowing the adversary $\mathcal{A}$ having the arbitrator's secret key as an input.

4.2.1 Implications and Limitations of The Enhanced Model

The model captures the case when the arbitrator's key is leaked to the attacker, or that someone having access to the arbitration key is colluding with the attacker. One limitation of the model is that it only captures the behavior of a malicious arbitrator whose nature is honest-but-curious. Specifically, it is assumed that the arbitration key is generated in complete accordance with the setup algorithm. This includes deleting all randomness used during key generation but not explicitly contained within the arbitration key. This is analogous to the situation for certificateless cryptosystems in which the attacker could be the key generation centre (KGC) itself. The nature of the KGC is modelled in two ways, namely, honest-but-curious and fully malicious. The former assumes the attacker has access to the KGC's master key which is generated honestly while the latter assumes that the KGC's master key is created by the attacker. Readers are referred to [Den08] for the discussion of the various models of certificateless cryptosystem.

There are two reasons for modeling an honest-but-curious arbitrator. Firstly, the observation is made that existing models already fall short in capturing attacks from such an honest-but-curious arbitrator. Secondly, existing schemes following a certain design approach are immune against this kind of attack. The formalization of a model that captures the behavior of a malicious arbitrator and the construction of schemes secure in this sense is left as an open problem.

In an orthogonal direction, it should be remarked that a similar extension could not be applied to the security against verifiers. In other words, the arbitrator must be completely honest to the signer. Otherwise, the verifier could just convert the partial signature from the signer with the help of the arbitrator without the need to fulfill the obligation.

4.3 Separation of the Proposed Model and the Existing Model

In this section, a separation of the security against signers in the enhanced model and that in the existing model [HYWS08b], which was reviewed in Section 2.3.2, is demonstrated. In order to do this, a concrete optimistic fair exchange scheme,

called A*-OFE[1], that is secure in the multi-user setting and chosen key model reviewed in Section 2.3.2 is presented. Then, a concrete attack against this scheme in the enhanced model is shown. This shows that the security against signers in the enhanced model captures strictly stronger security than that in the existing model, as it is trivial to show that the security in the enhanced model implies that in the multi-user setting and chosen key model.

4.3.1 High Level Description

Prior to proposing the concrete construction that will serve as a counterexample, the high level description will be firstly provided. In this example, the full signature is a Schnorr signature [Sch91] (c, s) on a message m under the signer's public key X such that $c = H(X||g||g^s X^c||m)$, where g is a generator of the group G in the Schnorr signature setting and H is a hash function. Suppose the order of g is of ℓ-bit and in the following γ is used to denote the value $\ell - 1$. The partial signature comprises $c, S = g^s$, an encryption of the exponent s and a proof that the encryption is done correctly. That is, to generate a partial signature, the signer does the following.

1. The signer releases c and $S = g^s$.
2. Let b_j for $j = 0$ to γ be the binary representation of s. That is, $s = \sum_{j=0}^{\gamma} 2^j b_j$, where $b_j \in \{0, 1\}$.
3. Since no efficient encryption of exponents in the Schnorr signature setting is known, the signer encrypts the individual bits b_j of s.
   - The signer encodes bit 0 as the identity element in group G while bit 1 is encoded as the generator g. Thus to encrypt the j-th binary bit b_j, the signer equivalently encrypts the group element g^{b_j}.
   - For $j = 0$ to γ, the signer independently and randomly chooses r_j and computes
     $$A_j = g^{r_j}, B_j = h^{r_j}, C_j = g^{b_j} Y^{r_j}.$$
     Note that (A_j, B_j, C_j) is in fact the encryption of the bit b_j under the arbitrator's public key (h, Y).

[1]The name A*-OFE represents that the arbitrator may implicitly collude with the signer in the enhanced model.

- The set of tuples (A_j, B_j, C_j) for $j = 0$ to γ constitute the encryption of the exponent s.

4. The signer then makes a proof of knowledge that the ciphertexts have been generated correctly. Naturally the proof of knowledge consists of two parts.

   - The first part ensures that the discrete logarithm of the value encrypted in (A_j, B_j, C_j), when added together after applying the proper weights, is the exponent s. More specifically, the first part itself is a zero knowledge proof of knowledge of a set of values $\{s_j, r_j\}_{j=0}^{\gamma}$ such that $A_j = g^{r_j}, B_j = h^{r_j}, C_j = g^{s_j} Y^{r_j}$ and $S = g^{\sum_{j=0}^{\gamma} 2^j s_j}$.
   - The second part guarantees that the discrete logarithm of the value encrypted in (A_j, B_j, C_j) can only be 0 or 1. It can be viewed as a zero knowledge proof of knowledge of values R_j such that either $C_j = Y^{R_j}$ or $C_j = gY^{R_j}$ holds.

To convert a partial signature into a full one, the arbitrator decrypts the ciphertexts and gains a sequence of plaintexts. If all the plaintexts are either the identity element in group G or the generator g, the arbitrator decodes the identity element and g as bit 0 and 1, respectively, and outputs the value whose binary representation is exactly the sequence of these bits. Otherwise the arbitrator returns $\perp$ to indicate failure in making a resolution.

Since the signer has no access to the arbitrator's secret key, the proof of knowledge of a set of values $\{s_j, r_j\}_{j=0}^{\gamma}$ such that $C_j = g^{s_j} Y^{r_j}, S = g^{\sum_{j=0}^{\gamma} 2^j s_j}$ and the proof of knowledge of values R_j such that $C_j = Y^{R_j}$ or $C_j = gY^{R_j}$ together would imply that $s_j \in \{0, 1\}$ and $r_j = R_j$. Thus the above proof of the set of relationships achieve what the signer would like to convince the verifier. That is, the sum of s_j after applying the appropriate weights would be the component s of the Schnorr signature and that s_j can only be 0 or 1. Thus the arbitrator can decrypt the ciphertexts and output the exponent s such that $S = g^s$.

4.3.2 Construction of A*-OFE

Formally it is assumed that a public group G with a generator g of prime order p of ℓ bits has been generated. Denote by γ the value $\ell - 1$. Let $H : \{0, 1\}^* \to \mathbb{Z}_p$ be a cryptographic hash function.

The construction of A*-OFE is as follows.

- $\mathsf{Setup}^{\mathsf{TTP}}$: The arbitrator chooses random elements $h \in G$ and $y \in \mathbb{Z}_p$, and computes $Y = g^y$. The public key is set as $\mathsf{APK} = (h, Y)$, and the arbitrator keeps $\mathsf{ASK} = y$ as private.

- $\mathsf{Setup}^{\mathsf{User}}$: Each user U_i chooses a secret value $x_i \in \mathbb{Z}_p$, and calculates $X_i = g^{x_i}$. The user sets $(\mathsf{SK}_i, \mathsf{PK}_i) = (x_i, X_i)$.

- PSig: To partially sign a message m, a user U_i does as follows:

  1. chooses a random $t \in \mathbb{Z}_p$ and then computes $c \in \mathbb{Z}_p$ and $s \in \mathbb{Z}_p$ such that
  $$c = H(\mathsf{PK}_i||g||g^t||m) \text{ and } s = t - cx_i \text{ (in } \mathbb{Z}_p).$$

  2. let $s = \sum_{j=0}^{\gamma} 2^j b_j$, where for each $0 \leq j \leq \gamma$, $b_j \in \{0, 1\}$. Note that this binary representation always exists and is unique in $\mathbb{Z}_p$[2]. Let further $S = g^s$ and $E_j = g^{2^j}$ for $0 \leq j \leq \gamma$. Thus
  $$S = \prod_{j=0}^{\gamma} (E_j)^{b_j}.$$

  3. U_i chooses uniformly at random $0 \leq r_j \leq \mathbb{Z}_p$ for each $0 \leq j \leq \gamma$, and computes
  $$A_j = g^{r_j},\ B_j = h^{r_j},\ C_j = g^{b_j} Y^{r_j}.$$

  4. For $0 \leq j \leq \gamma$, if $b_j = 0$, U_i chooses uniformly at random elements $t_{j0}, c_{j1}, s_{j1} \in \mathbb{Z}_p$ and sets
  $$T_{j0} = Y^{t_{j0}},\ T_{j1} = (\frac{C_j}{g})^{c_{j1}} Y^{s_{j1}}.$$
  Otherwise, U_i chooses uniformly at random $c_{j0}, s_{j0}, t_{j1} \in \mathbb{Z}_p$, and sets
  $$T_{j0} = (C_j)^{c_{j0}} Y^{s_{j0}},\ T_{j1} = (\frac{C_j}{g})^{t_{j1}}.$$
  Besides, for $0 \leq j \leq \gamma$, U_i chooses uniformly at random $t_j, \tilde{t_j} \in \mathbb{Z}_p$, and sets
  $$\tilde{S} = \prod_{j=0}^{\gamma} (E_j)^{\tilde{t_j}},\ \tilde{A}_j = g^{t_j},\ \tilde{B}_j = h^{t_j},\ \tilde{C}_j = g^{\tilde{t_j}} Y^{t_j}.$$

[2]While the value s is unique in $\mathbb{Z}_p$, the set $\{b_j\}_{j=0}^{l-1}$ such thta $g^s = g^{\sum_{j=0}^{\gamma} 2^j b_j}$ may not be unique. For example, suppose $s = 1 \in \mathbb{Z}_p$ and that $p < 2^l - 1$, two sets of $\{b_j\}_{j=0}^{\gamma}$ exist. They corresponds to the binary representation of 1 and $p+1$. However, later it will be shown that this will not affect the security of the scheme.

5. For $0 \leq j \leq \gamma$, denote $\tilde{T_j} = \tilde{A_j}||\tilde{B_j}||\tilde{C_j}$ and $\tilde{T_j}' = T_{j0}||T_{j1}$. Let $A = \tilde{T_0}||\cdots||\tilde{T_\gamma}$, and $B = \tilde{T_0}'||\cdots||\tilde{T_\gamma}'$. The user U_i computes

$$\tilde{c} = H\big(\ \mathsf{PK}_i||c||S||m||\ \tilde{S}\ ||\ A\ ||\ B\ \big).$$

For $0 \leq j \leq \gamma$, if $b_j = 0$, U_i computes

$$c_{j0} = \tilde{c} - c_{j1}(\text{in } \mathbb{Z}_p),\ s_{j0} = t_{j0} - c_{j0}r_j\ (\text{in } \mathbb{Z}_p).$$

Otherwise, U_i computes

$$c_{j1} = \tilde{c} - c_{j0}\ (\text{in } \mathbb{Z}_p),\ s_{j1} = t_{j1} - c_{j1}r_j\ (\text{in } \mathbb{Z}_p).$$

Furthermore, for $0 \leq j \leq \gamma$, U_i computes

$$\tilde{s_j} = \tilde{t_j} - \tilde{c}b_j\ (\text{in } \mathbb{Z}_p),\ s_j = t_j - \tilde{c}r_j\ (\text{in } \mathbb{Z}_p).$$

6. For $0 \leq j \leq \gamma$, denote $T_j = A_j||B_j||C_j$. The partial signature is set as $\sigma_P := (c,\ S,\ T_0,\ \cdots,\ T_\gamma,\ c_{00},\ c_{01},\ s_{00},\ s_{01},\ \cdots,\ c_{\gamma 0},\ c_{\gamma 1},\ s_{\gamma 0},\ s_{\gamma 1},\ \tilde{c},\ \tilde{s_0},\ s_0,\ \cdots,\ \tilde{s_\gamma},\ s_\gamma)$.

- PVer: Given a partial signature σ_P from user U_i, a verifier does as follows.

1. The verifier checks whether $c, \tilde{c} \in \mathbb{Z}_p$, and for $0 \leq j \leq \gamma$,

$$c_{j0}, c_{j1}, s_{j0}, s_{j1}, \tilde{s_j}, s_j \in \mathbb{Z}_p,\ c_{j0} + c_{j1} = \tilde{c}\ (\text{in } \mathbb{Z}_p).$$

2. The verifier checks whether

$$c = H(\mathsf{PK}_i||g||S(X_i)^c||m).$$

3. The verifier computes $\tilde{S} = S^{\tilde{c}} \cdot \prod_{j=0}^{\gamma}(E_j)^{\tilde{s_j}}$ and for $0 \leq j \leq \gamma$,

$$T_{j0} = (C_j)^{c_{j0}}Y^{s_{j0}},\ T_{j1} = (\frac{C_j}{g})^{c_{j1}}Y^{s_{j1}}.$$

4. Besides, for $0 \leq j \leq \gamma$, the verifier decomposes T_j as $A_j||B_j||C_j$ and computes

$$\tilde{A_j} = g^{s_j}(A_j)^{\tilde{c}},\ \tilde{B_j} = h^{s_j}(B_j)^{\tilde{c}},\ \tilde{C_j} = g^{\tilde{s_j}}Y^{s_j}(C_j)^{\tilde{c}}.$$

5. Denote $\tilde{T_j} = \tilde{A_j}||\tilde{B_j}||\tilde{C_j}$ and $\tilde{T_j}' = T_{j0}||T_{j1}$. Let $A = \tilde{T_0}||\cdots||\tilde{T_\gamma}$, and $B = \tilde{T_0}'||\cdots||\tilde{T_\gamma}'$. The verifier checks whether

$$\tilde{c} = H\big(\ \mathsf{PK}_i||c||S||m||\ \tilde{S}\ ||\ A\ ||\ B\ \big).$$

6. If all the above equations hold, then the verifier returns $\top$, otherwise $\perp$ is returned.

- Sig: To fully sign a message m, a user U_i chooses a random $t \in \mathbb{Z}_p$ and then computes $c \in \mathbb{Z}_p$ and $s \in \mathbb{Z}_p$ such that

$$c = H(\mathsf{PK}_i||g||g^t||m) \quad \text{and} \quad s = t - cx_i \text{ (in } \mathbb{Z}_p).$$

  The full signature is set as $\sigma := (c, s)$.

- Ver: Given a full signature $\sigma := (c, s)$ from user U_i, a verifier verifies whether $c \in \mathbb{Z}_p, s \in \mathbb{Z}_p$ and

$$c = H(\mathsf{PK}_i||g||g^s(X_i)^c||m).$$

  If so, $\top$ is returned and otherwise, $\perp$ is returned.

- Res: For the user U_i's partial signature σ_P, the arbitrator

  1. first checks whether $\mathsf{PVer}(m, \sigma_P, \mathsf{PK}_i) = \top$. If so, continues; otherwise, returns $\perp$.
  2. for $0 \leq j \leq \gamma$, compute $D_j = C_j/(A_j)^y$. If $D_j = g$, sets $b_j = 1$, and if $D_j = 1$, sets $b_j = 0$. Otherwise, it responds to $\mathcal{A}$ with $\perp$.
  3. If $\perp$ is not returned, returns $s = \sum_{j=0}^{\gamma} 2^j b_j \bmod p$[3].

4.3.3 Security Analysis.

It is not hard to verify that, in the above construction, any (partial) signature created by Sig (PSig) will be valid under Ver (PVer), and that any signature created by the arbitrator using Res based on a partial signature generated by PSig will be valid under Ver. Thus the correctness property of the above construction holds.

Since the signature generated by Sig and that generated by the arbitrator using Res based on a valid partial signature are both Schnorr signatures, the resolution ambiguity property also holds.

Next the specific construction is secure in the multi-user setting and chosen-key model reviewed in Section 2.3.2 will be shown.

[3]Note that even if the set of binary values $\{b_j\}$ is not unique, the value s computed by the arbitrator modulo p will be unique as long as $S = g^{\sum_{j=0}^{\gamma} 2^j b_j}$ and that $b_j \in \{0, 1\}$.

Theorem 4.1 *The A*-OFE scheme is secure against signers in the multi-user setting and chosen-key model under the discrete logarithm assumption.*

Proof. Suppose an adversary $\mathcal{A}$ breaks the security against signers. It will be shown how to construct an algorithm $\mathcal{R}$ that solves the discrete logarithm problem. This will contradict with the discrete logarithm assumption.

Note that algorithm $\mathcal{R}$ is given random elements $u \in G$. Its goal is to output an integer $\alpha \in \mathbb{Z}_p$ such that $u = g^\alpha$. Algorithm $\mathcal{R}$ simulates the challenger and interacts with adversary $\mathcal{A}$ as follows.

Setup. Algorithm $\mathcal{R}$ chooses a random integer $y \in \mathbb{Z}_p$ and sets $h = u$, $Y = h^y$. Note that the distributions of h, Y are uniform on G. $\mathcal{R}$ forwards the values h, Y to adversary $\mathcal{A}$, who returns its public key $\mathsf{PK}^* = X_A$.

Hash Queries. At any time adversary $\mathcal{A}$ can query the random oracle H. To response to these queries, $\mathcal{R}$ maintains a list of tuples $\langle string^{(i)}, c_i \rangle$ as explained below. This list is referred to as H-list. The list is initially empty. When $\mathcal{A}$ queries the oracle H at a point $string \in \{0,1\}^*$, algorithm $\mathcal{R}$ responds as follows:

1. If the query $string$ already appears on the H-list in some tuple $\langle string, c \rangle$, then algorithm $\mathcal{R}$ responds with $H(string) = c \in \mathbb{Z}_p$.

2. Otherwise, $\mathcal{R}$ generates a random $c \in \mathbb{Z}_p$, adds the tuple $\langle string, c \rangle$ to the H-list and responds to $\mathcal{A}$ as $H(string) = c \in \mathbb{Z}_p$.

Note that c is uniform in $\mathbb{Z}_p$ and is independent of $\mathcal{A}$'s current view as required.

Res Queries. Given a resolution query $(m, \sigma_P, \mathsf{PK}_i)$ where $\mathsf{PK}_i = X_i$ is the signer's public key and $\sigma_P := (c, S, T_0, \cdots, T_\gamma, \cdots)$, algorithm $\mathcal{R}$ responds to this query as follows:

1. checks whether $\mathsf{PVer}(\sigma_P, m, \mathsf{PK}_i) = \top$. If so, continues; otherwise, $\mathcal{R}$ responds to $\mathcal{A}$ with a special symbol $\perp$.

2. for $0 \leq j \leq \gamma$, decompose T_j as $A_j || B_j || C_j$, and compute $D_j = C_j/(B_j)^y$. If $D_j = g$, sets $b_j = 1$, and if $D_j = 1$, sets $b_j = 0$. Otherwise, $\mathcal{R}$ responds to $\mathcal{A}$ with $\perp$.

3. If $\perp$ is not returned, $\mathcal{R}$ forwards $s = \sum_{j=0}^{\gamma} 2^j b_j$ to $\mathcal{A}$ as the response.

Output. It is easy to see that the above hash queries and resolution queries are perfectly simulated. Finally, $\mathcal{A}$ halts. It either admits failure, in which case so does $\mathcal{R}$, or it returns a partial signature σ_P^* on message m^*, where $\sigma_P^* := (c^*, S^*, T_0^*, \cdots, T_\gamma^*, c_{00}^*, c_{01}^*, s_{00}^*, s_{01}^*, \cdots, c_{\gamma 0}^*, c_{\gamma 1}^*, s_{\gamma 0}^*, s_{\gamma 1}^*, \tilde{c}^*, \tilde{s_0}^*, s_0^*, \cdots, \tilde{s_\gamma}^*, s_\gamma^*)$ and $T_j^* = A_j^* || B_j^* || C_j^*$ for $0 \leq j \leq \gamma$, such that $\mathsf{PVer}(\sigma_P^*, m^*, \mathsf{PK}^*) = \top$, but it can not be resolved to a valid full signature by the resolution algorithm Res.

By the General Forking Lemma [BN06] (a standard rewinding technique in random oracle model), with non-negligible probability algorithm $\mathcal{R}$ is able to gain another partial signature σ_P' on the same message m^*, where $\sigma_P' := (c^*, S^*, T_0^*, \cdots, T_\gamma^*, c_{00}', c_{01}', s_{00}', s_{01}', \cdots, c_{\gamma 0}', c_{\gamma 1}', s_{\gamma 0}', s_{\gamma 1}', \tilde{c}', \tilde{s_0}', s_0', \cdots, \tilde{s_\gamma}', s_\gamma')$ and $\tilde{c}^* \neq \tilde{c}$, such that $\mathsf{PVer}(\sigma_P', m^*, \mathsf{PK}^*) = \top$, but it can not be resolved to a valid full signature by the resolution algorithm Res.

Without loss of generality, it is assumed that $\tilde{c}' > \tilde{c}^*$. For $0 \leq j \leq \gamma$, it has $(C_j^*)^{\tilde{c}^*} g^{\tilde{s_j}^*} Y^{s_j^*} = (C_j^*)^{\tilde{c}'} g^{\tilde{s_j}'} Y^{s_j'}$. It follows that $(C_j^*)^{\tilde{c}'-\tilde{c}^*} = g^{\tilde{s_j}^*-\tilde{s_j}'} Y^{s_j^*-s_j'}$. Let $\tau_j = (\tilde{s_j}^* - \tilde{s_j}')(\tilde{c}' - \tilde{c}^*)^{-1}$, $\upsilon_i = (s_j^* - s_j')(\tilde{c}' - \tilde{c}^*)^{-1}$. Hence $C_j^* = g^{\tau_j} Y^{\upsilon_j}$.

For $0 \leq j \leq \gamma$, because $(A_j^*)^{\tilde{c}^*} g^{s_j^*} = (A_j^*)^{\tilde{c}'} g^{s_j'}$, $(B_j^*)^{\tilde{c}^*} h^{s_j^*} = (B_j^*)^{\tilde{c}'} h^{s_j'}$, it follows $A_j^* = g^{\upsilon_i}, B_j^* = h^{\upsilon_i}$.

Moreover, for $0 \leq j \leq \gamma$, because $(C_j^*)^{c_{j0}^*} Y^{s_{j0}^*} = (C_j^*)^{c_{j0}'} Y^{s_{j0}'}$, $(\frac{C_i^*}{g})^{c_{j1}^*} Y^{s_{j1}^*} = (\frac{C_j^*}{g})^{c_{j1}'} Y^{s_{j1}'}$, $c_{j0}^* + c_{j1}^* = \tilde{c}^*$ and $c_{j0}' + c_{j1}' = \tilde{c}'$, it has the following three cases:

case1 : If $c_{j0}^* \neq c_{j0}'$ and $c_{j1}^* \neq c_{j1}'$, then it has $(C_j^*)^{c_{j0}'-c_{j0}^*} = Y^{s_{j0}^*-s_{j0}'}$ and $(\frac{C_j^*}{g})^{c_{j1}'-c_{j1}^*} = Y^{s_{j1}^*-s_{j1}'}$. Thus it has $C_j^* = Y^{\frac{s_{j0}^*-s_{j0}'}{c_{j0}'-c_{j0}^*}} = gY^{\frac{s_{j1}^*-s_{j1}'}{c_{j1}'-c_{j1}^*}}$. Denote $\beta = y(\frac{s_{j1}^*-s_{j1}'}{c_{j1}'-c_{j1}^*} - \frac{s_{j0}^*-s_{j0}'}{c_{j0}'-c_{j0}^*})$. It follows $1 = gh^\beta$. Algorithm $\mathcal{R}$ can simply output $\alpha = -\beta^{-1}$ and breaks the discrete logarithm assumption when this case happens. Thus, this case can be safely omitted.

case2 : If $c_{j0}^* = c_{j0}'$, and $c_{j1}^* \neq c_{j1}'$. It follows that $(\frac{C_j^*}{g})^{c_{j1}'-c_{j1}^*} = Y^{s_{j1}^*-s_{j1}'}$. Denote $\upsilon_j' = (s_{j1}^* - s_{j1}')(c_{j1}' - c_{j1}^*)^{-1}$. Then it has $C_j^* = gY^{\upsilon_j'}$. Since $C_j^* = g^{\tau_j} Y^{\upsilon_j}$, it follows $1 \equiv g^{\tau_j - 1} Y^{\upsilon_j - \upsilon_j'}$. Thus it must be the case $\tau_j = 1$ and $\upsilon_i = \upsilon_i{}'$, as otherwise algorithm $\mathcal{R}$ can simply output $\alpha = (1-\tau_j)(y\upsilon_j - y\upsilon_j')^{-1}$ and breaks the discrete logarithm problem. Therefore $C_j^* = gY^{\upsilon_j}$ when this case happens.

case3 : If $c_{j0}^* \neq c_{j0}'$ and $c_{j1}^* = c_{j1}'$, then likewise under the discrete logarithm assumption one can gain that $\tau_j = 0$ and $C_j^* = Y^{\upsilon_j}$ when this case happens.

Furthermore, since $(S^*)^{\tilde{c}^*} \prod_{i=0}^{\gamma} (E_i)^{\tilde{s_j}^*} = (S^*)^{\tilde{c}'} \prod_{j=0}^{\gamma} (E_i)^{\tilde{s_j}'}$, it has $(S^*)^{\tilde{c}'-\tilde{c}^*} = \prod_{j=0}^{\gamma} (E_j)^{\tilde{s_j}^*-\tilde{s_j}'}$. It follows that $(S^*) = \prod_{j=0}^{\gamma} (E_j)^{\tau_j}$.

From the above analysis, it is readily to see that the arbitrator always returns a

value $s^* = \sum_{j=0}^{\gamma} 2^j b_j \bmod p$ where $b_j = 0$ if $\tau_j = 0$, and $b_j = 1$ if $\tau_i = 1$. It is easy to verify $S^* = \prod_{j=0}^{\gamma}(E_j)^{\tau_j} = \prod_{j=0}^{\gamma} g^{2^j b_j} = g^{s^*}$, which means that under the discrete logarithm assumption, the scheme is secure against signers. □

Theorem 4.2 *The A*-OFE scheme is secure against verifiers in the multi-user setting and chosen-key model under the decisional Diffie-Hellman assumption.*

Proof. Suppose an adversary $\mathcal{A}$ makes Q_h hash queries, Q_p partial signing queries and Q_r resolution queries, and then wins by producing a new full signature forgery $\sigma^* := (c^*, s^*)$ on message m^*. For $1 \leq i \leq Q_p$, let $m^{(i)}$ be the ith message submitted to the partial signing oracle and $\sigma_P^{(i)} := (c^{(i)}, S^{(i)} = g^{s^{(i)}}, \cdots)$ be the reply for this query. To show the security, a sequence of games are firstly defined, and then the security is shown via a series of claims that if $\mathcal{A}$ is successful against Game x, then it will also be successful in Game $x + 1$.

Game 0. This is the real experiment between the challenger and an adversary $\mathcal{A}$ as defined in security against verifiers in Section 2.3.2, in the random oracle model. This means that the challenger firstly correctly generates the TTP's and target signer's key pairs by running algorithms $\mathsf{Setup}^{\mathsf{TTP}}$ and $\mathsf{Setup}^{\mathsf{User}}$, respectively. The challenger forwards the TTP's public key $\mathsf{APK} := (h, Y)$ and target signer's public key $\mathsf{PK}^* := X$ to the adversary $\mathcal{A}$, and then provides accesses to the hash oracle, partial signing oracle and resolution oracle as well.

Game 1. This is the same as Game 0, with the exception that one makes it a rule that $\mathcal{A}$ fails if there does not exist an index $1 \leq i \leq Q_p$ such that $m^{(i)} = m^*, c^{(i)} = c^*, S^{(i)} = g^{s^*}$.

Game 2. This is the same as Game 1, with the exception that the challenger is simulated as follows. Choose at random $y_1, y_2 \in \mathbb{Z}_p$ and set $Y = g^{y_1} h^{y_2}$.

Hash Queries. The simulation of hash queries is exactly the same as in the above theorem.

PSig Queries. At any time $\mathcal{A}$ can request a partial signature under challenge key $\mathsf{PK}^* = X$. Responds to the ith partial signing query on message $m^{(i)}$ as follows:

1. $\mathcal{R}$ chooses $c^{(i)}, s^{(i)} \in \mathbb{Z}_p$ at random, sets $c^{(i)} := H(\mathsf{PK}^* || g || g^{s^{(i)}} X^{c^{(i)}} || m^{(i)})$, and adds $\langle \mathsf{PK}^* || \ g || \ S^{(i)} X^{c^{(i)}} || \ m^{(i)}, c^{(i)} \rangle$ where $S^{(i)} = g^{s^{(i)}}$ to the H-list.

2. Let $s^{(i)} = \sum_{j=0}^{\gamma} 2^j b_j^{(i)}$, where for each $0 \leq j \leq \gamma$, $b_j^{(i)} \in \{0,1\}$. Choose uniformly at random $0 \leq r_j^{(i)} \leq \mathbb{Z}_p$ for each $0 \leq j \leq \gamma$, and compute

$$A_j^{(i)} = g^{r_j^{(i)}},\ B_j^{(i)} = h^{r_j^{(i)}},\ C_j^{(i)} = g^{b_j^{(i)}} (A_j^{(i)})^{y_1} (B_j^{(i)})^{y_2}.$$

3. Randomly choose $\tilde{c}^{(i)} \in \mathbb{Z}_p$. For $0 \leq j \leq \delta$, choose uniformly at random $c_{j0}^{(i)}, s_{j0}^{(i)}, s_{j1}^{(i)} \in \mathbb{Z}_p$, and compute $c_{j1}^{(i)} = \tilde{c}^{(i)} - c_{j0}^{(i)}$ and

$$T_{j0}^{(i)} = (C_j)^{c_{j0}^{(i)}} Y^{s_{j0}^{(i)}},\ T_{j1}^{(i)} = (\frac{C_j}{g})^{c_{j1}^{(i)}} Y^{s_{j1}^{(i)}}.$$

   Besides, for $0 \leq j \leq \delta$, choose uniformly at random $0 \leq \tilde{s_j}^{(i)}, s_j^{(i)} \leq \mathbb{Z}_p$. Compute $\tilde{S}^{(i)} = S^{\tilde{c}^{(i)}} \cdot \prod_{j=0}^{\gamma} (E_j)^{\tilde{s_j}^{(i)}}$, and for $0 \leq j \leq \delta$, $\tilde{A_j}^{(i)} = g^{s_j^{(i)}} (A_j^{(i)})^{\tilde{c}^{(i)}}$, $\tilde{B_j}^{(i)} = g^{s_j^{(i)}} (B_j^{(i)})^{\tilde{c}^{(i)}}$, $\tilde{C_j}^{(i)} = g^{\tilde{s_j}^{(i)}} Y^{s_j^{(i)}} (C_j^{(i)})^{\tilde{c}^{(i)}}$.

4. For each $0 \leq j \leq \gamma$, denote $\tilde{T_j}^{(i)} = \tilde{A_j}^{(i)} || \tilde{B_j}^{(i)} || \tilde{C_j}^{(i)}$, $\tilde{T_j}'^{(i)} = T_{j0}^{(i)} || T_{j1}^{(i)}$ and $T_j^{(i)} = A_j^{(i)} || B_j^{(i)} || C_j^{(i)}$. Let $A^{(i)} = \tilde{T_0}^{(i)} || \cdots || \tilde{T_\gamma}^{(i)}$, and $B^{(i)} = \tilde{T_0}'^{(i)} || \cdots || \tilde{T_\gamma}'^{(i)}$. Set

$$\tilde{c}^{(i)} := H\big(\mathsf{PK}^* || c^{(i)} || S^{(i)} || m^{(i)} || \tilde{S}^{(i)} || A^{(i)} || B^{(i)}\big)$$

   and add

$$\big\langle \mathsf{PK}^* || c^{(i)} || S^{(i)} || m^{(i)} || \tilde{S}^{(i)} || A^{(i)} || B^{(i)},\ \tilde{c}^{(i)} \big\rangle$$

   to the H-list.

5. The partial signature is returned to $\mathcal{A}$ as $\sigma_P^{(i)} := (c^{(i)}, S^{(i)}, T_0^{(i)}, \cdots, T_\gamma^{(i)}, c_{00}^{(i)}, c_{01}^{(i)}, s_{j0}^{(i)}, s_{j1}^{(i)}, \cdots, c_{\gamma 0}^{(i)}, c_{\gamma 1}^{(i)}, s_{\gamma 0}^{(i)}, s_{\gamma 1}^{(i)}, \tilde{c}^{(i)}, \tilde{s_0}^{(i)}, s_0^{(i)}, \cdots, \tilde{s_\gamma}^{(i)}, s_\gamma^{(i)})$.

Res Queries. $\mathcal{A}$ can request a resolution query $(m, \sigma_P, \mathsf{PK}_i)$ where $\mathsf{PK}_i = X_i$ is a signer's public key and $\mathcal{A}$ may not know its corresponding secret key, and $\sigma_P := (c, S, T_0, \cdots, T_\gamma, c_{00}, c_{01},\ s_{00}, s_{01}, \cdots, c_{\gamma 0}, c_{\gamma 1}, s_{\gamma 0}, s_{\gamma 1}, \tilde{c}, \tilde{s_0}, s_0,\ \cdots, \tilde{s_\gamma}, s_\gamma)$. Responds to this query as follows:

- checks whether $\mathsf{PVer}(m, \sigma_P, \mathsf{PK}_i) = \top$. If so, continues; otherwise, returns $\perp$.

  1. for $0 \leq j \leq \delta$, decompose T_j as $A_j || B_j || C_j$, and compute $D_j = C_j / (A_j)^{y_1} (B_j)^{y_2}$. If $D_j = g$, sets $b_j = 1$, and if $D_j = 1$, sets $b_j = 0$. Otherwise, $\mathcal{R}$ responds to $\mathcal{A}$ with $\perp$.
  2. If $\perp$ is not returned, $\mathcal{R}$ forwards $s = \sum_{j=0}^{\gamma} 2^i b_i \bmod p$ to $\mathcal{A}$ as the response.

- When $\mathsf{PVer}(\sigma_P, m, \mathsf{PK}_i) = \perp$, responds to $\mathcal{A}$ with $\perp$.

Game 3. This is the same as Game 2, with the exception that at the beginning of the game one guesses an index $1 \leq i^* \leq Q_p$, and $\mathcal{A}$ fails if $m^{(i^*)} \neq m^*$, or $c^{(i^*)} \neq c^*$, or $S^{(i^*)} \neq g^{s^*}$.

Game 4. This is the same as Game 3, with the exception that one makes some changes to the i^*th partial signing oracle in the above game. Namely, for each $0 \leq j \leq \gamma$, instead of choosing uniformly at random $r_j^{(i^*)} \in \mathbb{Z}_p$, and computing

$$A_j^{(i^*)} = g^{r_j^{(i^*)}},\ B_j^{(i^*)} = h^{r_j^{(i^*)}},\ C_j^{(i^*)} = g^{b_j^{(i^*)}}(A_j^{(i^*)})^{y_1}(B_j^{(i^*)})^{y_2},$$

Choose independently and uniformly at random $\alpha_j, \beta_j, \delta_j \in \mathbb{Z}_p$, and compute

$$A_j^{(i^*)} = g^{\alpha_j},\ B_j^{(i^*)} = g^{\beta_j},\ C_j^{i^*} = g^{b_j^{(i^*)}} g^{\delta_j}.$$

Game 5. This is the same as Game 4, with the exception that for $0 \leq j \leq \gamma$, one randomly and independently chooses $\mu_j \in \mathbb{Z}_p$ and replaces $C_j^{(i^*)} = g^{b_j^{(i^*)}} g^{\delta_j}$ with $C_j^{(i^*)} = g^{\mu_j}$.

Next the probability that $\mathcal{A}$ wins in these games is linked via a series of claims. Let S_x be the event that $\mathcal{A}$ wins in Game x.

Claim 1

$$\Pr[S_1] = \Pr[S_0] - \epsilon_{\mathrm{SAA}}, \tag{4.1}$$

where ϵ_{SAA} is the advantage of an adversary $\mathcal{A}'$ in the security against the arbitrator discussed later.

Suppose $\mathcal{A}$ generates a new full signature forgery $\sigma^* := (c^*, s^*)$ on message m^*, but there does not exist an index $1 \leq i \leq Q_p$ such that $m^{(i)} = m^*, c^{(i)} = c^*, S^{(i)} = g^{s^*}$. Then an adversary $\mathcal{B}$ in the security against the arbitrator is built. $\mathcal{B}$ correctly generates its key pair $(\mathsf{ASK}, \mathsf{APK})$ by running algorithm $\mathsf{Setup}^{\mathsf{TTP}}$ and then receives its own target signer's public key X. $\mathcal{B}$ forwards X and APK to $\mathcal{A}$. Whenever $\mathcal{A}$ requests a hash query or a partial signing query, $\mathcal{B}$ submits this query to its own hash oracle or partial signing oracle, respectively, and forwards the reply to $\mathcal{A}$. Whenever $\mathcal{A}$ requests a resolution query, $\mathcal{B}$ makes a resolution using its secret key ASK. Note that the simulation is perfect. Finally $\mathcal{B}$ outputs $\mathcal{A}$'s forgery $\sigma^* := (c^*, s^*)$ on message m^* and wins in the security against the arbitrator that will be discussed in the next theorem.

Claim 2

$$|\Pr[S_2] - \Pr[S_1]| \leq negl(k). \tag{4.2}$$

Define F to be the event that in a resolution query in Game 2, $\mathsf{PVer}(\sigma_P, m, \mathsf{PK}_i) = \top$ but there exists some $1 \leq j \leq \gamma$ such that $A_j = g^r, B_j = g^{r'}$ but $r \neq r' \mod q$. If F does not occur, with $A_j = g^{r_j}, B_j = h^{r_j}$, then $Y^{r_j} = (A_j)^{y_1} B_j{}^{y_2}$; therefore the decryption oracle computes $D_j = C_j / Y^{r_j}$, just as it should be. That is to say, the Game 1 and Game 2 proceed statistically indistinguishably, unless event F occurs. Thus it has $|\Pr[S_0] - \Pr[S_1]| \leq \Pr[F]$, and only need to show that $\Pr[F] \leq negl(k)$. To prove this inequality, it suffices to notice that in a valid partial signature σ_P, a non-interactive proof of knowledge $PK\{(r_j) : A_j = g^{r_j} \wedge B_j = h^{r_j}\}$ was essentially conducted, and the knowledge error is negligible in k even for an adversary with infinite power.

Claim 3

$$\Pr[S_3] = \frac{\Pr[S_2]}{Q_p}. \tag{4.3}$$

The only difference between these two games is that one guesses a random index i^*. This value is used no where in the game, and independent to the adversary. Once the adversary makes a forgery, one only declares him successful if the index of his forgery matches one's guess, which will occur with $1/Q_p$ probability.

Claim 4

$$|\Pr[\mathsf{S}_4] - \Pr[\mathsf{S}_3]| \leq l \cdot \epsilon_{\text{ddh}}, \tag{4.4}$$

where ϵ_{ddh} is the DDH-advantage of some efficient algorithm $\mathcal{R}$ (and hence negligible under the DDH assumption).

To show this, divide the transition from Game 3 to Game 4 into a sequences of sub-games from Game $0'$ to Game $(\gamma+1)'$. Game $0'$ is exactly the same as Game 3. Game $(j+1)'$ is the same as Game j', with the exception that in the i^*th partial signing query instead of choosing uniformly at random $r_j^{(i^*)} \in \mathbb{Z}_p$, and computing

$$A_j^{(i^*)} = g^{r_j^{(i^*)}},\ B_j^{(i^*)} = h^{r_j^{(i^*)}},\ C_j^{(i^*)} = g^{b_j^{(i^*)}} (A_j^{(i^*)})^{y_1} (B_j^{(i^*)})^{y_2},$$

one chooses uniformly at random $\alpha_j, \beta_j, \delta_j \in \mathbb{Z}_p$, and computes

$$A_j^{(i^*)} = g^{\alpha_j},\ B_j^{(i^*)} = g^{\beta_j},\ C_j^{i^*} = g^{b_j^{(i^*)}} g^{\delta_j}.$$

Note that Game $(\gamma+1)'$ is exactly the same as Game 4. Let F_j be the event that $\mathcal{A}$ wins in Game j'.

Next it will be shown that any difference between $\Pr[\mathsf{F}_j]$ and $\Pr[\mathsf{F}_{j+1}]$ can be parlayed into a corresponding DDH-advantage of an algorithm $\mathcal{R}$. Algorithm $\mathcal{R}$ runs as follows. It takes as input (g_2, u_1, u_2), and interacts with $\mathcal{A}$. $\mathcal{R}$ sets $h = g_2$. It simulates the oracles as in Game j' except that in the i^*th partial signing query it computes $A_j^{(i^*)} = u_1, B_j^{(i^*)} = u_2, C_j^{(i^*)} = g^{b_j^{(i^*)}}(u_1)^{y_1}(u_2)^{y_2}$. If $\mathcal{A}$ outputs a full signature $\sigma^* := (c^*, s^*)$ on message m^* such that $m^* = m^{(i^*)}, c^* = c^{(j^*)}, g^{s^*} = S^{(j^*)}$, $\mathcal{R}$ output 1, else output a random bit $b \in \{0, 1\}$.

When the input to $\mathcal{R}$ is of the form (g_2, g^r, g_2^r) where r is randomly chosen, then computation proceeds just as in Game j'. When the input to $\mathcal{R}$ is of the form $(g_2, g^r, g_2^{r'})$, where $r, r' \in \mathbb{Z}_p$ are independently uniformly chosen, one only needs to argue that $\varepsilon = (u_1)^{y_1}(u_2)^{y_2}$ is random to adversary A's view. If so, the simulation of $\mathcal{R}$ proceeds just the same as in Game $(j+1)'$.

To see this, consider the point $\mathbf{Q} = (y_1, y_2) \in (\mathbb{Z}_p)^2$. Let $g_2 = g^w$. At the beginning of the attack, this is a random point on the line

$$\log_g Y = y_1 + wy_2, \tag{4.5}$$

determined by the TTP's public key. Note that the i^*th partial signing query should not be submitted to the resolution oracle, otherwise $\sigma^* := (c^*, s^*)$ will not be a new forgery on message $m^* = m^{(i^*)}$. Thus, due to the same reason as in Claim 2, with overwhelming probability the resolution oracle only decrypts with respect to (A_i, B_i, C_i) of the form $\left(g^{r_i}, h^{r_i}, g^{b_i}(A_i)^{y_1}(B_i)^{y_2}\right)$, and the adversary $\mathcal{A}$ obtains only linear dependent relations $r_i \log_g Y = r_i y_1 + r_i w y_2$ (since $(A_i)^{y_1}(B_i)^{y_2} = g^{r_i y_1} h^{r_i y_1} = Y^{r_i}$). That means, no further information about $\mathbf{Q}$ is leaked.

Consider now the output $(A_j^{(i^*)}, B_j^{(i^*)}, C_j^{(i^*)})$ of $\mathcal{R}$'s partial signing reply for $\mathcal{A}$'s i^*th partial signing query. It has

$$\log_g \varepsilon = ry_1 + r'y_2. \tag{4.6}$$

Clearly, (4.5) and (4.6) are linearly independently when $r \neq r'$. ε is a perfect one time pad, and thus random to the adversary $\mathcal{A}$'s view.

Therefore, for each $0 \leq j \leq \gamma$, it has $|\Pr[\mathsf{F}_j] - \Pr[\mathsf{F}_{j+1}]| \leq \epsilon_{\text{ddh}}$, which is followed by $|\Pr[\mathsf{S}_4] - \Pr[\mathsf{S}_3]| = |\Pr[\mathsf{F}_{\gamma+1}] - \Pr[\mathsf{F}_0]| \leq (\gamma+1) \cdot \epsilon_{\text{ddh}}$. Note that $l = \gamma + 1$, the claim holds.

Claim 5

$$\Pr[\mathsf{S}_5] = \Pr[\mathsf{S}_4]. \tag{4.7}$$

Note that in game 5, g^{μ_j} is a randomly chosen element from G, the change from Game 5 to Game 4 was purely conceptual. Thus it has $\Pr[\mathsf{S}_5] = \Pr[\mathsf{S}_4]$.

Claim 6

$$\Pr[\mathsf{S}_5] \leq \epsilon_{\mathrm{dl}}, \tag{4.8}$$

where ϵ_{dl} is the discrete logarithm advantage, (and hence negligible under the discrete logarithm assumption).

To show this, an algorithm $\mathcal{R}$ that employs the adversary $\mathcal{A}$ in Game 5 is built to solve the discrete logarithm problem. Algorithm $\mathcal{R}$ runs as follows. It takes as input S. Its goal is to output $s \in \mathbb{Z}_p$ such that $g^s = S$. It interacts with $\mathcal{A}$, playing the role of the simulator in Game 5. $\mathcal{R}$ simulates the environments in the same way as in Game 5 with the exception that it sets $S^{(i^*)} = S$. The simulation is perfect. Finally when $\mathcal{A}$ wins in Game 5 by outputting (c^*, s^*) on message m^*, $\mathcal{R}$ outputs $s = s^*$ and solves the discrete logarithm problem.

Claim 7

$$\mathsf{Adv}_{\mathcal{A}}(1^k) \leq \epsilon_{\mathrm{SAA}} + Q_p(l \cdot \epsilon_{\mathrm{ddh}} + \epsilon_{\mathrm{dl}}) + negl(k), \tag{4.9}$$

where $\mathsf{Adv}_{\mathcal{A}}(1^k)$ is advantage of $\mathcal{A}$ in the real experiment defined in the multi-user setting and chosen key model.

As a sequence of equations (4.1), (4.2), (4.3), (4.4), (4.7) and (4.8) gained above, it has equation (4.9). The whole proof is done. □

Theorem 4.3 *The A*-OFE scheme is secure against the arbitrator in the multi-user setting and chosen-key model under the discrete logarithm assumption.*

Proof. Suppose an adversary $\mathcal{A}$ breaks the security against the arbitrator. It will be shown how to construct an algorithm $\mathcal{R}$ that solves the discrete logarithm problem. This will contradict with the discrete logarithm assumption. Algorithm $\mathcal{R}$ is given a random element $X \in G$ as input, and its goal is to output $x \in \mathbb{Z}_p$ such that $g^x = X$.

Setup : $\mathcal{R}$ sets $\mathsf{PK}^* := X$ and sends it to the arbitrator.

Hash Queries. Hash queries are simulated in the same way as described in the above theorems.

PSig Queries. Partial signing queries are the same as in Game 2 in the above theorem.

Output. It is obvious that the above hash queries and resolution queries are perfectly simulated. Finally, $\mathcal{A}$ halts. It either admits failure, in which case so does

$\mathcal{R}$, or it returns a full signature σ^* on message m^*, where $\sigma^* := (c^*, s^*)$ such that $\mathsf{Ver}(\sigma^*, m^*, \mathsf{PK}^*) = \top$.

Suppose adversary $\mathcal{A}$ makes Q_h hash queries and Q_p partial signing queries, and then produces a new full signature forgery. For $1 \leq i \leq Q_p$, let $m^{(i)}$ be the ith message submitted to the partial signing oracle and $\sigma_P^{(i)} := (c^{(i)}, S^{(i)} = g^{s^{(i)}}, \cdots)$ be the reply for this query. Let $\sigma^* := (c^*, s^*)$ be the forgery on message m^*. If there does not exist an index $1 \leq i \leq Q_p$ such that $m^{(i)} = m^*, c^{(i)} = c^*, S^{(i)} = g^{s^*}$, then due to the randomness of the outputs of hash oracle, with overwhelming probability $\mathcal{A}$ has submitted the string $X||g||g^{s^*}X^c||m^*$ to the hash query. Therefore by the General Forking Lemma [BN06] (a rewinding technology in the random oracle model), with non-negligible probability adversary $\mathcal{A}$ is able to produces another full signature forgery σ' on the same message m^*, where $\sigma_P := (c', s')$ such that $c^* \neq c'$. $\mathcal{R}$ thus have $g^{s^*}X^{c^*} = g^{s'}X^{c'}$. This enable $\mathcal{R}$ to output $x = (s^* - s')(c' - c^*)^{-1}$ and break the discrete logarithm assumption. □

4.3.4 A Concrete Attack against A*-OFE in the Enhanced Model

It will be shown that the above A*-OFE is insecure in the enhanced model. When the arbitrator reveals its secret key $y \in \mathbb{Z}_p$ to a signer whose secret/public key pair is $(x, X = g^x)$, the malicious signer $\mathcal{A}$ can create a partial signature that passes the inspection of partial signature verification algorithm, but it can not be resolved to a valid full signature.

Taking further investigation into the above concrete construction, one may observe that there is a loophole about the proof made in the conterexample: the knowledge r_j used in the first part of the proof such that $C_j = g^{s_j}Y^{r_j}$ could be *different* with the knowledge R_j used in the second part of proof such that $C_j = Y^{R_j}$ or $C_j = gY^{R_j}$. If that is true, the value s_j would not be 0 or 1. That is, the plaintext decrypted by the arbitrator in this case would be g^{s_j} rather than the identity element or the generator g of group G as it is supposed to be, and thus the arbitrator will fail to make a resolution.

Note that finding r_j, R_j such that $C_j = g^{s_j}Y^{r_j}$ and that $C_j = Y^{R_j}$ (or $C_j = gY^{R_j}$) while $r_j \neq R_j$ is hard if the discrete logarithm of Y to the base g is unknown. However, in the enhanced model, this discrete logarithm value is known, and therefore

finding such values r_j, R_j is feasible. This is the rationale of how the counterexample can be attacked when an adversary is allowed to have access to the arbitrator's secret key.

The concrete attack by $\mathcal{A}$ is as follows.

1. Choose a random $t \in \mathbb{Z}_p$ and compute $c \in \mathbb{Z}_p$ and $s \in \mathbb{Z}_p$ such that

$$c = H(X||g||g^t||m) \text{ and } s = t - cx \text{ (in } \mathbb{Z}_p).$$

   Note that s and y are independent and both random, thus with probability $1/2$ it has $s \geq y$.

2. Let $s = \sum_{j=0}^{\gamma} 2^j b_j$ and $s - y = \sum_{j=0}^{\gamma} 2^j d_j$, where for each $0 \leq j \leq \gamma$, $b_j, d_j \in \{0, 1\}$. Note that the binary representation always exist and can be uniquely determined. Denote $S = g^s$, $\gamma = \ell - 1$ and $E_j = g^{2^j}$ for $0 \leq j \leq \gamma$. Let further $b'_j = d_j$ for $1 \leq j \leq \gamma$, and $b'_0 = d_0 + y$. Thus

$$S = \prod_{j=0}^{\gamma} (E_j)^{b'_j}.$$

3. $\mathcal{A}$ chooses uniformly at random $0 \leq r_j \leq \mathbb{Z}_p$ for each $0 \leq j \leq \gamma$, and computes

$$A_j = g^{r_j},\ B_j = h^{r_j},\ C_j = g^{b'_j} Y^{r_j}.$$

   Note that a trick here that will be used by the malicious signers is that

$$C_0 = g^{b'_0} Y^{r_0} = g^{d_0} Y^{r_0+1}.$$

4. If $d_0 = 0$, $\mathcal{A}$ chooses uniformly at random elements $t_{00}, c_{01}, s_{01} \in \mathbb{Z}_p$ and sets

$$T_{00} = Y^{t_{00}},\ T_{01} = (\frac{C_0}{g})^{c_{01}} Y^{s_{01}}.$$

   Otherwise, $\mathcal{A}$ chooses uniformly at random $c_{00}, s_{00}, t_{01} \in \mathbb{Z}_p$, and sets

$$T_{00} = (C_j)^{c_{00}} Y^{s_{00}},\ T_{01} = (\frac{C_0}{g})^{t_{01}}.$$

5. For $1 \leq j \leq \gamma$, if $b'_j = 0$, $\mathcal{A}$ chooses uniformly at random elements $t_{j0}, c_{j1}, s_{j1} \in \mathbb{Z}_p$ and sets

$$T_{j0} = Y^{t_{j0}},\ T_{j1} = (\frac{C_j}{g})^{c_{j1}} Y^{s_{j1}}.$$

Otherwise, $\mathcal{A}$ chooses uniformly at random $c_{j0}, s_{j0}, t_{j1} \in \mathbb{Z}_p$, and sets

$$T_{j0} = (C_j)^{c_{j0}} Y^{s_{j0}},\ T_{j1} = (\frac{C_j}{g})^{t_{j1}}.$$

Besides, for $0 \leq j \leq \gamma$, $\mathcal{A}$ chooses uniformly at random $t_j, \tilde{t_j} \in \mathbb{Z}_p$, and sets

$$\tilde{S} = \prod_{j=0}^{\gamma} (E_j)^{\tilde{t_j}},\ \tilde{A_j} = g^{t_j},\ \tilde{B_j} = h^{t_j},\ \tilde{C_j} = g^{\tilde{t_j}} Y^{t_j}.$$

6. For $0 \leq j \leq \gamma$, denote $\tilde{T_j} = \tilde{A_j}||\tilde{B_j}||\tilde{C_j}$ and $\tilde{T_j}' = T_{j0}||T_{j1}$. Let $A = \tilde{T_0}||\cdots||\tilde{T_\gamma}$, and $B = \tilde{T_0}'||\cdots||\tilde{T_\gamma}'$. $\mathcal{A}$ computes

$$\tilde{c} = H(\ X||c||S||m||\ \tilde{S}\ ||\ A\ ||\ B\ ).$$

If $d_j = 0$, the $\mathcal{A}$ computes

$$c_{00} = \tilde{c} - c_{01} (\text{in } \mathbb{Z}_p),\ s_{00} = t_{00} - c_{00}(r_0 + 1)\ (\text{in } \mathbb{Z}_p).$$

Otherwise, $\mathcal{A}$ computes

$$c_{01} = \tilde{c} - c_{00}\ (\text{in } \mathbb{Z}_p),\ s_{01} = t_{01} - c_{01}(r_0 + 1)\ (\text{in } \mathbb{Z}_p).$$

For $1 \leq j \leq \gamma$, if $b'_j = 0$, $\mathcal{A}$ computes

$$c_{j0} = \tilde{c} - c_{j1} (\text{in } \mathbb{Z}_p),\ s_{j0} = t_{j0} - c_{j0} r_j\ (\text{in } \mathbb{Z}_p).$$

Otherwise, $\mathcal{A}$ computes

$$c_{j1} = \tilde{c} - c_{j0}\ (\text{in } \mathbb{Z}_p),\ s_{j1} = t_{j1} - c_{j1} r_j\ (\text{in } \mathbb{Z}_p).$$

Furthermore, for $0 \leq j \leq \gamma$, $\mathcal{A}$ computes

$$\tilde{s_j} = \tilde{t_j} - \tilde{c} b'_j\ (\text{in } \mathbb{Z}_p),\ s_j = t_j - \tilde{c} r_j\ (\text{in } \mathbb{Z}_p).$$

7. For $0 \leq j \leq \gamma$, denote $T_j = A_j||B_j||C_j$. The partial signature is set as $\sigma_P := (c, S, T_0, \cdots, T_\gamma, c_{00},\ c_{01}, s_{00}, s_{01}, \cdots, c_{\gamma 0}, c_{\gamma 1}, s_{\gamma 0}, s_{\gamma 1}, \tilde{c}, \tilde{s_0}, s_0, \cdots, \tilde{s_\gamma}, s_\gamma)$.

It is easy to check that $\mathsf{PVer}(\sigma_P, m, \mathsf{PK}_i) = \top$. Now, it needs to show the resolution about this partial signature will return the symbol $\perp$, indicating failure. Indeed, the arbitrator will compute $D_j = C_j/(A_j)^y$ for $0 \leq j \leq \gamma$ in a resolution process. Since $D_0 = C_0/(A_0)^y = g^{b'_0} = g^{d_i} Y$, rather than g or 1 as it should be. Thus the arbitrator will return $\perp$ by following the resolution algorithm. In this case, without any doubt, verifiers would be disadvantaged with probability 1/2. The malicious signer can also repeat step 1 several times, for example n times, until a value s is found such that $s \geq y$. In this case, the signer can cheat with probability as least $1 - 1/2^n$.

4.4 Previous Paradigms Revisited

From the knowledge in Chapter 3 and the above discussion in this chapter, it is clear that the enhanced chosen-key model has strengthened all the three aspects of security in the multi-user setting and chosen-key model. The security against signers is strengthened by allowing the dishonest signer having access to the arbitrator's secret key. The security against verifiers and the security against the arbitrator are strengthened by allowing the an adversary having access to the target signer's signing oracle (this part has been discussed in Chapter 3). In this section, the basic theoretical question is addressed, namely whether or not a scheme satisfying the security notions exists. Two previous known paradigms for constructing optimistic fair exchange schemes are revisited and their security in the enhanced model is evaluated.

4.4.1 Verifiably Encrypted Signature Paradigm

Dodis *et al.* [DLY07] showed optimistic fair exchange schemes secure in the multi-user setting can be constructed from verifiably encrypted signatures. Later, Huang *et al.* [HYWS08b] showed this generic construction also works in the chosen-key model. The generic construction of optimistic fair exchange schemes based on verifiably encrypted signatures [DLY07] is as follows.

Let $\mathcal{E} = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ be an encryption scheme, and $\mathcal{S} = (\mathsf{KGen}, \mathsf{Sig}, \mathsf{Ver})$ be a signature scheme. Given an encryption/decryption key pair (ek, dk) and a signature key pair (sk, vk), let Π be a simulation-sound [Sah99] non-interactive zero-knowledge (NIZK) proof system for the NP-language $L = \{(c, m, ek, vk) | \exists \sigma \ [c = \mathcal{E}.\mathsf{Enc}(\sigma, ek) \wedge \mathcal{S}.\mathsf{Ver}(m, \sigma, vk) = 1]\}$. Intuitively, the partial signature of the generic optimistic fair exchange scheme is the encryption of the signer's signature plus a non-interactive zero-knowledge proof showing that the ciphertext is correctly generated. The full signature is the signer's signature. Below is the generic construction of optimistic fair exchange schemes from verifiably encrypted signatures.

- $\mathsf{Setup}^{\mathsf{TTP}}$: The arbitrator runs $(dk, ek) \leftarrow \mathcal{S}.\mathsf{KGen}(1^k)$ and sets $\mathsf{ASK} = dk$ and $\mathsf{APK} = ek$.
- $\mathsf{Setup}^{\mathsf{User}}$: Each user U_i runs $(sk_i, vk_i) \leftarrow \mathcal{S}.\mathsf{KGen}(1^k)$ and sets $(\mathsf{SK}_i, \mathsf{PK}_i) = (sk_i, vk_i)$.

- Sig: To sign a message m, the user U_i runs $s \leftarrow \mathcal{S}.\mathsf{Sig}(sk_i, m)$. The full signature is set as $\sigma = s$.

- Ver: To verify whether a full signature σ is generated by user U_i, a verifier checks whether the output of $\mathcal{S}.\mathsf{Ver}(m, \sigma, vk_i)$ is valid or not. If valid, return $\top$. Otherwise, return $\bot$.

- PSig: To partially sign a message m, the user U_i runs $s \leftarrow \mathcal{S}.\mathsf{Sig}(sk_i, m)$ and $c \leftarrow \mathcal{E}.\mathsf{Enc}(ek, s)$ and generates a proof π showing $(c, m, ek, vk_i) \in L$. The partial signature is set as $\sigma_P = (c, \pi)$.

- PVer: To verify whether a partial signature $\sigma_P = (c, \pi)$ is generated by user U_i on message m, the verifier checks whether the proof π for the statement $(c, m, ek, vk_i) \in L$ is valid or not. If valid, return $\top$. Otherwise, return $\bot$.

- Res: For a partial signature $\sigma_P = (c, \pi)$ generated by the user U_i on message m, the arbitrator checks whether $\mathsf{PVer}(m, \sigma_P, \mathsf{ASK}, \mathsf{PK}_i) = \top$. If so, the arbitrator runs $s \leftarrow \mathcal{E}.\mathsf{Dec}(dk, c)$ and returns $\sigma = s$. Otherwise, the arbitrator returns $\bot$.

The correctness property of this construction follows from the correctness of the signature scheme $\mathcal{S}$, the correctness of the encryption scheme $\mathcal{E}$, and the completeness of the NIZK proof system Π, which states that if the statement is true, the honest verifier will be convinced of this fact by an honest prover.

The resolution ambiguity property holds due to the fact that a partial signature is a verifiable encryption of a full signature and the resolution process is just a decryption of the partial signature.

Typically the security against signers in the multi-user setting and chosen-key model is due to the soundness of the NIZK proof system Π, which was reviewed in Section 2.2.4. The soundness property of ordinary proof systems states that the prover should not be able to convince the verifier of a false statement with non-negligible probability. Normally the adversary in the soundness model is only given the public parameters as input. Here, to make the optimistic fair exchange schemes constructed from verifiably encrypted signatures also secure in the enhanced model, it is emphasized that the soundness property of Π must hold even when the adversary is also explicitly given the secret decryption key dk as input, like the model of verifiable encryption discussed in [CS03].

For security against verifiers in the multi-user setting and chosen key model, the simulation-sound property [Sah99] is required. Intuitively, the simulation-sound property requires that a polynomially bounded party is incapable of convincing others of a false statement even after having seen a simulated proof of its choice. This property guarantees that an adversary in the security against verifiers experiment is not able to submit a resolution query containing an accepting proof of a false statement even after seeing a simulated proof π generated by the simulator. Note that the soundness property of Π does not need to hold in a stronger sense that the adversary is also given the secret decryption key dk as input, as a malicious verifier in an optimistic fair exchange schemes never should have the arbitrator's secret key ASK. Otherwise the verifier will simply use it to decrypt a partial signature. Hence, it has the following theorem:

Theorem 4.4 *If $\mathcal{S}$ is* UF-CMA*-secure signature scheme, $\mathcal{E}$ is a* IND-CCA2*-secure encryption scheme, and Π is a simulation-sound NIZK proof system, then the above optimistic fair exchange scheme constructed from verifiably encrypted signatures is secure in the enhanced model.*

Proof. Security against signers. To break the security against signers, an adversary has to generate a partial signature $\sigma_P = (c, \pi)$ on message m, where π is an accepting proof but $(c, m, ek, vk) \notin L$. However, this is infeasible even when the adversary has the secret decryption key $\mathsf{ASK} = dk$ as input, because the soundness property of Π now holds even when a dishonest prover is explicitly given the secret decryption key dk as input.

Security against verifiers. Suppose an adversary $\mathcal{A}$ breaks the security against verifiers by outputting (m^*, σ^*). If $\mathcal{A}$ has not made a partial signing query on message m^*, the analysis of this type of attack is covered in the security against the arbitrator to be discussed later. Thus without loss of generality, assume that $\mathcal{A}$ has made the query m^* to the partial signing oracle O_{PSig}.

In this setting, it will be shown how to construct an algorithm $\mathcal{B}$ that breaks the IND-CCA2-security of the encryption scheme $\mathcal{E}$. Recall that $\mathcal{B}$ gets ek as input and has access to the decryption oracle O_{Dec}. $\mathcal{B}$ runs $(sk, vk) \leftarrow \mathcal{S}.\mathsf{KGen}(1^k)$ and sets $\mathsf{PK} = vk$, $\mathsf{APK} = ek$. $\mathcal{B}$ forwards PK and APK to $\mathcal{A}$ and simulates the oracles for $\mathcal{A}$.

Let q be the total number of O_{PSig} queries made by $\mathcal{A}$. To reply to the i-th query m_i to the partial signing oracle O_{PSig}, $\mathcal{B}$ uniformly chooses j from $\{1, 2, \cdots, q\}$ and does as follows.

- If $i = j$, $\mathcal{B}$ randomly chooses a message $\bar{m}_0$, and computes $\bar{s}_0 = \mathcal{S}.\mathsf{Sig}(sk, \bar{m}_0)$ and $\bar{s}_1 = \mathcal{S}.\mathsf{Sig}(sk, m_j)$. $\mathcal{B}$ submits $(\bar{s}_0, \bar{s}_1)$ to its own challenger. Let $\bar{c}_b$ be the challenge ciphertext returned by $\mathcal{B}$'s own challenger, which is either $\mathcal{E}.\mathsf{Enc}(ek, \bar{s}_0)$ or $\mathcal{E}.\mathsf{Enc}(ek, \bar{s}_1)$. $\mathcal{B}$ further simulates a proof π_j showing $(\bar{c}_b, m_j, ek, vk) \in L$. $\mathcal{B}$ forwards $(\bar{c}_b, \pi_j)$ to $\mathcal{A}$ the reply of this query.

- If $i \neq j$, $\mathcal{B}$ computes $s_i = \mathcal{S}.\mathsf{Sig}(sk, m_i)$, $c_i = \mathcal{E}.\mathsf{Enc}(ek, s_i)$ and generates a proof showing that $(c_i, m_i, ek, vk) \in L$. The reply of this query is (c_i, π_i).

To simulate a query m to the signing oracle O_{Sig}, $\mathcal{B}$ computes $s = \mathcal{S}.\mathsf{Sig}(sk, m)$ and returns s as the reply.

To simulate a query $(m, (c, \pi), \mathsf{PK}_i)$ to the resolution oracle O_{Res}, $\mathcal{B}$ does as follows.

- If π is valid and $c \neq \bar{c}_b$, $\mathcal{B}$ submits c to its own decryption oracle O_{Dec}. Let the reply be s. $\mathcal{B}$ forwards s to $\mathcal{A}$ as the reply.

- If π is valid and $c = \bar{c}_b$. $\mathcal{B}$ further checks whether $m = m_j$, $\mathsf{PK}_i = \mathsf{PK}$ and $\pi = \pi_j$. If so, $\mathcal{B}$ aborts and outputs a random bit to its $\mathsf{CCA2}$ challenger. Otherwise, by the simulation-sound property of Π, the decryption of c must be a valid signature on message m under the public key PK_i. Suppose $\mathcal{S}.\mathsf{Ver}(m, \bar{s}_b, \mathsf{PK}_i) = \top$ where $b \in \{0, 1\}$. $\mathcal{B}$ returns $\bar{s}_b$ to $\mathcal{A}$, and at the same time breaks the $\mathsf{IND\text{-}CCA2}$-security by outputting the bit b to its own $\mathsf{CCA2}$ challenger.

- If π is invalid, $\mathcal{B}$ returns $\perp$.

Note that if the challenge ciphertext $\bar{c}_b$ is the encryption of $\bar{s}_0$ (i.e., b=0), c_j has no information on the signature of m_j and $\mathcal{A}$'s chance of outputting a signature on message m_j under PK is negligible. If the challenge ciphertext is the encryption of $\bar{s}_1$ (i.e., b=1), then the simulated environment is indistinguishable with that in the real attack environment. Since j is chosen uniformly at random and independent of the adversary $\mathcal{A}$'s behavior, $\mathcal{A}$ outputs a signature on message m_j in this case with non-negligible probability at least $1/q$. Thus if $\mathcal{A}$'s final output (m^*, σ^*) satisfy $m^* = m_j$, $\mathcal{B}$ outputs 1 and otherwise $\mathcal{B}$ outputs a random bit to its $\mathsf{CCA2}$ challenger. It is easy to see that if $\mathcal{A}$ can forge a full signature with non-negligible probability, then $\mathcal{B}$ can also break the $\mathsf{IND\text{-}CCA2}$-security of the encryption scheme $\mathcal{E}$ with non-negligible probability.

Security against the arbitrator. To show security against the arbitrator, one converts any adversary $\mathcal{A}$ that breaks the security against the arbitrator into an adversary $\mathcal{B}$ that forges a new signature for the underlying signature scheme $\mathcal{S}$. The adversary $\mathcal{B}$, on input vk, sets $\mathsf{PK} = vk$, receives APK from $\mathcal{A}$ and simulates the environments for $\mathcal{A}$. To respond to a partial signing query on message m, $\mathcal{B}$ asks a signature s from its own signing oracle on message m, computes $c = \mathcal{E}.\mathsf{Enc}(\mathsf{APK}, s)$ and generates a proof showing $(c, m, \mathsf{APK}, vk) \in L$. To respond to a signing query on message m, $\mathcal{B}$ asks a signature s from its own signing oracle on message m and returns s to $\mathcal{A}$ as the reply. The simulation is perfect and finally $\mathcal{A}$ outputs (m^*, σ^*) such that m^* is not queried to neither the partial signing oracle nor the signing oracle. Thus $\mathcal{B}$ can simply win by outputting (m^*, σ^*). □

4.4.2 Conventional Signature and Ring signature Paradigm

Since the optimistic fair exchange schemes resulted from verifiably encrypted signatures may not be efficient in the standard model due to the involvement of a simulation-sound proof, Huang *et al.* [HYWS08b] suggested another generic methodology for constructing optimistic fair exchange schemes secure in the multi-user setting and chosen-key model. The generic construction is built on conventional signatures and ring signature.

Let $\mathcal{S} = (\mathsf{KGen}, \mathsf{Sig}, \mathsf{Ver})$ be a conventional signature scheme, and $\mathsf{RS} = (\mathsf{KGen}, \mathsf{Sig}, \mathsf{Ver})$ be a ring signature scheme. In Huang et al.'s generic construction, each signer has two key pairs, one for the conventional signature scheme $\mathcal{S}$ and the other for the ring signature scheme RS. The arbitrator has only a key pair for the ring signature scheme RS. The partial signature is a conventional signature σ' generated by the signature scheme $\mathcal{S}$. The full signature σ is the partial signature σ' plus a 2-user ring signature σ_r generated by the ring signature scheme RS with the ring members consisting of the signer and the arbitrator. To convert a partial signature into a full one, the arbitrator simply produces a ring signature σ_r using its secret key with the ring numbers being the signer and the arbitrator. the construction is reviewed as follows.

- $\mathsf{Setup}^{\mathsf{TTP}}$: The arbitrator runs $(ask, apk) \leftarrow \mathsf{RS.KGen}(1^k)$ and sets $(\mathsf{ASK}, \mathsf{APK}) := (ask, apk)$.

- $\mathsf{Setup}^{\mathsf{User}}$: Each user U_i runs $(sk_i, pk_i) \leftarrow \mathcal{S}.\mathsf{KG}(1^k)$, $(rsk_i, rpk_i) \leftarrow \mathsf{RS.KGen}(1^k)$, and sets $(\mathsf{SK}_i, \mathsf{PK}_i) := ((sk_i, rsk_i), (pk_i, rpk_i))$.

- Sig: To sign a full signature on message m, user U_i runs $\sigma' \leftarrow \mathcal{S}.\mathsf{Sig}(sk_i, m)$ and $\sigma_r \leftarrow \mathsf{RS.Sig}(rsk_i, m||\mathsf{PK}_i, R)$[4] where $R := \{rpk_i, apk\}$. The full signature is then set as $\sigma := (\sigma', \sigma_r)$.

- Ver: To verify whether a full signature $\sigma := (\sigma', \sigma_r)$ is generated by user U_i on message m, the verifier checks whether both $\mathcal{S}.\mathsf{Ver}(m, \sigma', pk_i) = \top$ and $\mathsf{RS.Ver}(m||\mathsf{PK}_i, \sigma_r, R) = \top$ where $R := \{rpk_i, apk\}$. If so, it returns $\top$; otherwise, it returns $\perp$.

- PSig: To partially sign a full signature on message m, user U_i runs $\sigma' \leftarrow \mathcal{S}.\mathsf{Sig}(sk_i, m)$, and returns σ' as the partial signature.

- PVer: To verify whether a partial signature $\sigma_P := \sigma'$ is generated by user U_i on message m, a verifier returns $\mathcal{S}.\mathsf{Ver}(m, \sigma', pk_i)$.

- Res: For a partial signature $\sigma_P := \sigma'$ generated by the user U_i on message m, the arbitrator first checks the validity of σ' by running $\mathcal{S}.\mathsf{Ver}(m, \sigma', pk_i)$. If σ' is invalid, it returns $\perp$; otherwise, it computes $\sigma_r \leftarrow \mathsf{RS.Sig}(ask, m||\mathsf{PK}_i, R)$, where $R := \{rpk_i, apk\}$. The arbitrator returns $\sigma := (\sigma', \sigma_r)$.

The correctness property of the construction from this paradigm holds due to correctness of the conventional signature scheme $\mathcal{S}$ and the ring signature scheme RS. The resolution ambiguity property follows the anonymity requirement of the ring signature scheme RS. For the security analysis, the ring signature is required to be unforgeable under an adaptive attack against a static adversary, a ring signature model which was proposed in [Boy07]. The reader is referred to Section 2.2.6 for the security of ring signatures.

Theorem 4.5 *If $\mathcal{S}$ is a conventional signature scheme that is* $\mathsf{UF\text{-}CMA}$*-secure, and* RS *is a secure ring signature scheme that is with basic anonymity and existential unforgeability under an adaptive attack against a static adversary, then the above*

[4] In Huang et al.'s construction, the ring signature is computed as $\sigma_r \leftarrow \mathsf{RS.Sig}(rsk_i, m||\sigma'||\mathsf{PK}_i, R)$. It will be shown later that it would suffice even if signing on a shorter message string $m||\mathsf{PK}_i$ rather than $m||\sigma'||\mathsf{PK}_i$.

optimistic fair exchange scheme constructed from conventional signatures and ring signatures is secure in the enhanced model.

Proof. Security against signers. To break the security against signers, an adversary has to generate a partial signature σ' on message m, such that the arbitrator is not able to generate a ring signature with respect to this message m and the partial signature σ'. However, this is infeasible even if the adversary has the arbitrator's public key, as the arbitrator can always do so under the ring consisting of the adversary and itself, which holds unconditionally.

Security against verifiers. Suppose that an adversary $\mathcal{A}$ breaks the security against verifiers. it will be shown how to construct an algorithm $\mathcal{B}$ that breaks the unforgeability of RS under an adaptive attack against a static adversary.

Given two public keys rpk_0 and rpk_1, which are challenge public keys, $\mathcal{B}$ runs $(sk, pk) \leftarrow \mathcal{S}.\mathsf{KGen}(1^k)$, and sets $\mathsf{APK} := rpk_1$ and $\mathsf{PK} := (pk, rpk_0)$. $\mathcal{B}$ forwards to $\mathcal{A}$ $(\mathsf{PK}, \mathsf{APK})$ and simulates the oracles for $\mathcal{A}$.

When $\mathcal{A}$ makes a partial signing query m to oracle O_{PSig}, $\mathcal{B}$ computes and returns $\mathcal{S}.\mathsf{Sig}(sk, m)$ to $\mathcal{A}$.

When $\mathcal{A}$ makes a signing query m to oracle O_{PSig}, $\mathcal{B}$ firstly runs $\sigma' \leftarrow \mathcal{S}.\mathsf{Sig}(sk, m)$ and gains a ring signature σ_r on message $m||\mathsf{PK}$ under the ring $\{rpk_0, rpk_1\}$ generated using the secret key corresponding to rpk_0 from its own ring signing oracle. $\mathcal{B}$ forwards (σ', σ_r) to $\mathcal{A}$ as the reply.

When $\mathcal{A}$ makes a resolution query $(m, \sigma', \mathsf{PK}'_i)$ where $\mathsf{PK}'_i := (pk'_i, rpk'_i)$ to oracle O_{Res}, $\mathcal{B}$ checks whether $\mathsf{PVer}(m, \sigma', \mathsf{PK}'_i, \mathsf{APK}) = \top$. If not, $\mathcal{B}$ returns $\perp$ to $\mathcal{A}$. Otherwise, $\mathcal{B}$ gains a ring signature σ_r on message $m||\mathsf{PK}'_i$ under the ring $\{rpk'_i, rpk_1\}$ generated using the secret key corresponding to rpk_1 from its own ring signing oracle. $\mathcal{B}$ forwards (σ', σ_r) to $\mathcal{A}$.

The simulation is perfect. Finally, $\mathcal{A}$ outputs its forgery $(\tilde{m}, \tilde{\sigma})$, where $\tilde{\sigma} = (\tilde{\sigma}', \tilde{\sigma}_r)$. Thus it has $\mathsf{RS.Ver}(\tilde{m}||\mathsf{PK}, \tilde{\sigma}_r, R) = \top$ where $R := \{rpk_0, rpk_1\}$. Since $(\tilde{m}, \cdot, \mathsf{PK}) \notin Query(\mathcal{A}, O_{\mathsf{Sig}})$ and $(\tilde{m}, \cdot, \mathsf{PK}) \notin Query(\mathcal{A}, O_{\mathsf{Res}})$, $\mathcal{B}$ has never issued the query on message $\tilde{m}||\mathsf{PK}$ under the ring $\{rpk_0, rpk_1\}$ to its own ring signing oracle. Therefore $\tilde{\sigma}_r$ is a valid ring signature on a new message $\tilde{m}||\mathsf{PK}$ under the ring $\{rpk_0, rpk_1\}$. $\mathcal{B}$ can simply output $(\tilde{m}||\mathsf{PK}, \tilde{\sigma}_r)$ and break the existential unforgeability under an adaptive attack against a static adversary.

Security against the arbitrator. Suppose that an adversary $\mathcal{A}$ breaks the security against the arbitrator. It will be shown how to construct an algorithm $\mathcal{B}$ that breaks

the unforgeability of $\mathcal{S}$. Recall that $\mathcal{B}$ gets vk as input and has access to its own signing oracle O_{sk}. $\mathcal{B}$ runs $(rsk, rpk) \leftarrow \mathsf{RS.KGen}(1^k)$ and forwards $\mathsf{PK} := (vk, rpk)$ to $\mathcal{A}$, who returns to $\mathcal{B}$ the public arbitration key APK. To respond to a partial signing query on message m, $\mathcal{B}$ asks a signature s from its own signing oracle O_{sk} on message m, and returns s to $\mathcal{A}$ as the reply. To respond to a signing query on message m, $\mathcal{B}$ asks a signature s from its own signing oracle on message m, runs $\sigma_r \leftarrow \mathsf{RS.Sig}(rsk, m||\mathsf{PK}, R)$ where $R = \{rpk, \mathsf{APK}\}$ and returns (s, σ_r) to $\mathcal{A}$ as the reply. The simulation is perfect and finally $\mathcal{A}$ outputs $(\tilde{m}, \tilde{\sigma})$, where $\tilde{\sigma} = (\tilde{\sigma}', \tilde{\sigma}_r)$ such that $\tilde{m}$ is not queried to neither the partial signing oracle nor the signing oracle. Thus $\mathcal{B}$ can simply break the unforgeability of $\mathcal{S}$ by outputting $(\tilde{m}, \tilde{\sigma}')$. □

4.5 Chapter Summary

In this chapter the issue in optimistic fair exchange that a potentially dishonest arbitrator may implicitly collude with a signer by sharing its secret arbitration key with the signer was addressed. The security against signers in the enhanced chosen-key model was updated to capture this issue. A separation between the security against signers in the existing model and that in the new model is showed. Together with the security against verifiers and security against the arbitrator proposed in chapter 3, the enhanced chosen-key model have strengthened all the three aspects of multi-user setting and chosen-key security for OFE. After that, the two well known paradigm of constructing optimistic fair exchange were revisited, namely verifiably encrypted signature paradigm and the conventional signature and ring signature paradigm, and it was shown that constructions based on these two paradigms remain valid in the enhanced chosen-key model.

Part II

OFE with Better Efficency

Chapter 5

Optimistic Fair Exchange based on Ring Signatures

In this chapter a new security model for 2-user ring signatures called unforgeability against restricted adaptive attacks is proposed. An observation that ring signatures secure in this model will suffice to construct OFE schemes secure in the enhanced chosen-key model is made. Based on this observation, more efficient OFE schemes secure in the standard model can be constructed.

5.1 Introduction

The highest level of security of optimistic fair exchange in the literature is the multi-user security in the chosen-key model, proposed by Huang, Yang, Wong and Susilo in CT-RSA 2008. They showed that an efficient optimistic fair exchange scheme secure in this sense can be constructed generically from a conventional signature and a ring signature. In particular, they showed that the ring signature did not need to satisfy the highest level of existential unforgeability considered in [BKM06], namely unforgeability with respect to insider corruption. Instead, unforgeability under an adaptive attack, against a static adversary [Boy07] in the 2-user setting will suffice. Since there already exist a number of efficient conventional signatures and ring signatures that are secure without random oracles, efficient OFE schemes whose security does not rely on random oracles [BR93, PS96] can be built by following Huang et al.'s generic construction. In fact, they also demonstrated the flexibility of their generic construction and discussed three efficient OFE instantiations without random oracles.

In Chapter 3 and Chapter 4, Huang et al.'s model was further strengthened to the enhanced chosen-key model, and in Chapter 4, it was shown that the generic construction based on conventional signatures and ring signatures still works in the

enhanced chosen-key model. The security requirement for the ring signature scheme is the same as that in Huang et al.'s model, i.e., the underlying ring signature scheme has to be with basic anonymity and existential unforgeability under an adaptive attack against a static adversary. This chapter will focus on improving the efficiency of the OFE schemes secure in the enhanced chosen-key model.

5.1.1 The Contributions

In this chapter, Huang et al.'s generic construction of OFE schemes from conventional signatures and ring signatures and the security proof in the enhanced chosen-key model is revisited. The contributions comprise three aspects.

Firstly, a new security model for 2-user ring signatures is propose, which is named 'unforgeability against restricted adaptive attacks', and compared with the existing model unforgeability under an adaptive attack, against a static adversary in the 2-user setting. These two models are separated by presenting a concrete ring signature scheme, and it is shown that the new model is strictly weaker. This means that any ring signature that is unforgeable under an adaptive attack, against a static adversary will also be unforgeable in the new model, but the converse statement does not hold.

Next, Huang et al.'s generic construction of OFE schemes from conventional signatures and ring signatures is revisited, and it is proved that a ring signature secure in the new ring signature model would suffice to guarantee the resulting OFE scheme's security in the enhanced chosen-key model, rather than the previous understanding that the ring signature should be unforgeable under an adaptive attack, against a static adversary. This observation can provide alternatives for the ring signatures to be adopted in the OFE construction.

Finally, to demonstrate the impact of the observation, a concrete OFE instantiation is offered based on a ring signature scheme that is only secure in the weaker model, i.e., unforgeability against restricted adaptive attacks. This concrete OFE instantiation is *the most efficient* OFE scheme secure in the standard model in terms of signature size, generation as well as verification. Also, the OFE instantiation relies on a weak assumption, namely, the computational Diffie-Hellman assumption.

5.1.2 Chapter Organization

In the next section a new 2-user ring signature model is defined and separated from existing ring models. Specifically, it is proved that the well-known Bender, Katz and Morselli's 2-user ring signature is secure in the weakened model. In Section 5.3, it is shown that 2-user ring signatures secure in this weaker model will suffice to guarantee the security of the resulting OFE scheme following the aforementioned generic construction. By adopting a ring signature scheme only secure in the weaker model, in Section 5.4, a new instantiation of OFE schemes secure in the enhanced chosen-key model is proposed, and compared with other efficient OFE instantiations. Finally, in Section 5.5, this chapter is summarized.

5.2 A New Model for 2-User Ring Signatures

A new security model named 'unforgeability against restricted adaptive attacks' for 2-user ring signatures is now defined, and then it is shown that it is strictly weaker than the model Huang et al. considered in [HYWS08b]. Consider the following experiment.

$$
\begin{aligned}
(rsk_0, rpk_0) &\leftarrow \mathsf{RS.KG}(1^k) \\
(rsk_1, rpk_1) &\leftarrow \mathsf{RS.KG}(1^k) \\
R &:= \{rpk_0, rpk_1\} \\
(R, m, \sigma) &\leftarrow \mathcal{A}^{O_{\mathsf{RS.Sig}}}(R) \\
\text{success of } \mathcal{A} &:= \left[\wedge \begin{array}{c} \mathsf{RS.Ver}(m, \sigma, R) = \top \\ (m, \{\cdot, rpk_1\}) \notin Query(\mathcal{A}, O_{\mathsf{RS.Sig}}) \end{array} \right]
\end{aligned}
$$

where $\mathcal{A}$ is a PPT adversary, $O_{\mathsf{RS.Sig}}$ is the ring signing oracle which takes as input a message m, a list of public keys S where $|S| = 2$ and $rpk_1 \in S$, and outputs a ring signature σ on message m under the ring S using the signing key rsk_1, and $Query(\mathcal{A}, O_{\mathsf{RS.Sig}})$ is the set of ring signature queries issued by $\mathcal{A}$. By "restricted adaptive", it means that the adversary can adaptively choose a public key rpk where $rpk \neq rpk_1$ and makes a query to the ring signing oracle with respect to a ring $\{rpk, rpk_1\}$, but the adversary is restricted from making queries with respect to a ring $\{rpk, rpk_0\}$. Besides, the adversary is not allowed to make a query on the challenge message with respect to any ring (however, in the model considered by

Huang et al., the adversary can make a query on the challenge message with respect to a 2-user ring S such that $S \neq R$ but $S \cap R \neq \emptyset$). A ring signature scheme is said to be *(existentially) unforgeable against restricted adaptive attacks* if there is no adversary wins the experiment with non-negligible probability.

5.2.1 Separating Adaptive Model from Restricted Adaptive Model

In the following, a concrete example is given for showing that a 2-user ring signature scheme secure in the restricted adaptive model may not be secure in the adaptive model considered by Huang et al. The instantiation is a simple modification of Bender, Katz and Morselli's 2-user ring signature scheme [BKM06]. More specifically, in Bender, Katz and Morselli's 2-user ring signature scheme, each user chooses his own Waters hash generators [Wat05]. In the instantiation, all users share the same Waters hash generators to make things simpler to understand. The modified 2-user ring signature scheme $\mathsf{RS} = (\mathsf{KGen}, \mathsf{Sig}, \mathsf{Ver})$ is as follows.

Let G be a multiplicative cyclic group of prime order p, g be a generator of G, and $e : G \times G \to G_T$ be a bilinear pairing where G_T is a multiplicative group of order p. Let further $u', u_1, u_2, \cdots, u_n \leftarrow G$ be the Waters hash generators that are uniformly and independently chosen at random from G. Let $H : \{0,1\}^* \to \{0,1\}^n$ be a collision-resistant hash function. It is assumed that the public parameter $(G, p, e, H, g, u', u_1, \cdots, u_n)$ is shared by all users.

$\mathsf{RS.KGen}$: Choose a random exponent $\alpha \in \mathbb{Z}_p$; set $rpk = g^\alpha$ and $rsk = \alpha$.

$\mathsf{RS.Sig}(rsk, M, R)$: To sign a message $M \in \{0,1\}^*$ with respect to the ring $R = \{rpk, rpk'\}$ where $rsk = \alpha$ is the corresponding secret key of rpk, proceed as follows: compute $(m_1, \cdots, m_n) \leftarrow H(M)$, where $m_i \in \{0,1\}$ for $i = 1$ to n, and choose random $r \leftarrow \mathbb{Z}_p$. The ring signature σ is set as

$$S_1 = (rpk')^\alpha \cdot (u' \prod_{j=1}^{n} u_j^{m_j})^r, \quad \text{and} \quad S_2 = g^r.$$

$\mathsf{RS.Ver}(M, \sigma, R)$: To verify the signature (S_1, S_2) on message M with respect to the ring $R = \{rpk, rpk'\}$, compute $(m_1, \cdots, m_n) \leftarrow H(M)$ and check whether

$$e(rpk, rpk') = e(S_1, g) \cdot e(S_2^{-1}, u' \prod_{j=1}^{n} u_j^{m_j}).$$

Bender, Katz and Morselli [BKM06] proved that their ring signature scheme is unconditionally anonymous against full key exposure, the highest level of anonymity considered in [BKM06]. They also showed that their ring signature scheme is unforgeable against chosen-subring attacks, in which the adversary is not allowed to making signing queries with respect to a ring containing adversarially-generated public keys.

On the other hand, the same scheme is known to be insecure in the adaptive model as shown in [SW07]. The adaptive attack described in [SW07] is also applicable to the modified version presented above. Indeed, to generate a signature on message $M \in \{0,1\}^*$ with respect to the ring $R = \{rpk, rpk'\}$ where $rsk = \alpha$ is the corresponding secret key of rpk, an adversary randomly chooses $s \in \mathbb{Z}_p$, sets a public key $\bar{rpk} = (rpk')^s$, and asks the signing oracle to generate a signature on message M with respect to the ring $R' = \{rpk, \bar{rpk}\}$. Let $(m_1, \cdots, m_n) \leftarrow H(M)$, the signature with respect to the ring R' will be

$$S_1 = \bar{rpk}^{\alpha} \cdot (u' \prod_{j=1}^{n} u_j^{m_j})^r, \quad \text{and} \quad S_2 = g^r.$$

Therefore $(S_1^{1/s}, S_2^{1/s})$ will be a ring signature with respect to the ring R on message M.

It is easy to see that the above instantiation of ring signature scheme is unconditionally anonymous against full key exposure, as the only value $(rpk')^{\alpha}$ (where $rpk = g^{\alpha}$) is needed to sign, and either of the two users can compute this value.

Next, it is shown that the above instantiation of ring signature scheme is in fact secure in the new unforgeability model, i.e., unforgeable under a restricted adaptive attack.

Theorem 5.1 *The 2-user ring signature scheme above is unforgeable against restricted adaptive attacks if the CDH assumption holds in G.*

Proof. Suppose an adversary $\mathcal{A}$ breaks the unforgeability under a restricted adaptive attack. It will be shown how to construct an algorithm $\mathcal{B}$ that produces a solution for a CDH instance. This will contradict with the CDH assumption. Let the CDH instance be $(G, p, g, A = g^a, B = g^b)$. Let q be the number of different messages contained in the queries $\mathcal{A}$ has made to the ring signing oracles.

At the start, the simulator sets an integer, $m = 4q$, and chooses uniformly at random an integer, k^* between 0 and n. It then chooses an n-length vector, $\vec{x} = (x_i)$,

where the elements of $\vec{x}$ are chosen uniformly at random between 0 and $m-1$ and a value, x', chosen uniformly at random between 0 and $m-1$. Besides, the simulator chooses a random $y' \in \mathbb{Z}_p$ and an n-length vector, $\vec{y} = (y_i)$, where the elements of $\vec{y}$ are chosen at random in $\mathbb{Z}_p$. The simulator keep these values private. let $e : G \times G \to G_T$ be a bilinear pairing where G_T is a multiplicative group of order p and $H : \{0,1\}^* \to \{0,1\}^n$ be a collision-resistant hash function. The simulator sets $rpk_0 = A$ and $rpk_1 = B$ and assigns the Waters hash generators parameters $u' = B^{p-k^*m+x'}g^{y'}$ and $u_i = B^{x_i}g^{y_i}$. The simulator forwards rpk_0, rpk_1 and the public parameters $(G, p, e, H, u', u_1, \cdots, u_n)$ to the adversary. From the view of the adversary, the distribution of the simulated public parameters is identical to the real construction.

For ease of analysis the following functions are defined where $\tilde{M}$ is the set of indices i such that $m_i = 1$ while $H(M) = (m_1, \cdots, m_n)$:

$$F(M) = (p - mk^*) + x' + \sum_{i \in \tilde{M}} x_i, \quad \text{and} \quad J(M) = y' + \sum_{i \in \tilde{M}} y_i.$$

A binary function $K(M)$ is defined as

$$K(M) = \begin{cases} 0, & \text{if } \; x' + \sum_{i \in \tilde{M}} x_i \equiv 0 \pmod{m} \\ 1, & \text{otherwise.} \end{cases}$$

When the adversary issues a signing query on message M with respect to a ring $\{g_1, rpk_1\}$ where g_1 could be rpk_0 or could be adversarially-generated by the adversary, the simulator checks whether $K(M) = 0$. If so, the simulator aborts. Otherwise, the simulator chooses a random $r \in \mathbb{Z}_p$, and computes the signature as

$$S_1 = g_1^{\frac{-J(M)}{F(M)}}(u' \prod_{i \in \tilde{M}} u_i)^r, \quad S_2 = g_1^{\frac{-1}{F(M)}} g^r.$$

Let $\tilde{r} = r - \frac{\alpha}{F(M)}$ and $g_1 = g^\alpha$. Then it has

$$\begin{aligned} S_1 &= g_1^{\frac{-J(M)}{F(M)}}(u' \textstyle\prod_{i \in \tilde{M}} u_i)^r \\ &= g_1^{\frac{-J(M)}{F(M)}}(B^{F(M)} g^{J(M)})^r \\ &= B^\alpha (B^{F(M)} g^{J(M)})^{-\frac{\alpha}{F(M)}} (B^{F(M)} g^{J(M)})^r \\ &= B^\alpha (u' \textstyle\prod_{i \in \tilde{M}} u_i)^{r - \frac{\alpha}{F(M)}} \\ &= B^\alpha (u' \textstyle\prod_{i \in \tilde{M}} u_i)^{\tilde{r}} \end{aligned}$$

Additionally, it has

$$S_2 = g_1^{\frac{-1}{F(M)}} g^r = g^{r-\frac{\alpha}{F(M)}} = g^{\tilde{r}}.$$

The simulator will be able to perform this computation if and only if $F(M) \neq 0 \mod p$. Finally the adversary outputs a signature (S_1^*, S_2^*) on a message M^* with respect to the target ring $\{rpk_0, rpk_1\}$. Due to the collision-resistant property of the hash function, $H(M^*) \neq H(M)$ for any message M that had been submitted to the signing query before. Let $\tilde{M}^*$ be the set of indices i such that $m_i^* = 1$ where $H(M^*) = (m_1^*, \cdots, m_n^*)$. If $x' + \sum_{i \in \tilde{M}^*} x_i = k^* m$, then it has

$$e(\frac{S_1^*}{(S_2^*)^{y+\sum_{i\in\tilde{M}^*} y_i}}, g) = \frac{e(A,B)e(S_2^*, u' \prod_{j=1}^n u_j^{m_j^*})}{e((S_2^*)^{y+\sum_{i\in\tilde{M}^*} y_i}, g)} = e(A,B).$$

Therefore the simulator can solve the CDH problem by computing g^{ab} as

$$g^{ab} = \frac{S_1^*}{(S_2^*)^{y+\sum_{i\in\tilde{M}^*} y_i}}.$$

Similar to [Wat05], the probability that the simulator does not abort during simulating the signing oracle and the equation $x' + \sum_{i \in \tilde{M}^*} x_i = k^* m$ holds is at least $\lambda = \frac{1}{8(n+1)q}$, which is non-negligible. This completes the proof. □

For completeness, the relations between the restricted adaptive model and existing ring signature models are also studied. Specifically, it is further shown that, for 2-user ring signatures satisfying basic anonymity, the model is strictly stronger than the existing model *unforgeability against chosen-subring attacks* [BKM06] reviewed below.

Let $\mathsf{RS} = (\mathsf{KGen}, \mathsf{Sig}, \mathsf{Ver})$ be a ring signature scheme. The model *unforgeability against chosen-subring attackes in 2-user setting* is defined by the following experiment:

1. Key pairs $\{rpk_i, rsk_i\}_{i=1}^{n(k)}$ are generated by $\mathsf{RS.KGen}(1^k)$, and the set of public keys $R := \{rpk_i\}_{i=1}^{n(k)}$ is given to an adversary $\mathcal{A}$.

2. $\mathcal{A}$ is provided with a ring signing oracle $\mathsf{OSign}(\cdot,\cdot,\cdot)$, where $\mathsf{OSign}(s, m, S)$ outputs $\mathsf{RS.Sig}(rsk_s, m, S)$, and it requires that $|S| = 2$, $S \subset R$ and $rpk_s \in S$.

3. $\mathcal{A}$ outputs (m^*, R^*, σ^*) where $|R^*| = 2$, and succeeds if $R^* \subset R$, $\mathsf{RS.Ver}(m^*, \sigma^*, R^*) = \top$, and $\mathcal{A}$ had not queried $(\cdot, m^*, R^*)$ to the ring signing oracle.

A ring signature scheme is said to be *(existentially) unforgeable under a chosen-subring attacks in the 2-user setting* if there is no adversary wins the above experiment with non-negligible probability.

It is emphasized that, by a hybrid argument, the above model is in fact equivalent to a model in which $\mathcal{A}$ is only given the set of two public keys $R := \{prk_0, rpk_1\}$, all the ring signing queries should be queried with respect to the challenge ring R, and the final forgery of a ring signature should be with respect to the same ring R.

For 2-user ring signatures with basic anonymity, the argument that unforgeability against restricted adaptive attacks is a stronger model than unforgeability against chosen-subring attacks follows from the following two claims.

Claim 1 *If a 2-user ring signature scheme achieves basic anonymity and is unforgeable against restricted adaptive attacks, then it is unforgeable against chosen-subring attacks.*

Proof of the claim is straightforward, since, guaranteed by the basic anonymity property, any ring signature queries in the chosen-subring model with respect to the signing key rsk_0 can be simulated by the adversary in the restricted adaptive model by generating a ring signature using secret key rsk_1. □

Claim 2 *If there exists 2-user ring signature scheme which achieves basic anonymity and is unforgeable against chosen-subring attacks, then there exists a scheme which achieves basic anonymity, but is not unforgeable against restricted adaptive attacks.*

Proof. Let $\mathsf{RS} = (\mathsf{KGen}, \mathsf{Sig}, \mathsf{Ver})$ be a ring signature scheme satisfying the conditions stated in the claim. Construct the following scheme RS' from RS as follows.

- $\mathsf{KGen}'(1^k)$: Randomly pick a bit $b \in \{0,1\}$ and run $(rsk, rpk) \leftarrow \mathsf{KGen}(1^k)$. Output $rpk' = b||rpk$ and $rsk' = rsk$.
- $\mathsf{Sig}'(rsk_d, m, S)$: On input a message m, a 2-user set $S := \{b_0||rpk_0, b_1||rpk_1\}$, and a secret key rsk_d which is the corresponding secret key of $b_d||rpk_d$ where $d \in \{0,1\}$, if $b_0 = b_1$, it outputs $\sigma \leftarrow \mathsf{Sig}(rsk_d, m, \bar{S})$ where $\bar{S} := \{rpk_0, rpk_1\}$; otherwise, it outputs $\sigma \leftarrow \mathsf{Sig}(rsk_d, \bar{m}, \bar{S})$, where $\bar{m}$ is the complementary message of m, i.e. $\bar{m}$ and m are of the same bit-length but each i-th bits of them are different.

- $\mathsf{Ver}'(m, \sigma, S)$: On input a message m, a ring signature σ, and a 2-user set $S := \{b_0||rpk_0, b_1||rpk_1\}$, if $b_0 = b_1$, it outputs $\mathsf{Ver}(m, \sigma, \bar{S})$ where $\bar{S} := \{rpk_0, rpk_1\}$; otherwise, it outputs $\mathsf{Ver}(\bar{m}, \sigma, \bar{S})$ where $\bar{m}$ is the complementary message of m.

Clearly, the above scheme RS' still achieves basic anonymity. It is also not difficult to see that it remains unforgeable against chosen-subring attacks. However, it is not unforgeable against restricted adaptive attacks, since given a challenge ring $R := \{b_0||rpk_0, b_1||rpk_1\}$, the adversary can always asks a query on message m with respect to a ring $R' := \{b'_0||rpk_0, b_1||rpk_1\}$ where $b'_0 = (b_0 + 1) \mod 2$. In this case, the adversary can easily forge a ring signature on message $\bar{m}$ with respect to the challenge ring R where message $\bar{m}$ is the complementary message of m. □

Based on the above analysis, the security guaranteed by the restricted adaptive model lies between the securities guaranteed by unforgeability against chosen-subring attacks and by unforgeability under an adaptive attack, against a static adversary.

5.3 Security of the Generic Construction

Huang et al.'s generic construction of OFE schemes based on conventional signatures and ring signatures was reviewed in Chapter 4. It has been proved that the above generic construction of OFE is secure in the enhanced chosen-key model, provided that $\mathcal{S}$ is existentially unforgeable against adaptive chosen message attacks and RS is a secure ring signature scheme with basic anonymity and existential unforgeability under an adaptive attack, against a static adversary in the 2-user setting.

Here it will be shown that the construction is still secure if a ring signature scheme that is secure in the weaker ring model, i.e., unforgeable against restricted adaptive attacks, is adopted.

Theorem 5.2 *The generic construction of optimistic fair exchange scheme is secure in the enhanced chosen-key model, if $\mathcal{S}$ is existentially unforgeable against chosen message attacks and RS is a secure ring signature with basic anonymity and existential unforgeable against restricted adaptive attacks.*

Proof. Theorem 5.2 follows from the following lemmas. □

Lemma 5.3 *The generic construction of optimistic fair exchange scheme is unconditionally secure against signers.*

Proof. Obviously, for any message m and any valid conventional signature σ' on m under the verification key pk_i, the arbitrator can always produce a ring signature σ_r on $m||\mathsf{PK}_i$ using its own secret key, under the ring $R := \{rpk_i, apk\}$. Therefore, no adversary can win the game. □

Lemma 5.4 *The generic construction of optimistic fair exchange scheme is secure against verifiers if* RS *is unforgeable against restricted adaptive attacks.*

Proof. Suppose that an adversary $\mathcal{A}$ breaks the security against verifiers. It is shown how to construct an algorithm $\mathcal{B}$ that breaks the unforgeability against restricted adaptive attacks.

Given two public keys rpk_0 and rpk_1, which are challenge public keys, $\mathcal{B}$ randomly generates a key pair (sk, pk) of $\mathcal{S}$ by running $(sk, pk) \leftarrow \mathcal{S}.\mathsf{KGen}(1^k)$, and sets $\mathsf{APK} := rpk_1$ and $\mathsf{PK} := (pk, rpk_0)$. It then runs $\mathcal{A}$ as a subroutine with input $(\mathsf{PK}, \mathsf{APK})$.

When $\mathcal{A}$ makes a partial signing query m to oracle O_{PSig}, $\mathcal{B}$ computes and returns $\mathcal{S}.\mathsf{Sig}(sk, m)$ to $\mathcal{A}$.

When $\mathcal{A}$ makes a signing query m to oracle O_{PSig}, $\mathcal{B}$ firstly runs $\sigma' \leftarrow \mathcal{S}.\mathsf{Sig}(sk, m)$ and gains a ring signature σ_r on message $m||\mathsf{PK}$ under the ring $\{rpk_0, rpk_1\}$ from its own ring signing oracle. $\mathcal{B}$ forwards (σ', σ_r) to $\mathcal{A}$ as the reply. Due to the basic anonymity, even though the ring signature is generated using the secret key corresponding to rpk_1, the answer to the signing query is indistinguishable from that in the real environment.

When $\mathcal{A}$ makes a resolution query $(m, \sigma', \mathsf{PK}'_i)$ where $\mathsf{PK}'_i := (pk'_i, rpk'_i)$ to oracle O_{Res}, $\mathcal{B}$ checks whether $\mathsf{PVer}(m, \sigma', \mathsf{PK}'_i, \mathsf{APK}) = \top$. If not, $\mathcal{B}$ returns $\perp$ to $\mathcal{A}$. Otherwise, $\mathcal{B}$ gains a ring signature σ_r on message $m||\mathsf{PK}'_i$ under the ring $\{rpk'_i, rpk_1\}$ from its own ring signing oracle. $\mathcal{B}$ forwards (σ', σ_r) to $\mathcal{A}$.

Finally, $\mathcal{A}$ outputs its forgery (m^*, σ^*), where $\sigma^* = (\sigma'^*, \sigma_r^*)$. Thus it must be $\mathsf{RS}.\mathsf{Ver}(m^*||\mathsf{PK}, \sigma_r^*, R) = \top$ where $R := \{rpk_0, rpk_1\}$. Since $(m^*, \cdot, \mathsf{PK})$ is not submitted to the signing oracle or the resolution oracle, $\mathcal{B}$ has never issued a query $(\sigma^*||\mathsf{PK}, \{\cdot, rpk_1\})$ to its own ring signing oracle. Therefore σ_r^* is a valid ring signature on a new message $m^*||\mathsf{PK}$ under the ring $\{rpk_0, rpk_1\}$. $\mathcal{B}$ can simply output $(m^*||\mathsf{PK}, \sigma_r^*)$ and break the existential unforgeability against restricted adaptive attacks. □

Lemma 5.5 *The generic construction of optimistic fair exchange scheme is secure against the arbitrator if $\mathcal{S}$ is unforgeable under adaptive chosen message attacks.*

Proof. Since this proof only relies on the property of the conventional signature scheme $\mathcal{S}$, the proof is the same as that in Theorem 4.5 in Chapter 4. Here it will just be omitted. □

5.4 A new efficient OFE scheme

In the following, an OFE instantiation is provided to demonstrate the significance of the generic construction when adopting a ring signature scheme that is unforgeable against restricted adaptive attacks. Water's signature scheme [Wat05] is used as $\mathcal{S}$ and the modified Bender-Katz-Morselli ring signature (in Section 5.2.1) as RS. Note that in Water's signature scheme, each user's public key includes g_1, g_2 and a set of Waters hash generators $u', u_1, \cdots, u_n$. In the OFE instantiation, all the users share the same Waters hash generators so that $\mathcal{S}$ and RS can share the same public parameters. Besides, it is required that all users share the same g_2 so that each user can combine two key pairs for $\mathcal{S}$ and RS respectively into a single key pair.

Global Setup : On input 1^k where k is a security parameter, the setup algorithm generates a multiplicative cyclic group G of prime order p and a bilinear pairing $e : G \times G \to G_T$ where G_T is a multiplicative group of order p. Let g be a generator of G. It then chooses random exponents $a \in \mathbb{Z}_p$ and sets $A := g^a$. Let $H : \{0,1\}^* \to \{0,1\}^n$ be a collision-resistant hash function. The setup algorithm picks independently and uniformly at random Waters hash generators $u', u_1, \cdots, u_n \leftarrow G$. The public parameters shared by all users are set as $(G, p, e, H, A, u', u_1, \cdots, u_n)$.

After the global setup is finished, the algorithms in OFE can be executed as follows.

$\mathsf{Setup}^{\mathsf{TTP}}$: The TTP chooses exponents $y \leftarrow \mathbb{Z}_p$ and sets $Y = g^y$. APK is set as Y. ASK is set as y.

$\mathsf{Setup}^{\mathsf{User}}$: User U_i randomly chooses an exponent $x_i \leftarrow \mathbb{Z}_p$ and sets $\mathsf{SK}_i = x_i$ and $\mathsf{PK}_i = g^{x_i}$.

$\mathsf{PSig}(M, \mathsf{SK}_i, \mathsf{APK})$: It computes $(m_1, \cdots, m_n) \leftarrow H(M)$, chooses $r \leftarrow \mathbb{Z}_p$, and

computes

$$S_1 = A^{x_i} \cdot (u' \prod_{j=1}^{n} u_j^{m_j})^r, \quad \text{and} \quad S_2 = g^r.$$

The partial signature is set as $\sigma_P = (S_1, S_2)$.

$\mathsf{PVer}(M, \sigma_P, \mathsf{PK}_i, \mathsf{APK})$: It verifies

$$e(A, \mathsf{PK}_i) = e(S_1, g) \cdot e(S_2^{-1}, u' \prod_{j=1}^{n} u_j^{m_j}).$$

If so, returns $\top$; otherwise it returns $\perp$.

$\mathsf{Sig}(M, \mathsf{SK}_i, \mathsf{APK})$: It computes $\sigma_P \leftarrow \mathsf{PSig}(M, \mathsf{SK}_i, \mathsf{APK})$ and $(m'_1, \cdots, m'_n) \leftarrow H(M||\mathsf{PK}_i)$. It then chooses $r' \leftarrow \mathbb{Z}_p$ and compute

$$S'_1 = \mathsf{APK}^{x_i} \cdot (u' \prod_{j=1}^{n} u_j^{m'_j})^{r'}, \quad \text{and} \quad S'_2 = g^{r'}.$$

The full signature is set as $\sigma = (\sigma_P, S'_1, S'_2)$.

$\mathsf{Ver}(M, \sigma_P, \mathsf{PK}_i, APK)$: It verifiers whether $\mathsf{PVer}(M, \sigma_P, \mathsf{PK}_i, \mathsf{APK}) = \top$ and whether

$$e(\mathsf{APK}, \mathsf{PK}_i) = e(S'_1, g) \cdot e(S_2'^{-1}, u' \prod_{j=1}^{n} u_j^{m'_j}).$$

If both hold, it returns $\top$; otherwise, it returns $\perp$.

$\mathsf{Res}(M, \sigma_P, \mathsf{ASK}, \mathsf{PK}_i)$: It first verifies whether σ_P is a valid partial signature by running $\mathsf{PVer}(M, \sigma_P, \mathsf{PK}_i, \mathsf{APK})$. If σ_P is invalid, it returns $\perp$. Otherwise, it computes $(m'_1, \cdots, m'_n) \leftarrow H(M||\mathsf{PK}_i)$, chooses $r' \leftarrow \mathbb{Z}_p$, computes

$$S'_1 = \mathsf{PK}_i^y \cdot (u' \prod_{j=1}^{n} u_j^{m_j})^{r'}, \quad \text{and} \quad S'_2 = g^{r'},$$

and returns $\sigma = (\sigma_P, S'_1, S'_2)$.

Since the securities of both Waters' signature scheme and that of the modified Bender-Katz-Morselli ring signature are based on the computational Diffie-Hellman assumption, the instantiation is secure under the computational Diffie-Hellman assumption [DH76], a well-established assumption on which many cryptographic primitives are based and was reviewed in Definition 2.4.

5.4.1 Comparison

There are only three efficient OFE schemes that are known to be secure in the multi-user setting and chosen-key model without random oracles. They are the three instantiations proposed by Huang et al. based on their generic construction. **Instantiation** $\mathcal{I}_1$ uses Waters' signature scheme [Wat05] as $\mathcal{S}$ and Shacham-Waters's ring signature scheme [SW07] as RS. The security is based on sub-group decision assumption [BGN05, SW07] and computational Diffie-Hellman assumption [DH76]. **Instantiation** $\mathcal{I}_2$ employs Boneh-Boyen's weakly secure signature scheme [BB04] plus a one-time signature scheme as $\mathcal{S}$, and Chandran-Groth-Sahai ring signature scheme [CGS07] as RS. The security follows from a stronger assumption, i.e., strong Diffie-Hellman assumption [BB04, CGS07].

In these two instantiations, each user has two key pairs, one for the conventional signature and the other one for ring signature. To make the instantiations more practical and efficient, it may be more desirable to combine the two key pairs into one. Thus in **Instantiation** $\mathcal{I}_3$, Boyen's ring signature [Boy07] (or, say, his mesh signature) is employed. In Boyen's ring signature scheme, each user owns a single key pair, and the adversary can ask not only ring signature queries, but also atomic (or conventional) signature queries. The security is based on Poly Strong Diffie-Hellman assumption introduced by Boyen [Boy07], which is a stronger variant of the Strong Diffie-Hellman assumption.

Compared with the three instantiations suggested by Huang et al., the new instantiation has the advantage of relying on simpler cryptography assumption. Besides, in both **Instantiation** $\mathcal{I}_3$ and the new instantiation, each user owns a single key pair. However, the public key of a user in **Instantiation** $\mathcal{I}_3$ consists of three group elements, while only one in the new instantiation.

Next the performance of the instantiation is compared with Huang et al.'s three instantiations. Since pairing and exponentiation operations take more time than multiplication operations do, the costs of multiplication computations and hash evaluations will be simply ignored. Let "E" denote an exponentiation operation, and "P" denote a pairing operation. By "$\mathsf{OFE.Sig}$ elements", "$\mathsf{OFE.Sig}$ costs" and "$\mathsf{OFE.Ver}$ costs", they mean the number of group elements of a full signature, the cost of generating a full signature, and the cost of verifying a full signature, respectively. The table 5.1 summarizes the performances of Huang et al.'s instantiations and the new instantiation.

Instantiations:	OFE.Sig elements	OFE.Sig costs	OFE.Ver costs
$\mathcal{I}_1$	8	7E	8P
$\mathcal{I}_2$	10	12E	9P
$\mathcal{I}_3$	8	10E	8E + 4P
New	4	4E	4P

Table 5.1: Performance Comparison

From the table, it is clear that, compared with Huang et al.'s instantiations, the new instantiation saves almost 50% or even more of the costs in both generating a full signature and verifying a full signature. Besides, the number of group elements of the full signature in the new instantiation is only half of those in Huang et al.'s instantiations. It should also be noted that **Instantiation** $\mathcal{I}_1$ requires the use of composite order groups equipped with a bilinear map, which is known to be less efficient compared with prime-order groups equipped with bilinear map.

As a side note, the most efficient scheme in the random oracle model under the CDH assumption is based on the verifiably-encrypted signature [BGLS03]. For the the generation and verification of a partial signature, the costs are 3 exponentiations and 3 pairing. The corresponding figures for full signatures are 1 exponentiation and 2 pairings. The partial signature size and full signature size are 2 and 1 group elements respectively. It is fair to say the construction performs comparably to the most efficient scheme secure in the random oracle model.

5.5 Chapter Summary

It is well-known that efficient optimistic fair exchange schemes without random oracles can be built from conventional signatures and ring signatures. To guarantee the resulting OFE scheme's security in the enhanced chosen-key model, it was previously believed that the ring signature scheme should be unforgeable under an adaptive attack, against a static adversary in the 2-user setting. In this chapter, a new weaker model for ring signatures named "unforgeability against restricted adaptive attacks" was proposed, and it was proved that a 2-User ring signature secure in the weaker model was sufficient to guarantee the resulting OFE scheme's security. This observation makes it feasible to construct more efficient OFE schemes whose security relies on a weaker assumption.

Part III

OFE with New Features

Chapter 6

Threshold-Oriented Optimistic Fair Exchange

In this chapter a new scenario of OFE, namely exchange of threshold signatures, is considered.

6.1 Introduction

In Chapter 3 and 4 the OFE security model was extended to better capture possible attacks in reality, and in Chapter 5 more efficient OFE schemes in the enhanced chosen-key model were proposed. In this chapter, a different aspect of OFE will be investigated.

Since the introduction of OFE, some desirable properties such as setup-free [ZB06], stand-alone [ZB06], abuse-free [GJM99], signer ambiguity [HYWS08a], resolution ambiguity [MK01] and accountability [HMS+11] has been proposed in the literature.

In [AV04] and [KL10], OFE employing multiple arbitrators are discussed to reduce the trust placed on the single arbitrator. Unfortunately, the existing techniques are either expensive or rely on synchronized clocks, which is undesirable as achieving synchronization in a peer-to-peer setting in which the arbitrators do not even know each other is hard.

Traditionally in OFE the digital items to be exchanged are digital signatures. Most of the previous works on OFE are done in the individual setting, in which the two involving parties are individual users and they represent themselves. An interesting scenario in OFE is that either party consists of a group of users. In such a scenario, every single user in the group can represent its party to execute transactions with another party. In [QWM12], the authors employ a ring signature such that all the group of users' public keys are involved in the ring to ensure that each signer

can sign on behalf of the party. Later, optimistic fair exchange of group signatures is considered in [HWS11b]. The difference between optimistic fair exchange of ring signatures and that of group signatures is similar with the difference between ring signatures and group signatures. For instance, the user in ring signature setting can choose its key pair by himself while in the group signature setting, a group manager is responsible to the generation of a user's key pair. Furthermore, optimistic fair exchange of group signatures has an additional feature that the anonymity of a signer can be revoked by its group manager. One common characteristic that the above two kinds of optimistic fair exchange share is that each single user in the party can sign on behalf of the party.

To the best of my knowledge, there is no previous work on OFE discussing about the scenario that at least a number of users together can represent a party. That is, for a party involving a group of n users, only at least t users of them together can sign on behalf of the party and make exchanges with other parties. The notion of threshold-oriented optimistic fair exchange (TOFE) will be introduced, which in essence is optimistic fair exchange of threshold signatures. This can be viewed as a natural way to reduce the trust placed on every single user of the group.

Besides, TOFE has other practical applications. For example, consider the case in which two parties intend to exchange a secret key of an identity-based encryption (IBE) [BF01]. In an identity-based setting, the key generation centre (KGC) is a high value target to adversaries as compromising the master key will break the whole system. Thus the master key is typically split amongst a set of authorities so that only when a threshold of authorities together can create a secret key for an identity [KG10]. Remember that the secret key of an identity can be viewed as a digital signature on the user's identity from the KGC [BF01]. Thus, fair exchange of secret key of an identity-based encryption also falls within the model of OFE. In case when the master key is split amongst a set of authorities and two KGCs, perhaps each for a certain geographic location, would like to exchange a secret key of a specific identity, TOFE would be useful.

The table below summarizes the categories of exchanged digital items that have been discussed in the literatures.

Schemes	Digial Items Exchanged
traditional OFE	individual signatures
Qu et al. [QWM12]	ring signatures
Huang et al. [HWS11b]	group signatures
The Scheme	threshold signatures / secret keys of an IBE

Table 6.1: Digital items that are exchanged in OFE

6.1.1 Contribution.

In this chapter, optimistic fair exchange in a threshold-oriented setting is studied. Specifically, a formal definition for TOFE is presented. A concrete construction is proposed and it is demonstrated that the construction is secure in the random oracle model.

6.2 Definition of TOFE

6.2.1 Syntax

The definitions and security models of OFE from various literatures are adapted for the TOFE. For efficiency consideration, the definition of TOFE consists of non-interactive algorithms only. The following is the syntax of a construction of TOFE, which consists of seven algorithms. In addition, the common reference string model is adopted.

- *Common Reference String Generation* On input a security parameter 1^k, this algorithm outputs a common reference string param_{CRS} which includes the security parameter 1^k. It is assumed param_{CRS} is an implicit input to all algorithms described below.

- $(\mathsf{pk}_A, \mathsf{sk}_A) \leftarrow \mathsf{AGen}()$ This algorithm outputs the arbitrator key pairs (pk_A, sk_A).

- $(\mathsf{pk}_U, \{\mathsf{sk}_{U,i}\}_{i=1}^n) \leftarrow \mathsf{UGen}(n, t)$ This algorithm takes as input the required number of signers n, the threshold t and output the public key of the user pk_U, together with n secret signing keys for the signers $\mathsf{sk}_{U,i}$.

- $\mathsf{PSign} = (\mathsf{PSign}_{(s)}, \mathsf{PSign}_{(v)}, \mathsf{PSign}_{(g)})$ This is a suite of three algorithms which allows a subset of signers to create a partial signature.
  - $\hat{\sigma}_i \leftarrow \mathsf{PSign}_{(s)}(\mathsf{pk}_A, M, \mathsf{sk}_{U,i})$ On input the public key of the arbitrator pk_A, a message M and a secret signing key of signer i, this algorithm outputs a partial signature share for signer i.
  - $\mathit{valid}/\mathit{invalid} \leftarrow \mathsf{PSign}_{(v)}(\mathsf{pk}_A, \mathsf{pk}_U, M, \hat{\sigma}_i, i)$ On input the public key of the arbitrator pk_A and that of the user pk_U, a message M, a partial signature share $\hat{\sigma}_i$ from signer i, this algorithm checks the validity of the partial signature share created by signer i.
  - $\hat{\sigma} \leftarrow \mathsf{PSign}_{(g)}(\mathsf{pk}_A, \mathsf{pk}_U, M, \{\hat{\sigma}_i\}_{i\in\mathcal{I}}, \mathcal{I})$ On input the public key of the arbitrator pk_A and that of the user pk_U, a message M, t partial signature shares $\{\hat{\sigma}_i\}$ for $i \in \mathcal{I}$ such that $\mathcal{I} \subset [n]$ and $|\mathcal{I}| = t$, this algorithm outputs a partial signature.
- $\mathit{valid}/\mathit{invalid} \leftarrow \mathsf{PVer}(\mathsf{pk}_A, \mathsf{pk}_U, M, \hat{\sigma})$ This algorithm checks the validity of a partial signature $\hat{\sigma}$ on message M based on the public key of the arbitrator pk_A, the public key of the user pk_U.
- $\mathsf{Sign} = (\mathsf{Sign}_{(s)}, \mathsf{Sign}_{(v)}, \mathsf{Sign}_{(g)})$ Similar to the partial signature generation process, the signing algorithm is also a set of three algorithms which allows a subset of signers to create a signature.
  - $\sigma_i \leftarrow \mathsf{Sign}_{(s)}(\mathsf{pk}_A, M, \mathsf{sk}_{U,i})$ On input public key of the arbitrator pk_A, message M and secret signer key of signer i, this algorithm outputs a signature share for signer i.
  - $\mathit{valid}/\mathit{invalid} \leftarrow \mathsf{Sign}_{(v)}(\mathsf{pk}_A, \mathsf{pk}_U, M, \sigma_i, i)$ This algorithm checks the validity of the signature share σ_i created by signer i based on the public key of the arbitrator pk_A, the public key of the user pk_U and message M.
  - $\sigma \leftarrow \mathsf{Sign}_{(g)}(\mathsf{pk}_A, \mathsf{pk}_U, M, \{\sigma_i\}_{i\in\mathcal{I}}, \mathcal{I})$ On input the public key of the arbitrator pk_A and that of the user pk_U, a message M, t signature shares $\{\sigma_i\}$ for $i \in \mathcal{I}$ such that $\mathcal{I} \subset [n]$ and $|\mathcal{I}| = t$, this algorithm outputs a signature.
- $\mathit{valid}/\mathit{invalid} \leftarrow \mathsf{Ver}(\mathsf{pk}_A, \mathsf{pk}_U, M, \sigma)$ This algorithm checks the validity of a signature σ on message M based on the public key of the arbitrator pk_A and that of the user pk_U.

- $\sigma \leftarrow \mathsf{Res}(\mathsf{pk}_A, \mathsf{pk}_U, M, \hat{\sigma}, \mathsf{sk}_A)$ Given a valid partial signature $\hat{\sigma}$, a message M, public key of the user pk_U, key pair of the arbitrator $(\mathsf{pk}_A, \mathsf{sk}_A)$, this algorithm allows the arbitrator to output a signature on message M. Note that $\perp$ is returned if $\texttt{invalid} \leftarrow \mathsf{PVer}(\mathsf{pk}_A, \mathsf{pk}_U, M, \hat{\sigma})$.

Correctness A construction of TOFE is correct if the following conditions hold:

1. Any partial signature created by any t honest signers using PSign will be valid under PVer.
2. Any signature created by any t honest signers using Sign will be valid under Ver.
3. Any signature created by the arbitrator using Res based on a valid partial signature will be valid under Ver.

Furthermore, it is required that any signature created by the arbitrator using Res based on a valid partial signature will be indistinguishable from the signature created by any t honest signers using Sign.

6.2.2 A Typical Usage of the TOFE Algorithms

Note that in OFE with three message flows between the initiator Alice and the receiver Bob, the item to be sent by Bob is not restricted to any format. It could be a digital item such as electronic money. For simplicity it is assumed that the item to be sent by Bob is a digital signature. Nonetheless, it could be a ring signature, a group signature or a threshold signature. Below it is showed how Alice and Bob can conduct an exchange based on the definition of TOFE. Note that the party Alice in TOFE consists of a group of n signers, and an exchange is possible only when at least t-out-of-n signers agree to participate.

The definition of TOFE does not require the set of t signers to communicate with each other. Below is a typical usage of the definition of TOFE algorithms.

1. *Partial Signature Shares Collection* Bob approaches each signer independently and the signers agree on the items to be exchanged. The signer, say signer i, invokes $\mathsf{PSign}_{(s)}$ and sends the share of the partial signature $\hat{\sigma}_i$ to Bob. Bob uses $\mathsf{PSign}_{(v)}$ to verify the share.

2. *Partial Signature Generation* Upon collecting t partial signature shares, Bob invokes $\mathsf{PSign}_{(g)}$ to generate a partial signature $\hat{\sigma}$. He invokes PVer to ensure its validity.

3. *Obligation Fulfillment* If the partial signature Bob obtained is valid, he fulfills his obligations. In this example, Bob sends his digital signature to all the signers involved.

4. *Signature Shares Collection* Each signer validates that Bob has fulfilled his obligations. In this example, each signer checks that the digital signature sent by Bob is valid. If yes, each signer, say signer i, invokes $\mathsf{Sign}_{(s)}$ and sends the share of the signature σ_i to Bob, who checks its validity with $\mathsf{Sign}_{(v)}$.

5. *Signature Generation* Upon collecting t signature shares, Bob invokes $\mathsf{Sign}_{(g)}$ to generate a signature σ. He invokes Ver to ensure its validity. If yes, the exchange process is completed.

6. *Resolution* Suppose some signers refuse to send their signature shares, or that the signature created in signature generation is invalid, Bob can approach the arbitrator for assistance. Specifically, he approaches the arbitrator and proves that he has fulfilled his obligation. After that, Bob submits the valid partial signature $\hat{\sigma}$ to the arbitrator. The arbitrator sends back the signature σ by invoking Res and this completes the exchange.

7. *Remarks* In this example, Bob can send his digital signature to the arbitrator as a proof of obligation fulfillment. Even if Bob is lying, the arbitrator can still give this digital signature to the signers should they also complain and thus the exchange could be completed regardless of what happens afterwards.

6.2.3 Security Model

Traditionally, any construction of optimistic fair exchange should be secure in three aspects, namely, security against signers, security against verifiers and security against the arbitrator respectively. As suggested by the respective names, they intend to cover the scenarios when the named party is dishonest. The traditional model is modified to suit the threshold setting. Specifically, the verifier can collude with $t-1$ malicious signers in the consideration of security against verifiers.

Security Against Signers

This property guarantees that even when all the signers collude together, they cannot create a partial signature that passes the partial signature verification algorithm PVer yet it cannot be resolved into a full signature by the arbitrator. This property intends to protect honest verifiers. Specifically, the following three-phase game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$ is used to define this property.

Initialization $\mathcal{A}$ specifies the number of signers n and the threshold t. $\mathcal{C}$ creates the common reference string param_{CRS} and invokes

$$(\mathsf{pk}_A, \mathsf{sk}_A) \leftarrow \mathsf{AGen}(),$$

$$(\mathsf{pk}_U, \{\mathsf{sk}_{U,i}\}_{i=1}^n) \leftarrow \mathsf{UGen}(n, t).$$

$\mathcal{C}$ gives $(\mathsf{param}_{CRS}, \mathsf{pk}_A, \mathsf{pk}_U, \{\mathsf{sk}_{U,i}\}_{i=1}^n)$ to $\mathcal{A}$.

Query $\mathcal{A}$ can adaptively issue the following query to $\mathcal{C}$.

- Res Query. $\mathcal{A}$ gives $(\hat{\sigma}, M)$ to $\mathcal{C}$, who invokes

$$\sigma \leftarrow \mathsf{Res}(\mathsf{pk}_A, \mathsf{pk}_U, M, \hat{\sigma}, \mathsf{sk}_A)$$

  and returns σ to $\mathcal{A}$.

End-Game $\mathcal{A}$ submits $(M^*, \hat{\sigma}^*)$ and wins the game if

$$\mathtt{valid} \leftarrow \mathsf{PVer}(\mathsf{pk}_A, \mathsf{pk}_U, M^*, \hat{\sigma}^*)$$

$$\mathtt{invalid} \leftarrow \mathsf{Ver}\big(\mathsf{pk}_A, \mathsf{pk}_U, M^*, \mathsf{Res}(\mathsf{pk}_A, \mathsf{pk}_U, M, \hat{\sigma}^*, \mathsf{sk}_A)\big)$$

Security Against Verifiers

This property guarantees that even when the verifier colludes with $t-1$ signers, they cannot create a valid full signature. This property intends to protect honest signers. The model is static in the sense that the subset of signers to be controlled by the attacker is fixed during the initialization phase. Specifically, the following three-phase game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$ is used to define this property.

Initialization $\mathcal{A}$ specifies the number of signers n and the threshold t, together with an index set $\mathcal{I}' \subset [n]$ such that $|\mathcal{I}'| = t - 1$. $\mathcal{C}$ creates the common reference string param_{CRS} and invokes

$$(\mathsf{pk}_A, \mathsf{sk}_A) \leftarrow \mathsf{AGen}(),$$

$$(\mathsf{pk}_U, \{\mathsf{sk}_{U,i}\}_{i=1}^{n}) \leftarrow \mathsf{UGen}(n, t).$$

$\mathcal{C}$ gives $(\mathsf{param}_{CRS}, \mathsf{pk}_A, \mathsf{pk}_U, \{\mathsf{sk}_{U,i}\}_{i\in\mathcal{I}'})$ to $\mathcal{A}$.

Query $\mathcal{A}$ can adaptively issue the following query to $\mathcal{C}$.

- $\mathsf{PSign}_{(s)}$ Query. $\mathcal{A}$ gives (M, i) to $\mathcal{C}$, who invokes $\hat{\sigma}_i \leftarrow \mathsf{PSign}_{(s)}(\mathsf{pk}_A, M, \mathsf{sk}_{U,i})$ and returns $\hat{\sigma}_i$ to $\mathcal{A}$.
- $\mathsf{Sign}_{(s)}$ Query. $\mathcal{A}$ gives (M, i) to $\mathcal{C}$, who invokes $\sigma_i \leftarrow \mathsf{Sign}_{(s)}(\mathsf{pk}_A, M, \mathsf{sk}_{U,i})$ and returns σ_i to $\mathcal{A}$.
- Res Query. $\mathcal{A}$ gives $(\hat{\sigma}, M)$ to $\mathcal{C}$, who invokes $\sigma \leftarrow \mathsf{Res}(\mathsf{pk}_A, \mathsf{pk}_U, M, \hat{\sigma}, \mathsf{sk}_A)$ and returns σ to $\mathcal{A}$.

End-Game $\mathcal{A}$ submits $(M^*, \hat{\sigma}^*)$ and wins the game if

$$\mathtt{valid} \leftarrow \mathsf{Ver}(\mathsf{pk}_A, \mathsf{pk}_U, M^*, \hat{\sigma}^*)$$

and that $(M^*, \cdot)$ did not appear in any $\mathsf{Sign}_{(s)}$ query. Furthermore, if there exists a $\mathsf{PSign}_{(s)}$ query with input $(M^*, \cdot)$, $(\cdot, M^*)$ should not appear as input in any Res query.

Security Against the Arbitrator

This property guarantees that the arbitrator cannot create a signature on behalf of the user unless it is given a valid partial signature. In TOFE, the arbitrator is allowed to collude with $t - 1$ signers. As in the case of security against verifiers, the model is static in the sense that the subset of signers to be controlled by the attacker is fixed during the initialization phase. Specifically, the following three-phase game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$ is used to define this property.

Initialization $\mathcal{A}$ specifies the number of signers n and the threshold t, together with an index set $\mathcal{I}' \subset [n]$ such that $|\mathcal{I}'| = t - 1$. $\mathcal{C}$ creates the common reference string param_{CRS} and invokes

$$(\mathsf{pk}_A, \mathsf{sk}_A) \leftarrow \mathsf{AGen}(),$$

$$(\mathsf{pk}_U, \{\mathsf{sk}_{U,i}\}_{i=1}^n) \leftarrow \mathsf{UGen}(n, t).$$

$\mathcal{C}$ gives $(\mathsf{param}_{CRS}, \mathsf{pk}_A, \mathsf{pk}_U, \{\mathsf{sk}_{U,i}\}_{i\in\mathcal{I}'}, \mathsf{sk}_A)$ to $\mathcal{A}$.

Query $\mathcal{A}$ can adaptively issue the following query to $\mathcal{C}$.

- $\mathsf{PSign}_{(s)}$ Query. $\mathcal{A}$ gives (M, i) to $\mathcal{C}$, who invokes $\hat{\sigma}_i \leftarrow \mathsf{PSign}_{(s)}(\mathsf{pk}_A, M, \mathsf{sk}_{U,i})$ and returns $\hat{\sigma}_i$ to $\mathcal{A}$.
- $\mathsf{Sign}_{(s)}$ Query. $\mathcal{A}$ gives (M, i) to $\mathcal{C}$, who invokes $\sigma_i \leftarrow \mathsf{Sign}_{(s)}(\mathsf{pk}_A, M, \mathsf{sk}_{U,i})$ and returns σ_i to $\mathcal{A}$.

End-Game $\mathcal{A}$ submits $(M^*, \hat{\sigma}^*)$ and wins the game if

$$\mathtt{valid} \leftarrow \mathsf{Ver}(\mathsf{pk}_A, \mathsf{pk}_U, M^*, \hat{\sigma}^*)$$

and that $(M^*, \cdot)$ did not appear in any $\mathsf{Sign}_{(s)}$ query nor $\mathsf{PSign}_{(s)}$ query.

6.3 Construction

The TOFE is motivated by the ordinary OFE by [BGLS03]. Indeed, when $t = n = 1$, the construction degenerates to their scheme.

Common Reference String The construction works in the common reference string model. For a security parameter 1^k, let G, G_T be cyclic groups of prime order p with g as a generator of G, where p is a k-bit prime. Further, let $e : G \times G \to G_T$ be a bilinear map. The common reference string is defined to be

$$\mathsf{param}_{CRS} := (1^k, G, G_T, p, g, e).$$

AGen On input param_{CRS}, the arbitrator picks at random $y \in_R \mathbb{Z}_p$ and computes $Y = g^y$. The public key and secret key of the arbitrator is defined as

$$(\mathsf{pk}_A, \mathsf{sk}_A) := \big(Y, y\big).$$

UGen On input param_{CRS}, the required number of signers n and the threshold t, the user picks at random a polynomial of degree $t-1$ in $\mathbb{Z}_p$, say f. Assume the signers are indexed by i, for $i = 1$ to n, with $n \geq t \geq 1$. The user further picks a hash function $H : \{0,1\}^* \to G$. Note that H is to be modelled as a random oracle.

For $i = 1$ to n, the secret signing key of signer i is defined as $f(i)$.

The user computes the public key as

$$\mathsf{pk}_U := (H, X, X_1, \ldots, X_n) := (H, g^{f(0)}, g^{f(1)}, \ldots, g^{f(n)}).$$

The value $f(0)$, which is the actual master secret, should be deleted. This ensures only a set of t signers together could create a threshold signature.

PSign The partial signature generation process consists of three sub-algorithms.

- *Generation of a Partial Signature Share* On input param_{CRS}, pk_A, a message M and the signing key of signer i $f(i)$, signer i randomly picks $r_i \in_R \mathbb{Z}_p$ and outputs the partial signature share as

$$\hat{\sigma}_i := (\alpha_i, \beta_i) := \big(H(M)^{f(i)} Y^{r_i}, g^{r_i}\big).$$

- *Verification of a Partial Signature Share* The partial signature share $\hat{\sigma}_i$ can be verified by evaluating the following relation:

$$e(\alpha_i, g) \stackrel{?}{=} e(H(M), X_i) e(Y, \beta_i).$$

- *Generation of a Partial Signature* When t partial signature shares, say, $\hat{\sigma}_i$ for $i \in \mathcal{I} \subset [n]$ such that $|\mathcal{I}| = t$ on the same message, say M, have been collected, anyone can output the partial signature on message M as:

$$\hat{\sigma} := (\alpha, \beta) := (\prod_{i \in \mathcal{I}} \alpha_i^{\lambda_i}, \prod_{i \in \mathcal{I}} \beta_i^{\lambda_i}).$$

  where λ_i is defined as

$$\lambda_i := \prod_{j \in \mathcal{I} \setminus \{i\}} \frac{-j}{i - j}.$$

  As discussed, $f(0) = \sum_{i \in \mathcal{I}} f(i) \lambda_i$.

PVer On input param_{CRS}, pk_A, pk_U, a message M and a partial signature $\hat{\sigma}$, the algorithm outputs `valid` if and only if the following equality holds:

$$e(\alpha, g) = e\big(H(M), X\big) e(Y, \beta).$$

Sign The full signature generation process consists of three sub-algorithms as well.

- *Generation of a Signature Share* On input param_{CRS}, pk_A, a message M and the signing key of signer i $f(i)$, signer i outputs the signature share as

$$\sigma_i := H(M)^{f(i)}.$$

- *Verification of a Signature Share* The signature share σ_i can be verified by evaluating the following relation:

$$e(\sigma_i, g) \stackrel{?}{=} e(H(M), X_i).$$

- *Generation of a Signature* When t signature shares, say, σ_i for $i \in \mathcal{I} \subset [n]$ such that $|\mathcal{I}| = t$ on the same message, say M, have been collected, anyone can output the signature on message M as:

$$\sigma := \prod_{i \in \mathcal{I}} \sigma_i^{\lambda_i}$$

where λ_i is defined as

$$\lambda_i := \prod_{j \in \mathcal{I} \setminus \{i\}} \frac{-j}{i-j}.$$

Ver On input param_{CRS}, pk_A, pk_U, a message M and a signature σ, the algorithm outputs `valid` if and only if the following equality holds:

$$e(\sigma, g) = e(H(M), X).$$

Res On input param_{CRS}, pk_A, pk_U, a message M, a partial signature $\hat{\sigma}$ and the secret key of the arbitrator y, the full signature can be computed as follows.

- Check that $\hat{\sigma}$ is a valid partial signature by evaluating the relation

$$e(\alpha, g) \stackrel{?}{=} e(H(M), X) e(Y, \beta).$$

- Output σ as

$$\sigma := \alpha / \beta^y.$$

Regarding the security of the construction of TOFE, the following theorem is presented.

Theorem 6.1 *The construction of TOFE is secure against signers, verifiers and the arbitrator under the CDH assumption in the random oracle model.*

Proof: Security against signers. Given a valid partial signature $\hat{\sigma}^* := (\alpha^*, \beta^*)$ on message M^*, such that

$$e(\alpha^*, g) = e(H(M^*), X)e(Y, \beta^*),$$

the resolved signature σ is defined as $\alpha^*/(\beta^*)^y$ where $Y = g^y$.

Note that

$$e(\sigma, g) = \frac{e(\alpha^*, g)}{e((\beta^*)^y, g)} = \frac{e(H(M^*), X)e(Y, \beta^*)}{e((\beta^*), g^y)} = e(H(M^*), X),$$

any valid partial signature will always be resolved to a valid full signature.

Security against verifiers. Suppose the final output of $\mathcal{A}$ is (M^*, σ^*). If $\mathcal{A}$ has not made a $\mathsf{PSign}_{(s)}$ query with input $(M^*, \cdot)$, the analysis of this type of attack is covered in the security against the arbitrator to be discussed later. Thus without loss of generality, it is safely assumed that $\mathcal{A}$ has made a $\mathsf{PSign}_{(s)}$ query with input $(M^*, \cdot)$. In this setting, it is shown how to construct a simulator $\mathcal{S}$ that is given $A = g^a$, $B = g^b$ and tries to solve the CDH problem by outputting g^{ab}.

Initialization $\mathcal{A}$ specifies the number of signers n and the threshold t, together with an index set $\mathcal{I}' \subset [n]$ such that $|\mathcal{I}'| = t-1$. $\mathcal{S}$ sets the common reference string param_{CRS}, $\mathsf{pk}_A = B^y$ for some randomly picked $y \in_R \mathbb{Z}_p$. For each $i \in \mathcal{I}'$, $\mathcal{S}$ picks $s_i \in_R \mathbb{Z}_p$ and computes $X_i = g^{s_i}$. $\mathcal{S}$ sets $X = g^a$. Consider a degree $t-1$ polynomial $f(x)$ such that $f(0) = a$ and $f(i) = s_i$ for $i \in \mathcal{I}'$. Note that the set of points $(0, a) \cup \{(i, s_i)\}_{i \in \mathcal{I}'}$ uniquely determines this polynomial yet the coefficients are unknown to $\mathcal{S}$. However, $\mathcal{S}$ can still compute $X_i = g^{f(i)}$ for $i \in [n] \setminus \mathcal{J}$ where $\mathcal{J} := 0 \cup \mathcal{I}'$ using the Lagrange polynomial interpolation technique discussed in Section 2.2.1. Specifically, for $i \in [n] \setminus \mathcal{J}$,

$$g^{f(i)} = g^{\sum_{j \in \mathcal{J}} f(j)\lambda_j(i)} = \prod_{j \in \mathcal{J}} \left(g^{f(j)}\right)^{\lambda_j(i)}.$$

Note that both $\lambda_j(i)$ and $g^{f(j)}$ for all $j \in \mathcal{J}$ are computable by $\mathcal{S}$ and thus $\mathcal{S}$ can compute $X_i = g^{f(i)}$ for all $i = 1$ to n. $\mathcal{S}$ also specifies the random oracle H. pk_U is set to be $(H, X, X_1, \ldots, X_n)$. $\mathcal{S}$ gives $(\mathsf{param}_{CRS}, \mathsf{pk}_A, \mathsf{pk}_U, \{s_i\}_{i \in \mathcal{I}'})$ to $\mathcal{A}$.

Query $\mathcal{A}$ can adaptively issue the following query to $\mathcal{S}$.

- Random Oracle H Query. Suppose $\mathcal{A}$ makes q queries of this type. $\mathcal{S}$ picks an index $z \in [q]$ at random. For the h-th query, $\mathcal{A}$ submits a value M_h and is expecting the value of $H(M_h)$. If $h \neq z$, $\mathcal{S}$ replies with g^{d_h} for a random $d_h \in_R \mathbb{Z}_p$. For the z-th query, $\mathcal{S}$ replies with g^b.
- $\mathsf{PSign}_{(s)}$ Query. $\mathcal{A}$ gives (M, i) to $\mathcal{S}$. Then, $\mathcal{S}$ locates the random oracle H query for M. If there exists h such that $M = M_h$ and that $h \neq z$, $\mathcal{S}$ picks $r \in_R \mathbb{Z}_p$ at random and responds with $(\alpha, \beta) = (Y^r X_i^{d_h}, g^r)$. If M has not been queried, $\mathcal{S}$ makes such a random oracle query on input M. If $M = M_z$, $\mathcal{S}$ responds as follows.
  - Note that each X_i for $i \in [n] \setminus \mathcal{I}'$ is of the form $g^{u_i a + v_i}$ for some constant $u_i \neq 0$, v_i known by $\mathcal{S}$.
  - $\mathcal{S}$ computes r such that $u_i = -yr$.
  - $\mathcal{S}$ randomly picks $t_i \in_R \mathbb{Z}_p$, computes $\beta_i = (g^a)^r g^{t_i}$ and $\alpha_i = (g^b)^{v_i + y t_i}$ and returns (α_i, β_i) to $\mathcal{A}$.
- $\mathsf{Sign}_{(s)}$ Query. $\mathcal{A}$ gives (M, i) to $\mathcal{S}$. If $M = M_z$, $\mathcal{S}$ aborts. Otherwise, $\mathcal{S}$ can locate h such that $H(M) = g^{d_h}$. Next, $\mathcal{S}$ computes $\sigma_i = X_i^{d_h}$ and returns σ_i to $\mathcal{A}$.
- Res Query. $\mathcal{A}$ gives $(\hat{\sigma}, M)$ to $\mathcal{S}$. $\mathcal{S}$ first checks the validity of $\hat{\sigma}$ and proceeds if it is valid. Otherwise it returns $\perp$. Then, $\mathcal{S}$ locates the random oracle H query for M. If $M = M_z$, $\mathcal{S}$ aborts. Otherwise, there exists h such that $H(M) = g^{d_h}$. Next, $\mathcal{S}$ computes $\sigma = X^{d_h}$ and returns σ to $\mathcal{A}$.

End-Game $\mathcal{A}$ submits $(M^*, \hat{\sigma}^*)$. If $M^* \neq M_z$, $\mathcal{S}$ aborts. In the random oracle model, M must have been submitted as an input in the random oracle H-query. Thus, with probability $1/q$, $\mathcal{S}$ does not abort. $\mathcal{S}$ outputs σ as the solution to the CDH problem. Note that in order to win,

$$e(\sigma, g) = e\big(H(M^*), X\big).$$

It implies that $\sigma = g^{ab}$.

Security against the arbitrator. It will be shown any adversary $\mathcal{A}$ that breaks the security against the arbitrator can be converted into a simulator $\mathcal{S}$ that solves the CDH problem. $\mathcal{S}$ is given $A = g^a$, $B = g^b$ and its goal is to output g^{ab}.

Initialization $\mathcal{A}$ specifies the number of signers n and the threshold t, together with an index set $\mathcal{I}' \subset [n]$ such that $|\mathcal{I}'| = t-1$. $\mathcal{S}$ sets the common reference string param_{CRS}, $\mathsf{pk}_A = g^y$ for some randomly picked $y \in_R \mathbb{Z}_p$. For each $i \in \mathcal{I}'$, $\mathcal{S}$ picks $s_i \in_R \mathbb{Z}_p$ and computes $X_i = g^{s_i}$. $\mathcal{S}$ sets $X = g^a$. Consider a degree $t-1$ polynomial $f(x)$ such that $f(0) = a$ and $f(i) = s_i$ for $i \in \mathcal{I}'$. Note that the set of points $(0, a) \cup \{(i, s_i)\}_{i \in \mathcal{I}'}$ uniquely determines this polynomial yet the coefficients are unknown to $\mathcal{S}$. However, $\mathcal{S}$ can still compute $X_i = g^{f(i)}$ for $i \in [n] \setminus \mathcal{J}$ where $\mathcal{J} := 0 \cup \mathcal{I}'$ as

$$g^{f(i)} = g^{\sum_{j \in \mathcal{J}} f(j)\lambda_j(i)} = \prod_{j \in \mathcal{J}} \big(g^{f(j)}\big)^{\lambda_j(i)}.$$

$\mathcal{S}$ also specifies the random oracle H. pk_U is set to be $(H, X, X_1, \ldots, X_n)$. $\mathcal{S}$ gives $(\mathsf{param}_{CRS}, \mathsf{pk}_A, \mathsf{pk}_U, \{s_i\}_{i \in \mathcal{I}'}, y)$ to $\mathcal{A}$.

Query $\mathcal{A}$ can adaptively issue the following query to $\mathcal{S}$.

- Random Oracle H Query. Suppose $\mathcal{A}$ makes q queries of this type. $\mathcal{S}$ picks an index $z \in [q]$ at random. For the h-th query, $\mathcal{A}$ submits a value M_h and is expecting the value of $H(M_h)$. If $h \neq z$, $\mathcal{S}$ replies with g^{d_h} for a random $d_h \in_R \mathbb{Z}_p$. For the z-th query, $\mathcal{S}$ replies with g^b.
- $\mathsf{PSign}_{(s)}$ Query. $\mathcal{A}$ gives (M, i) to $\mathcal{S}$. Then, $\mathcal{S}$ locates the random oracle H query for M. If there exists h such that $M = M_h$ and that $h \neq z$, $\mathcal{S}$ picks $r \in_R \mathbb{Z}_p$ at random and responds with $(\alpha, \beta) = (Y^r X_i^{d_h}, g^r)$. If M has not been queried, $\mathcal{S}$ makes such a random oracle query on input M. If $M = M_z$, $\mathcal{S}$ aborts.
- $\mathsf{Sign}_{(s)}$ Query. $\mathcal{A}$ gives (M, i) to $\mathcal{S}$. If $M = M_z$, $\mathcal{S}$ aborts. Otherwise, $\mathcal{S}$ can locate h such that $H(M) = g^{d_h}$. Next, $\mathcal{S}$ computes $\sigma_i = X_i^{d_h}$ and returns σ_i to $\mathcal{A}$.

End-Game $\mathcal{A}$ submits $(M^*, \hat{\sigma}^*)$. If $M^* \neq M_z$, $\mathcal{S}$ aborts. In the random oracle model, M must have been submitted as an input in the random oracle H-query. Thus, with probability $1/q$, $\mathcal{S}$ does not abort. $\mathcal{S}$ outputs σ as the solution to the CDH problem. Note that in order to win,

$$e(\sigma, g) = e\big(H(M^*), X\big).$$

It implies that $\sigma = g^{ab}$.

This completes the proof of the theorem. □

6.4 Chapter Summary

In this chapter the first threshold-oriented fair exchange protocol which allows a subset of signers to exchange a digital item with a counter party was presented. Indeed, in the specific construction, the item being exchanged is a threshold signature. Formal security model for TOFE was defined, an efficient construction was present and it was shown that it is secure in the random oracle model under well-known assumptions. Construction of TOFE in the standard model is left as an open problem. The construction of TOFE secure in a non-static model is left as the future work.

Chapter 7

Perfect Ambiguous Optimistic Fair Exchange

In this chapter a new notion named perfect ambiguous optimistic fair exchange (PAOFE) is proposed to prevent the premature leakage of information about an exchange to take place. The new notion can be viewed as an extension of ambiguous optimistic fair exchenge (AOFE) with a new property *perfect anonymity*.

7.1 Introduction

Consider a scenario in which Apple engages Intel in a fair exchange protocol to sign a contract that pays an amount of money for the early termination of the use of Intel technology in the next generation of Macbook and iMac desktop computers. In this situation, reveal of the will of signing this contract, prior to the final signing date will be potentially harmful to the companies. For instance, Apple may be reluctant to expose prematurely the changes it is introducing to its next generation products, which may possibly affect the sales of the current generation of the products. On the other hand, the potential termination of cooperation with Apple may lead to a decline of Intel's shares value. Therefore, it is necessary that the fair exchange of signatures protocol should not leak information prematurely before both parties agree on the exchange.

To the best of my knowledge, ambiguous optimistic fair exchange [HYWS08a] is the closest cryptographic solution to the above problem. An AOFE protocol comprises three parties, namely, signer Alice, verifier Bob, and a semi-trusted third party known as the "arbitrator". In a typically execution of an AOFE protocol, Alice delivers a "commitment" of her signature, called ambiguous partial signature, to Bob. Upon successful verification of the ambiguous partial signature, Bob delivers his full signature to Alice. After verifying the full signature from Bob, Alice sends

to Bob her own full signature. This completes the protocol.

Bob can approach the arbitrator for assistance in the situation in which Alice refuses to send her full signature at the end of the exchange protocol. The ambiguous partial signature is designed in such a way that the arbitrator can turn it into Alice's full signature, which is indistinguishable to a "real" signature created by Alice. In this way, as long as the arbitrator is trusted to carry out its duty, Bob can always be assured he can obtain a full signature from Alice, either from Alice or the arbitrator. In addition, the arbitrator is not required to take part in typical executions of the protocol.

AOFE differs from traditional optimistic fair exchange (OFE) schemes, for example [ASW97, Mic03, BGLS03, DR03, Wan05, ZM07, DLY07, HYWS08b, HV11], in the sense that the ambiguous partial signature does not reveal the exact identity of its creator. Specifically, in OFE, everyone can verify that Alice has created a commitment of her signature in the first step. This may create an unfair situation to Alice as Bob can simply use Alice's commitment as a mean to his advantage. For instance, if Alice's signature represents her contract tender for Bob's service, Bob can use Alice's commitment as a way to ask for a higher price from another party. On the other hand, the ambiguous partial signature in AFOE has the extra property that it can be created by either Alice or Bob. Thus, while Bob can be assured that this is Alice's commitment of her signature, he cannot convince anybody that this is Alice's commitment since from an outsider's view Bob could have been the creator of the ambiguous partial signature as well. Nonetheless, in AOFE, the arbitrator knows who is the creator of the ambiguous signature.

Unfortunately, AOFE is inadequate to the aforementioned problem raised earlier. If AOFE is employed in the above scenario, Apple will transmit the ambiguous partial signature to Intel on the contract of the termination of the use of Intel technology in its next generation of computers as the first step of the exchange. This ambiguous partial signature itself leaks sufficient information to be valuable. The reason is that in this scenario, it does not matter who is the signer of this contract. The valuable information to an outsider is that these two companies are discussing about a potential termination, which is the partial signature. The ambiguous partial signature created by Apple or Intel is sufficient evidence to prove the authenticity of the information.

One key observation about the existing exchange protocol is that the ambiguous partial signature in AOFE, as well as the regular partial signature in OFE, is

publicly verifiable. This is not strictly a necessary functional requirement of an exchange protocol. In fact, this may have an undesirable effect as illustrated in the case earlier. In general, if Bob is known to be trustworthy, for example, if Bob is a government department, then malicious observer Oven who obtains an ambiguous partial signature submitted to Bob knows the intention of Alice. Besides, the observation is made that the arbitrator in AOFE knows who the creator of an ambiguous partial signature is, and is capable of converting it into a full signature. A high level of trust has to be placed on the arbitrator.

Hence, a new notion, called Perfect Ambiguous Optimistic Fair Exchange (PAOFE), is introduced as a practical cryptographic solution to the aforementioned scenario. Indeed, the solution builds on top of AOFE and it also fulfills all the security requirements of an AOFE. In addition, PAOFE enjoys a new property called *Perfect Ambiguity* in which the equivalent of an "ambiguous partial signature" leaks no information about the actual signer, intended recipient and the signature itself, and not even in the view of the arbitrator. Thus, no outsider can tell if an exchange is in progress.

7.1.1 The Contributions

In this chapter the following contributions are maked.

1. The notion of Perfect Ambiguous Optimistic Fair Exchange is proposed, which allows a signer Alice to generate a partial signature in such a way that no outsider, not even the arbitrator, is able to infer any useful information about the signature. Indeed, a partial signature in PAOFE generated by the signer Alice with Bob being the receiver is indistinguishable to a random bit string chosen from the signature space. In other words, any partial signature is indistinguishable from a partial signature on a random message with respect to a random signer and a random receiver. To realize this notion, Bob's secret key is required in the verification of the partial signature in PAOFE. Thus, only Bob is able to verify the partial signature, and an outsider gains nothing about the transaction. Both the identities of the signer and receiver and the content of an transaction are perfectly hidden.

2. A security model for PAOFE in the multi-user setting under chosen-key attack is defined. The model captures the existing security requirements for AOFE,

namely, signer ambiguity, resolution ambiguity, security against signers, security against verifiers and security against the arbitrator. In addition, PAOFE covers an additional requirement: *perfect ambiguity*. It is required that any user can generate a partial signature whose distribution is indistinguishable from that of a partial signatures generated by Alice. In other words, a specific partial signature generated by Alice with recipient Bob is indistinguishable from a partial signature uniformly randomly chosen from the whole signature space.

3. A generic construction of PAOFE is proposed from two well established cryptographic primitives, namely, AOFE and key-private encryption and provide the security proof of the proposal in the proposed model. The generic construction works in the standard model and does not involve any extra assumptions.

7.1.2 Chapter Organization

In the next section, a notion for public-key encryption that guarantees the data-privacy and key-privacy simultaneously is proposed. In Section 7.3, a formal definition of PAOFE, together with the security model in the multi-user and chosen key setting is proposed. Then, a generic construction of PAOFE and the security proof of the scheme under the model in Section 7.4 is provided. Finally, the chapter is summarised in Section 7.6.

7.2 A new Encryption Notion

One building block that will be used in this chapter is the public-key encryption schemes. For a public-key encryption scheme $\mathcal{E} = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$, it is known that IND-CCA2-security captures the privacy of data, and IK-CCA2-security which was reviewed in Definition 2.15 captures privacy of user's key. Though the goals of data-privacy and key-privacy are orthogonal, it is very desirable, from a practical point of view, that an encryption scheme satisfies both sides. To guarantee both the message-privacy and key-privacy properties at the same time, the existing notions IND-CCA2-security and IK-CCA2-security are combined into one. Formally, consider the following experiment conducted between a challenger $\mathcal{C}$ and an the adversary $\mathcal{A}$.

Setup : The challenger $\mathcal{C}$ runs $\mathsf{KGen}(1^k)$ to generate a decryption/encryption key pair (dk, ek), gives the encryption key ek to the adversary $\mathcal{A}$, and keeps dk as

private.

Phase 1 Queries : $\mathcal{A}$ can adaptively make a polynomial number of decryption queries to $\mathcal{C}$. For each query, $\mathcal{A}$ submits a ciphertext c to $\mathcal{A}$, who returns a message m outputted by $\mathsf{Dec}(dk, c)$.

Challenge phase : $\mathcal{A}$ submits a messages m^*. $\mathcal{C}$ randomly flips a coin $b \in \{0, 1\}$. If $b = 0$ and computes $c_0 = \mathsf{Enc}(ek, m^*)$. If $b = 1$, $\mathcal{C}$ uniformly at random chooses a ciphertext from the whole ciphertext space with respect to any message and any public key. $\mathcal{C}$ returns c_b to $\mathcal{A}$.

Phase 2 Queries : $\mathcal{A}$ can further adaptively make a polynomial number of decryption queries to $\mathcal{C}$ with the only limitation that the challenge ciphertext c_b should not be queried. For each query, $\mathcal{A}$ submits a ciphertext c to $\mathcal{A}$, who returns a message m outputted by $\mathsf{Dec}(dk, c)$.

Guess : $\mathcal{A}$ outputs a guess $b' \in \{0, 1\}$. $\mathcal{A}$ wins the game if $b' = b$.

Definition 7.1 *A public-key encryption scheme is* IND-IK-CCA2*-secure if no PPT adversary wins the above experiment with non-negligible probability.*

By using the hybrid technique described in [Gol00], it is easy to see that any public key encryption scheme that is both IND-CCA2 secure and IK-CCA2 secure will be IND-IK-CCA2 secure. Since Cramer-Shoup encryption scheme [CS98] is both IND-CCA2 secure and IK-CCA2 secure [BBDP01], it is IND-IK-CCA2 secure.

7.3 Perfect Ambiguous Optimistic Fair Exchange

In a PAOFE scheme, it is required that given a partial signature, no outsider should be able to learn any information about it. Specifically, the message on which the partial signature was generated, in addition to the identities of both the signer and the receiver should be completely hidden. To achieve this, the verification algorithm in PAOFE is required to involve the secret key of the receiver, rather than the case that the partial signature is publicly verifiable in AOFE. Besides, the resolution algorithm in AOFE is extended to the resolution protocol in PAOFE. Since an algorithm can be seen as a non-interactive protocol, the model is more general and could capture a larger class of schemes.

Definition 7.2 *A perfect ambiguous optimistic fair exchange scheme involves the users (signers and verifiers) and the arbitrator, and consists of the following (probabilistic) polynomial-time algorithms/protocols:*

- **PMGen**: *On input* 1^k *where* k *is a security parameter, this algorithm outputs a system parameter* PM.

- $\mathbf{Setup}^{\mathsf{TTP}}$: *On input* PM, *the algorithm generates a secret key* ASK, *and a public key* APK *of the arbitrator.*

- $\mathbf{Setup}^{\mathsf{User}}$: *On input* PM *and (optionally)* APK, *it outputs a secret/public key pair* $(\mathsf{SK}, \mathsf{PK})$. *For a user* U_i, $(\mathsf{SK}_i, \mathsf{PK}_i)$ *is used to denote the user's key pair.*

- **Sig** *and* **Ver:** $\mathbf{Sig}(M, \mathsf{SK}_i, \mathsf{PK}_i, \mathsf{PK}_j, \mathsf{APK})$, *outputs a (full) signature* σ *on message* M *of user* U_i *with the designated verifier* U_j, *while* $\mathbf{Ver}(M, \sigma, \mathsf{PK}_i, \mathsf{PK}_j, \mathsf{APK})$ *outputs* $\top$ *or* $\perp$, *indicating* σ *is* U_i*'s valid full signature on* M *with the designated verifier* U_j *or not.*

- **PSig** *and* **PVer:** *These are partial signing and verification algorithms respectively.* $\mathbf{PSig}(M, \mathsf{SK}_i, \mathsf{PK}_i, \mathsf{PK}_j, \mathsf{APK})$, *run by a signer* U_i, *outputs a partial signature* σ_P, *while* $\mathbf{PVer}(M, \sigma_P, \mathsf{SK}_j, \mathsf{PK}_i, \mathsf{PK}_j, \mathsf{APK})$, *run by a verifier* U_j, *outputs* $\top$ *or* $\perp$.

- **Res:** *This is a resolution protocol between the verifier* U_j *and the arbitrator, involving a pair of interactive algorithms* $(\mathsf{Res}_{\mathcal{V}}, \mathsf{Res}_{\mathcal{T}})$. $\mathsf{Res}_{\mathcal{V}}(M, \sigma_P, \mathsf{SK}_j, \mathsf{PK}_i, \mathsf{PK}_j, \mathsf{APK})$, *run by the verifier, outputs a full signature* σ, *or* $\perp$ *indicating the failure of resolving a partial signature.*

Resolution ambiguity property states that any "resolved signature" $\mathsf{Res}_{\mathcal{V}}(M, \mathbf{PSig}(M, \mathsf{SK}_i, \mathsf{PK}_i, \mathsf{PK}_j, \mathsf{APK}), \mathsf{SK}_j, \mathsf{PK}_i, \mathsf{PK}_j, \mathsf{APK})$ *is computationally indistinguishable from the "actual signature"* $\mathbf{Sig}(M, \mathsf{SK}_i, \mathsf{PK}_i, \mathsf{PK}_j, \mathsf{APK})$.

7.3.1 PAOFE Models

- **Perfect ambiguity**: Intuitively, it is required that no outsiders, even the arbitrator, should be able to learn any information about a partial signature such as the content of the message or the identities of the signer and receiver. This ensures the privacy for both the signer and the receiver. To achieve this property, it is required that in the view of an outsider, the partial signature is indistinguishable to a signature randomly sampled from the signature space. Since the verifier is able to forge a partial signature, here it is needed to provide the dishonest signer a fake partial signing oracle, which outputs a fake partial

signature forged by the verifier. Formally, it is required no PPT distinguisher $\mathcal{A}$ succeeds with non-negligible probability in the following experiment:

$$
\begin{aligned}
\mathsf{PM} &\leftarrow \mathbf{PMGen}(1^k) \\
(\mathsf{APK}, \mathsf{ASK}^*) &\leftarrow \mathcal{A}(\mathsf{PM}) \\
(\mathsf{SK}_B, \mathsf{PK}_B) &\leftarrow \mathbf{Setup}^{\mathsf{User}}(\mathsf{PM}, \mathsf{APK}) \\
(M, (\mathsf{SK}_A, \mathsf{PK}_A), \Upsilon) &\leftarrow \mathcal{A}^{O^B_{\mathbf{PSig}}, O^B_{\mathbf{FakePSig}}, O^B_{\mathbf{PVer}}}(\mathsf{ASK}^*, \mathsf{APK}, \mathsf{PK}_B) \\
b &\leftarrow \{0,1\} \\
\sigma_P &\leftarrow \begin{cases} \mathbf{PSig}(M, \mathsf{SK}_A, \mathsf{PK}_A, \mathsf{PK}_B, \mathsf{APK}) \text{ if } b = 0 \\ \sigma'_P \leftarrow \mathcal{S} \text{ if } b = 1 \end{cases} \\
b' &\leftarrow \mathcal{A}^{O^B_{\mathbf{PSig}}, O^B_{\mathbf{FakePSig}}, O^B_{\mathbf{PVer}}}(\sigma_P, \Upsilon) \\
\text{success of } \mathcal{A} &:= [b' = b \\
&\quad \wedge (M, \sigma_P, \mathsf{PK}_A) \notin \mathit{Query}(\mathcal{A}, O^B_{\mathbf{PVer}})]
\end{aligned}
$$

where Υ is $\mathcal{A}$'s state information, $\mathcal{S}$ is the whole partial signature space, oracle $O^B_{\mathbf{PSig}}$ takes as input (M, PK_j) and outputs a partial signature of PK_B's on M with the receiver's public key being PK_j, oracle $O^B_{\mathbf{FakePSig}}$ takes as input (M, PK_i) and returns a fake partial signature of user U_i's generated using SK_B on M with the receiver's public key being PK_B[1], oracle $O^B_{\mathbf{PVer}}$ takes as input a partial signature σ_P of user PK_i's on message M with the verifier being PK_B, i.e., $(M, \sigma_P, \mathsf{PK}_i)$, and outputs $\top$ or $\bot$, and $\mathit{Query}(\mathcal{A}, O^B_{\mathbf{PVer}})$ is the set of queries $\mathcal{A}$ issued to oracle $O^B_{\mathbf{PVer}}$. Note that in previous ambiguous optimistic fair exchange models, the partial verification oracle $O^B_{\mathbf{PVer}}$ was not provided, as a partial signature is publicly verifiable. To cope with the change in PAOFE that partial signature is no longer publicly verifiable, a partial signature verification oracle is provided to the adversary in the security model.

[1]The key pair $(\mathsf{SK}_B, \mathsf{PK}_B)$ involved in this oracle $O^B_{\mathbf{FakePSig}}$ is the key pair generated in the third step of this experiment.

- **Signer Ambiguity**: Informally, *signer ambiguity* means that B may forge partial signatures that look indistinguishable from those generated by A. Formally, it is required no PPT distinguisher $\mathcal{A}$ succeeds with non-negligible probability in the following experiment:

$$\begin{aligned}
\mathsf{PM} &\leftarrow \mathbf{PMGen}(1^k) \\
(\mathsf{ASK}, \mathsf{APK}) &\leftarrow \mathbf{Setup}^{\mathsf{TTP}}(\mathsf{PM}) \\
(M, (\mathsf{SK}_0, \mathsf{PK}_0), (\mathsf{SK}_1, \mathsf{PK}_1), \Upsilon) &\leftarrow \mathcal{A}^{O_{\mathbf{Res}}}(\mathsf{APK}) \\
b &\leftarrow \{0,1\} \\
\sigma_P &\leftarrow \begin{cases} \mathbf{PSig}(M, \mathsf{SK}_0, \mathsf{PK}_0, \mathsf{PK}_1, \mathsf{APK}), & b = 0 \\ \mathbf{FakePSig}(M, \mathsf{SK}_1, \mathsf{PK}_0, \mathsf{PK}_1, \mathsf{APK}), & b = 1 \end{cases} \\
b' &\leftarrow \mathcal{A}^{O_{\mathbf{Res}}}(\sigma_P, \Upsilon) \\
\text{success of } \mathcal{A} &:= [b' = b \\
&\quad \wedge (M, \mathsf{PK}_0, \mathsf{PK}_1) \notin \mathit{Query}(\mathcal{A}, O_{\mathbf{Res}})
\end{aligned}$$

  where Υ is $\mathcal{A}$'s state information, oracle $O_{\mathbf{Res}}$ takes an input $(M, \mathsf{PK}_i, \mathsf{PK}_j)$ and starts an execution of the **Res** protocol with the adversary running the interactive algorithm $\mathsf{Res}_{\mathcal{R}}$, algorithm **FakePSig** is a fake partial signature signing algorithm and $\mathbf{FakeSig}(M, \mathsf{SK}_j, \mathsf{PK}_i, \mathsf{PK}_j, \mathsf{APK})$ outputs a forged partial signature σ_P on M of user U_i with the designated verifier U_j generated using SK_j, and $\mathit{Query}(\mathcal{A}, O_{\mathbf{Res}})$ is the set of queries $\mathcal{A}$ issued to the resolution oracle $O_{\mathbf{Res}}$.

- **Security Against Signers**: It is required that any PPT adversary $\mathcal{A}$, who models a dishonest signer, succeeds with at most negligible probability in the

following experiment:

$$\begin{aligned}
\mathsf{PM} &\leftarrow \mathbf{PMGen}(1^k)\\
(\mathsf{ASK},\mathsf{APK}) &\leftarrow \mathbf{Setup}^{\mathsf{TTP}}(\mathsf{PM})\\
(\mathsf{SK}_B,\mathsf{PK}_B) &\leftarrow \mathbf{Setup}^{\mathsf{User}}(\mathsf{PM},\mathsf{APK})\\
(M,\sigma_P,\mathsf{PK}_A) &\leftarrow \mathcal{A}^{O^B_{\mathbf{PSig}},O^B_{\mathbf{FakePSig}},O^B_{\mathbf{PVer}},O_{\mathbf{Res}}}(\mathsf{APK},\mathsf{PK}_B)\\
\mathsf{Input}_{\mathcal{T}} &:= (M,\mathsf{ASK},\mathsf{PK}_A,\mathsf{PK}_B)\\
\mathsf{Input}_{\mathcal{V}} &:= (M,\sigma_P,\mathsf{SK}_B,\mathsf{PK}_A,\mathsf{PK}_B,\mathsf{APK})\\
[\mathsf{Res}_{\mathcal{T}}(\mathsf{Input}_{\mathcal{T}})\rightarrow \mathsf{state}_{\mathcal{T}}] &\overset{\mathbf{Res}}{\Longleftrightarrow} [\mathsf{Res}_{\mathcal{V}}(\mathsf{Input}_{\mathcal{V}})\rightarrow \sigma]\\
\text{success of } \mathcal{A} &:= [\mathbf{PVer}(M,\sigma_P,\mathsf{SK}_B,\mathsf{PK}_A,\mathsf{PK}_B,\mathsf{APK})=\top\\
&\quad \wedge\ \mathbf{Ver}(M,\sigma,\mathsf{PK}_A,\mathsf{PK}_B,\mathsf{APK})=\bot\\
&\quad \wedge\ (M,\mathsf{PK}_A)\notin Query(\mathcal{A},O^B_{\mathbf{FakePSig}})]
\end{aligned}$$

where all the four oracles are described in the previous experiments, $Query(\mathcal{A}, O^B_{\mathbf{FakePSig}})$ is the set of queries made by $\mathcal{A}$ to oracle $O^B_{\mathbf{FakePSig}}$. Note that the adversary is not allowed to corrupt PK_B, otherwise it can easily success in the experiment by simply using SK_B to produce a fake partial signature under public keys PK_A, PK_B and outputting it.

- **Security Against Verifiers**: It is required that any PPT adversary $\mathcal{A}$, who models a dishonest verifier, succeeds with at most negligible probability in the following experiment:

$$\begin{aligned}
\mathsf{PM} &\leftarrow \mathbf{PMGen}(1^k)\\
(\mathsf{ASK},\mathsf{APK}) &\leftarrow \mathbf{Setup}^{\mathsf{TTP}}(\mathsf{PM})\\
(\mathsf{SK}_A,\mathsf{PK}_A) &\leftarrow \mathbf{Setup}^{\mathsf{User}}(\mathsf{PM},\mathsf{APK})\\
(M,\sigma,\mathsf{PK}_B) &\leftarrow \mathcal{A}^{O_{\mathbf{PSig}},O_{\mathbf{FakePSig}},O_{\mathbf{PVer}},O_{\mathbf{Res}}}(\mathsf{APK},\mathsf{PK}_A)\\
\text{success of } \mathcal{A} &:= [\mathbf{Ver}(M,\sigma,\mathsf{PK}_A,\mathsf{PK}_B,\mathsf{APK})=\top\\
&\quad \wedge\ (M,\mathsf{PK}_A,\mathsf{PK}_B)\notin Query(\mathcal{A},O_{\mathbf{Res}})]
\end{aligned}$$

where oracle $O_{\mathbf{Res}}$ is described in the previous experiments, oracle $O_{\mathbf{PSig}}$ takes as input (M,PK_j) and outputs a partial signature of PK_A's on M with the receiver's public key being PK_j generated using SK_A, oracle $O_{\mathbf{FakePSig}}$ takes as input (M,PK_i) and returns a fake partial signature of user U_i's generated using SK_A on M with the receiver's public key being PK_A, oracle $O_{\mathbf{PVer}}$ takes as input

a partial signature σ_P of user U_i's on message M with the receiver's public key being PK_A, i.e., $(M, \sigma_P, \mathsf{PK}_i)$, and outputs $\top$ or $\bot$, and $Query(\mathcal{A}, O_{\mathbf{Res}})$ is the set of queries $\mathcal{A}$ issued to the resolution oracle.

- **Security Against the Arbitrator**: It is required that any PPT adversary $\mathcal{A}$, who models a dishonest arbitrator, succeeds with at most negligible probability in the following experiment:

$$\begin{aligned}
\mathsf{PM} &\leftarrow \mathbf{PMGen}(1^k) \\
(\mathsf{APK}, \mathsf{ASK}^*) &\leftarrow \mathcal{A}(\mathsf{PM}) \\
(\mathsf{SK}_A, \mathsf{PK}_A) &\leftarrow \mathbf{Setup}^{\mathsf{User}}(\mathsf{PM}, \mathsf{APK}) \\
(M, \sigma, \mathsf{PK}_B) &\leftarrow \mathcal{A}^{O_{\mathbf{PSig}}, O_{\mathbf{FakePSig}}, O_{\mathbf{PVer}}}(\mathsf{ASK}^*, \mathsf{APK}, \mathsf{PK}_A) \\
\text{success of } \mathcal{A} &:= [\mathbf{Ver}(M, \sigma, \mathsf{PK}_A, \mathsf{PK}_B, \mathsf{APK}) = \top \\
&\quad \wedge (M, \mathsf{PK}_B) \notin Query(\mathcal{A}, O_{\mathbf{PSig}})]
\end{aligned}$$

  where all the three oracles are described in the previous experiment, ASK^* is $\mathcal{A}$'s state information, which might not be the corresponding secret key of APK, and $Query(\mathcal{A}, O_{\mathbf{PSig}})$ is the set of queries $\mathcal{A}$ issued to oracle $O_{\mathbf{PSig}}$.

7.4 Generic Construction

In this section, a generic construction of PAOFE is presented. Let $\Gamma = (\mathsf{PMGen}, \mathsf{Setup}^{\mathsf{TTP}}, \mathsf{Setup}^{\mathsf{User}}, \mathsf{Sig}, \mathsf{Ver}, \mathsf{PSig}, \mathsf{PVer}, \mathsf{Res})$ be an ambiguous optimistic fair exchange scheme. Let $\mathcal{E} = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ be a public key encryption scheme.
A perfect ambiguous optimistic fair exchange can be constructed as follows:

- **PMGen**: This algorithm calls $\Gamma.\mathsf{PMGen}(1^k) \rightarrow PM$ where k is a security parameter, and outputs $\mathsf{PM} := PM$.

- **Setup**$^{\mathsf{TTP}}$: The arbitrator runs $\Gamma.\mathsf{Setup}^{\mathsf{TTP}}(\mathsf{PM}) \rightarrow (ASK, APK)$, and sets $(\mathsf{ASK}, \mathsf{APK}) := (ASK, APK)$.

- **Setup**$^{\mathsf{User}}$: Each user U_i runs $\Gamma.\mathsf{Setup}^{\mathsf{User}}(\mathsf{PM}, APK) \rightarrow (SK_i, PK_i)$ and $\mathcal{E}.\mathsf{KGen}(1^k) \rightarrow (ek_i, dk_i)$ respectively, and sets $(\mathsf{SK}_i, \mathsf{PK}_i) := ((SK_i, dk_i), (PK_i, ek_i))$.

- **PSig**: To partially sign a message M with the verifier U_j, U_i runs $\Gamma.\mathsf{PSig}(M || \mathsf{PK}_i || \mathsf{PK}_j, SK_i, PK_i, PK_j, APK) \rightarrow \sigma'_P$ and then encrypts it under U_j's

public encryption key ek_j by running $c = \mathcal{E}.\mathsf{Enc}(ek_j, \sigma'_P)$. The partial signature is set as $\sigma_P := c$.

- **PVer**: On receiving a partial signature σ_P on message M from the signer U_i, user U_j decrypts it using its own decryption key dk_j, i.e., $\sigma'_P = \mathcal{E}.\mathsf{Dec}(dk_j, \sigma_P)$, and then checks if $\Gamma.\mathsf{PVer}(M||\mathsf{PK}_i||\mathsf{PK}_j, \sigma'_P, PK_i, PK_j, APK) = \top$. If so, it accepts; otherwise, it rejects.

- **Sig**: To fully sign a message M for the verifier U_j, U_i calls $\Gamma.\mathsf{Sig}(M||\ \mathsf{PK}_i||\ \mathsf{PK}_j,\ SK_i,\ PK_i,\ PK_j,\ APK) \rightarrow \sigma$ and sends σ to U_j.

- **Ver**: On receiving a full signature σ from U_i, U_j outputs $\Gamma.\mathsf{Ver}(M||\ \mathsf{PK}_i||\ \mathsf{PK}_j,\ \sigma,\ PK_i,\ PK_j,\ APK)$.

- **Res**: Given a partial signature σ_P on message M from the signer U_i, user U_j decrypts it using its own decryption key dk_j, i.e., $\sigma'_P = \mathcal{E}.\mathsf{Dec}(dk_j, \sigma_P)$, and sends $(M, \sigma'_P, \mathsf{PK}_i, \mathsf{PK}_j)$ to the arbitrator.The arbitrator first checks the validity of σ'_P by running $\Gamma.\mathsf{PVer}(M||\ \mathsf{PK}_i||\ \mathsf{PK}_j,\ \sigma'_P,\ PK_i,\ PK_j,\ APK)$. If it's invalid, it returns $\perp$ to U_j. Otherwise, it returns $\Gamma.\mathsf{Res}(M||\ \mathsf{PK}_i||\ \mathsf{PK}_j,\ \sigma'_P,\ ASK,\ PK_i,\ PK_j)$ to U_j.

7.4.1 Security Analysis

Obviously the resolution ambiguity property in PAOFE follows from that in AOFE. The generic construction is secure according to the model in Section 7.3.1, guaranteed by the following theorems.

Theorem 7.1 *The generic construction is perfect ambiguous if* $\mathcal{E} = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ *is an* IND-IK-CCA2 *secure encryption.*

Proof. To show perfect ambiguity, one converts any adversary $\mathcal{A}$ that wins the experiment into an adversary $\mathcal{A}'$ that breaks the IND-IK-CCA2 security of $\mathcal{E}$. Recall that $\mathcal{A}'$ gets ek as input and has access to oracle O_{Dec}. Suppose the public parameter PM is generated. $\mathcal{A}$ first chooses a public adjudication key APK and outputs it, and keeps a corresponding secret state information ASK^* private. $\mathcal{A}'$ sets $PM :=$ PM, $APK :=$ APK, and runs $\Gamma.\mathsf{Setup}^{\mathsf{User}}(PM, APK) \rightarrow (SK_B, PK_B)$ and invokes $\mathcal{A}$ on input $\mathsf{PK}_B := (PK_B, ek)$.

Given a partial signature signing query $(M, \mathsf{PK}_j = (PK_j, ek_j))$ to oracle $O^B_{\mathbf{PSig}}$, $\mathcal{A}'$ runs $\Gamma.\mathsf{PSig}\ (M||\mathsf{PK}_B||\mathsf{PK}_j, SK_B, PK_B, PK_j, APK) \rightarrow \sigma'_P$, and then encrypts σ'_P under ek_j by running $c = \mathcal{E}.\mathsf{Enc}(ek_j, \sigma'_P)$. $\mathcal{A}'$ returns c to $\mathcal{A}$ as the answer.

Given a fake partial signing query (M, PK_i) where $\mathsf{PK}_i = (PK_i, ek_i)$ to oracle $O^B_{\mathbf{FakePSig}}$, $\mathcal{A}'$ runs $\Gamma.\mathsf{PSig}(M||\mathsf{PK}_i||\mathsf{PK}_B, SK_B, PK_B, PK_i, APK) \rightarrow \sigma'_P$, and then encrypts σ'_P under ek by running $c = \mathcal{E}.\mathsf{Enc}(ek, \sigma'_P)$. $\mathcal{A}'$ returns c to $\mathcal{A}$ as the answer.

Given a partial signature verification query $(M, \sigma_P, \mathsf{PK}_i = (PK_i, ek_i))$ to $O^B_{\mathbf{PVer}}$, $\mathcal{A}'$ makes a decryption query σ_P to its own oracle O_{Dec}. Denote the answer from O_{Dec} is σ'_P. $\mathcal{A}'$ returns $\Gamma.\mathsf{PVer}\ (M||\mathsf{PK}_i||\mathsf{PK}_B, \sigma'_P, PK_i,\ PK_B, APK)$ to $\mathcal{A}$.

At some time, $\mathcal{A}$ submits $(M^*, (\mathsf{SK}_A, \mathsf{PK}_A))$, where $\mathsf{SK}_A := (SK_A, dk_A)$, and SK_A matches PK_A. $\mathcal{A}'$ runs $\Gamma.\mathsf{PSig}(M^*||\mathsf{PK}_A||\mathsf{PK}_B, SK_A,\ PK_A, PK_B,\ APK) \rightarrow \sigma'_P$, and submits σ'_P to its own challenger, which returns a ciphertext c^*. $\mathcal{A}'$ forwards $\sigma_P := c^*$ to $\mathcal{A}$, and then continues to simulate the oracles $O^B_{\mathbf{PSig}}$ and $O^B_{\mathbf{FakePSig}}$ in the same way as above. About the further queries $(M, \sigma_p, \mathsf{PK}_i)$ where $\mathsf{PK}_i = (PK_i, ek_i)$ to oracle $O^B_{\mathbf{PVer}}$, the following two cases are distinguished:

1. $\sigma_P \neq c^*$. In this case, $\mathcal{A}'$ simulates $O^B_{\mathbf{PVer}}$ in the same way as above.

2. $\sigma_P = c^*$. In this case, $\mathcal{A}'$ just returns $\perp$ to the adversary A. First of all, one can exclude the subcase where $(M, c^*, \mathsf{PK}_i) = (M^*, c^*, \mathsf{PK}_A)$, because $(M^*, c^*, \mathsf{PK}_A)$ is prohibited from being queried to oracle $O^B_{\mathbf{PVer}}$. If $(M,\ c^*,\ \mathsf{PK}_i) \neq (M^*, c^*, \mathsf{PK}_A)$ and $\mathbf{PVer}(M,\ c^*,\ \mathsf{SK}_B,\ \mathsf{PK}_i,\ \mathsf{PK}_B,\ \mathsf{APK}) = \top$, the probability of this happening is negligible.

It can be seen that the oracles $O^B_{\mathbf{PSig}}$, $O^B_{\mathbf{FakePSig}}$ and $O^B_{\mathbf{PVer}}$ are simulated properly by $\mathcal{A}'$. Finally, $\mathcal{A}$ outputs a bit d. $\mathcal{A}'$ outputs a bit $b' = d$, and A' has never issued a query to its decryption oracle O_{Dec} on input c^*. If $\mathcal{A}$ succeeds in the experiment, $\mathcal{A}'$ also succeeds in outputting the bit b'. Therefore $\mathcal{A}'$'s advantage is also non-negligible. □

Theorem 7.2 *The generic construction is signer ambiguous if* $\Gamma = (\mathsf{PMGen}, \mathsf{Setup}^{\mathsf{TTP}}, \mathsf{Setup}^{\mathsf{User}}, \mathsf{Sig}, \mathsf{Ver}, \mathsf{PSig}, \mathsf{PVer}, \mathsf{Res})$ *is signer ambiguous.*

Proof. To show signer ambiguity, one converts any adversary $\mathcal{A}$ that wins the experiment into an adversary $\mathcal{A}'$ that breaks the signer ambiguity security of Γ. Recall that $\mathcal{A}'$ gets APK as input and has access to oracle O'_{Res}. Suppose the public parameter PM is generated. $\mathcal{A}'$ invokes $\mathcal{A}$ on input $\mathsf{APK} := APK$.

Given a resolution query $(M, \mathsf{PK}_i, \mathsf{PK}_j)$ where $\mathsf{PK}_i = (PK_i, ek_i)$ and $\mathsf{PK}_j = (PK_j, ek_j)$ to $O_{\mathbf{Res}}$, suppose $\mathcal{A}$ sends σ'_P to $\mathcal{A}'$ in the first run of the protocol. $\mathcal{A}'$ makes a query $(M||\mathsf{PK}_i||\mathsf{PK}_j, \sigma'_P, PK_i, PK_j)$ to its own oracle O'_{Res} and returns the answer to $\mathcal{A}$.

At some time, $\mathcal{A}$ submits $(M^*, (\mathsf{SK}_0, \mathsf{PK}_0), (\mathsf{SK}_1, \mathsf{PK}_1))$, where $\mathsf{SK}_0 := (SK_0, dk_0)$, $\mathsf{PK}_0 := (PK_0, ek_0)$, $\mathsf{SK}_1 := (SK_1, dk_1)$, $\mathsf{PK}_1 := (PK_1, ek_1)$ and SK_b matches PK_b for $b = 0, 1$. $\mathcal{A}'$ submits $(M^*||\mathsf{PK}_0||\mathsf{PK}_1, (SK_0, PK_0), (SK_1, PK_1))$ to its own challenger, which returns a partial signature σ^*_P with respect to the secret key SK_b for some random choice $b \in \{0, 1\}$. $\mathcal{A}'$ encrypts σ^*_P under the public encryption key ek_1, i.e., $c = \mathcal{E}.\mathsf{Enc}(ek_1, \sigma^*_P)$, and forwards $\sigma_P := c^*$ to $\mathcal{A}$ as the answer. $\mathcal{A}'$ then continues to simulate the oracle $O_{\mathbf{Res}}$ in the same way as above.

It can be seen that the oracle $O_{\mathbf{Res}}$ is simulated properly by $\mathcal{A}'$. Finally, $\mathcal{A}$ outputs a bit d. $\mathcal{A}'$ outputs a bit $b' = d$ and halts. Since $\mathcal{A}$ is not allowed to issue a query $(M^*, \mathsf{PK}_0, \mathsf{PK}_1)$ to the resolution oracle $O_{\mathbf{Res}}$, $\mathcal{A}'$ has never made a query to its oracle O'_{Res} with respect to message $M^*||\mathsf{PK}_0||\mathsf{PK}_1$. If $\mathcal{A}$ succeeds in the experiment, $\mathcal{A}'$ also succeeds in outputting the bit b' with the same probability. $\mathcal{A}'$'s advantage is also non-negligible. □

Theorem 7.3 *The generic construction is secure against signers if* $\Gamma = (\mathsf{PMGen}, \mathsf{Setup}^{\mathsf{TTP}}, \mathsf{Setup}^{\mathsf{User}}, \mathsf{Sig}, \mathsf{Ver}, \mathsf{PSig}, \mathsf{PVer}, \mathsf{Res})$ *is secure against signers.*

Proof. To show security against signers, one converts any adversary $\mathcal{A}$ that wins the experiment into an adversary $\mathcal{A}'$ that breaks the security against signers of Γ. Recall that $\mathcal{A}'$ gets (APK, PK_B) as input and has access to oracles O'^{B}_{PSig} and O'_{Res}. $\mathcal{A}'$ runs $\mathcal{E}.\mathsf{KGen}(1^k) \rightarrow (ek_B, dk_B)$ and invokes $\mathcal{A}$ on input $\mathsf{APK} := APK$ and $\mathsf{PK}_B := (PK_B, ek_B)$.

Given a partial signing query (M, PK_j) where $\mathsf{PK}_j = (PK_j, ek_j)$ to oracle $O^B_{\mathbf{PSig}}$, $\mathcal{A}'$ makes a query $(M||\mathsf{PK}_B||\mathsf{PK}_j, PK_j)$ to its own oracle O'^{B}_{PSig}. Denote the answer from O'^{B}_{PSig} is σ'_P. $\mathcal{A}'$ then encrypt it under the public encryption key ek_j, i.e., $c = \mathcal{E}.\mathsf{Enc}(ek_j, \sigma'_P)$, and returns $\sigma_P := c$ to $\mathcal{A}$ as the answer.

Given a fake partial signing query (M, PK_i) where $\mathsf{PK}_i = (PK_i, ek_i)$ to oracle $O^B_{\mathbf{FakePSig}}$, $\mathcal{A}'$ makes a query $(M||\mathsf{PK}_i||\mathsf{PK}_B, PK_i)$ to its own oracle O'^{B}_{PSig}. Denote the answer from O'^{B}_{PSig} is σ'_P. $\mathcal{A}'$ then encrypt it under the public encryption key ek_B, i.e., $c = \mathcal{E}.\mathsf{Enc}(ek_B, \sigma'_P)$, and returns $\sigma_P := c$ to $\mathcal{A}$ as the answer.

Given a partial signature verification query $(M, \sigma_P, \mathsf{PK}_i = (PK_i, ek_i))$ to $O^B_{\mathbf{PVer}}$, $\mathcal{A}'$ decrypts σ_P using its own decryption key dk_B, i.e., $\sigma'_P = \mathcal{E}.\mathsf{Dec}(dk_B, \sigma_P)$, and

returns $\Gamma.\mathsf{PVer}\ (M||\mathsf{PK}_i||\mathsf{PK}_B, \sigma'_P,\ PK_i, PK_B, APK)$ to $\mathcal{A}$.

Given a resolution query $(M, \mathsf{PK}_i, \mathsf{PK}_j)$ where $\mathsf{PK}_i = (PK_i, ek_i)$ and $\mathsf{PK}_j = (PK_j, ek_j)$ to $O^B_{\mathbf{Res}}$, suppose $\mathcal{A}$ sends σ'_P to $\mathcal{A}'$ in the first run of the protocol. $\mathcal{A}'$ makes a query $(M||\mathsf{PK}_i||\mathsf{PK}_j, \sigma'_P,\ PK_i, PK_j)$ to its own oracle O'_{Res} and returns the answer to $\mathcal{A}$.

It can be seen that the oracles $O_{\mathbf{PSig}}$, $O^B_{\mathbf{FakePSig}}$, $O^B_{\mathbf{PVer}}$ and $O_{\mathbf{Res}}$ are simulated properly by $\mathcal{A}'$. Finally, $\mathcal{A}$ outputs a partial signature σ^*_P on message M^* under PK_A, PK_B where $\mathsf{PK}_A = (PK_A, ek_A)$. $\mathcal{A}'$ decrypts σ^*_P under the decryption key dk_B, i.e., $\tilde{\sigma_P} = \mathcal{E}.\mathsf{Dec}(dk_B, \sigma^*_P)$ and outputs $(M^*||\mathsf{PK}_A||\mathsf{PK}_B, \tilde{\sigma_P}, PK_A)$. Notice that $\tilde{\sigma_P}$ is a valid partial signature on message $M^*||\mathsf{PK}_A||\mathsf{PK}_B$ under PK_A and PK_B, i.e. $\Gamma.\mathsf{PVer}(M^*||\mathsf{PK}_A||\mathsf{PK}_B, \tilde{\sigma_P}, PK_A, PK_B, APK) = \top$, but it can not be resolved to a valid full signature by the resolution algorithm $\Gamma.\mathsf{Res}$, i.e. $\Gamma.\mathsf{Res}(M^*||\mathsf{PK}_A||\mathsf{PK}_B, \tilde{\sigma_P}$, $ASK, PK_A, PK_B) = \perp$. Since $\mathcal{A}$ is prohibited from making a query (M^*, PK_A) to oracle $O^B_{\mathbf{FakePSig}}$, $\mathcal{A}'$ has never issued a query $(M^*||\mathsf{PK}_A||\mathsf{PK}_B, PK_A)$ to its own oracle O'^B_{PSig}, thus if $\mathcal{A}$ succeeds in the experiment, $\mathcal{A}'$ also succeeds with the same probability in breaking the security against signers of Γ. □

Theorem 7.4 *The generic construction is secure against verifiers if* $\Gamma = (\mathsf{PMGen}, \mathsf{Setup}^{\mathsf{TTP}}, \mathsf{Setup}^{\mathsf{User}}, \mathsf{Sig}, \mathsf{Ver}, \mathsf{PSig}, \mathsf{PVer}, \mathsf{Res})$ *is secure against verifiers.*

Proof. To show security against verifiers, one converts any adversary $\mathcal{A}$ that wins the experiment into an adversary $\mathcal{A}'$ that breaks the security against verifiers of Γ. Recall that $\mathcal{A}'$ gets (APK, PK_A) as input and has access to oracles O'_{PSig} and O'_{Res}. $\mathcal{A}'$ runs $\mathcal{E}.\mathsf{KGen}(1^k) \rightarrow (ek_A, dk_A)$ and invokes $\mathcal{A}$ on input $\mathsf{APK} := APK$ and $\mathsf{PK}_A := (PK_A, ek_A)$.

Given a partial signing query (M, PK_j) where $\mathsf{PK}_j = (PK_j, ek_j)$ to oracle $O_{\mathbf{PSig}}$, $\mathcal{A}'$ makes a query $(M||\mathsf{PK}_A||\mathsf{PK}_j,\ PK_j)$ to its own oracle O'_{PSig}. Denote the answer from O'_{PSig} is σ'_P. $\mathcal{A}'$ then encrypt it under the public encryption key ek_j, i.e., $c = \mathcal{E}.\mathsf{Enc}(ek_j, \sigma'_P)$, and returns $\sigma_P := c$ to $\mathcal{A}$ as the answer.

Given a fake partial signing query (M, PK_i) where $\mathsf{PK}_i = (PK_i, ek_i)$ to oracle $O_{\mathbf{FakePSig}}$, $\mathcal{A}'$ makes a query $(M||\mathsf{PK}_i||\mathsf{PK}_A, PK_i)$ to its own oracle O'_{PSig}. Denote the answer from O'_{PSig} is σ'_P. $\mathcal{A}'$ then encrypt it under the public encryption key ek_A, i.e., $c = \mathcal{E}.\mathsf{Enc}(ek_A, \sigma'_P)$, and returns $\sigma_P := c$ to $\mathcal{A}$ as the answer.

Given a partial signature verification query $(M, \sigma_P, \mathsf{PK}_i = (PK_i, ek_i))$ to oracle $O_{\mathbf{PVer}}$, $\mathcal{A}'$ decrypts σ_P using its own decryption key dk_A, i.e., $\sigma'_P = \mathcal{E}.\mathsf{Dec}(dk_A, \sigma_P)$, and returns $\Gamma.\mathsf{PVer}\ (M||\mathsf{PK}_i||\mathsf{PK}_A, \sigma'_P,\ PK_i, PK_A, APK)$ to $\mathcal{A}$.

Given a resolution query $(M, \mathsf{PK}_i, \mathsf{PK}_j)$ where $\mathsf{PK}_i = (PK_i, ek_i)$ and $\mathsf{PK}_j = (PK_j, ek_j)$ to $O_{\mathbf{Res}}$, suppose $\mathcal{A}$ sends σ'_P to $\mathcal{A}'$ in the first run of the protocol. $\mathcal{A}'$ makes a query $(M||\mathsf{PK}_i||\mathsf{PK}_j, \sigma'_P, PK_i, PK_j)$ to its own oracle O'_{Res} and returns the answer to $\mathcal{A}$.

It can be seen that the oracles $O_{\mathbf{FakePSig}}$, $O_{\mathbf{PSig}}$, $O_{\mathbf{PVer}}$ and $O_{\mathbf{Res}}$ are simulated properly by $\mathcal{A}'$. Finally, $\mathcal{A}$ returns a full signature σ^* on message M^* under PK_A, PK_B where $\mathsf{PK}_B = (PK_B, ek_B)$ such that $\mathbf{Ver}(M^*, \sigma^*, \mathsf{PK}_A, \mathsf{PK}_B, \mathsf{APK}) = \top$, which means $\Gamma.\mathsf{Ver}(M^*||\mathsf{PK}_A||\mathsf{PK}_B, \sigma^*, PK_A, PK_B, APK) = \top$. $\mathcal{A}'$ outputs $(M^*||\mathsf{PK}_A||\mathsf{PK}_B, \sigma^*, PK_B)$ and aborts. Since $\mathcal{A}$ is prohibited from making a query $(M^*, \mathsf{PK}_A, \mathsf{PK}_B)$ to oracle $O_{\mathbf{Res}}$, $\mathcal{A}'$ has never made a query with respect to message $M^*||\mathsf{PK}_A||\mathsf{PK}_B$ to its own oracle O'_{Res}. If $\mathcal{A}$ succeeds in the experiment, $\mathcal{A}'$ also succeeds in breaking the security against verifiers of Γ. Thus $\mathcal{A}'$'s advantage is also non-negligible. □

Theorem 7.5 *The generic construction is secure against the arbitrator if* $\Gamma = (\mathsf{PMGen}, \mathsf{Setup}^{\mathsf{TTP}}, \mathsf{Setup}^{\mathsf{User}}, \mathsf{Sig}, \mathsf{Ver}, \mathsf{PSig}, \mathsf{PVer}, \mathsf{Res})$ *is secure against the arbitrator.*

Proof. To show security against the arbitrator, one converts any adversary $\mathcal{A}$ that wins the experiment into an adversary $\mathcal{A}'$ that breaks the security against the arbitrator of Γ. Suppose the public parameter PM is generated. $\mathcal{A}$ first chooses a public adjudication key APK and outputs it, and keeps a corresponding secret state information ASK^* private. $\mathcal{A}'$ sets $APK := \mathsf{APK}$, gets PK_A as input, and has access to oracles O'_{PSig}. $\mathcal{A}'$ runs $\mathcal{E}.\mathsf{KGen}(1^k) \rightarrow (ek_A, dk_A)$ and invokes $\mathcal{A}$ on input $\mathsf{PK}_A := (PK_A, ek_A)$.

Given a partial signing query (M, PK_j) where $\mathsf{PK}_j = (PK_j, ek_j)$ to oracle $O_{\mathbf{PSig}}$, $\mathcal{A}'$ makes a query $(M||\mathsf{PK}_A||\mathsf{PK}_j, PK_j)$ to its own oracle O'_{PSig}. Suppose the answer from O'_{PSig} is σ'_P. $\mathcal{A}'$ then encrypt it under the public encryption key ek_j, i.e., $c = \mathcal{E}.\mathsf{Enc}(ek_j, \sigma'_P)$, and returns $\sigma_P := c$ to $\mathcal{A}$ as the answer.

Given a fake partial signing query (M, PK_i) where $\mathsf{PK}_i = (PK_i, ek_i)$ to oracle $O_{\mathbf{FakePSig}}$, $\mathcal{A}'$ makes a query $(M||\mathsf{PK}_i||\mathsf{PK}_A, PK_i)$ to its own oracle O'_{PSig}. Denote the answer from O'_{PSig} is σ'_P. $\mathcal{A}'$ then encrypt it under the public encryption key ek_A, i.e., $c = \mathcal{E}.\mathsf{Enc}(ek_A, \sigma'_P)$, and returns $\sigma_P := c$ to $\mathcal{A}$ as the answer.

Given a partial signature verification query $(M, \sigma_P, \mathsf{PK}_i = (PK_i, ek_i))$ to oracle $O_{\mathbf{PVer}}$, $\mathcal{A}'$ decrypts σ_P using its own decryption key dk_A, i.e., $\sigma'_P = \mathcal{E}.\mathsf{Dec}(dk_A, \sigma_P)$, and returns $\Gamma.\mathsf{PVer}$ $(M||\mathsf{PK}_i||\mathsf{PK}_A, \sigma'_P, PK_i, PK_A, APK)$ to $\mathcal{A}$.

It can be seen that the oracles $O_{\mathbf{FakePSig}}$, $O_{\mathbf{PSig}}$ and $O_{\mathbf{PVer}}$ are simulated properly by $\mathcal{A}'$. Finally, $\mathcal{A}$ returns a full signature σ^* on message M^* under PK_A, PK_B where $\mathsf{PK}_B = (PK_B, ek_B)$, such that $\mathbf{Ver}(M^*, \sigma^*, \mathsf{PK}_A, \mathsf{PK}_B, \mathsf{APK}) = \top$, which means $\Gamma.\mathsf{Ver}(M^*||\,\mathsf{PK}_A||\,\mathsf{PK}_B, \sigma^*, PK_A, PK_B, APK) = \top$. $\mathcal{A}'$ outputs $(M^*||\mathsf{PK}_A||\mathsf{PK}_B, \sigma^*, PK_B)$ and aborts. Since $\mathcal{A}$ is prohibited from making a query $(M^*, \mathsf{PK}_A, \mathsf{PK}_B)$ to oracle $O_{\mathbf{PSig}}$, $\mathcal{A}'$ has not made a query with respect to message $M^*||\mathsf{PK}_A||\mathsf{PK}_B$ to its own oracle O'_{PSig}. If $\mathcal{A}$ succeeds in the experiment, $\mathcal{A}'$ also succeeds with the same probability in breaking the security against the arbitrator of Γ, as σ^* is a valid full signature on message $M^*||\mathsf{PK}_A||\mathsf{PK}_B$ under PK_A, PK_B. Thus $\mathcal{A}'$'s advantage is also non-negligible. □

7.5 Comparison with Designated Verifier Signature

A designated verifier signature scheme is a signature scheme in which signatures can only be verified by a single designated verifier who is chosen by the signer. This concept was first introduced by Jakobsson, Sako and Impagliazzo [JSI96]. After getting a designated verifier signature, the verifier can not convince others that it if from the signer, as it can also be generated by the verifier in the view of an outsider. PAOFE is similar with designated verifier signatures in this sense, since the verifier can not convince outsider of the authorship of the partial signature either. However, in PAOFE, the partial signature reveals no information about the identities of the signer or the verifier, but the designated verifier signatures do not necessarily have to be so. Besides, there is a third party in PAOFE, who can help identify the exact signer of the partial signature. Such a third party does not exist for designated verifier signature schemes.

7.6 Chapter Summary

In this chapter, the notion of perfect ambiguous optimistic fair exchange was proposed, and a formal security model was given. A generic construction of PAOFE was then proposed, and its security was proved under the proposed model in the standard model.

The generic construction involves an encryption and an AOFE scheme. Compared with that of an AOFE scheme, the computation cost of the PAOFE scheme contains only one more encryption operation, which is quite acceptable. Construction of more efficient concrete PAOFE schemes is left as the future work.

Chapter 8

Attribute-based Optimistic Fair Exchange: How to Restrict Brokers with Policies

In this chapter a new notion named attribute-based optimistic fair exchange (ABOFE) is introduced to solve the fair exchange problem in the attribute-based setting.

8.1 Introduction

The brokerage business model has been used since the pre-Internet era, where the intermediary buys from the supplier (or producer) and owns the goods first, and then sells it. This model plays a very important role in the online business nowadays, as it enables fast and secure transactions without relying on a single merchant's connection. Brokers have been known to be active in business-to-business (B2B), businesss-to-consumer (B2C) and consumer-to-consumer (C2C) or peer-to-peer (P2P) markets. Although this model is known to be very useful and practical, some issues have happened since the broker may incorporate some certain strategies to increase his/her sales, and it may damage even the supplier.

Consider the following real life case study. A broker B buys games from the game developer G and sells these games to its customer. Since B is considered as a broker, the price that has been set to B is certainly lower than the retail price of the game itself. In order to maximise its sales, B is happy to make its margin to be very low, and this way B will gain popularity among the customers and attract more buyers. In particular, this is certainly possible if B purchases the games from different countries, since usually the price for each country will be different, due to the sales tax applied. This action is certainly damaging the market and G will not be able to sell that particular game with the retail price to the customers as they would have preferred to acquire it from B instead. This issue can be solved by

placing required policies for the brokers. First of all, each broker must be licensed in order to sell the games. The license is limited to the country of residence. The game will be playable, if and only if, the required CD key sold by the broker matches with the country where the game will be played. To give an example, someone resides in the US will purchase a CD key from the broker. First of all, he/she needs to be sure that the broker is a licensed retailer for US. Once the CD key is issued by the broker, this CD key is only usable if it is used in the US and not in other countries. Another scenario involves limitation of the age restriction towards the game itself. For instance, consider the game "God of War" that is restricted to people over 18 years old. A US online reseller selles the activation key (or CD key) for this game. This price is lower than the price of the same game available in Canada. There are two issues need to be solved here. First, the buyer needs to be ensured that the reseller is a genuine reseller, and the reseller needs to be sure that the buyer is at least 18 years old. Subsequently, the activation key sold can only be used in the US and not elsewhere. Hence, the buyer from Canada will not be able to make use of this particular activation key.

The Approach. In this chapter, a cryptographic primitive that aims to solve the above scenario is to be presented. The notion of optimistic fair exchange (OFE) is used and enhanced to enable policies for both parties, namely the seller and the buyer (or the signer and the verifier, resp.). A trivial solution for enabling the above scenario would be that prior to involving an OFE protocol, each participant will simply present evidence, for instance a copy of the identity card or license, to the other participant, hence confirming that they satisfy the requirements. Nevertheless, this solution compromises the privacy of the users, and therefore it is not ideal. Furthermore, providing such kind of evidence over the network securely is rather challenging as well.

8.1.1 The Contributions

In this chapter, motivated by idea of attribute-based encryption (ABE) [SW05] and ciphertext policy attribute-based encryption (CP-ABE) [Wat11], the notion of attribute-based optimistic fair exchange (ABOFE) is introduced, which can be viewed as an extension of OFE, as a practical cryptographic solution to the aforementioned scenario.

In ABOFE, each user satisfying a set of attributes is assigned a credential by

the credential center. This allows the signer Alice to generate a credential-protected package in such a way that only verifiers that possess appropriate credentials can convert it into a full signature, which naturally guarantees that the verifier should satisfy some particular set of attributes.

The syntax is proposed and a security model for ABOFE is defined in the multi-user setting under chosen-key attack. The model captures the existing security requirements for OFE, namely, security against signers, security against verifiers and security against the arbitrator. As suggested by the respective names, they intend to cover the scenarios when the named party is dishonest.

Finally, a generic construction of ABOFE is proposed from the two well established cryptographic primitives, OFE and CP-ABE, and the security proof of the proposal is provided in the proposed model. The generic construction works in the standard model and does not involve any extra assumptions. The efficiency of an instantiation of the generic construction is also discussed.

8.1.2 Chapter Organization

In the next section, the syntax of ABOFE and its security definitions are presented. Then the construction is presented in Section 8.3. An instantiation is descussed in Section 8.4. Finally, this chapter is summarised in Section 8.5.

8.2 Attribute-based Optimistic Fair Exchange

The definitions and security models of OFE are adapted for the attribute-based OFE. In ABOFE, besides the traditional secret/public key pair, each user also possesses a credential, generated by the credential center, corresponding to a set of attributes the user satisfies, like in the notion of ciphertext-policy attribute-based encryption (CP-ABE) which was reviewed in Section 2.2.8. It is required that a user that does not possess a set of attributes should not be able to gain other users' full signatures. Thus a full signature will not be sent by the signer in ABOFE, which is different from the case in OFE.

With this in mind, the algorithms of which ABOFE scheme consists are introduced.

Definition 8.1 *An attribute-based optimistic fair exchange scheme involves the users, the credential center and the arbitrator, and consists of the following (probabilistic)*

polynomial-time algorithms:

- **PMGen**: *On input* 1^k *and attribute universe description* U *where* k *is a security parameter, this algorithm outputs a system parameter* PM *and a credential secret key* CK.
- $\mathbf{Setup}^{\mathsf{TTP}}$: *On input* 1^k, *the algorithm generates a secret key* ASK, *and a public key* APK *of the arbitrator.*
- $\mathbf{Setup}^{\mathsf{User}}$: *On input* 1^k *and (optionally)* APK, *it outputs a secret/public key pair* $(\mathsf{SK}, \mathsf{PK})$. *For a user* U_i, $(\mathsf{SK}_i, \mathsf{PK}_i)$ *is used to denote the user's key pair.*
- $\mathbf{Setup}^{\mathsf{Cred}}$: *On input the credential secret key* CK *and a set of attributes* S, *the attribute key generation algorithm outputs a credential* CD_S *corresponding to the set of attributes* S.
- $\mathbf{Tran}_{\mathsf{P}}$: *The transaction promise generation algorithm* $\mathbf{Tran}_{\mathsf{P}}(m_1, m_2, \mathsf{SK}_i, \mathsf{APK}, \mathsf{PM}, \mathbb{A}_1, \mathbb{A}_2)$ *takes as input two messages* m_1 *and* m_2, *respectively, user* U_i*'s secret key* SK_i, *the arbitrator's public key* APK, *the system parameter* PM *and two access structure* $\mathbb{A}_1$ *and* $\mathbb{A}_2$ *over the universe of attributes, respectively. This algorithm outputs a transaction promise* ω.
- **TPVer**: *The transaction promise verification algorithm* $\mathbf{TPVer}(m_1, m_2, \omega, \mathsf{PK}_i, \mathsf{APK}, \mathsf{PM}, \mathbb{A}_1, \mathbb{A}_2)$ *takes as input two message* m_1 *and* m_2, *respectively, a transaction promise* ω, *user* U_i*'s public key* PK_i, *the arbitrator's public key* APK, *the system parameter* PM, *and two access structure* $\mathbb{A}_1$ *and* $\mathbb{A}_2$, *respectively. The algorithm outputs* $\top$ *indicating valid or* $\perp$ *indicating invalid.*
- $\mathbf{Tran}_{\mathsf{S}}$: *The transaction for signer algorithm* $\mathbf{Tran}_{\mathsf{S}}(m, \mathsf{SK}_i, \mathsf{APK}, \mathsf{PM}, \mathbb{A})$ *takes as input a message* m, *user* U_i*'s secret key* SK_i, *the arbitrator's public key* APK, *the system parameter* PM, *and an access structure* $\mathbb{A}$. *The algorithm outputs a credential-protected package* π.
- $\mathbf{Tran}_{\mathsf{V}}$: *The transaction for verifier algorithm* $\mathbf{Tran}_{\mathsf{V}}(m, \pi, \mathsf{PK}_i, \mathsf{APK}, \mathsf{PM}, \mathsf{CD}_S)$ *takes a message* m, *a credential-protected package* π, *user* U_i*'s public key* PK_i, *the arbitrator's public key* APK, *the system parameter* PM, *and a credential* CD_S. *This algorithm outputs a full signature* σ *or* $\perp$ *indicating failure.*
- **Ver:** *The full signature verification algorithm* $\mathbf{Ver}(m, \sigma, \mathsf{PK}_i, \mathsf{APK})$ *outputs* $\top$ *indicating valid or* $\perp$ *indicating invalid.*

- $\mathbf{Res}_{\mathsf{User}}$: *The resolution for user algorithm* $\mathbf{Res}_{\mathsf{User}}(m_1, m_2, \omega, \mathsf{SK}_j, \mathsf{PK}_i, \mathsf{APK}, \mathsf{PM}, \mathbb{A}_1, \mathbb{A}_2)$ *takes as input two message* m_1 *and* m_2*, respectively, a transaction promise* ω*, and user* U_j*'s secret key* SK_j*, the system parameter* PM*, and two access structure* $\mathbb{A}_1$ *and* $\mathbb{A}_2$*, respectively. This algorithm outputs a resolution request* request.

- $\mathbf{Res}_{\mathsf{TTP}}$: *The resolution for arbitrator algorithm* $\mathbf{Res}_{\mathsf{TTP}}(m, \mathsf{PK}_i, \mathbb{A}, \mathsf{request}, \mathsf{ASK})$ *takes as input a message* m*, user* U_i*'s public key* PK_i*, an access structure* $\mathbb{A}$*, a resolution request* request *and the arbitrator's secret key* ASK*. It outputs two credential-protected packages* π_1 *and* π_2*, respectively.*

Correctness states that, for $\mathbf{PMGen}(1^k) \to (\mathsf{PM}, \mathsf{CK})$, $\mathbf{Setup}^{\mathsf{TTP}}(1^k) \to (\mathsf{ASK}, \mathsf{APK})$, $\mathbf{Setup}^{\mathsf{User}}(1^k) \to (\mathsf{SK}_i, \mathsf{PK}_i)$, $\mathbf{Setup}^{\mathsf{User}}(1^k) \to (\mathsf{SK}_j, \mathsf{PK}_j)$, $\mathbf{Tran}_{\mathsf{P}}$ $(m_1, m_2, \mathsf{SK}_i, \mathsf{APK}, \mathsf{PM}, \mathbb{A}_1, \mathbb{A}_2) \to \omega$, $\mathbf{Tran}_{\mathsf{S}}$ $(m_1, \mathsf{SK}_i, \mathsf{APK}, \mathsf{PM}, \mathbb{A}_2) \to \pi_1$, $\mathbf{Tran}_{\mathsf{S}}$ $(m_2, \mathsf{SK}_j, \mathsf{APK}, \mathsf{PM}, \mathbb{A}_1) \to \pi_2$, $\mathbf{Res}_{\mathsf{User}}$ $(m_1, m_2, \omega, \mathsf{SK}_j, \mathsf{PK}_i, \mathsf{APK}, \mathsf{PM}, \mathbb{A}_1, \mathbb{A}_2) \to \mathsf{request}$, $\mathbf{Res}_{\mathsf{TTP}}(m_1, \mathsf{PK}_i, \mathbb{A}_2, \mathsf{request}, \mathsf{ASK}) \to (\tilde{\pi}_1, \tilde{\pi}_2)$, $\mathbf{Setup}^{\mathsf{Cred}}$ $(\mathsf{CK}, S) \to \mathsf{CD}_S$ where the sets of attributes S satisfies the access structures $\mathbb{A}_1$, and $\mathbf{Setup}^{\mathsf{Cred}}(\mathsf{CK}, S') \to \mathsf{CD}_{S'}$ where the sets of attributes S' satisfies the access structures $\mathbb{A}_2$, the following conditions hold:

- $\mathbf{TPVer}(m_1, m_2, \omega, \mathsf{PK}_i, \mathsf{APK}, \mathsf{PM}, \mathbb{A}_1, \mathbb{A}_2) = \top$.
- $\mathbf{Ver}(m_1, \mathbf{Tran}_{\mathsf{V}}(m_1, \pi_1, \mathsf{PK}_i, \mathsf{APK}, \mathsf{PM}, \mathsf{CD}_{S'}), \mathsf{PK}_i, \mathsf{APK}) = \top$.
- $\mathbf{Ver}(m_2, \mathbf{Tran}_{\mathsf{V}}(m_2, \pi_2, \mathsf{PK}_j, \mathsf{APK}, \mathsf{PM}, \mathsf{CD}_S), \mathsf{PK}_j, \mathsf{APK}) = \top$.
- $\mathbf{Ver}(m_1, \mathbf{Tran}_{\mathsf{V}}(m_1, \tilde{\pi}_1, \mathsf{PK}_i, \mathsf{APK}, \mathsf{PM}, \mathsf{CD}_{S'}), \mathsf{PK}_i, \mathsf{APK}) = \top$.
- $\mathbf{Ver}(m_2, \mathbf{Tran}_{\mathsf{V}}(m_2, \tilde{\pi}_2, \mathsf{PK}_j, \mathsf{APK}, \mathsf{PM}, \mathsf{CD}_S), \mathsf{PK}_j, \mathsf{APK}) = \top$.

Resolution ambiguity states that, any "resolved credential-protected packages" $\tilde{\pi}_1$ and $\tilde{\pi}_2$ outputted by $\mathbf{Res}_{\mathsf{Abr}}$ are computationally indistinguishable from the "actual credential-protected packages" π_1 and π_2 generated by $\mathbf{Tran}_S$, respectively.

8.2.1 A Typical Usage of ABOFE

For simplicity, a typical usage of ABOFE between two users Alice and Bob will be described. Before the exchange phase, the credential center runs **PMGen** to set up the public parameters and keep the credential secret key as private. Alice and Bob acquire their respective credentials generated by $\mathbf{Setup}^{\mathsf{Cred}}$ from the credential

center. Besides, the arbitrator sets its key pair by invoking $\mathbf{Setup}^{\mathsf{TTP}}$. Alice and Bob generate their own key pairs by invoking $\mathbf{Setup}^{\mathsf{User}}$, respectively.

Suppose Alice intends to exchange her own full signature σ_1 on message m_1 for Bob's full signature σ_2 on message m_2. Besides, Alice should satisfy an access structure $\mathbb{A}_1$, and Bob should satisfy an access structure $\mathbb{A}_2$. The exchange phase consists of three steps.

1. Alice invokes $\mathbf{Tran}_{\mathsf{P}}$ to generate a transaction promise ω on message m_1 and m_2 with respect to access structures $\mathbb{A}_1$ and $\mathbb{A}_2$. Alice sends ω to Bob.

2. Bob invokes **TPVer** to ensure the validity of ω. Bob then invokes $\mathbf{Tran}_{\mathsf{S}}$ to generate a credential-protected package π_B on message m_2 with respect to access structure $\mathbb{A}_1$. Bob sends π_B to Alice.

3. If satisfying the access structure $\mathbb{A}_1$, Alice can invoke $\mathbf{Tran}_{\mathsf{V}}$ to gain a full signature σ_B of Bob's. Then Alice invokes $\mathbf{Tran}_{\mathsf{S}}$ to generate a credential-protected package π_A on message m_1 with respect to access structures $\mathbb{A}_2$. Alice sends π_A to Bob, who can invoke $\mathbf{Tran}_{\mathsf{V}}$ to gain Alice's full signature σ_A if satisfying the access structure $\mathbb{A}_2$.

Note that in the normal cases, the exchange will finish and both Alice and Bob can gain the full signature of the other's. In the case Alice refuses to send her credential-protected package π_A in the third step, or that credential-protected package is created with respect to improper access structures, Bob can approach the arbitrator for assistance. Specifically, he approaches the arbitrator and sends a resolution request generated by invoking $\mathbf{Res}_{\mathsf{User}}$. To make a resolution, the arbitrator invokes $\mathbf{Res}_{\mathsf{TTP}}$ to generate two credential-protected packages $\tilde{\pi}_1$ and $\tilde{\pi}_2$. $\tilde{\pi}_1$ is sent to Bob, and $\tilde{\pi}_2$ is sent to Alice. Thus both Alice and Bob can invoke $\mathbf{Tran}_{\mathsf{V}}$ to gain the other's full signature if satisfying the expectant attributes.

8.2.2 Security Model

The traditional OFE model is modified to make it suitable in the attribute-based setting.

- **Security Against Signers**: This property guarantees that a transaction promise generated by a signer can always be resolved to a full signature of the signer's

if the arbitrator honestly makes a resolution and the verifier satisfies the expectant attributes. Formally, consider the following experiment, in which the adversary $\mathcal{A}$ models a dishonest signer that can even have access to the credential secret key and the verifier's secret key.

Experiment SAS:

$$\begin{aligned}
(\mathsf{PM}, \mathsf{CK}) &\leftarrow \mathbf{PMGen}(1^k, U) \\
(\mathsf{ASK}, \mathsf{APK}) &\leftarrow \mathbf{Setup}^{\mathsf{TTP}}(1^k) \\
(\mathsf{SK}_B, \mathsf{PK}_B) &\leftarrow \mathbf{Setup}^{\mathsf{User}}(1^k) \\
(m_1^*, m_2^*, \omega^*, \mathsf{PK}_A, \mathbb{A}_1, \mathbb{A}_2, S) &\leftarrow \mathcal{A}^{O_{\mathbf{Res}}}(\mathsf{PM}, \mathsf{CK}, \mathsf{APK}, \mathsf{SK}_B, \mathsf{PK}_B) \\
\mathsf{CD}_S &\leftarrow \mathbf{Setup}^{\mathsf{Cred}}(\mathsf{CK}, S) \\
\mathsf{request}^* &\leftarrow \mathbf{Res}_{\mathsf{User}}(m_1^*, m_2^*, \omega^*, \mathsf{SK}_B, \mathsf{APK}, \mathsf{PM}, \mathbb{A}_1, \mathbb{A}_2) \\
(\pi_1^*, \pi_2^*) &\leftarrow \mathbf{Res}_{\mathsf{TTP}}(m_1^*, \mathsf{PK}_A, \mathbb{A}_2, \mathsf{request}^*, \mathsf{ASK}) \\
\sigma^* &\leftarrow \mathbf{Tran}_{\mathsf{V}}(m_1^*, \pi_1^*, \mathsf{PK}_A, \mathsf{APK}, \mathsf{PM}, \mathsf{CD}_S) \\
\text{success of } \mathcal{A} &:= [\mathbf{TPVer}(m_1^*, m_2^*, \omega^*, \mathsf{PK}_A, \mathsf{APK}, \mathsf{PM}, \mathbb{A}_1, \mathbb{A}_2) = \top \\
&\quad \wedge \mathbf{Ver}(m_1^*, \sigma^*, \mathsf{PK}_A, \mathsf{APK}) = \bot]
\end{aligned}$$

where S is a set of attributes that satisfies the access structure $\mathbb{A}_2$, the resolution for arbitrator oracle $O_{\mathbf{Res}}$ takes as input $(m, \mathsf{PK}_i, \mathbb{A}, \mathsf{request})$, and outputs $\mathbf{Res}_{\mathsf{TTP}}(m, \mathsf{PK}_i, \mathbb{A}, \mathsf{request}, \mathsf{ASK})$. In this experiment, the adversary can arbitrarily choose a public key PK_i, and it may not know the corresponding private key of PK_i.

- **Security Against Verifiers**: This property captures two cases: the first case is that no verifier, even given the credential secret key, should be able to generate a full signature of the signer's while not possessing a credential-protected package. The second case is that the verifier that does not possess the expectant attributes should not be able to generate a full signature even if given a corresponding credential-protected package. Formally, it is required that no PPT adversary $\mathcal{A}$, who models a dishonest verifier, succeeds with non-negligible probability in either of the following two experiments:

Experiment SAV1:

$$\begin{aligned}
(\mathsf{PM}, \mathsf{CK}) &\leftarrow \mathbf{PMGen}(1^k, U) \\
(\mathsf{ASK}, \mathsf{APK}) &\leftarrow \mathbf{Setup}^{\mathsf{TTP}}(1^k) \\
(\mathsf{SK}_A, \mathsf{PK}_A) &\leftarrow \mathbf{Setup}^{\mathsf{User}}(1^k, \mathsf{APK}) \\
(m^*, \sigma^*) &\leftarrow \mathcal{A}^{O_{\mathbf{Tran_P}}, O_{\mathbf{Res}}}(\mathsf{PM}, \mathsf{CK}, \mathsf{APK}, \mathsf{PK}_A) \\
\text{success of } \mathcal{A} &:= [\mathbf{Ver}(m^*, \sigma^*, \mathsf{PK}_A, \mathsf{APK}) = \top \\
&\quad \wedge (m^*, \mathsf{PK}_A, \cdot, \cdot) \notin Query(\mathcal{A}, O_{\mathbf{Res}})]
\end{aligned}$$

where oracle $O_{\mathbf{Res}}$ is described in the previous experiment, oracle $O_{\mathbf{Tran_P}}$ takes as input $(m_1, m_2, \mathbb{A}, \mathbb{A}')$ and outputs $\mathbf{Tran_P}(m_1, m_2, \mathsf{SK}_A, \mathsf{APK}, \mathsf{PM}, \mathbb{A}, \mathbb{A}')$, and $Query(\mathcal{A}, O_{\mathbf{Res}})$ is the set of queries $\mathcal{A}$ issued to the resolution oracle. Note that there is no need to provide $\mathcal{A}$ with access to the transaction for signer oracle $O_{\mathbf{Tran_S}}$, as resolution ambiguity property guarantees that its functionality could be achieved by executing $O_{\mathbf{Tran_P}}$ and $O_{\mathbf{Res}}$.

Experiment SAV2:

$$\begin{aligned}
(\mathsf{PM}, \mathsf{CK}) &\leftarrow \mathbf{PMGen}(1^k, U) \\
(\mathsf{ASK}, \mathsf{APK}) &\leftarrow \mathbf{Setup}^{\mathsf{TTP}}(1^k) \\
(\mathsf{SK}_A, \mathsf{PK}_A) &\leftarrow \mathbf{Setup}^{\mathsf{User}}(1^k, \mathsf{APK}) \\
(m^*, \sigma^*) &\leftarrow \mathcal{A}^{O_{\mathbf{Setup}^{\mathsf{Cred}}}, O_{\mathbf{Tran_P}}, O_{\mathbf{Res}}}(\mathsf{PM}, \mathsf{APK}, \mathsf{PK}_A) \\
\text{success of } \mathcal{A} &:= [\mathbf{Ver}(m^*, \sigma^*, \mathsf{PK}_A, \mathsf{APK}) = \top \\
&\quad \wedge \text{ for any } S \in Query(\mathcal{A}, O_{\mathbf{Setup}^{\mathsf{Cred}}}), \\
&\quad\quad \text{for any } (m^*, \mathsf{PK}_A, \mathbb{A}, \cdot) \in Query(\mathcal{A}, O_{\mathbf{Res}}), \\
&\quad\quad S \text{ does not satisfy } \mathbb{A}]
\end{aligned}$$

where oracles $O_{\mathbf{Tran_P}}$, $O_{\mathbf{Res}}$ and $Query(\mathcal{A}, O_{\mathbf{Res}})$ are described in the previous experiment, oracle $O_{\mathbf{Setup}^{\mathsf{Cred}}}$ takes as input a set of attributes S, and outputs a credential CD_S, and $Query(\mathcal{A}, O_{\mathbf{Setup}^{\mathsf{Cred}}})$ is the set of queries $\mathcal{A}$ issued to the oracle $O_{\mathbf{Setup}^{\mathsf{Cred}}}$.

- **Security Against the Arbitrator**: This property guarantees that even when given the credential secret key, the arbitrator should not be able to create a

valid full signature unless it has seen a corresponding transaction promise. Formally, consider the following experiment, in which the two-stage adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ models the dishonest arbitrator that may even have access the credential key.

Experiment SAA:

$$\begin{aligned}
(\mathsf{PM}, \mathsf{CK}) &\leftarrow \mathbf{PMGen}(1^k, U) \\
(\mathsf{APK}, \mathsf{ASK}^*) &\leftarrow \mathcal{A}_1(\mathsf{PM}, \mathsf{MK}) \\
(\mathsf{SK}_A, \mathsf{PK}_A) &\leftarrow \mathbf{Setup}^{\mathsf{User}}(1^k, \mathsf{APK}) \\
(m^*, \sigma^*) &\leftarrow \mathcal{A}_2^{O_{\mathbf{Tran_P}}}(\mathsf{PM}, \mathsf{CK}, \mathsf{ASK}^*, \mathsf{APK}, \mathsf{PK}_A) \\
\text{success of } \mathcal{A} &:= [\mathbf{Ver}(m^*, \sigma^*, \mathsf{PK}_A, \mathsf{APK}) = \top \\
&\quad \wedge (m^*, \cdot, \cdot, \cdot) \notin \mathit{Query}(\mathcal{A}, O_{\mathbf{Tran_P}})]
\end{aligned}$$

where the partial signing oracle $O_{\mathbf{Tran_P}}$ is described in the previous experiment, ASK^* is $\mathcal{A}$'s state information, which might not be the corresponding secret key of APK, and $\mathit{Query}(\mathcal{A}, O_{\mathbf{Tran_P}})$ is the set of queries $\mathcal{A}$ issued to oracle $O_{\mathbf{Tran_P}}$.

Definition 8.2 *An attribute-based optimistic fair exchange scheme is said to be secure in the multi-user setting and chosen-key model if there is no PPT adversary that wins any of the experiments above with non-negligible probability.*

8.3 Construction

In this section, a generic construction of ABOFE that is secure in the defined model is given.

Let $\mathcal{E} := (\mathsf{Setup}, \mathsf{Encrypt}, \mathsf{KeyGen}, \mathsf{Decrypt})$ be a ciphertext-policy attribute-based encryption scheme and $\mathsf{OFE} = (\mathsf{Setup}^{\mathsf{TTP}}, \mathsf{Setup}^{\mathsf{User}}, \mathsf{PSig}, \mathsf{PVer}, \mathsf{Sig}, \mathsf{Ver}, \mathsf{Res})$ be a conventional optimistic fair exchange scheme. Below are the details of the generic construction of an attribute-based optimistic fair exchange scheme.

- **PMGen**: On input 1^k and attribute universe description U, this algorithm runs $\mathcal{E}.\mathsf{Setup}(1^k, U) \to (\mathrm{PM}, \mathrm{MK})$. The public parameter and credential secret key are set as $\mathsf{PM} := \mathrm{PM}$ and $\mathsf{CK} := \mathrm{MK}$, respectively.

- **Setup**$^{\mathsf{TTP}}$: On input 1^k, this algorithm runs $\mathsf{OFE.Setup}^{\mathsf{TTP}}(1^k) \rightarrow (ASK, APK)$. The arbitrator's secret and public key pair is set as $\mathsf{ASK} := ASK$ and $\mathsf{APK} := APK$.

- **Setup**$^{\mathsf{User}}$: This algorithm runs $\mathsf{OFE.Setup}^{\mathsf{User}}(1^k, APK) \rightarrow (SK_i, PK_i)$. The user's secret and public key pair is set as $\mathsf{SK}_i := SK_i, \mathsf{PK}_i := PK_i$.

- **Setup**$^{\mathsf{Cred}}$: On input the credential secret key CK and a set of attributes S, this algorithm runs $\mathcal{E}.\mathsf{KeyGen}\ (\mathsf{MK}, S) \rightarrow \mathrm{SK}$. The credential is set as $\mathsf{CD}_S := \mathrm{SK}$.

- **Tran**$_{\mathsf{P}}$: Taking as input $(m_1, m_2, \mathsf{SK}_i, \mathsf{APK}, \mathsf{PM}, \mathbb{A}_1, \mathbb{A}_2)$, user U_i runs $\mathsf{OFE.PSig}\ (m_1, \mathsf{SK}_i,\ \mathsf{APK}) \rightarrow \sigma_P$, and generates a non-repudiation information θ about the transaction[1] with respect to the tuple $(\sigma_P, m_1,\ m_2, \mathbb{A}_1, \mathbb{A}_2)$. The transaction promise ω is set as (σ_P, θ).

- **TPVer**: Taking as input $(m_1, m_2, \omega, \mathsf{PK}_i, \mathsf{APK}, \mathsf{PM}, \mathbb{A}_1, \mathbb{A}_2)$, it outputs $\mathsf{OFE.PVer}\ (m_1, \sigma_P,\ \mathsf{PK}_i,\ \mathsf{APK})$.

- **Tran**$_{\mathsf{S}}$: Taking as input $(m, \mathsf{SK}_i,\ \mathsf{APK}, \mathsf{PM}, \mathbb{A})$, user U_i runs $\mathsf{OFE.Sig}(m,\ \mathsf{SK}_i,\ \mathsf{APK}) \rightarrow s$, and then encrypts s under the access structure $\mathbb{A}$, i.e., $\mathcal{E}.\mathsf{Encrypt}\ (\mathsf{PM}, s, \mathbb{A}) \rightarrow \mathrm{CT}$. The credential-protected package is set as $\pi := \mathrm{CT}$.

- **Tran**$_{\mathsf{V}}$: Taking as input $(m, \pi, \mathsf{PK}_i,\ \mathsf{APK}, \mathsf{PM}, \mathsf{CD}_S)$, the verifier that possesses a set of attributes S that satisfies the access structure $\mathbb{A}$ will be able to convert the credential-protected package to a full signature. The verifier outputs $\sigma \leftarrow \mathcal{E}.\mathsf{Decrypt}(\mathsf{PM}, \pi, \mathsf{CD}_S)$.

- **Ver**: Taking as input $(m, \sigma, \mathsf{PK}_i, \mathsf{APK})$, it outputs $\mathsf{OFE.Ver}(m, \sigma,\ \mathsf{PK}_i,\ \mathsf{APK})$.

- **Res**$_{\mathsf{User}}$: Taking as input $(m_1, m_2, \omega, \mathsf{SK}_j, \mathsf{PK}_i, \mathsf{APK}, \mathsf{PM}, \mathbb{A}_1, \mathbb{A}_2)$, it runs $\mathsf{OFE.Sig}\ (m_2, \mathsf{SK}_j\ \mathsf{APK}) \rightarrow \sigma_2$. The resolution request is set as $\mathsf{request} := (m_2, \omega, \mathsf{PK}_j, \sigma_2, \mathbb{A}_1)$.

[1]This is used to identify the transaction that is going on and will be necessary when the arbitrator makes a resolution. Typically this can be achieved by signing $\sigma_P||m_1||m_2||\mathbb{A}_1||\mathbb{A}_2$ using an independent signature key pair.

- $\mathbf{Res}_{\mathsf{TTP}}$: Taking as input $(m, \mathsf{PK}_i, \mathbb{A}, \mathsf{request}, \mathsf{ASK})$, the arbitrator firstly parses $\mathsf{request}$ as $(m_2, \omega, \mathsf{PK}_j, \sigma_2, \mathbb{A}_1)$, and checks whether $\mathbf{TPVer}$ $(m_1, m_2, \omega, \mathsf{PK}_i, \mathsf{APK}, \mathsf{PM}, \mathbb{A}_1, \mathbb{A}_2) = \top$ and whether $\mathbf{Ver}(m_2, \sigma_2, \mathsf{PK}_j, \mathsf{APK}) = \top$. If either does not hold, it returns $\perp$. Otherwise, the arbitrator parses ω as (σ_P, θ), computes $\sigma_1 \leftarrow \mathsf{OFE.Res}(m, \sigma_P, \mathsf{ASK}, \mathsf{PK}_i)$ and then encrypts σ_1 under the access structure $\mathbb{A}_2$, i.e., $\mathcal{E}.\mathsf{Encrypt}(\mathsf{PM}, \sigma_1, \mathbb{A}_2) \rightarrow \mathrm{CT}_1$. The arbitrator also encrypts σ_2 under the access structure $\mathbb{A}_1$, i.e., $\mathcal{E}.\mathsf{Encrypt}(\mathsf{PM}, \sigma_2, \mathbb{A}_1) \rightarrow \mathrm{CT}_2$. The credential-protected packages are set as $\pi_1 := \mathrm{CT}_1$ and $\pi_2 := \mathrm{CT}_2$.

8.3.1 Security Analysis

Regarding the security of the generic construction of ABOFE, it has the following theorem.

Theorem 8.1 *The generic construction of ABOFE is secure in the multi-user setting and chosen-key model if the optimistic fair exchange scheme* OFE *is secure in the multi-user setting and chosen-key model and the ciphertext-policy attribute-based encryption scheme* $\mathcal{E}$ *is fully secure.*

Theorem 8.1 is proved by the following three lemmas.

Lemma 8.2 (Security against Signers) *The generic construction is secure against signers if* $\mathsf{OFE} = (\mathsf{Setup}^{\mathsf{TTP}}, \mathsf{Setup}^{\mathsf{User}}, \mathsf{PSig}, \mathsf{PVer}, \mathsf{Sig}, \mathsf{Ver}, \mathsf{Res})$ *is secure against signers.*

Proof. To show security against signers, one converts any adversary $\mathcal{A}$ that wins the experiment $\mathtt{SAS}$ into an adversary $\mathcal{A}'$ that breaks the security against signers of OFE. Recall that $\mathcal{A}'$ gets APK as input and has access to oracle O'_{Res}. $\mathcal{A}'$ runs $\mathbf{PMGen}(1^k, U) \rightarrow (\mathsf{PM}, \mathsf{CK})$ and invokes $\mathcal{A}$ on input PM, CK and $\mathsf{APK} := APK$.

Given a resolution for arbitrator query $(m, \mathsf{PK}_i, \mathbb{A}, \mathsf{request})$ to $O_{\mathbf{Res}}$, $\mathcal{A}'$ firstly parses $\mathsf{request}$ as $(m_2, \omega, \mathsf{PK}_j, \sigma_2, \mathbb{A}')$ and checks whether $\mathbf{TPVer}$ $(m_1, m_2, \omega, \mathsf{PK}_i, \mathsf{APK}, \mathsf{PM}, \mathbb{A}', \mathbb{A}) = \top$ and whether $\mathbf{Ver}(m_2, \sigma_2, \mathsf{PK}_j, \mathsf{APK}) = \top$. If either does not hold, it returns $\perp$. Otherwise, $\mathcal{A}'$ parses ω as (σ_P, θ), and makes a query $(m, \sigma_P, \mathsf{PK}_i)$ to its own oracle O'_{Res}. Denote the answer from O'_{Res} is σ_1. $\mathcal{A}'$ encrypts σ_1 under the access structure $\mathbb{A}$, i.e., $\mathcal{E}.\mathsf{Encrypt}(\mathsf{PM}, \sigma_1, \mathbb{A}) \rightarrow \mathrm{CT}_1$. $\mathcal{A}'$ also encrypts σ_2 under the access structure $\mathbb{A}'$, i.e., $\mathcal{E}.\mathsf{Encrypt}(\mathsf{PM}, \sigma_2, \mathbb{A}') \rightarrow \mathrm{CT}_2$. The credential-protected packages are set as $\pi_1 := \mathrm{CT}_1$ and $\pi_2 := \mathrm{CT}_2$. $\mathcal{A}'$ returns (π_1, π_2) to $\mathcal{A}$.

It can be seen that the resolution for arbitrator oracle $O_{\mathbf{Res}}$ is perfectly simulated by $\mathcal{A}'$. Finally, $\mathcal{A}$ outputs a tuple $(m_1^*, m_2^*, (\sigma_P^*, \theta^*), \mathsf{PK}_A, \mathbb{A}_1, \mathbb{A}_2, S)$ where the set of attributes S satisfies the access structure $\mathbb{A}_2$. $\mathcal{A}'$ outputs $(m_1^*, \sigma_P^*, \mathsf{PK}_A)$.

Denote (π_1^*, π_2^*) as the output of $\mathbf{Res}_{\mathsf{TTP}}(m_1^*, \mathsf{PK}_A, \mathbb{A}_2, \mathsf{Request}^*, \mathsf{ASK})$, and σ^* as the output of $\mathbf{Tran}_{\mathsf{V}}$ $(m_1^*, \pi_1^*, \mathsf{PK}_A, \mathsf{APK}, \mathsf{PM}, \mathsf{CD}_S)$ where $\mathsf{CD}_S = \mathbf{Setup}^{\mathsf{Cred}}(\mathsf{CK}, S)$. Due to the fact that the set of attributes S satisfies the access structure $\mathbb{A}_2$, σ^* is in fact also the output of $\mathsf{OFE.Res}(m_1^*, \sigma_P^*, \mathsf{ASK}, \mathsf{PK}_A)$. Since $\mathbf{TPVer}(m_1^*, m_2^*, (\sigma_P^*, \theta^*)$, $\mathsf{PK}_A, \mathsf{APK}, \mathsf{PM}, \mathbb{A}_1, \mathbb{A}_2) = \top$ but $\mathbf{Ver}(m_1^*, \sigma^*, \mathsf{PK}_A, \mathsf{APK}) = \bot$, it means $\mathsf{OFE.PVer}$ $(m_1^*, \sigma_P^*, PK_A, APK) = \top$ but $\mathsf{OFE.Ver}$ $(m_1^*, \sigma^*, PK_A, APK) = \bot$. Thus if $\mathcal{A}$ succeeds in the experiment, $\mathcal{A}'$ also succeeds with the same probability in breaking the security against signers of OFE. □

Lemma 8.3 (Security against Verifiers) *The generic construction is secure against verifiers if* $\mathsf{OFE} = (\mathsf{Setup}^{\mathsf{TTP}}, \mathsf{Setup}^{\mathsf{User}}, \mathsf{PSig}, \mathsf{PVer}, \mathsf{Sig}, \mathsf{Ver}, \mathsf{Res})$ *is secure against verifiers and the ciphertext-policy attribute-based encryption* $\mathcal{E} := (\mathsf{Setup}, \mathsf{Encrypt}, \mathsf{KeyGen}, \mathsf{Decrypt})$ *is fully secure.*

Proof. To show security against verifiers, one considers the experiments `SAV1` and `SAV2`, respectively. One firstly converts any adversary $\mathcal{A}$ that wins the experiment `SAV1` into an adversary $\mathcal{A}'$ that breaks the security against verifiers of OFE.

Recall that $\mathcal{A}'$ gets (APK, PK_A) as input and has access to oracles O'_{PSig} and O'_{Res}. $\mathcal{A}'$ runs $\mathbf{PMGen}(1^k, U) \to (\mathsf{PM}, \mathsf{MK})$ and invokes $\mathcal{A}$ on input PM, CK, $\mathsf{APK} := APK$ and $\mathsf{PK}_A := PK_A$.

Given a transaction promise query $(m_1, m_2, \mathbb{A}, \mathbb{A}')$ to oracle $O_{\mathbf{Tran_P}}$, $\mathcal{A}'$ makes a query m to its own oracle O'_{PSig}. Denote the answer from O'_{PSig} is σ_P. $\mathcal{A}'$ returns (σ_P, θ) to $\mathcal{A}$ where θ is a non-repudiation information about the transaction with respect to the tuple $(\sigma_P, m_1, m_2, \mathbb{A}_1, \mathbb{A}_2)$.

Given a resolution for arbitrator query $(m, \mathsf{PK}_i, \mathbb{A}, \mathsf{request})$ to $O_{\mathbf{Res}}$, $\mathcal{A}'$ firstly parses $\mathsf{request}$ as $(m_2, \omega, \mathsf{PK}_j, \sigma_2, \mathbb{A}')$ and checks whether $\mathbf{TPVer}$ $(m_1, m_2, \omega, \mathsf{PK}_i$, $\mathsf{APK}, \mathsf{PM}, \mathbb{A}', \mathbb{A}) = \top$ and whether $\mathbf{Ver}(m_2, \sigma_2, \mathsf{PK}_j, \mathsf{APK}) = \top$. If either does not hold, it returns $\bot$. Otherwise, $\mathcal{A}'$ parses ω as (σ_P, θ), and makes a query $(m, \sigma_P$, $\mathsf{PK}_i)$ to its own oracle O'_{Res}. Denote the answer from O'_{Res} is σ_1. $\mathcal{A}'$ encrypts σ_1 under the access structure $\mathbb{A}$, i.e., $\mathcal{E}.\mathsf{Encrypt}(\mathsf{PM}, \sigma_1, \mathbb{A}) \to \mathrm{CT}_1$. $\mathcal{A}'$ also encrypts σ_2 under the access structure $\mathbb{A}'$, i.e., $\mathcal{E}.\mathsf{Encrypt}(\mathsf{PM}, \sigma_2, \mathbb{A}') \to \mathrm{CT}_2$. The credential-protected packages are set as $\pi_1 := \mathrm{CT}_1$ and $\pi_2 := \mathrm{CT}_2$. $\mathcal{A}'$ returns (π_1, π_2) to $\mathcal{A}$.

It can be seen that the oracles $O_{\mathbf{PSig}}$ and $O_{\mathbf{Res}}$ are perfectly simulated by $\mathcal{A}'$. Finally, $\mathcal{A}$ outputs a tuple (m^*, σ^*) such that $\mathbf{Ver}(m^*, \sigma^*, \mathsf{PK}_A, \mathsf{APK}) = \top$. This means $\mathsf{OFE.Ver}(m^*, \sigma^*, PK_A, APK) = \top$. $\mathcal{A}'$ outputs (m^*, σ^*). Since $\mathcal{A}$ is prohibited from making a query $(m^*, \mathsf{PK}_A, \cdot, \cdot)$ to oracle $O_{\mathbf{Res}}$, $\mathcal{A}'$ has never made a query with respect to a tuple $(m^*, \cdot, \mathsf{PK}_A)$ to its own oracle O'_{Res}. If $\mathcal{A}$ succeeds in the experiment `SAV1`, $\mathcal{A}'$ also succeeds in breaking the security against verifiers of OFE. Thus $\mathcal{A}'$'s advantage is also non-negligible.

Next, one considers the experiment `SAV2`. One converts any adversary $\mathcal{A}$ that wins the experiment `SAV2` into an algorithm $\mathcal{B}$ that breaks the full security of the encryption scheme $\mathcal{E}$.

Recall that $\mathcal{B}$ gets PM as input and can query its own challenger for private keys corresponding to a sequence of sets of attributes. $\mathcal{B}$ runs $\mathbf{Setup}^{\mathsf{TTP}}(1^k) \rightarrow (\mathsf{ASK}, \mathsf{APK})$ and $\mathbf{Setup}^{\mathsf{User}}(1^k, \mathsf{APK}) \rightarrow (\mathsf{SK}_A, \mathsf{PK}_A)$, and invokes $\mathcal{A}$ as a subroutine by forwarding $\mathsf{PM} :=$ PM, APK and PK_A.

When $\mathcal{A}$ makes a query S to oracle $O_{\mathbf{Setup}^{\mathsf{Cred}}}$, $\mathcal{B}$ supplies S to its own challenger and forwards the reply to $\mathcal{A}$. When $\mathcal{A}$ makes a transaction promise query $(m_1, m_2, \mathbb{A}, \mathbb{A}')$ to oracle $O_{\mathbf{Tran}_{\mathsf{P}}}$, $\mathcal{B}$ runs $\mathbf{Tran}_{\mathsf{P}}(m_1, m_2, \mathsf{SK}_A, \mathsf{APK}, \mathsf{PM}, \mathbb{A}_1, \mathbb{A}_2)\omega$ and send ω to $\mathcal{A}$ as the reply.

Suppose $\mathcal{A}$ makes q valid resolution for arbitrator queries to $O_{\mathbf{Res}}$. $\mathcal{B}$ chooses a random value $z \in \{1, \cdots, q\}$. When $\mathcal{A}$ makes a j-th resolution query $(m_j, , \mathsf{PK}_j, \mathbb{A}_j, \mathsf{request}_j)$ to $O_{\mathbf{Res}}$, if $j \neq z$, $\mathcal{B}$ runs $\mathbf{Res}_{\mathsf{TTP}}(m_z, \mathsf{PK}_j, \mathbb{A}_j, \mathsf{request}_j, \mathsf{ASK})$ and sends the outputs to $\mathcal{A}$ as the reply. When $j = z$ and $\mathsf{PK}_z \neq \mathsf{PK}_A$, $\mathcal{B}$ aborts and returns failure. When $j = z$ and $\mathsf{PK}_z = \mathsf{PK}_A$, $\mathcal{B}$ parses $\mathsf{request}_z$ as $(m_2, \omega_j, \mathsf{PK}_i, \sigma_2, \mathbb{A}'_z)$ and parses ω_z as (σ_{Pz}, θ_z). $\mathcal{B}$ runs $\mathsf{OFE.Res}(m_z, \sigma_{Pz}, \mathsf{ASK}, \mathsf{PK}_A) \rightarrow \sigma_z$, randomly chooses M_1 that is of the same bit length with σ_z, and sends $M_0 := \sigma_z$, M_1 and the access structure $\mathbb{A}_z$ to its own challenger. Denote the challenge ciphertext is CT. $\mathcal{B}$ also encrypts σ_2 under the access structure $\mathbb{A}'_z$, i.e., $\mathcal{E}.\mathsf{Encrypt}(\mathsf{PM}, \sigma_2, \mathbb{A}') \rightarrow \mathrm{CT}'$. The credential-protected packages are set as $\pi_1 :=$ CT and $\pi_2 :=$ CT$'$. $\mathcal{A}'$ returns (π_1, π_2) to $\mathcal{A}$ as the reply to the z-th resolution for arbitrator query.

Finally $\mathcal{A}$ either outputs failure or wins by outputting a tuple (m^*, σ^*) such that $\mathbf{Ver}(m^*, \sigma^*, \mathsf{PK}_A, \mathsf{APK}) = \top$. Note that when $\mathcal{A}$ wins, $\mathcal{A}$ must have made a resolution query $(m^*, \mathsf{PK}_A, \cdot, \cdot)$ to $O_{\mathbf{Res}}$. Otherwise the analysis of this type of attack is covered in the experiment `SAV1` discussed above. If $m^* = m_z$, $\mathsf{PK}_A = \mathsf{PK}_z$, and $\mathbb{A}_2 = \mathbb{A}'_z$, then $\mathcal{B}$ outputs 0. Otherwise, $\mathcal{B}$ outputs a random bit.

If the challenge ciphertext CT is the encryption of σ_z (i.e., $b = 0$), CT is a valid attribute-related signature and the distribution of B's view in the simulated environment is identical with that in the real attack environment. If the challenge ciphertext CT is the encryption of M_1 (i.e., $b = 1$), CT has no information on the full signature of a message m_z and B's chance of forging a valid full signature on message m_z (with respect to PK_A) is negligible. Thus if $\mathcal{A}$ in the real experiment SAV2 wins with non-negligible probability, then $\mathcal{B}$ wins with non-negligible probability in breaking the full security of the encryption scheme $\mathcal{E}$. □

Lemma 8.4 (Security against the Arbitrator) *The generic construction is secure against the arbitrator if* $\mathsf{OFE} = (\mathsf{Setup}^{\mathsf{TTP}}, \mathsf{Setup}^{\mathsf{User}}, \mathsf{PSig}, \mathsf{PVer}, \mathsf{Sig}, \mathsf{Ver}, \mathsf{Res})$ *is secure against the arbitrator.*

Proof. To show security against the arbitrator, one converts any adversary $\mathcal{A}$ that wins the experiment SAA into an adversary $\mathcal{A}'$ that breaks the security against the arbitrator of OFE. $\mathcal{A}$ firstly chooses a public adjudication key APK and outputs it, keeps a corresponding secret state information ASK^* private. $\mathcal{A}'$ sets $APK := \mathsf{APK}$, gets PK_A as input, and has access to oracles O'_{PSig}. $\mathcal{A}'$ runs $\mathbf{PMGen}(1^k, U) \rightarrow (\mathsf{PM}, \mathsf{CK})$ and forwards PM, CK and $\mathsf{PK}_A := PK_A$ to $\mathcal{A}$.

Given a transaction promise query $(m_1, m_2, \mathbb{A}, \mathbb{A}')$ to oracle $O_{\mathbf{Tran_P}}$, $\mathcal{A}'$ makes a query m to its own oracle O'_{PSig}. Denote the answer from O'_{PSig} is σ_P. $\mathcal{A}'$ returns (σ_P, θ) to $\mathcal{A}$ where θ is a non-repudiation information about the transaction with respect to the tuple $(\sigma_P, m_1, m_2, \mathbb{A}_1, \mathbb{A}_2)$.

It can be seen that the oracle $O_{\mathbf{Tran_P}}$ is perfectly simulated by $\mathcal{A}'$. Finally, $\mathcal{A}$ outputs a tuple (m^*, σ^*) such that $\mathbf{Ver}\ (m^*, \sigma^*, \mathsf{PK}_A, \mathsf{APK} = \top$. This means $\mathsf{OFE.Ver}$ $(m^*, \sigma^*, PK_A, APK) = \top$. $\mathcal{A}'$ outputs (m^*, σ^*). Since $\mathcal{A}$ is prohibited from making a query $(m^*, \cdot, \cdot, \cdot)$ to oracle $O_{\mathbf{Tran_P}}$, $\mathcal{A}'$ has never made a query message m^* to its own oracle O'_{PSig}. If $\mathcal{A}$ succeeds in the experiment SAV2, $\mathcal{A}'$ also succeeds with the same probability in breaking the security against the arbitrator of OFE. □

8.4 Instantiation

In the following, an instantiation is provided to demonstrate the flexibility of the generic construction. It was shown in [HYWS08b] that OFE secure in the multi-user setting and chosen key model can be constructed from conventional signatures and ring signatures. More specifically, in this paradigm, the partial signature in OFE

is a conventional signature, and the full signature in OFE is the partial signature, together with a two party ring signature generated between the signer and the arbitrator. The authors [HYWS08b] suggest a concrete OFE scheme can be built on Waters signature scheme [Wat05] in a group of composite order and Shacham-Waters' ring signature scheme [SW07] so that they share the same set of system parameters. For the CP-ABE scheme, the one proposed by Lewko and Waters in [LW12] is employed, which is known to be fully secure. Note that there is a global setup process before execution of the scheme due to the requirement of having such a setup process of Shacham-Waters' ring signature.

Global Setup : On input 1^k where k is a security parameter, the setup algorithm generates a multiplicative cyclic group G of composite order n_1 where $n_1 = pq$ and a bilinear pairing $e : G \times G \to G_T$ where G_T is a multiplicative group of order n_1. Let G_p and G_q be the cyclic order-p and order-q subgroups of G, respectively. Let g be a generator of G and h be a generator of G_q. It then chooses random exponents $a, b \in \mathbb{Z}_{n_1}$ and sets

$$A := g^a, B := g^b, \hat{A} := h^a.$$

Let $H_0 : \{0,1\}^* \to \{0,1\}^k$ be a collision-resistant hash function. The setup algorithm picks Waters hash generators

$$u', u_1, \cdots, u_k \leftarrow G.$$

The common reference string is set as $(n_1, A, B, \hat{A}, u', u_1, \cdots, u_k, H_0)$.

After the global setup is finished, the algorithms in ABOFE can be executed as follows.

- **PMGen**: On input 1^k where k is a security parameter, the setup algorithm outputs a multiplicative cyclic group $\hat{G}$ of order n_2 where $n_2 = p_1p_2p_3$ (3 distinct primes) and a bilinear pairing $\hat{e} : \hat{G} \times \hat{G} \to \hat{G}_T$ where $\hat{G}_T$ is a multiplicative group of order n_2. Let $\hat{G}_{p_i}$ denote the subgroup of order p_i in $\hat{G}$. Let g_i be a generator of $\hat{G}_{p_i}$ and $H_1 : \hat{G}_T \to \{0,1\}^l$ be a collision-resistant hash function. It then chooses random exponents $\alpha, c, \gamma \in \mathbb{Z}_n$, and for each attribute $i \in U$, it chooses a random group element $h_i \in \hat{G}_{p_1}$. Let SE be a symmetric encryption scheme whose symmetric key space is $\{0,1\}^l$. The public parameters PM are set as

$$n_2, g_1, \hat{A}, g_1^c, g_1^\gamma, e(g_1, g_1)^\alpha, H_1, h_1, \cdots, h_{|U|}, \mathsf{SE}.$$

The master key are set as $\mathsf{MK} := (g_1^{\alpha}, g_3)$.

- **Setup**$^{\mathsf{TTP}}$: The arbitrator chooses a random exponent $y \in \mathbb{Z}_{n_1}$, and sets $\mathsf{APK} := g^y \in G$ and $sk := A^y \in G$.

- **Setup**$^{\mathsf{User}}$: Each user U_i randomly chooses exponents $x_1, x_3 \in \mathbb{Z}_{n_1}$ and computes $\bar{g}_1 = g^{x_1}$ and $\bar{g}_3 = g^{x_3}$. Besides, user U_i randomly chooses $\bar{g}_2, \bar{u}', \bar{u}_1, \cdots, \bar{u}_k \in G$ and sets $\mathsf{PK}_i := (\bar{g}_1, \bar{g}_2, \bar{g}_3, \bar{u}', \bar{u}_1, \cdots, \bar{u}_k)$ and $\mathsf{SK} := (\bar{g}_2^{x_1}, A^{x_3})$.

- **Setup**$^{\mathsf{Cred}}$: On input the credential secret key CK and a set of attributes S, the algorithm chooses random exponents $t, u \in \mathbb{Z}_{n_2}$, and random elements $R, R', R'', \{R_i\}_{i \in S} \in G_{p_3}$ (this can be done by raising a generator of G_{p_3} to random exponents modulo n_2). The credential is:

$$S,\ K = g_1^{\alpha} g_1^{ct} g_1^{\gamma u} R,\ K' = g_1^{u} R',\ K'' = g^{t} R'',\ K_i = h_i^{t} R_i \text{ for } i \in S.$$

- **Tran**$_{\mathsf{P}}$: **Tran**$_{\mathsf{P}}(m_A, m_B, \mathsf{SK}_i, \mathsf{APK}, \mathsf{PM}, (M, \rho), (M', \rho'))$, where M is an $L \times N$ matrix and ρ a map from each row M_j of M to an attribute $\rho(j)$, and M' is an $L' \times N'$ matrix and ρ' a map from each row M'_j of M' to an attribute $\rho'(j)$, does as follows:

  1. Compute $(m_1, \cdots, m_k) \leftarrow H_0(m_A)$,
  2. Choose a random exponent $r \in \mathbb{Z}_{n_1}$ and computes

$$S_1 = \bar{g}_2^{x_1} \cdot (\bar{u}' \prod_{i=1}^{k} \bar{u}_i^{m_i})^r, \quad \text{and} \quad S_2 = g^r.$$

  3. σ_P is set as (S_1, S_2).
  4. Generate a non-repudiation information θ about the transaction with respect to the tuple $(\sigma_P, m_A, m_B, (M, \rho), (M', \rho'))$. The transaction promise ω is set as (σ_P, θ).

- **TPVer**: **TPVer**$(m_A, m_B, \omega, \mathsf{PK}_i, \mathsf{APK}, \mathsf{PM}, (M, \rho), (M', \rho'))$ does as follows.

  1. Parses ω as (σ_P, θ).

2. Compute $(m_1, \cdots, m_k) \leftarrow H_0(m_A)$, and verify whether

$$e(S_1, g) \cdot e(S_2^{-1}, \bar{u}' \prod_{j=1}^{k} \bar{u}_j^{m_j}) = e(\bar{g}_1, \bar{g}_2)$$

holds. If so, output $\top$; if not, output $\perp$.

- **Tran$_\mathsf{S}$**: **Tran$_\mathsf{S}$**$(m, \mathsf{SK}_i, \mathsf{APK}, \mathsf{PM}, (M, \rho))$, where M is an $L \times N$ matrix and ρ a map from each row M_j of M to an attribute $\rho(j)$, does as follows:

1. Compute $(m_1, \cdots, m_k) \leftarrow H_0(m)$,
2. Choose a random exponent $r \in \mathbb{Z}_{n_1}$ and computes

$$S_1 = \bar{g}_2^{x_1} \cdot (\bar{u}' \prod_{i=1}^{k} \bar{u}_i^{m_i})^r, \quad \text{and} \quad S_2 = g^r.$$

3. Compute $(m'_1, \cdots, m'_k) \leftarrow H_0(m||S_1||S_2||\mathsf{PK}_i)$.
4. Choose two random exponents $t_0, t_1 \in \mathbb{Z}_{n_1}$ and sets

$$C'_0 = (\bar{g}_3/B)h^{t_0}, \pi_0 = ((\bar{g}_3/B)h^{t_0})^{t_0}, C'_1 = h^{t_1}, \pi_1 = ((\mathsf{APK}/B)^{-1}h^{t_1})^{t_1}.$$

5. Choose $r' \leftarrow \mathbb{Z}_{n_1}$, and compute first $t = t_1 + t_2$ and then

$$S'_1 = A^{x_3} \cdot (u' \prod_{j=1}^{k} u_j^{m'_j})^{r'} \cdot \hat{A}^t, \quad \text{and} \quad S'_2 = g^{r'}.$$

6. Choose a random vector $v \in \mathbb{Z}_{n_2}^N$, denoted $v = (s, v_2, \cdots, v_N)$. For each row M_j of M, it chooses a random $r_j \in \mathbb{Z}_{n_2}$.
7. Choose a random element $D \in G_T$, computes $sk = H_1(D)$, and uses sk as a symmetric key of SE to encrypt the bit strings $S_1||S_2||S'_1||S'_2||C'_0||C'_1||\pi_0||\pi_1$ and gains a ciphertext $\tilde{c}$.
8. The credential-protected package π is set as

$$\tilde{c},\ C_0 = De(g_1, g_1)^{\alpha s},\ C = g_1^s,\ C' = (g_1^\gamma)^s,$$

$$C_j = (g_1^c)^{M_j \cdot v} h_{\rho(j)}^{-r_j},\ D_j = g_1^{r_j},\ \text{for } j = 1, \cdots, L.$$

- **Tran$_\mathsf{V}$**: **Tran$_\mathsf{V}$**$(m, \pi, \mathsf{PK}_i, \mathsf{APK}, \mathsf{PM}, \mathsf{CD}_S)$ does as follows.

1. Computes constants $\omega_j \in \mathbb{Z}_{n_2}$ such that $\sum_{\rho(j)\in S} \omega_j M_j = (1, 0, \cdots, 0)$. It then computes

$$\hat{e}(C, K)\hat{e}(C', K')^{-1} / \prod_{\rho(j)\in S} (\hat{e}(C_j, K'')\hat{e}(D_j, K_{\rho(j)}))^{\omega_j} = \hat{e}(g_1, g_1)^{\alpha s}.$$

2. Recover D as $C_0/\hat{e}(g_1, g_2)^{\alpha s}$, compute $sk = H_1(D)$ and uses sk as a symmetric key of SE to decrypt the ciphertext c to gain a full signature σ.

- **Ver**: $\mathbf{Ver}(m, \sigma, \mathsf{PK}_i, \mathsf{APK})$ does as follows.

1. Parse σ as $(S_1, S_2, S_1', S_2', C_0, C_1, \pi_0, \pi_1)$, compute $(m_1, \cdots, m_k) \leftarrow H_0(m)$, and verify whether

$$e(S_1, g) \cdot e(S_2^{-1}, \bar{u}' \prod_{j=1}^{k} \bar{u}_j^{m_j}) = e(\bar{g}_1, \bar{g}_2)$$

holds. If so, output $\top$; if not, output $\bot$.

2. Check whether

$$e(C_0', C_0'/(\bar{g}_3/B)) = e(h, \pi_0) \quad \text{and} \quad e(C_1', C_1'/(\mathsf{APK}/B)) = e(h, \pi_1)$$

hold. If either does not hold, output $\bot$.

3. Compute $(m_1', \cdots, m_k') \leftarrow H_0(m||S_1||S_2||\mathsf{PK}_i)$ and $C_2' = C_0' \cdot C_1'$, and verify whether

$$e(A, BC_2') = e(S_1', g) \cdot e((S_2')^{-1}, u' \prod_{j=1}^{k} u_j^{m_j'})$$

holds. If so, output $\top$; if not, output $\bot$.

- $\mathbf{Res}_{\mathsf{User}}$: $\mathbf{Res}_{\mathsf{User}}(m_A, m_B, \omega, \mathsf{SK}_j, \mathsf{PK}_i, \mathsf{APK}, \mathsf{PM}, (M, \rho), (M', \rho'))$ does as follows.

1. Parses PK_j as $(\tilde{g}_1, \tilde{g}_2, \tilde{g}_3, \tilde{u}', \tilde{u}_1, \cdots, \tilde{u}_k)$ and $\mathsf{SK}_j := (\tilde{g}_2^{x_1'}, A^{x_3'})$.
2. Compute $(m_1, \cdots, m_k) \leftarrow H_0(m_B)$,
3. Choose a random exponent $r \in \mathbb{Z}_{n_1}$ and computes

$$S_1 = \tilde{g}_2^{x_1'} \cdot (\bar{u}' \prod_{i=1}^{k} \tilde{u}_i^{m_i})^r, \quad \text{and} \quad S_2 = g^r.$$

4. Compute $(m'_1, \cdots, m'_k) \leftarrow H_0(m||S_1||S_2||\mathsf{PK}_j)$.
5. Choose two random exponents $t_0, t_1 \in \mathbb{Z}_{n_1}$ and sets

$$C'_0 = (\tilde{g}_3/B)h^{t_0}, \pi_0 = ((\tilde{g}_3/B)h^{t_0})^{t_0}, C'_1 = h^{t_1}, \pi_1 = ((\mathsf{APK}/B)^{-1}h^{t_1})^{t_1}.$$

6. Choose $r' \leftarrow \mathbb{Z}_{n_1}$, and compute first $t = t_1 + t_2$ and then

$$S'_1 = A^{x_3} \cdot (u' \prod_{j=1}^{k} u_j^{m'_j})^{r'} \cdot \hat{A}^t, \quad \text{and} \quad S'_2 = g^{r'}.$$

7. Set $\sigma_B := (S_1, S_2, S'_1, S'_2, C_0, C_1, \pi_0, \pi_1)$.
8. Set $\mathsf{request} := (\omega, m_B, \sigma_B, \mathsf{PK}_j, (M, \rho))$.

- **$\mathbf{Res}_{\mathsf{TTP}}$**: $\mathbf{Res}_{\mathsf{TTP}}(m_A, \mathsf{PK}_i, (M', \rho'), \mathsf{request}, \mathsf{ASK})$ does as follows.

1. Parse $\mathsf{request}$ as $(\omega, m_B, \sigma_B, \mathsf{PK}_j, (M, \rho))$ and ω as σ_P and θ. If the non-repudiation information θ is not with respect to the tuple $(\sigma_P, m_A, m_B, (M, \rho), (M', \rho'))$, outputs $\perp$.
2. Parse σ_P as (S_1, S_2), compute $(m_1, \cdots, m_k) \leftarrow H_0(m_A)$, and verify whether

$$e(S_1, g) \cdot e(S_2^{-1}, \bar{u}' \prod_{j=1}^{k} \bar{u}_j^{m_j}) = e(\bar{g}_1, \bar{g}_2)$$

   holds. If not, output $\perp$.
3. Check whether $\mathbf{Ver}(m_B, \sigma_B, \mathsf{PK}_j, \mathsf{APK}) = \top$. If not, output $\perp$.
4. Compute $(m'_1, \cdots, m'_k) \leftarrow H_0(m_A||S_1||S_2||\mathsf{PK}_i)$.
5. Choose two random exponents $t_0, t_1 \in \mathbb{Z}_{n_1}$ and sets

$$C'_0 = h^{t_0}, \pi_0 = ((\bar{g}_3/B)^{-1}h^{t_0})^{t_0}, C'_1 = (\mathsf{APK}/B)h^{t_1}, \pi_1 = ((\mathsf{APK}/B)h^{t_1})^{t_1}.$$

6. Choose $r' \leftarrow \mathbb{Z}_{n_1}$, and compute first $t = t_1 + t_2$ and then

$$S'_1 = A^{y} \cdot (u' \prod_{j=1}^{k} u_j^{m'_j})^{r'} \cdot \hat{A}^t, \quad \text{and} \quad S'_2 = g^{r'}.$$

7. Choose a random vector $v \in \mathbb{Z}_{n_2}^{N'}$, denoted $v = (s, v_2, \cdots, v_{N'})$. For each row M_j of M, it chooses a random $r_j \in \mathbb{Z}_{n_2}$.

8. Choose a random element $D \in G_T$, computes $sk = H_1(D)$, and uses sk as a symmetric key of SE to encrypt the bit strings $S_1||S_2||S_1'||S_2'||C_0'||C_1'||\pi_0||\pi_1$ and gains a ciphertext $\tilde{c}_A$.
9. The credential-protected package π_A is set as

$$\tilde{c}_A,\ C_0 = De(g_1, g_1)^{\alpha s},\ C = g_1^s,\ C' = (g_1^{\gamma})^s,$$

$$C_j = (g_1^c)^{M_j' \cdot v} h_{\rho'(j)}^{-r_j},\ D_j = g_1^{r_j},\ \text{for } j = 1, \cdots, L'.$$

10. Choose a random vector $v' \in \mathbb{Z}_{n_2}^N$, denoted $v' = (s', v_2', \cdots, v_N')$. For each row M_j' of M', it chooses a random $r_j' \in \mathbb{Z}_{n_2}$.
11. Choose a random element $D' \in G_T$, computes $sk' = H_1(D')$, and uses sk' as a symmetric key of SE to encrypt the σ_B and gains a ciphertext $\tilde{c}_B$.
12. The credential-protected package π_B is set as

$$\tilde{c}_B,\ C_0 = De(g_1, g_1)^{\alpha s'},\ C = g_1^{s'},\ C' = (g_1^{\gamma})^{s'},$$

$$C_j = (g_1^c)^{M_j \cdot v'} h_{\rho(j)}^{-r_j'},\ D_j = g_1^{r_j'},\ \text{for } j = 1, \cdots, L.$$

13. The output is (π_A, π_B).

8.4.1 Efficency Analysis

Since pairing and exponentiation operations take more time than multiplication operations do, the costs of multiplication computations will be simply ignored here. Let $\mathrm{SE}, \mathrm{SD}, \mathrm{P}$ and E denote a symmetric encryption operation, a symmetric decryption operation and a pairing operation, and an exponentiation operation, respectively. Let M be an $L \times N$ matrix, ρ a map from each row M_j of M to an attribute $\rho(j)$, and S a set of attributes that satisfies the access structure (M, ρ). Denote s as the number of elements of the set $\{1 \leq j \leq L | \rho(j) \in S\}$.

The costs of the instantiation of ABOFE is presented in Table 8.1, in which $\mathbf{Tran}_{\mathsf{S}}$ is evaluated with respect to the access structure (M, ρ), and $\mathbf{Tran}_{\mathsf{V}}$ is evaluated with respect to the set of attributes S.

8.5 Chapter Summary

The notion of attribute-based optimistic fair exchange was proposed, and a formal security model was given. A generic construction of PAOFE was then proposed,

Algorithms:	$\mathbf{Tran_P}$	**TPVer**	$\mathbf{Tran_S}$	$\mathbf{Tran_V}$	**Ver**
Cost:	2E	3P	1SE, $(12 + 3L)$E	1SD, $(2s + 2)$P, (s)E	10P

Table 8.1: Costs of algorithms in the instantiation of ABOFE.

and its security is proved under the proposed model in the standard model. Since ABOFE can be used to solve the fair exchange problem when the attributes of the participants need to be taken into account, ABOFE can be viewed as an extension of OFE in the attribute-based setting. A more elegant solution based on fair exchange of attribute-based signatures is left as the future work.

Part IV

Fairness of Concurrent Signatures

Chapter 9

Fairness in Concurrent Signatures Revisited

In this chapter the fairness of concurrent signatures is revisited, and it is shown that the initial signer enjoys more advantages than it was acknowledged.

9.1 Introduction

Fairness in the exchange of digital signatures is normally achieved with the help of a trusted third party (TTP), which is often off-line. It is well-known that the role of the TTP cannot be replaced by a normal certification authority.

In Eurocrypt 2004 [CKP04], Chen, Kudla and Paterson presented a new cryptographic primitive called "concurrent signature" to allow two parties to produce two ambiguous signatures, such that both signatures do not bind to their true signers. Upon the release of the signatures, any third party cannot identify the true signer who generated the signature. However, upon the release of an extra piece of information called the keystone, both signatures will bind concurrently. Further, Chen, Kudla and Paterson presented a concrete concurrent signature scheme based on a variant of Schnorr based ring signature scheme [AOS02].

In a concurrent signature protocol run, there are two parties involved, namely Alice and Bob (or A and B, respectively). Since one party is required to create a keystone and sends the initial signature to the other party, this party is called the *initial signer*. A party who responds to the initial signature by creating another signature is called a *matching signer*. Without losing generality, throughout this chapter, it is assumed that Alice (or A, resp.) is the initial signer and Bob (or B, resp.) is the matching signer.

Due to the merit of not requiring a TTP, concurrent signatures have been promising in many electronic commerce applications, especially in the scenarios where a

TTP may not be available. A widely-used example is that concurrent signatures provide a novel solution to the old problem of fair tendering of contracts.

It is acknowledged that concurrent signatures fall short to fully solve the problem of fair exchange of digital signatures, since Alice is in control of the time at which the keystone is released and thus, control *when* the ambiguous signatures become binding to their respective signers. Alice can even decide not to complete a concurrent signature protocol run if Alice decided not to release the keystone ultimately. Nonetheless, the concept of *fairness* in a concurrent signature is defined as follows: "once Alice releases the keystone, both ambiguous signatures from Alice and Bob binds concurrently". Specifically, it is required that Alice cannot output a maliciously crafted keystone so that the ambiguous signature from Bob together with this keystone passes the verification algorithm, yet verification of the ambiguous signature created by Alice (perhaps also maliciously) together with this keystone would output failure. In this thesis this definition of fairness is called a "white-box" guarantee, as the malicious Alice is required to convert the ambiguous signature corresponding to Bob to a "well-formed" publicly verifiable signature under Bob to be considered successful. This is possibly sufficient for legal contract signing purpose, since the contract is valid if and only if a "well-formed" signature is present. This definition is adopted in Chen et al.'s paper and the subsequent works.

9.1.1 Fairness in Practice

In this chapter, the observation is made that the formal definition of fairness does *not* necessarily capture the fairness in practice completely. For example, there is *no guarantee* that Alice could not convince a third party Carol that Bob has signed a message, M_B, that is, committed to M_B, *without* revealing the keystone. It is identified that a malicious initial signer may enjoy three levels of advantages in concurrent signature.

(a) Level 0 advantage is inherent in concurrent signature. The initial signer can always choose to abort or complete a concurrent signature protocol run.

(b) Level 1 advantage allows the initial signer to demonstrate the fact that she is capable of making both signatures valid if she wanted to, without actually making both signatures publicly verifiable.

(c) Level 2 advantage allows the initial signer to convince a third party that the matching signer has committed to a certain message, for example, M_B, without revealing anything else.

These advantage levels have concrete implications regarding the use of concurrent signatures in practical scenarios. As discussed by Chen et al. in their seminal paper [CKP04], it is very often the case that the matching signer would not mind sacrificing level 0 advantage. However, regarding level 1 and 2 advantages, concurrent signatures may not be suitable in some other scenarios such as tendering systems (c.f. [CKP04] as it will be shown with details in the later part of this chapter). Hence, in those scenarios, the OFE [ASW97, ASW98, ASW00, DLY07] or Ambiguous OFE [HYWS08a, HWS11a, HWS12] systems are indeed more suitable compared to concurrent signatures.

9.1.2 The Contributions

Firstly, it is shown that any constructions of concurrent signature following Chen, Kudla and Peterson is *always* subject to abuse by the initial signer, with advantage level 1 and 2, in addition to the commonly acknowledged advantage level 0. Generic methods are presented that allow a malicious initial signer to convince any third party that he/she has the ability to make both signatures verifiable (level 1), or that the matching signer has committed to his message (level 2). Secondly, one variant of concurrent signatures, namely asymmetric concurrent signature [Ngu05], is examined and it is demonstrated how a malicious initial signer can exhibit his/her level 1 advantage in an effective manner. The attack is practical and its implication may discourage the adoption of concurrent signatures in some application scenarios, and particularly, when it is undesirable to allow a malicious initial signer to convince anyone of the binding of the matching signer's signature without making the signature publicly verifiable.

9.1.3 Related Work

Following the seminal work by Chen et al., many subsequent concurrent signature schemes have been proposed (such as [SMZ04, WBZ06, Ngu05, TSSN05]). Nguyen [Ngu05] proposed an interesting variant that embraces the asymmetric property of concurrent signatures (c.f. the symmetric property of all the previously known

concurrent signature schemes). Furthermore, Nguyen noted that the construction techniques of an asymmetric concurrent signature scheme can be used for constructing a multi-party concurrent signature scheme, which is the solution to the open problem stated in Chen et al.'s seminal work [CKP04]. Subsequently, Tonien, Susilo and Safavi-Naini [TSSN05] proposed a multi-party concurrent signature scheme that uses a different model from the construction achieved from [Ngu05].

In an orthogonal direction, Susilo, Mu and Zhang [SMZ04] investigated the privacy issue in concurrent signatures. They observed that prior the release of a keystone, and just from the two ambiguous signatures, any third party can already conclude that the two ambiguous signatures must be created by these two possible signers. At the same time, if the possible signers are believed to be honest, the outsider can already tell who the actual signer is corresponding to each ambiguous signature. They then introduced a stronger requirement called *perfect ambiguity*, which requires the ambiguous signatures to remain "ambiguous" even if the two potential signers are known to be honest. Unfortunately, their scheme is shown to be insecure by Wang, Bao and Zhou [WBZ06], and subsequently, Wang et al. proposed a modified scheme that is proven secure under this stronger notion.

Yuen, Wong, Susilo and Huang [YWSH11] constructed a concurrent signature variant that supports negotiation between the initial signer and the matching signer on who will control the final binding of the ambiguous signatures. They showed that their model is compatible with the original definition by Chen et al. [CKP04].

Very recently, Tan, Huang and Wong [THW12] presented the first concurrent signature scheme that is based on the standard assumption. Their ambiguity model is very similar to the one proposed by Yuen et al. [YWSH11].

9.1.4 Chapter Organization

In the next section, the classification of advantage levels to the initial signer is presented in detail, their implications are discussed and generic techniques which allows the initial signer to enjoy these advantages are presented. In Section 9.3, it is shown how an initial signer can enjoy level 1 advantage in the asymmetric concurrent signature scheme due to Nguyen [Ngu05]. Finally, the chapter is summarised in Section 9.5.

9.2 Abusing Fairness in Concurrent Signatures

In this section, the various advantage levels enjoyed by the initial signer, their implications and how they could be acquired are discussed. At the high level, it is often the case that exhibiting such advantage requires the use of zero-knowledge proof [GMR85].

9.2.1 Advantage Level 0

Level 0 advantage is inherent in concurrent signatures, as the initial signer, Alice, is in possession of the keystone, which is under the full control of Alice on when and whether the keystone will be released to the public. Thus, the primitive is not suitable if the matching signer will be at a disadvantage if withholding the keystone or delaying the release of the keystone would cause harm to the matching signer.

The Implications.

Consider the tendering systems as discussed in the seminal paper of Chen et al. [CKP04]. They described a scenario where A has a bridge-building contract that she would like to put out to tender. Suppose there are two companies B and C that wish to put in proposals to win this contract. In this scenario, B acts as the initial signer, as this is to prohibit A to show this contract to C to get a better proposal from C. Noted that in this particular scenario, B has the full control of the keystone, since the keystone is selected by B. Therefore, if at the end of the tender, if A would like to select B as the winner of the tender, B may still have the liberty for not completing the contract by not releasing the keystone, and hence, it is unfair to A.

9.2.2 Advantage Level 1

The Abuse.

Assume Alice is a malicious initial signer, whose purpose is to convince a third party Carol that the matching signer Bob and herself are about to exchange signatures on messages M_A and M_B. Assume Alice and Bob have completed step 3 of the concurrent signature protocol (as described in Section 2.5.1). That is, Alice is in possession of a keystone k and the ambiguous signature from Bob $\sigma_B := (s_B, h_B, f)$

on message M_B. At the same time, Bob is in possession of Alice's ambiguous signature $\sigma_A := (s_A, h_A, f)$ on message M_A. In the generic attack, Alice reveals σ_A, σ_B, M_A, M_B to Carol and then conducts a zero-knowledge proof-of-knowledge of the value k with Carol such that

$$f = \mathsf{KGEN}(k).$$

Carol is convinced that Alice and Bob are exchanging signatures on messages M_A and M_B and that Alice has the ability to complete the transaction.

The Implications.

Consider an open auction [WBZ06, YWSH11] in which Alice's ambiguous signature is a contract to sell a certain goods while Bob's ambiguous signature is a contract of a bid. Level-1 advantage allows Alice to convince Carol that she has the ability to seal the contract with Bob and the bid is specified in M_B. This allows Alice to safely urge Carol for a higher bid, and Bob is at a disadvantage.

Note that this implication is also applicable to the tendering systems as discussed previously. However, in this scenario, let us consider the case where A, who would like to put the bridge-building tender, is the initial signer. Hence, the company B and C will act as the matching signers. In this setting, A will take the advantage level 1 to convince C about B's tender, so that C will increase the value of her tender, and hence, disadvantaging B.

9.2.3 Advantage Level 2

The Abuse.

Assume Alice is a malicious initial signer, whose purpose is to convince a third party Carol that the matching signer Bob has committed to message M_B. Assume Alice and Bob have completed step 3 of the concurrent signature protocol (as described in Section 2.5.1). That is, Alice is in possession of a keystone k and the ambiguous signature from Bob $\sigma_B := (s_B, h_B, f)$ on message M_B. At the same time, Bob is in possession of Alice's ambiguous signature $\sigma_A := (s_A, h_A, f)$ on message M_A. The observation on the incompleteness of the original fairness definition in [CKP04] arises from the fact that to convince Carol about Bob's commitment to M_B *does not necessarily* involve outputting some maliciously crafted keystone $\hat{k}$. Specifically, in

the generic attack, Alice conducts a zero-knowledge proof-of-knowledge of the tuple (k, σ_B) with Carol such that the following statements are true:

1. $f = \mathsf{KGEN}(k)$, and
2. $\mathsf{AVERIFY}(\sigma_B, X_A, X_B, M_B) = 1$.

This would be sufficient to convince a third party about *Bob's intention* to sign a message M_B, without revealing anything about Alice's intention, thus undermining the fairness guarantees of the concurrent signature schemes, and put Bob into disadvantage.

The Implications.

Note that in the zero-knowledge proof, Alice does not reveal the keystone k, the keystone fix f, nor Bob's ambiguous signature. Thus, even if Carol is presented with the secret key of Bob, the ambiguous signatures σ_A and σ_B from Bob, she could not conclude that Alice has committed to M_A. In other words, the only thing that Carol can be assured of is that Alice is in possession of a signature from Bob on message M_B. Bob is left at an unfair position.

Note that both level 1 and 2 advantages are outside the security definition presented in Section 2.5.2. It is not claimed that existing concurrent signature schemes are broken for two reasons. Firstly, they are not within the security model. Secondly, even though they are theoretically possible, it is not always feasible. Thus, people should bear in mind any concurrent signature following this syntax *could* be abused by the initial signer, and hence, the adoption of concurrent signatures in application scenario should be examined to make sure that either the level 1 and 2 advantages of the initial signer are acceptable or that the advantages cannot be claimed efficiently.

9.3 Abusing Fairness in Asymmetric Concurrent Signatures

In this section, a practical abuse in advantage level 1 of the asymmetric concurrent signature due to Nguyen [Ngu05] is demonstrated. In this abuse, the initial signer

can convince any verifier that he/she has the ability to make both ambiguous signatures verifiable. It should be stressed again that the attack is *outside* their original model and therefore it is not claimed that Nguyen's original scheme is broken.

9.3.1 Review of Nguyen's asymmetric concurrent signatures

For completeness, Nguyen's scheme will be firstly reviewed.

Setup. Let $G = \langle g \rangle$ be a group of prime order p. The key pair of Alice and Bob are respectively $(X_A = g^{x_A}, x_A) \in G \times \mathbb{Z}_p$, $(X_B = g^{x_B}, x_B) \in G \times \mathbb{Z}_p$. Let $H : G \times \{0,1\}^* \to \mathbb{Z}_p$ be a hash function that would be modeled as random oracle.

Generation of Alice's ambiguous signature. On input a message $M_A \in \{0,1\}^*$ and Bob's public key X_B, Alice randomly generates $r \in_R \mathbb{Z}_p$ and computes $c = H(g^r, M_A)$, $s = g^{r+cx_A}$. Alice sets her ambiguous signature as (c, s) and sends it to Bob. The keystone is defined as $k = r + cx_A$ such that $s = g^k$.

Generation of Bob's ambiguous signature. On input a message $M_B \in \{0,1\}^*$ and an ambiguous signature (c, s) on message M_A from Alice, Bob first checks if $c = H(sX_A^{-c}, M_A)$. Bob continues if the check is successful. He computes $s' = s^{x_B}$, $c' = H(s'g^{r'}, M_B)$ and $k' = \frac{r'-c'}{x_B}$. He sets his ambiguous signature as (c', s', k') and sends it to Alice.

Binding of both signatures. Upon receiving (c', s', k') from Bob, Alice first checks if

$$c' = H(g^{c'} X_B^{k'} s', M_B)$$

and $s' = X_B^k$. If the check is successful and if Alice decides to have both signatures binded, she releases the value k. Both signatures are now publicly verifiable by checking the following verification equations.

Verification of Alice's signature:

$$c \stackrel{?}{=} H(X_A^{-c} s, M_A) \ \wedge \ s \stackrel{?}{=} g^k$$

Verification of Bob's signature:

$$c' \stackrel{?}{=} H(g^{c'} X_B^{k'} s', M_B) \ \wedge \ s' \stackrel{?}{=} X_B^k$$

Remark. Nguyen's construction is asymmetric in the sense that Alice's signature and Bob's signature are of different form with different verification equations.

9.3.2 A Concrete Level-1 Abuse on Nguyen's Scheme

Given the pair of ambiguous signatures (c, s), (c', s', k') on message M_A and M_B respectively, Alice, who is in possession of the keystone k satisfying the relationship $s = g^k$ and $s' = X_B^k$ can convince a third party Carol that she has the ability to bind both ambiguous signatures by proving she knows the value k in zero-knowledge. The full proof protocol is shown below.

1. Alice sends both (c, s), (c', s', k') M_A, M_B to Carol.
2. Carol checks $c = H(X_A^{-c}s, M_A)$ and $c' = H(g^{c'}X_B^{k'}s', M_B)$. If yes, she randomly generates two values $t_1, t_2 \in_R \mathbb{Z}_p$ and sends $T_0 = g^{t_1}h^{t_2}$ to Alice.
3. Alice randomly generates a value $t \in_R \mathbb{Z}_p$, computes $T_1 = g^t$, $T_2 = X_B^t$ and sends T_1, T_2 to Carol.
4. Carol sends t_1, t_2 to Alice
5. Alice checks if $T_0 = g^{t_1}h^{t_2}$. If yes, she computes $z = t - t_1 k$ and sends z to Carol.
6. Carol checks if $T_1 = s^{t_1}g^z$ and $T_2 = s'^{t_1}X_B^z$ and accepts the proof if both equations hold.

9.4 Comparison of Concurrent Signatures and OFE

In this section the properties of concurrent signatures and those of OFE are compared with respect to the identified levels of fairness. Remember that in OFE the verifier can always choose to abort or complete a fair exchange after having received the signer's partial signature. Thus for both concurrent signatures and OFE a user can enjoy advantage level 0. However, the work in this chapter demonstrates a clear gap between the notion of concurrent signature and optimistic fair exchange (OFE) in which no party enjoys advantage level 1. Furthermore, in the variant Ambiguous OFE, no party enjoys either advantage level 1 or advantage level 2.

9.5 Chapter Summary

The advantage gained by the initial signer in a concurrent signature scheme were pointed out, and classified into three levels. In fact, any concurrent signature satisfying Chen, Kudla and Paterson's syntax can be abused in different ways. This is a very important observation in particular where concurrent signatures are used in different scenarios. Cautions must be exercised when concurrent signatures are to be adopted in real applications to ensure either the matching signer can accept the inherent unfairness of concurrent signatures or it is hard for the initial signer to claim the advantage. In particular, it is demonstrated that concurrent signatures were not suitable for tendering systems (in contrast to the seminal paper of Chen et al. [CKP04]).

Part V

Conclusion and Future Work

Chapter 10

Conclusion and Future Work

10.1 Summary of the Thesis

Fair exchange of digital signatures has been considered as a fundamental problem in cryptography. The main contributions of this thesis are as follows.

10.1.1 OFE with Stronger Security

In Chapter 3 and Chapter 4, it was identified that the the existing security models for OFE were not practical enough. To make it complete and more practical, the enhanced chosen-key model was proposed, which allows a dishonest signer to have access to the arbitrator's secret key and explicitly provides the dishonest verifier and the dishonest arbitrator with the signing oracle. It was demonstrated that the enhanced chosen-key model were strictly stronger than the highest level of existing OFE security model. Two popular approaches in the construction of OFE protocols were revisited, namely, schemes based on verifiably encrypted signature and those based on conventional signature plus ring signature and it was shown that they could still yield schemes which remain secure in the enhanced model.

10.1.2 OFE with Better Efficiency

It is known that efficient OFE schemes can be constructed based on conventional signatures and ring signatures. To guarantee the security of the resulting OFE schemes, the security requirement for the ring signatures was previously believed to be unforgeable under an adaptive attack, against a static adversary in the 2-user setting. In Chapter 5, a strictly weaker model for 2-user ring signatures called unforgeability against restricted adaptive attacks was proposed and it was proved that 2-user ring signatures secure in this weaker model would suffice to guarantee the

security of the resulting OFE scheme. Based on this observation, a new instantiation of OFE scheme was proposed that achieved the best efficiency in the standard model in terms of signature size, generation as well as verification. Besides, the OFE instantiation relies on a weak assumption, and has comparable efficiency with the most efficient OFE scheme secure in the random oracle model.

10.1.3 OFE with New Features

To extend the applications of OFE, several situations that had not been researched before were studied. In Chapter 6, optimistic fair exchange of threshold signatures was considered. In Chapter 7, perfect ambiguous optimistic fair exchange (PAOFE) was introduced, which can be viewed as an extension of AOFE with a new property *perfect ambiguity*. In Chapter 8, OFE in the attribute-based setting was considered.

10.1.4 Concurrent Signatures Revisited

Concurrent signature is a new cryptographic primitive to solve the fair exchange of digital signatures problem without the help of a third party. In Chapter 9, it was identified that theoretically the initial signer in a concurrent signature scheme can gain more advantages except the basic one that was known. It was emphasized that cautions must be exercised when concurrent signatures are to be adopted in real applications and concurrent signatures may not be suitable in some applications such as tendering systems.

10.2 Future Work

It would be an interesting work to apply the ideas in the enhanced chosen-key model for OFE to AOFE, and propose the corresponding security model for AOFE, as well as new efficient constructions of AOFE schemes secure in the new enhanced model. Besides, construction of threshold optimistic fair exchange in the standard model and secure in a model that is not static is left as the future work. It would also be a meaningful work to find more elegant solutions based instead on fair exchange of attribute-based signatures to solve the fair exchange problem in the attribute-based setting.

Bibliography

[AOS02] Masayuki Abe, Miyako Ohkubo, and Koutarou Suzuki. 1-out-of-n Signatures from a Variety of Keys. *Advances in Cryptology - Asiacrypt 2002, Lecture Notes in Computer Science 2501*, pages 415 – 432, 2002.

[ASW97] N. Asokan, M. Schunter, and M. Waidner. Optimistic protocols for fair exchange. In *Proc. 4th ACM Conference on Computer and Communications Security*, pages 8–17, 1997.

[ASW98] N. Asokan, V. Shoup, and M. Waidner. Optimistic fair exchange of digital signatures. *Advances in Cryptology-Eurocrypt 1998, Lecture Notes in Computer Science 1403*, pages 591 – 606, 1998.

[ASW00] N. Asokan, V. Shoup, and M. Waidner. Optimistic fair exchange of digital signatures. *IEEE Journal of Selected Areas Communications, vol. 18, no. 4*, pages 593 – 610, Apr. 2000.

[AV04] Gildas Avoine and Serge Vaudenay. Optimistic fair exchange based on publicly verifiable secret sharing. In *ACISP'07*, volume 3108, pages 74–85. Springer, 2004.

[BB04] Dan Boneh and Xavier Boyen. Short signatures without random oracles. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT*, volume 3027 of *Lecture Notes in Computer Science*, pages 56–73. Springer, 2004.

[BB08] Dan Boneh and Xavier Boyen. Short signatures without random oracles and the sdh assumption in bilinear groups. *J. Cryptology*, 21(2):149–177, 2008.

[BBDP01] Mihir Bellare, Alexandra Boldyreva, Anand Desai, and David Pointcheval. Key-privacy in public-key encryption. In *Advances in Cryptology - ASIACRAPT '01*, volume 2248 of *LNCS*, pages 566–582. Springer-Verlag, 2001.

[BCDvdG87] Ernest F. Brickell, David Chaum, Ivan Damgard, and Jeroen van de Graaf. Gradual and verifiable release of a secret. *Proc. of Crypto'87, Lecture Notes in Computer Science*, 293:156–166, 1987.

[BCNP04] Boaz Barak, Ran Canetti, Jesper Buus Nielsen, and Rafael Pass. Universally composable protocols with relaxed set-up assumptions. In *FOCS'04*, pages 186–195, 2004.

[Bei96] Amos Beimel. *Secure Schemes for Secret Sharing and Key Distribution*. PhD thesis, Israel Institute of Technology, Technion, Haifa, Israel, 1996.

[Bel02] Mihir Bellare. A note on negligible functions. *Journal of Cryptology*, 15(4):271–285, 2002.

[BF98] Colin Boyd and Ernest Foo. Off-line fair payment protocols using convertible signatures. In Ohta and Pei [OP98], pages 271–285.

[BF01] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the weil pairing. In Joe Kilian, editor, *CRYPTO'01*, volume 2139, pages 213–229. Springer, 2001.

[BG93] Mihir Bellare and Oded Goldreich. On defining proofs of knowledge. In *Proceedings of the 12th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '92, pages 390–420, London, UK, UK, 1993. Springer-Verlag.

[BGLS03] Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. *Advances in cryptology - Eurocrypt 2003, Lecture Notes in Computer Science*, 2656:416–432, 2003.

[BGN05] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-dnf formulas on ciphertexts. In Joe Kilian, editor, *TCC*, volume 3378 of *Lecture Notes in Computer Science*, pages 325–341. Springer, 2005.

[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.

[BLS04] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pairing. *J. Cryptology*, 17(4):297–319, 2004.

[BN00] Dan Boneh and Moni Naor. Timed commitments. In *CRYPTO*, volume 1880 of *Lecture Notes in Computer Science*, pages 236–254, 2000.

[BN06] Mihir Bellare and Gregory Neven. Multi-signatures in the plain public-key model and a general forking lemma. In *Proceedings of the 13th ACM conference on Computer and communications security*, CCS '06, pages 390–399, New York, NY, USA, 2006. ACM.

[Bon98] Dan Boneh. The decision Diffie-Hellman problem. In *Proceedings of the Third International Symposium on Algorithmic Number Theory*, ANTS-III, pages 48–63, London, UK, UK, 1998. Springer-Verlag.

[Boy07] Xavier Boyen. Mesh signatures. In Moni Naor, editor, *EUROCRYPT*, volume 4515 of *Lecture Notes in Computer Science*, pages 210–227. Springer, 2007.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby, editors, *ACM Conference on Computer and Communications Security*, pages 62–73. ACM, 1993.

[Bra93] Stefan A. Brands. An efficient off-line electronic cash system based on the representation problem. Technical report, Amsterdam, The Netherlands, The Netherlands, 1993.

[BW07] Xavier Boyen and Brent Waters. Full-domain subgroup hiding and constant-size group signatures. In Okamoto and Wang [OW07], pages 1–15.

[CD00] Jan Camenisch and Ivan Damgård. Verifiable encryption, group encryption, and their applications to separable group signatures and signature sharing schemes. In *Proceedings of the 6th International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology*, ASIACRYPT '00, pages 331–345, London, UK, 2000. Springer-Verlag.

[CGH98] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited (preliminary version). In Jeffrey Scott Vitter, editor, *STOC*, pages 209–218. ACM, 1998.

[CGH04] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited. *J. ACM*, 51(4):557–594, July 2004.

[CGS07] Nishanth Chandran, Jens Groth, and Amit Sahai. Ring signatures of sub-linear size without random oracles. In Lars Arge, Christian Cachin, Tomasz Jurdzinski, and Andrzej Tarlecki, editors, *ICALP*, volume 4596 of *Lecture Notes in Computer Science*, pages 423–434. Springer, 2007.

[Cha94] David Chaum. Designated confirmer signatures. In Alfredo De Santis, editor, *EUROCRYPT*, volume 950 of *Lecture Notes in Computer Science*, pages 86–91. Springer, 1994.

[Che98] Liqun Chen. Efficient fair exchange with verifiable confirmation of signatures. In Ohta and Pei [OP98], pages 286–299.

[Che06] Jung Hee Cheon. Security analysis of the strong diffie-hellman problem. In *EUROCRYPT*, pages 1–11, 2006.

[CKP04] Liqun Chen, Caroline Kudla, and Kenneth G. Paterson. Concurrent signatures. *Advances in Cryptology - Eurocrypt 2004, Lecture Notes in Computer Science 3027*, pages 278 – 305, 2004.

[Cle89] Richard Cleve. Controlled gradual disclosure schemes for random bits and their applications. In *Proceedings on Advances in cryptology*, CRYPTO '89, pages 573–588, New York, NY, USA, 1989. Springer-Verlag New York, Inc.

[Cra05] Ronald Cramer, editor. *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings*, volume 3494 of *Lecture Notes in Computer Science*. Springer, 2005.

[CS98] Ronald Cramer and Victor Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext. In *CRYPTO '98*, pages 13–25, 1998.

[CS03] Jan Camenisch and Victor Shoup. Practical verifiable encryption and decryption of discrete logarithms. In Dan Boneh, editor, *CRYPTO*, volume 2729 of *Lecture Notes in Computer Science*, pages 126–144, 2003.

[CTS95] Benjamin Cox, J. D. Tygar, and Marvin Sirbu. Netbill security and transaction protocol. In *Proceedings of the 1st conference on USENIX Workshop on Electronic Commerce - Volume 1*, pages 6–6, Berkeley, CA, USA, 1995. USENIX Association.

[CW79] J.Lawrence Carter and Mark N. Wegman. Universal classes of hash functions. *Journal of Computer and System Sciences*, 18(2):143 – 154, 1979.

[Dam88] Ivan Bjerre Damgård. Collision free hash functions and public key signature schemes. In *Proceedings of the 6th annual international conference on Theory and application of cryptographic techniques*, EUROCRYPT'87, pages 203–216, Berlin, Heidelberg, 1988. Springer-Verlag.

[Dam94] Ivan Bjerre Damgård. Practical and provably secure release of a secret and exchange of signatures. In *Workshop on the theory and application of cryptographic techniques on Advances in cryptology*, EUROCRYPT '93, pages 200–217, Secaucus, NJ, USA, 1994. Springer-Verlag New York, Inc.

[Den08] Alexander W. Dent. A survey of certificateless encryption schemes and security models. *Int. J. Inf. Sec.*, 7(5):349–377, 2008.

[DGLW96] Robert H. Deng, Li Gong, Aurel A. Lazar, and Weiguo Wang. Practical Protocols for Certified Electronic Mail. *J. Network Syst. Manage.*, 4(3):279–297, 1996.

[DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[DLY07] Yevgeniy Dodis, Pil Joong Lee, and Dae Hyun Yum. Optimistic fair exchange in a multi-user setting. In *Proceedings of the 10th international conference on Practice and theory in public-key cryptography*, PKC'07, pages 118–133, Berlin, Heidelberg, 2007. Springer-Verlag.

[DR03] Yevgeniy Dodis and Leonid Reyzin. Breaking and repairing optimistic fair exchange from podc 2003. In *Proceedings of the 3rd ACM workshop on Digital rights management*, DRM '03, pages 47–54, New York, NY, USA, 2003. ACM.

[EGL85] Shimon Even, Oded Goldreich, and Abraham Lempel. A randomized protocol for signing contracts. *Communication of the ACM*, 28:637–647, 1985.

[ES05] Paul D. Ezhilchelvan and Santosh K. Shrivastava. A family of trusted third party based fair-exchange protocols. *IEEE Trans. Dependable Secur. Comput.*, 2:273–286, October 2005.

[FR97] Matthew K. Franklin and Michael K. Reiter. Fair exchange with a semi-trusted third party (extended abstract). In *Proceedings of the 4th ACM conference on Computer and communications security*, CCS '97, pages 1–5, New York, NY, USA, 1997. ACM.

[FS87] Amos Fiat and Adi Shamir. How to prove yourself: practical solutions to identification and signature problems. In *Proceedings on Advances in cryptology—CRYPTO '86*, pages 186–194, London, UK, UK, 1987. Springer-Verlag.

[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.

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *J. Comput. Syst. Sci.*, 28(2):270–299, 1984.

[GMR85] S Goldwasser, S Micali, and C Rackoff. The knowledge complexity of interactive proof-systems. In *Proceedings of the seventeenth annual ACM symposium on Theory of computing*, STOC '85, pages 291–304, New York, NY, USA, 1985. ACM.

[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.

[Gol84] Oded Goldreich. A simple protocol for signing contracts. *Proc. of Crypto'83*, pages 133–136, 1984.

[Gol90] Oded Goldreich. A note on computational indistinguishability. *Information Processing Letters*, pages 277–281, 1990.

[Gol00] Oded Goldreich. *Foundations of Cryptography: Basic Tools*. Cambridge University Press, New York, NY, USA, 2000.

[GP03] Juan A. Garay and Carl Pomerance. Timed fair exchange of standard signatures: [extended abstract]. In *Financial Cryptography*, volume 2742 of *Lecture Notes in Computer Science*, pages 190–207, 2003.

[HK12] Dennis Hofheinz and Eike Kiltz. Programmable hash functions and their applications. *J. Cryptology*, 25(3):484–527, 2012.

[HMS+11] Xinyi Huang, Yi Mu, Willy Susilo, Wei Wu, Jianying Zhou, and Robert H. Deng. Preserving transparency and accountability in optimistic fair exchange of digital signatures. *IEEE Transactions on Information Forensics and Security*, 6(2):498–512, 2011.

[HV11] Somayeh Heidarvand and Jorge L. Villar. Fair and abuse-free contract signing protocol from boneh-boyen signature. In *Proceedings of the 7th European conference on Public key infrastructures, services and applications*, EuroPKI'10, pages 125–140, Berlin, Heidelberg, 2011. Springer-Verlag.

[HW11] Qiong Huang and Duncan S. Wong. Short convertible undeniable signature in the standard model. In Feng Bao and Jian Weng, editors, *ISPEC*, volume 6672 of *Lecture Notes in Computer Science*, pages 257–272. Springer, 2011.

[HWS11a] Qiong Huang, Duncan S. Wong, and Willy Susilo. Efficient designated confirmer signature and DCS-based ambiguous optimistic fair exchange. *IEEE Transactions on Information Forensics and Security*, 6(4):1233–1247, 2011.

[HWS11b] Qiong Huang, Duncan S. Wong, and Willy Susilo. Group-oriented fair exchange of signatures. *Inf. Sci.*, 181:3267–3283, August 2011.

[HWS12] Qiong Huang, Duncan S. Wong, and Willy Susilo. The construction of ambiguous optimistic fair exchange from designated confirmer signature without random oracles. In *Proceedings of Public Key Cryptography 2012*, volume 7293 of *Lecture Notes in Computer Science*, pages 120–137. Springer, 2012.

[HYWS08a] Qiong Huang, Guomin Yang, Duncan S. Wong, and Willy Susilo. Ambiguous optimistic fair exchange. In *Advances in Cryptology - ASIACRYPT 2008*, pages 74–89, 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 *Proceedings of the 2008 The Cryptopgraphers' Track at the RSA conference on Topics in cryptology*, CT-RSA'08, pages 106–120, Berlin, Heidelberg, 2008. Springer-Verlag.

[JSI96] Markus Jakobsson, Kazue Sako, and Russell Impagliazzo. Designated verifier proofs and their applications. In Maurer [Mau96], pages 143–154.

[KG10] Aniket Kate and Ian Goldberg. Distributed private-key generators for identity-based cryptography. In *SCN'10*, pages 436–453, 2010.

[KL10] Alptekin Küpçü and Anna Lysyanskaya. Optimistic fair exchange with multiple arbiters. In *ESORICS'10*, pages 488–507, 2010.

[LOS+06] Steve Lu, Rafail Ostrovsky, Amit Sahai, Hovav Shacham, and Brent Waters. Sequential aggregate signatures and multisignatures without random oracles. In *EUROCRYPT, 2006.*, pages 465–485. Springer, 2006.

[LW12] Allison B. Lewko and Brent Waters. New proof methods for attribute-based encryption: Achieving full security through selective techniques. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO*, volume 7417 of *Lecture Notes in Computer Science*, pages 180–198. Springer, 2012.

[Mao03] Wenbo Mao. *Modern Cryptography: Theory and Practice*. Prentice Hall Professional Technical Reference, 2003.

[Mau96] Ueli M. Maurer, editor. *Advances in Cryptology - EUROCRYPT '96, International Conference on the Theory and Application of Cryptographic Techniques, Saragossa, Spain, May 12-16, 1996, Proceeding*, volume 1070 of *Lecture Notes in Computer Science*. Springer, 1996.

[Mic03] Silvio Micali. Simple and fast optimistic protocols for fair electronic exchange. In *Proceedings of the twenty-second annual symposium on Principles of distributed computing*, PODC '03, pages 12–19, New York, NY, USA, 2003. ACM.

[MK01] Olivier Markowitch and Steve Kremer. An optimistic non-repudiation protocol with transparent trusted third party. In *Information Security Conference 2001, Lecture Notes in Computer Science*, pages 363–378. Springer-Verlag, 2001.

[MS01] Olivier Markowitch and Shahrokh Saeednia. Optimistic fair exchange with transparent signature recovery. In Paul F. Syverson, editor, *Financial Cryptography*, volume 2339 of *Lecture Notes in Computer Science*, pages 329–340. Springer, 2001.

[MVO96] Alfred J. Menezes, Scott A. Vanstone, and Paul C. Van Oorschot. *Handbook of Applied Cryptography*. CRC Press, Inc., Boca Raton, FL, USA, 1st edition, 1996.

[Ngu05] K. Nguyen. Asymmetric concurrent signatures. *Information and Communications Security (ICICS 2005), Lecture Notes in Computer Science 3783*, pages 181 – 193, 2005.

[Odl85] A M Odlyzko. Discrete logarithms in finite fields and their cryptographic significance. In *Proc. of the EUROCRYPT 84 workshop on Advances in cryptology: theory and application of cryptographic techniques*, pages 224–314, New York, NY, USA, 1985. Springer-Verlag New York, Inc.

[OMO08] Yusuke Okada, Yoshifumi Manabe, and Tatsuaki Okamoto. An optimistic fair exchange protocol and its security in the universal composability framework. *Int. J. Appl. Cryptol.*, 1:70–77, February 2008.

[OP98] Kazuo Ohta and Dingyi Pei, editors. *Advances in Cryptology - ASIACRYPT '98, International Conference on the Theory and Applications of Cryptology and Information Security, Beijing, China, October 18-22, 1998, Proceedings*, volume 1514 of *Lecture Notes in Computer Science*. 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.

[PS96] David Pointcheval and Jacques Stern. Security proofs for signature schemes. In Maurer [Mau96], pages 387–398.

[QWM12] Lie Qu, Guilin Wang, and Yi Mu. Optimistic fair exchange of ring signatures. In *Security and Privacy in Communication Networks*, volume 96, pages 227–242. 2012.

[RS92] Charles Rackoff and Daniel R. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In *Proceedings of the 11th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '91, pages 433–444, London, UK, UK, 1992. Springer-Verlag.

[RS09] Markus Rückert and Dominique Schröder. Security of verifiably encrypted signatures and a construction without random oracles. In *Proceedings of the 3rd International Conference Palo Alto on Pairing-Based Cryptography*, Pairing '09, pages 17–34, Berlin, Heidelberg, 2009. Springer-Verlag.

[RST01] Ronald L. Rivest, Adi Shamir, and Yael Tauman. How to leak a secret. In Colin Boyd, editor, *ASIACRYPT*, volume 2248 of *Lecture Notes in Computer Science*, pages 552–565. Springer, 2001.

[Sah99] Amit Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *Proceedings of the 40th Annual Symposium on Foundations of Computer Science*, FOCS '99, pages 543–553, Washington, DC, USA, 1999. IEEE Computer Society.

[Sch91] Claus-Peter Schnorr. Efficient signature generation by smart cards. *J. Cryptology*, 4(3):161–174, 1991.

[Sha79] Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979.

[SMZ04] Willy Susilo, Yi Mu, and Fangguo Zhang. Perfect concurrent signature schemes. *Information and Communication Security(ICICS 2004), LNCS 3269*, pages 14 – 26, 2004.

[SW05] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In Cramer [Cra05], pages 457–473.

[SW07] Hovav Shacham and Brent Waters. Efficient ring signatures without random oracles. In Okamoto and Wang [OW07], pages 166–180.

[THW12] Xiao Tan, Qiong Huang, and Duncan S. Wong. Concurrent signature without random oracles. Cryptology ePrint Archive, Report 2012/576, 2012. `http://eprint.iacr.org/`.

[TSSN05] D. Tonien, Willy Susilo, and R. Safavi-Naini. Multi-party concurrent signatures. *The 9th Information Security Conference (ISC 2006), LNCS 4176*, 2005.

[Wan05] Guilin Wang. An abuse-free fair contract signing protocol based on the rsa signature. In *Proceedings of the 14th international conference on World Wide Web*, WWW '05, pages 412–421, New York, NY, USA, 2005. ACM.

[Wat05] Brent Waters. Efficient identity-based encryption without random oracles. In Cramer [Cra05], pages 114–127.

[Wat11] Brent Waters. Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. In Dario Catalano, Nelly Fazio, Rosario Gennaro, and Antonio Nicolosi, editors, *Public Key Cryptography*, volume 6571 of *Lecture Notes in Computer Science*, pages 53–70. Springer, 2011.

[WBZ06] Gulin Wang, Feng Bao, and Jianying Zhou. The fairness of perfect concurrent signature. *Information and Communication Security(ICICS 2006), LNCS 4307*, pages 435 – 451, 2006.

[YWSH11] Tsz Hon Yuen, Duncan S. Wong, Willy Susilo, and Qiong Huang. Concurrent signatures with fully negotiable binding control. In Xavier Boyen and Xiaofeng Chen, editors, *ProvSec*, volume 6980 of *Lecture Notes in Computer Science*, pages 170–187. Springer, 2011.

[ZB06] Huafei Zhu and Feng Bao. Stand-alone and setup-free verifiably committed signatures. In *CT-RSA'06*, volume 3860 of *Lecture Notes in Computer Science*, pages 159–173, 2006.

[ZM07] Jianhong Zhang and Jian Mao. A novel verifiably encrypted signature scheme without random oracle. In *Proceedings of the 3rd international conference on Information security practice and experience*, ISPEC'07, pages 65–78, Berlin, Heidelberg, 2007. Springer-Verlag.

[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.