論文 / 著書情報
Article / Book Information

題目(和文)	代理人再暗号化方式に対する概念と構成
Title(English)	Notions and Constructions on Proxy Re-Encryption
著者(和文)	ｸﾞｴﾝﾏｲﾝ ﾊｰ
Author(English)	Manh Ha Nguyen
出典(和文)	学位:博士（理学）, 学位授与機関:東京工業大学, 報告番号:甲第9613号, 授与年月日:2014年9月25日, 学位の種別:課程博士, 審査員:田中　圭介,小島　定吉,渡辺　治,鹿島　亮,脇田　建
Citation(English)	Degree:, Conferring organization: Tokyo Institute of Technology, Report number:甲第9613号, Conferred date:2014/9/25, Degree Type:Course doctor, Examiner:,,,,
学位種別(和文)	博士論文
Type(English)	Doctoral Thesis

Notions and Constructions on Proxy Re-Encryption

Manh Ha Nguyen

Doctoral Thesis

Department of Mathematical and Computing Sciences
Graduate School of Information Science and Engineering
Tokyo Institute of Technology

Supervisor: Keisuke Tanaka

August, 2014

Abstract

Public-key encryption is a fundamental cryptographic primitive with which we can communicate securely over possibly insecure network without shared secret information in advance. Proxy re-encryption (PRE) extends the traditional public-key encryption to support the delegation of the decryption rights, in which a proxy is allowed to transform ciphertexts computed under the public key of the delegator into other ciphertexts for the delegatee. The proxy, however, learns nothing about the underlying messages encrypted, and has no knowledge of the secret keys of the delegators and the delegatees.

PRE schemes have many applications in encrypted email forwarding, distributed file storage systems, law enforcement, digital rights management (DRM), and outsourced filtering of encrypted spam. For PRE schemes, security against chosen ciphertext attacks (CCA security) is nowadays considered as a standard security notion needed in most practical applications/situations where PRE schemes are used. There are many notions of CCA security for PRE proposed so far. However, these notions do not capture all of the essential aspects of the CCA security. In this thesis, we focus on CCA security notions for PRE and constructing schemes which are secure in the sense of these security notions.

We begin with Chapter 1 providing some backgrounds and our contributions on PRE. Going on, in Chapter 2, after introducing notations used in the whole thesis, we review the factoring assumption, the properties of bilinear maps and the intractability assumptions that our schemes rely on. Next, we review the definitions on public key encryption, symmetric key encryption, signature, and target-collision resistant hash function, which are the main tools in our constructions.

According to the direction of transformation, PRE can be classified into two types: *unidirectional* and *bidirectional*. In unidirectional PRE, the proxy can only transform ciphertexts from the delegator to the delegatees. While in bidirectional PRE, the proxy can transform ciphertexts in both directions. PRE can also be categorized into *multi-hop* PRE, in which the ciphertexts can be transformed from Alice to Bob and then to Charlie and so on, and *single-hop* PRE, in which the ciphertexts can only be transformed once.

In Chapter 3, we first recall the concepts and the security notions of bidirectional multi-hop, bidirectional single-hop, and unidirectional single-hop PRE. We then present new CCA security definitions for both bidirectional single-hop, and unidirectional single-hop PRE by extending those of the previous works. In order to point out why our security models are stronger than those of the previous works, we present concrete PRE schemes which are secure in their model, but are not secure in our models. In this chapter, we also review the model and security definitions of conditional PRE (CPRE) which is a variant of proxy re-encryption, where only the ciphertext satisfying one condition set by the delegator can be transformed by the proxy and then decrypted

by the delegatee.

In Chapter 4, we focus on constructing PRE schemes which are based on the hardness of the factoring problem. In particular, we present three PRE schemes. The first is a CPA-secure bidirectional multi-hop PRE scheme. The second is a CCA-secure bidirectional single-hop PRE scheme. The last is a CCA-secure unidirectional single-hop PRE scheme. The first is in the standard model, and the others in the random oracle model. In order to construct the second and the third schemes, we propose a new factoring-based strong signature scheme which is a variant of the Schnorr signature scheme.

In Chapter 5, we first show that the PRE scheme proposed by Shao, Liu, and Zhou in the Journal of Systems and Software 85(3) is vulnerable to chosen-ciphertext attack. We then propose a unidirectional single-hop PRE scheme which is secure in the full CCA security model. Our scheme relies on mild complexity assumptions in bilinear groups without random oracles.

In Chapter 6, we first show that the CPRE scheme proposed by Liang, Liu, Tan, Wong, and Tang in ICISC2012 is vulnerable to chosen-ciphertext attack. We then propose two CPRE schemes which are secure against the selective and adaptive condition chosen-ciphertext attack, respectively. These schemes are extended from the CCA-secure PRE scheme proposed in Chapter 5. Our schemes are the first CPRE schemes which are secure against condition chosen-ciphertext attack without random oracles.

We conclude the thesis and give several open problems in Chapter 7.

Acknowledgments

This thesis would not have been written without the helps and advices of my supervisor, Keisuke Tanaka. For that I would like to thank him first of all. I joined Keisuke's group because of his enthusiasm about his research and his willingness to work on problems outside his immediate area of expertise. I am now certain that this was the right decision for me. Keisuke taught me how to approach a difficult problem, to truly understand what is involved, and to find the correct tools and techniques to solve it. His suggestions and advices were always useful at each step of this process. His excellent and ideas have always been essential of my research. I would also like to thank Toshiyuki Isshiki for thoughtful discussions and supports, and the past that he have dragged me into proxy re-encryption.

I would like to thank all members of Tanaka Laboratory for their advice, encouragement, friendship, and delightful non-working hours. Thank you, Keiji Omura, Ryoutaro Hayashi, Toshiyuki Isshiki, Akihiro Mihara, Takao Onodera, Hiroki Hada, Manabu Suzuki, Naoyuki Yamashita, Shizu Kanauchi, Keita Xagawa, Takato Hirano, Christopher Alfred Portman, Harugana Hiwatari, Jun Nakajima, Chihiro Ohyama, Masatoshi Yashiro, Ryo Nishimaki, Mario Larangeira Junior, Kouich Sakumoto, Tatsunori Seki, Hirotoshi Takebe, Daisuke Inoue, Akira Numayama, Koichiro Wada, Hideaki Suzuki, Toshihide Matsuda, Yuki Tan, Hinako Kamimura, Hitoshi Namiki, Akihiro Yamada, Chiaki Minato, Hapuarachchillage D. P. S. S. Kumara, Takayuki Otsubo, Yusuke Kinoshita, Yoshihiro Koseki, Haruna Higo, Yuyu Wang, Ai Ishida, Tomoyuki Komatsu, Ryosuke Nakata, Fuyuki Kitagawa, Yuuki Sawai, Hiroyuki Tohyama, Ta Minh Thanh, Hajime Mimura, Soki Nishiyama, and Satoshi Yasuda.

I would like to express my sincere gratitude to the board of examiners, Professors Ryo Kashima, Sadayoshi Kojima, Ken Wakita, and Osamu Watanabe, for their valuable comments and challenging questions, which help me understand more about the topics.

I am thankful to Koichiro Akiyama for giving me a chance to intern in Toshiba. I would like to express deep gratitude for the members of Computer Architecture & Security Laboratory, Corporate R&D Center, Toshiba Corporation. I especially thank to Yoshikazu Hanatani, for his encouragements and fruitful discussions in the summer intern.

I would like to express my sincerest gratitude to many anonymous reviewers of my papers whose comments helped to improve this work. I would also like to thank many domestic and international researchers who I met while attending conferences. Special thanks must go to Le Trieu Phong for his inspirational chats during conferences. The discussion with him was always fruitful, and made it possible to achieve the results in and not in this thesis.

Last, but not least, I would like to thank my parents for their unconditional supports and loves in all stages of my life. Especially, I would like to express my sincerest gratitude to my

mother. She was my first teacher and mentor. Whatever I am today is because of her constant encouragement and inspiration. Special thanks go to my parents-in-law for helping me handle being both a researcher and as a father. I would also like to thank my host families, all my brothers and sisters for their kind support and encouragement. Above all, I would like to express my deepest gratitude to my wife Thi Hieu Pham and my little angel Ha Linh Nguyen for all their love, help, support, encouragement, and inspiration during my studies.

This work is supported by a grant of I-System Co. Ltd., NTT Secure Platform Laboratories, and the Ministry of Education, Culture, Sports, Science and Technology.

Contents

1 Introduction **1**
- 1.1 Background . . . 1
- 1.2 Contributions . . . 5
- 1.3 Organization . . . 8

2 Preliminaries **11**
- 2.1 Notations . . . 11
- 2.2 Cryptographic Assumptions . . . 11
  - 2.2.1 Factoring Assumption . . . 12
  - 2.2.2 Bilinear Maps and Complexity Assumptions . . . 13
  - 2.2.3 The Gap Hashed Diffie-Hellman Assumption . . . 14
- 2.3 Public Key Encryption . . . 14
- 2.4 Symmetric Key Encryption . . . 16
- 2.5 Signature . . . 17
- 2.6 Target-Collision Resistant Hash Function . . . 18

3 Models and Security Notions **19**
- 3.1 Bidirectional Multi-Hop Proxy Re-Encryption . . . 19
  - 3.1.1 Model . . . 19
  - 3.1.2 Security Notions . . . 20
- 3.2 Bidirectional Single-Hop Proxy Re-Encryption . . . 21
  - 3.2.1 Model . . . 21
  - 3.2.2 Security Notions . . . 22
- 3.3 Unidirectional Single-Hop Proxy Re-Encryption . . . 24
  - 3.3.1 Model . . . 24
  - 3.3.2 Security Notions . . . 25
  - 3.3.3 Discussion in the Comparison with the Previous Security Models . . . 28
- 3.4 Conditional Proxy Re-Encryption . . . 34
  - 3.4.1 Model . . . 34
  - 3.4.2 Security Notions . . . 35

4 Schemes Based on Factoring **41**
- 4.1 Introduction . . . 41
- 4.2 A Strongly Unforgeable Signature Scheme from Factoring . . . 42

4.2.1 Construction . . . 42
4.2.2 Security Analysis . . . 43
4.3 The Proposed Schemes . . . 44
4.3.1 The BM-CPA-Secure Scheme . . . 44
4.3.2 The BS-CCA-Secure Scheme . . . 45
4.3.3 Extension to a Unidirectional Single-Hop Scheme . . . 47
4.4 Proofs . . . 49
4.4.1 Proof of Theorem 4 . . . 49
4.4.2 Proof of Theorem 5 . . . 50
4.4.3 Proof of Theorem 6 . . . 60
4.5 Summary . . . 71

5 A Scheme with Bilinear Maps 73
5.1 Introduction . . . 73
5.2 Analysis of the Shao-Liu-Zhou Scheme . . . 74
5.2.1 Review of the Shao-Liu-Zhou Scheme . . . 74
5.2.2 Our Attack . . . 76
5.3 The Proposed Scheme . . . 77
5.3.1 Construction . . . 77
5.3.2 Security Analysis . . . 78
5.4 On the Hardness of the 6-AmDBDH Problem . . . 82
5.4.1 The DBDH Problem and Its Variant . . . 82
5.4.2 Equivalent Definitions of the 6-AmDBDH Problem . . . 82
5.4.3 The General Diffie-Hellman Problem . . . 83
5.4.4 Complexity Lower Bounds in Generic Bilinear Groups . . . 83
5.5 Proofs . . . 84
5.5.1 Proof of Theorem 7 . . . 84
5.5.2 Proof of Theorem 8 . . . 88
5.6 Summary . . . 91

6 CCA-Secure Conditional Schemes without Random Oracles 93
6.1 Introduction . . . 93
6.2 Analysis of the Liang-Liu-Tan-Wong-Tang Scheme . . . 94
6.2.1 Review of the Liang-Liu-Tan-Wong-Tang Scheme . . . 94
6.2.2 Our Attack . . . 96
6.3 The Proposed Schemes . . . 97
6.3.1 The 2nd-sCond-CCA-Secure Scheme . . . 97
6.3.2 The 2nd-aCond-CCA-Secure Scheme . . . 100
6.3.3 Comparisons . . . 101
6.4 Proofs . . . 101
6.4.1 Proof of Theorem 11 . . . 101
6.4.2 Proof of Theorem 12 . . . 105
6.4.3 Proof of Theorem 13 . . . 108
6.4.4 Proof of Theorem 14 . . . 113

6.5 Summary 116

7 Conclusion **117**

Bibliography **119**

Chapter 1

Introduction

Contents

1.1 Background 1
1.2 Contributions 5
1.3 Organization 8

1.1 Background

Public-key encryption (PKE) refers to a cryptographic system requiring two separate keys, one to lock or encrypt the *plaintext* and one to unlock or decrypt the *ciphertext*. They are called *public key* and *secret key*, respectively. Neither key will do both functions. The publicly available encrypting-key is widely distributed, while the private decrypting-key is known only to the recipient. Messages are encrypted with the recipient's public key and can only be decrypted with the corresponding secret key.

Proxy Re-Encryption. *Proxy re-encryption* (PRE), first introduced by Blaze, Bleumer, and Strauss in [8], extends the traditional PKE to support the delegation of the decryption rights, in which a proxy is allowed to transform ciphertexts computed under the public-key of Alice (the delegator) into other ciphertexts for Bob (the delegatee). The proxy, however, learns nothing about the underlying messages encrypted, and has no knowledge of the secret keys of the delegators and the delegatees.

PRE schemes have applications in encrypted email forwarding [8], distributed file storage systems [4], law enforcement [34], digital rights management (DRM), and outsourced filtering of encrypted spam [4]. In all these cases, the gist is that the process of re-encryption, i.e., decrypting under one key for encryption under another key, should not allow the re-encryptor module to compromise the secrecy of encrypted messages.

According to the direction of transformation, PRE can be classified into two types: *unidirectional* and *bidirectional* [34]. In unidirectional PRE, the proxy can only transform ciphertexts from Alice to Bob. While in bidirectional PRE, the proxy can transform ciphertexts in both

directions. PRE can also be categorized into *multi-hop* PRE, in which the ciphertexts can be transformed from Alice to Bob and then to Charlie and so on, and *single-hop* PRE, in which the ciphertexts can only be transformed once [34].

In 1998, Blaze, Bleumer, and Strauss [8] proposed the first bidirectional PRE scheme. It is based on a simple modification of the Elgamal encryption scheme [21]. This scheme is efficient and semantically secure under the decisional Diffie-Hellman (DDH) assumption.

In 2005, Ateniese, Fu, Green, and Hohenberger [3, 4] showed the first examples of unidirectional PRE schemes based on bilinear maps. Moreover, they obtained the master key security property in that the proxy is unable to collude with the delegatees in order to expose the delegator's secret. The constructions [3, 4] are also efficient, semantically secure assuming the intractability of decisional variants of the bilinear Diffie-Hellman problem [11]. These PRE schemes only ensure the chosen-plaintext (CPA) security, which seems definitely insufficient for many practical applications.

In 2007, Canetti and Hohenberger [16] gave a definition of security against chosen ciphertext attacks (CCA) for bidirectional PRE schemes and described an efficient construction satisfying this definition (for bidirectional schemes but easily adaptable for the unidirectional case, as explained in [42]). Their security analysis takes place in the standard model (without the random oracle heuristic [6]). Like the scheme in [8], their construction is bidirectional and they left as an open problem to come up with a CCA-secure unidirectional scheme.

In 2008, Libert and Vergnaud [42] partially resolved this problem by presenting a unidirectional single-hop PRE scheme without random oracles. They proved that their scheme is secure against the replayable chosen-ciphertext attack (RCCA) [15]. Here the RCCA security is a weaker variant of the CCA security in the sense that a harmless mauling of the challenge ciphertext is tolerated.

In 2010, Matsuda, Nishimaki, and Tanaka [45] proposed a bidirectional multi-hop PRE scheme without bilinear maps, and claimed that their scheme is CCA-secure in the standard model. However, in 2011, Weng, Zhao, and Hanaoka [67] presented a concrete attack, and indicated that the scheme proposed by Matsuda, Nishimaki, and Tanaka fails to achieve the CCA-security.

In 2011, Shao, Liu, and Zhou [55] proposed a construction of unidirectional single-hop PRE and claimed that their scheme achieves key privacy without losing the CCA security in the standard model, which is an open problem left by Ateniese et al. [2]. In this thesis, however, we will show that their scheme is vulnerable to chosen-ciphertext attack (see Section 5.2 for more details).

In 2012, Hanaoka, Kawai, Kunihiro, Matsuda, Weng, Zhang, and Zhao [25] proposed a CCA security definition for PRE, and claimed that it is stronger than all the previous works. They also presented a unidirectional single-hop scheme which is secure in their security model. As we will show, their model can be further strengthened (see Section 3.3.3 for more details).

Conditional Proxy Re-Encryption. The traditional PRE is too powerful since it allow the proxy to convert all of Alice's second-level ciphertexts, without any discrimination. Rather than performing the conversion of all ciphertexts, in many cases, it may be sufficient for the proxy to convert only the ciphertext with a specific keyword. To address this issue, in 2009, Weng,

Deng, Ding, Chu, and Lai [66] introduced the concept of conditional proxy re-encryption (CPRE), whereby Alice has a fine-grained control over the delegation. As a result, Alice can flexibly assign Bob the decryption capability based on the conditions attached to the messages using a proxy. For example, suppose Alice is on vacation. She can make Bob to read only those messages which have the keyword "urgent" in their subject. This flexible delegation is obviously not possible with PRE schemes. To realise a CPRE scheme, in [66], Weng et al. used bilinear pairings and random oracles. They claimed that their scheme is secure against chosen-ciphertext attack. However, later in 2009, Weng, Yang, Tang, Deng, and Lai [68] showed that the scheme of Weng et al. [66] fails to achieve the CCA security. They then proposed a CPRE scheme, and proved its CCA-security under the well-studied decisional bilinear Diffie-Hellman (DBDH) assumption in the random oracle model. In the same paper, they left an open question of constructing a CCA-secure CPRE scheme without random oracles.

Recently, employing CPRE in the identity-based cryptographic setting, Liang, Liu, Tan, Wong, and Tang [40] defined the notion of identity-based conditional proxy re-encryption (IBCPRE) which can be seen as an extension of identity-based proxy re-encryption. They then proposed a construction for this primitive and claimed that their scheme achieves secure against adaptive condition and adaptive identity chosen-ciphertext attack in the standard model. One might consider that this is an answer to the open problem left by Weng et al. [68]. However, it is not known that IBCPRE, in general, implies CPRE. This is because these primitives are in different settings: in the model of IBCPRE it needs a trusted private key generator to generate all secret keys for identities, but in that of CPRE each user can generate a secret key by himself. In addition, in this thesis, we show that the scheme proposed by Liang et al. [40] is vulnerable even to selective condition chosen-ciphertext attack (see Section 6.2 for details). Therefore, how to construct a CCA-secure (IB)CPRE scheme without random oracles is still an open problem so far.

Related Works. Many papers in the literature, the first one of which being [44] by Mambo and Okamoto, consider applications where data encrypted under a public key should eventually be encrypted under a different key. In proxy encryption schemes [34, 35], a receiver Alice allows a delegatee Bob to decrypt ciphertexts intended for her with the help of a proxy by providing them with shares of her private key. This requires delegatees to store an additional secret for each new delegation. Ivan and Dodis [34] present efficient proxy encryption schemes based on RSA, the decision Diffie-Hellman problem as well as in an identity-based setting [11, 12, 53] under bilinear-map-related assumptions.

Proxy re-encryption schemes are a special kind of proxy cryptosystems where delegatees only need to store their own decryption key. They find applications in secure e-mail forwarding, digital rights management (DRM) or distributed storage systems (e.g., [3, 4]). The signature analogue, also suggested by Blaze et al. [8] in 1998, of PRE systems was formalized by Ateniese and Hohenberger [5] in 2005. The two techniques were notably combined [60] to design interoperable DRM systems where digital content can be translated between devices from different DRM domains.

From a theoretical point of view, the first positive obfuscation result for a complex cryptographic functionality was presented by Hohenberger, Rothblum, Shelat and Vaikuntanathan [30]: they proved the existence of an efficient program obfuscator for a family of circuits implementing

re-encryption.

In [2], Ateniese, Benson, and Hohenberger analyzed the notion of ciphertext anonymity (a.k.a. key privacy) in proxy re-encryption. This notion demands that even the proxy performing translations be unable to infer useful information on the identities of the participants between which it re-encrypts ciphertexts. PRE with this property can be useful in various applications such as distributed file systems, digital rights management, credential system, and mailing lists. The first key-private PRE was proposed by Ateniese et al. [2]. This scheme is only CPA-secure. The first CCA-secure key-private PRE was proposed by Shao, Liu, Wei, and Ling [59]. This scheme is proven secure under the decisional Diffie-Hellman assumption in the random oracle model. More recently, Zheng, Zhu, Zhu, and Zhang [69] proposed an improved key-private PRE with CCA security. The first lattice based key-private PRE was proposed by Aono, Boyen, Phong, and Wang [1]. These two schemes are also CCA-secure in the random oracle model.

Recently, as cloud computing emerges, PRE gains much more attention as one of the key security components to provide secure cloud services. The security against corrupted proxies is especially important in such applications since the proxies may be out of control of honest users and the proxies are more likely to be attacked than those in on-premise systems. In [3], Ateniese et al. mentioned the security notion, *non-transferability*, with respect to the security against malicious proxies, which is described as "*The (malicious) proxy and a set of colluding delegatees cannot re-delegate decryption rights.*" They also note that *"achieving a proxy scheme that is non-transferable, in the sense that the only way for Bob to transfer offline decryption capabilities to Carol is to expose his own secret key, seems to be the main open problem left for proxy re-encryption.*" Until now, some attempts for non-transferable PRE have been taken. For example, in the scheme proposed by Libert and Vergnaud [41], a delegator can identify the malicious proxies by analyzing a re-encryption key to convert ciphertexts of the delegator into some malicious user's generated (forged) by the colluding proxies and delegatees. Although it is one possible approach to the non-transferable PRE, it still cannot prevent colluding proxies and delegatees from re-delegating the decryption rights. Further, the scheme is less efficient in the sense that the ciphertext size depends on the number of delegations and it is only proved to be secure against chosen plaintext attack (CPA). In the scheme by Wang et al. [63] which is an ID-based PRE scheme, a trusted third party (privatekey generator, PKG) takes part in generating re-encryption keys. This approach, however, is not a complete solution to non-transferable PRE because, as pointed out by He, Chim, Hui, and Yiu [27], it is just a transformation of "delegatee-proxy-collusion transferable problem" to "PKG alone transferable problem." In the ID-based scheme by He, Chim, Hui, and Yiu [27], the delegator and the delegatee communicate and send some information to each other to generate the re-encryption key (The delegator also communicates with PKG). Therefore, colluding proxies and delegatees cannot generate a new re-encryption key without delegator's help. In IWSEC 2011, Hayashi, Matsushita, Yoshida, Fujii, and Okada introduced the unforgeability of re-encryption keys against collusion attack (UFReKey-CA), which is a relaxed notion of the non-transferability. They also proposed a stronger security notion, the *strong* unforgeability of re-encryption keys against collusion attack (sUFReKey-CA). Since sUFReKey-CA implies UFReKey-CA and sUFReKey-CA is simpler (i.e. easier to treat) definition than UFReKey-CA, sUFReKey-CA is useful to prove UFReKey-CA. They then proposed two concrete constructions of PRE and claimed that they meet both replayable-CCA security and sUFReKey-CA under two new variants of the Diffie-Hellman inversion assumption. However, in [31], Isshiki, Nguyen, and

Tanaka presented two concrete attacks to their PRE schemes. The first attack is to the sUFReKey-CA property on their two schemes. The second attack is to the assumptions employed in the security proofs for sUFReKey-CA of their two schemes. Therefore, it is an open problem of constructing UFReKey-CA secure PRE schemes.

In [24] , Green and Ateniese studied the problem of identity-based PRE and proposed a unidirectional scheme that can be made chosen-ciphertext secure. Their security results are presented only in the random oracle model. Chu and Tzeng [18] proposed a new multi-hop unidirectional identity-based proxy re-encryption scheme in the standard model. However, their scheme is not chosen-ciphertext secure, Shao and Cao [54] pointed out that its transformed ciphertext can be modified to another well-formed transformed ciphertext by anyone. In [38], Lai, Zhu, Deng, Liu, and Kou gave new constructions on IB-PRE which are based on identity-based mediated encryption. Luo, Hu, and Chen [43] also gave a new generic IB-PRE construction based on an existing IBE scheme. All these schemes follow Green-Ateniese token paradigm, which makes the decryption cost and size of ciphertext grow linearly with the re-encryption times. In addition, Matsuo [46] proposed a new proxy re-encryption system for identity-based encryption, but his solution needs a re-encryption key generator to generate re-encryption keys. Wang, Cao, and Wang [62] proposed the first CCA-secure IBPRE scheme in the standard model. Shao, Wei, Ling, and Xie [56] proposed an identity-based conditional proxy re-encryption. Recently, Shao and Cao [58] proposed a generic construction for IBPRE from non-anonymous hierarchical identity-based encryption. However, none of the above IBPRE schemes are anonymous. The first anonymous IBPRE scheme was proposed by Shao [57].

1.2 Contributions

The contributions of the thesis are summarized below:

- **New CCA security notions for PRE:** In CT-RSA 2012, Hanaoka et al. [25] proposed a chosen-ciphertext (CCA) security definition for unidirectional single-hop PRE, and claimed that it is stronger than all of the previous works. However, in this work, we show that their definition is indeed only a somewhat strengthened variant of the replayable-CCA one, and does not fully capture the CCA security notion. We then present a new CCA security definition which is extended from theirs. In particular, our new definition naturally extends that of [14] by giving an adversary all the possible resources as the observation by Hanaoka et al. [25]. Then, we point out why our security model is stronger than that of Hanaoka et al. [25] by giving PRE schemes which are secure in the model of [25] but are not secure in our model (See Section 3.3.3). Moreover, our security model is the strongest one to date. By the same way, we also present a new CCA security definition for bidirectional single-hop PRE. Finally, we extend this CCA security notion to the framework of unidirectional single-hop CPRE and obtaining new selective and adaptive CCA security notions for CPRE, respectively.

- **Factoring-Based Proxy Re-Encryption Schemes:** In this work, we propose the first PRE schemes based on the hardness of the factoring problem. In particular, we make the following contributions:

1. We present a CPA-secure bidirectional and multi-hop PRE scheme, under the factoring assumption, in the standard model (i.e., without the random oracle idealization). Our scheme based on the public key encryption (PKE) scheme proposed by Wee [65] which is indistinguishable against chosen-plaintext attack (IND-CPA). (See Section 4.3.1).
2. We present a bidirectional and single-hop PRE scheme which is secure against the chosen-ciphertext attack, under the factoring assumption in the random oracle model. In order to construct the scheme, we modify the IND-CPA-secure PKE [65] such that it achieves the IND-CCA security by using Fujisaki-Okamoto transformation. We also propose a new factoring-based strongly unforgeable signature scheme which is a variant of the Schnorr scheme [50]. By combining these two primitives, we obtain a CCA-secure PRE scheme. Our scheme achieves the CCA security on both second-level and first-level (i.e., the original and transformed, respectively) ciphertext in the random oracle model. (See Section 4.3.2).
3. We present a CCA-secure unidirectional and single-hop PRE scheme by using "token-controlled encryption" technique to extend the above bidirectional and single-hop PRE scheme. Hence, the security of this scheme is also proven in the random oracle model, assuming the hardness of factoring. This scheme also achieves the CCA security on both second-level and first-level ciphertext. (See Section 4.3.3).

- **A Proxy Re-Encryption Scheme with Bilinear Maps:** In this work, we first show that the PRE scheme proposed by Shao et al. [55] is vulnerable to chosen-ciphertext attack, and present a concrete attack to break the CCA security of it. We then propose a unidirectional PRE scheme which is secure in the full CCA security model without random oracles. This is also the first CCA-secure unidirectional PRE scheme in the standard model. Our scheme is efficient and relies on a mild complexity assumption in bilinear groups. Additionally, our scheme also achieves the security in the sense of master secret security which is proposed by Ateniese et al. [4].

  See Table 1 for details of the comparison on our PRE schemes with related previous works. ROM stands for the random oracle model. "1st-CCA" means that the CCA security of the first-level ciphertext. "$\leftrightarrow$" and "$\rightarrow$" mean "bidirectional" and "unidirectional", respectively.

- **CCA-Secure Conditional Proxy Re-Encryption Schemes without Random Oracles:** In this work, we first show that the CPRE scheme proposed by Liang et al. [40] is vulnerable to chosen-ciphertext attack, and present a concrete attack to break the CCA security of it. We then propose two CPRE schemes which are secure against the selective and adaptive condition chosen-ciphertext attack (sCond-CCA and aCond-CCA, for short), respectively. These schemes are extended from the CCA-secure PRE scheme proposed in Section 5.3. Specifically, the first scheme is constructed by using the technique of constructing the selective-identity secure identity-based encryption proposed by Boneh and Boyen [13]. The other one is presented by using the technique of constructing the adaptive-identity secure identity-based encryption proposed by Waters [64]. Both of these two schemes can be proved secure without random oracles, which answers the problem left by Weng et al. [68].

Authors	Assumption	Security of 2nd-level CT	1st-CCA	Bilinear Map	ROM	Direction	Hop
BBS98 [8]	DDH	CPA	no	no	no	$\leftrightarrow$	multi
AFGH05 [3]	eDBDH	CPA	no	yes	no	$\rightarrow$	single
CH07 [16]	DBDH	RCCA	no	yes	no	$\leftrightarrow$	multi
LV08 [42]	3-wDBDHI	RCCA	no	yes	no	$\rightarrow$	single
DWLC08 [20]	CDH	CCA	no	no	yes	$\leftrightarrow$	single
CWYD10 [17]	CDH	CCA	no	no	yes	$\rightarrow$	single
CDL11 [14]	CDH	CCA	yes	no	yes	$\rightarrow$	single
HMY+11 [26]	3-wDBDHI	RCCA	no	yes	no	$\rightarrow$	single
HKK+12 [25]	DBDH	CCA	no	yes	no	$\rightarrow$	single
Ours(Sec. 4.3.1)	**Factoring**	**CPA**	**no**	**no**	**no**	$\leftrightarrow$	**multi**
Ours(Sec. 4.3.2)	**Factoring**	**CCA**	**yes**	**no**	**yes**	$\leftrightarrow$	**single**
Ours(Sec. 4.3.3)	**Factoring**	**CCA**	**yes**	**no**	**yes**	$\rightarrow$	**single**
Ours(Sec. 5.3)	**6-AmDBDH & 2-AmCDH**	**CCA**	**yes**	**yes**	**no**	$\rightarrow$	**single**

Table 1: Comparison of our PRE schemes with previous works.

Authors	Assumption	Security of 2nd-level CT	1st-CCA	ROM
WDD+09 [66]	3-QBDH	not Cond-CCA	no	yes
WYT+09 [68]	DBDH	adaptive Cond-CCA	yes	yes
FSW09 [22]	3-QDBDH & q-ABDHE	adaptive Cond-RCCA	—	no
CWC+09 [19]	n-BDHE	selective Cond-RCCA	—	no
VSRR11 [61]	CDH	adaptive Cond-CCA	—	yes
LLT+12 [40]	DBDH	not Cond-CCA	no	no
LHS+13 [39]	(P, Q, f)-GDDHE	adaptive Cond-CCA	—	yes
SYL13 [51]	q-wDBDHI	adaptive Cond-RCCA	—	no
Ours (Sec. 6.3.1)	**6-AmDBDH**	**selective Cond-CCA**	**yes**	**no**
Ours (Sec. 6.3.2)	**6-AmDBDH**	**adaptive Cond-CCA**	**yes**	**no**

Table 2: Comparison of our results with the existing CPRE schemes.

It is interesting that our schemes have the same length of a public key, a secret key, and a ciphertext as the PRE scheme in Section 5.3. Our schemes have also almost the same computational cost as the underlying scheme (see Section 6.3.3 for details).

See Table 2 for details of the comparison about our schemes with related previous works. "not Cond-CCA" means that the corresponding scheme fails to achieve the condition CCA-security. "Cond-RCCA" stands for condition replayable CCA-security. "—" means that the CCA security of the corresponding scheme is not considered.

1.3 Organization

The rest of the thesis is structured as follows:

In Chapter 2, after introducing notations used in the whole thesis, we review the factoring assumption, the properties of bilinear maps and the intractability assumptions that our schemes rely on. Next, we review the models and security definitions on public key encryption, symmetric key encryption, signature, and target-collision resistant hash function, which are the main tools in our constructions.

In Chapter 3, we first recall the concepts and the security notions of bidirectional multi-hop PRE. In Section 3.2, we then review the model of bidirectional single-hop PRE, and present new CCA security notions for it. In Section 3.3, we present new CCA security definitions for unidirectional single-hop PRE by extending that of the previous works. In order to point out why our security models are stronger than those of the previous works, we present concrete PRE schemes which are secure in their model, but are not secure in our models. In Section 3.4, we review the model and present new CCA security definitions of conditional PRE.

In Chapter 4, we focus on constructing PRE schemes which are based on the hardness of the factoring problem. In Section 4.2, we introduce a new factoring-based strongly unforgeable signature scheme which will be used to construct our schemes. In Section 4.3, we first propose a bidirectional multi-hop PRE scheme and show that it meets the CPA security under the factoring assumption, in the standard model (i.e., without using random oracles). We then present our main scheme which is CCA secure (in the random oracle model) under the factoring assumption. Finally, we show how to extend our main scheme to achieve a unidirectional and single-hop PRE scheme which is secure in the sense of chosen-ciphertext attack. The security proofs of these schemes are given in Section 4.4.

In Chapter 5, we first give an overview of CCA-secure PRE in the literature. We then show that the PRE scheme proposed by Shao et al. [55] is vulnerable to chosen-ciphertext attack in Section 5.2. In Section 5.3, we propose a unidirectional single-hop PRE scheme which is secure in the full CCA security model without random oracles. Our scheme is obtained by extending the PKE scheme proposed by Lai, Deng, Liu, and Kou [37]. The security proofs of these schemes are given in Section 5.5.

In Chapter 6, we first give introductions and backgrounds of CPRE. We then show that the CPRE scheme proposed by Liang et al. [40] is vulnerable to chosen-ciphertext attack in Section 6.2. In Section 6.3, we then propose two CPRE schemes which are secure against the selective and adaptive condition chosen-ciphertext attack, respectively. These schemes are extended from the CCA-secure PRE scheme proposed in Chapter 5. Our schemes are the first CPRE schemes

which are secure against condition chosen-ciphertext attack without random oracles. The security proofs of these schemes are given in Section 6.4.

Finally, we conclude in Chapter 7, and give several open problems.

Chapter 2

Preliminaries

Contents

2.1 Notations **11**
2.2 Cryptographic Assumptions **11**
2.2.1 Factoring Assumption 12
2.2.2 Bilinear Maps and Complexity Assumptions 13
2.2.3 The Gap Hashed Diffie-Hellman Assumption 14
2.3 Public Key Encryption **14**
2.4 Symmetric Key Encryption **16**
2.5 Signature **17**
2.6 Target-Collision Resistant Hash Function **18**

2.1 Notations

We denote by $\mathbb{N}$ the set of all integers, and for an integer $k \in \mathbb{N}$ we denote by $[k]$ the set $\{1, \ldots, k\}$. If $\mathbf{S}$ is a set then $s \leftarrow_R \mathbf{S}$ denotes the operation of picking an element s of $\mathbf{S}$ uniformly at random. A probabilistic polynomial-time (PPT) algorithm is a randomized algorithm which runs in strict polynomial time. For a probabilistic algorithm A, we denote $y = A(x; R)$ the process of running A on input x and with randomness R, and assigning y the result. We write $y = A(x; \cdot)$ if the randomness R is unknown. We write $y \leftarrow A(x)$ for $y = A(x; R)$ with uniformly chosen R from the randomness space of A. By λ we denote the security parameter, which indicates the "amount of security" we desire. We use $negl(n)$ to denote a negligible function in n, i.e., it always holds that $negl(n) < 1/n^c$ for any $0 < c \in \mathbb{Z}$ for sufficiently large n.

2.2 Cryptographic Assumptions

In this section, we review several cryptographic assumptions on which our schemes are based. We start by reviewing the factoring assumption. Then, we recall the properties of bilinear maps and

present a new assumption which is a variant of the modified decisional bilinear Diffie-Hellman assumption (a.k.a. mDBDH assumption). Finally, we review the gap hashed Diffie-Hellman assumption proposed by Kiltz [36].

2.2.1 Factoring Assumption

Let $N = PQ$ be a Blum integer for safe primes P, Q, i.e., $P, Q \equiv 3 \pmod 4$ where $p = (P-1)/2$ and $q = (Q-1)/2$ are both primes. Let $\mathbb{J}_N$ denote the subgroup of $\mathbb{Z}_N^*$ with Jacobi symbol $+1$, and let $\mathbb{QR}_N$ denote the subgroup of quadratic residues (i.e. $\mathbb{QR}_N := \{x \in \mathbb{Z}_N^* : \exists y \in \mathbb{Z}_N^* \text{ with } y^2 = x \bmod N\}$). Following [28, 29], we work over the cyclic group of signed quadratic residues, given by the quotient group $\mathbb{QR}_N^+ := \mathbb{QR}_N / \pm 1$. $\mathbb{QR}_N^+$ is a cyclic group of order pq and is efficiently recognizable (by verifying that the Jacobi symbol is $+1$). In addition, the map $x \mapsto x^2$ is a permutation over $\mathbb{QR}_N^+$. Furthermore, assuming that factoring Blum integers are hard on average and that safe primes are dense, the family of permutations $\mathsf{SQ} : x \mapsto x^2$ (indexed by N) acting on the groups $\mathbb{QR}_N^+$ is one-way.

Factoring Assumption. We assume a probabilistic polynomial time (PPT) algorithm $\mathbf{BlumGen}$ that, on input a security parameter 1^k, generates two random safe λ-bit primes $P = 2p + 1$ and $Q = 2q + 1$, then outputs a Blum integer $N = PQ$.

Definition 1 (Factoring Assumption [29])**.** *For an algorithm $\mathcal{A}$, we define its* factoring advantage *as*

$$\mathrm{Adv}^{\mathrm{Fac}}_{\mathbf{BlumGen},\mathcal{A}}(k) := \Pr\left[N = PQ : N \leftarrow \mathbf{BlumGen}(1^k), \mathcal{A}(N) = (P, Q)\right].$$

We say that the (t, ε)-Factoring assumption for $\mathbf{BlumGen}$ holds if no t-time algorithm $\mathcal{A}$ has factoring advantage at least ε.

Iterated Squaring. Following [65], in our constructions, we make use of (N, g) as a part of the public parameter, where N is a random 2λ-bit Blum integer and g is chosen uniformly from $\mathbb{QR}_N^+$. We will henceforth assume g is a generator for $\mathbb{QR}_N^+$, which happens with probability $1 - O(1/\sqrt{N})$. Assuming that factoring Blum integers are hard on average and that safe primes are dense, the family of permutations $\mathsf{ISQ} : x \mapsto x^{2^\lambda}$ (indexed by N) acting on the groups $\mathbb{QR}_N^+$ is one-way. Using the Blum-Blum-Shub (BBS) pseudorandom generator [9], we may extract λ hard-core bits from $x \in \mathbb{QR}_N^+$ that are pseudorandom even given x^{2^λ}, that is:

$$BBS_N(x) := \left(\mathsf{lsb}_N(x), \mathsf{lsb}_N(x^2), \ldots, \mathsf{lsb}_N(x^{2^{\lambda-1}})\right).$$

The pseudorandomness of BBS is defined as follows.

Definition 2 (Pseudorandomness of BBS Generator [29])**.** *For an algorithm $\mathcal{A}$, define*

$$\mathrm{Adv}^{\mathrm{BBS}}_{\mathcal{A}}(\lambda) = \left|\Pr[\mathcal{A}(N, z, BBS_N(u)) = 1] - \Pr\left[\mathcal{A}(N, z, U_{\{0,1\}^\lambda}) = 1\right]\right|,$$

where $N \leftarrow \mathbf{BlumGen}(1^\lambda), u \leftarrow_R \mathbb{QR}_N, z = u^{2^\lambda}$, and $U_{\{0,1\}^\lambda} \in \{0,1\}^\lambda$ is independently and uniformly chosen. We say that $\mathcal{A}$ (t, ε)-breaks BBS if $\mathcal{A}$'s running time is at most $t = t(\lambda)$ and $\mathrm{Adv}^{\mathrm{BBS}}_{\mathcal{A}}(\lambda) \geq \varepsilon = \varepsilon(\lambda)$.

The following lemma says that any BBS-distinguisher can be used to factor Blum integers. See [29] for the proof.

Lemma 1 (BBS-distinguisher $\Rightarrow$ Factoring Algorithm [9, 29])**.** *For every algorithm $\mathcal{A}$ that $(t_{BBS}, \varepsilon_{BBS})$-breaks BBS, there is an algorithm $\mathcal{B}$ the $(t_{fac}, \varepsilon_{fac})$-factors Blum integers, where $t_{fac} \approx k^4 t_{BBS}/\varepsilon_{BBS}^2$ and $\varepsilon_{fac} = \varepsilon_{BBC}/\lambda$.*

Therefore, since the factoring assumption implies the one-wayness of ISQ, we have the following trivial lemma.

Lemma 2 ([29])**.** ISQ *is a one way function $\Longleftrightarrow$ the factoring problem is hard.*

2.2.2 Bilinear Maps and Complexity Assumptions

Groups $(\mathbb{G}, \mathbb{G}_T)$ of prime order p are called *bilinear map groups* if there is a mapping $\mathbf{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ with the following properties: (1) bilinearity: $\mathbf{e}(g^a, h^b) = \mathbf{e}(g, h)^{ab}$ for any $(g, h) \in \mathbb{G} \times \mathbb{G}$ and $a, b \in \mathbb{Z}_p$, (2) efficient computability for any input pair, (3) non-degeneracy: $\mathbf{e}(g, h) \neq 1_{\mathbb{G}_T}$ whenever $g, h \neq 1_{\mathbb{G}}$.

In this thesis, the proposed scheme shall use a variant of the modified decisional bilinear Diffie-Hellman assumption (a.k.a. mDBDH assumption) [16, 49] named 6-augmented modified decisional bilinear Diffie-Hellman (6-AmDBDH) assumption. The details of this assumption is as follows.

Definition 3 (6-AmDBDH Assumption)**.** *The 6-augmented modified decisional bilinear Diffie-Hellman (6-*AmDBDH*) problem in groups $(\mathbb{G}, \mathbb{G}_T)$ is, given $(g, g^a, g^b, g^c, g^{a/c}, g^{c^2}, g^{ac}, g^{ac^2}, g^{ac^3}, g^{ac^4}, Q) \in \mathbb{G}^{10} \times \mathbb{G}_T$, where $a, b, c \leftarrow_R \mathbb{Z}_p$, decide whether $Q = \mathbf{e}(g, g)^{ab/c^2}$ holds. For an adversary $\mathcal{A}$, we define its advantage in solving the* 6-AmDBDH *problem in groups $(\mathbb{G}, \mathbb{G}_T)$ as*

$$\begin{aligned} & \mathrm{Adv}_{\mathcal{A}}^{6\text{-AmDBDH}} \stackrel{def}{=} \\ & \big|\Pr[\mathcal{A}(g, g^a, g^b, g^c, g^{a/c}, g^{c^2}, g^{ac}, g^{ac^2}, g^{ac^3}, g^{ac^4}, \mathbf{e}(g, g)^{ab/c^2}) = 1 | a, b, c \leftarrow_R \mathbb{Z}_p] \\ - \ & \Pr[\mathcal{A}(g, g^a, g^b, g^c, g^{a/c}, g^{c^2}, g^{ac}, g^{ac^2}, g^{ac^3}, g^{ac^4}, \mathbf{e}(g, g)^z) = 1 | a, b, c, z \leftarrow_R \mathbb{Z}_p]\big|. \end{aligned}$$

*We say that the (t, ε)-6-*AmDBDH *assumption holds in $(\mathbb{G}, \mathbb{G}_T)$ if no t-time adversary $\mathcal{A}$ has advantage at least ε in solving the 6-*AmDBDH *problem in $(\mathbb{G}, \mathbb{G}_T)$.*

The hardness of the 6-AmDBDH problem is implied by that of the general Diffie-Hellman problem in the generic bilinear group model [10]. In particular, a generic attacker's advantage in solving the 6-AmDBDH problem is less than 16 times of that in solving the DBDH problem. The details are given in Section 5.4.

We also use a variant of the computational Diffie-Hellman assumption (a.k.a. CDH assumption) named 2-augmented modified computational Diffie-Hellman (2-AmCDH) assumption, which is almost identical to the CDH assumption except the introduction of the additional terms $g^{ba^2}, g^{\frac{b}{a}}$.

Definition 4 (2-AmCDH Assumption)**.** *The 2-augmented modified computational Diffie-Hellman (2-*AmCDH*) problem is computing* g^{ab} *given* $(g, g^a, g^b, g^{ba^2}, g^{\frac{b}{a}}) \in \mathbb{G}^5$*, where* $a, b \leftarrow_R \mathbb{Z}_p$*. For an adversary* $\mathcal{A}$*, we define its advantage in solving the 2-*AmCDH *problem in* $\mathbb{G}$ *as* $\mathrm{Adv}_{\mathcal{A}}^{\text{2-AmCDH}} = \Pr[\mathcal{A}(g, g^a, g^b, g^{ba^2}, g^{\frac{b}{a}}) = g^{ab}]$*, where the probability is taken over the random choices of* a, b *and those made by* $\mathcal{A}$*. We say that the* (t, ε)*-2-AmCDH assumption holds in* $\mathbb{G}$ *if no t-time algorithm* $\mathcal{A}$ *has advantage at least* ε *in solving the 2-AmCDH problem in* $\mathbb{G}$*.*

2.2.3 The Gap Hashed Diffie-Hellman Assumption

Following [36], let Gen be a parameter generation algorithm that on input 1^k returns the description of a multiplicative cyclic group $\mathbb{G}$ of prime order p, where $2^k < p < 2^{k+1}$, and a random generator g of $\mathbb{G}$. Gen furthermore outputs the description of a random hash function $H : \mathbb{G} \to \{0,1\}^{\ell(k)}$ that outputs $\ell(k)$ bits for a fixed polynomial $\ell(\cdot)$.

The gap hashed Diffie-Hellman (GHDH) assumption [36] states, roughly, that the two distributions $(g^x, g^y, H(g^{xy}))$ and (g^x, g^y, R) are computationally indistinguishable when x, y are drawn at random from $\mathbb{Z}_p$ and R is drawn at random from $\{0,1\}^{\ell(k)}$. This assumption should hold relative to an oracle that efficiently solves the DDH problem (See [36] for more details).

More formally let Gen be a parameter generation algorithm. To an adversary $\mathcal{B}$ we associate the following experiment.

Experiment $\mathbf{Exp}^{\mathrm{ghdh}}_{\mathsf{Gen},\mathcal{B}}(1^k)$,

$\mathcal{HG} = (\mathbb{G}, g, p, H) \leftarrow \mathsf{Gen}(1^k)$,

$x, y \leftarrow_R \mathbb{Z}_p^*$, $Q_0 \leftarrow_R \{0,1\}^{\ell(k)}$, $Q_1 \leftarrow H(g^{xy})$,

$\gamma \leftarrow_R \{0,1\}$,

$\gamma' \leftarrow \mathcal{B}^{\mathrm{DDHsolve}_{\mathbb{G}}(\cdot,\cdot,\cdot,\cdot)}(1^k, \mathcal{HG}, g^x, g^y, Q_\gamma)$,

If $\gamma \neq \gamma'$ then return 0 else return 1.

Here the oracle $\mathrm{DDHsolve}_{\mathbb{G}}(g, g^a, g^b, g^c)$ returns 1 iff $ab = c \mod p$. Note that, since we make use of bilinear map, $\mathbf{e}(\cdot,\cdot)$ can be used to simulate the oracle $\mathrm{DDHsolve}_{\mathbb{G}}$. We define the advantage of $\mathcal{B}$ in the above experiment as

$$\mathrm{Adv}^{\mathrm{ghdh}}_{\mathrm{Gen},\mathcal{B}}(k) = \left| \Pr\left[\mathbf{Exp}^{\mathrm{ghdh}}_{\mathsf{Gen},\mathcal{B}}(1^k) = 1\right] - \frac{1}{2} \right|.$$

We say that the *GHDH assumption relative to group generator* Gen holds if $\mathrm{Adv}^{\mathrm{ghdh}}_{\mathrm{Gen},\mathcal{B}}(k)$ is a negligible function in k for all polynomial-time adversaries $\mathcal{B}$.

2.3 Public Key Encryption

In this section, we introduce the notions and attack models of public-key encryption. Public-key encryption is one of the most popular cryptographic primitives. It is used to send messages safely from the sender to the receiver using an open channel. Here an open channel means that anyone

can eavesdrop the communication between the sender and the receiver. Therefore, public-key encryption schemes require that messages are correctly sent from the sender to the receiver, and the messages cannot be guessed by others, even if the communication of the scheme is eavesdropped on.

Formally, a public-key encryption is a three tuple of algorithms $\Pi = (KGen, Enc, Dec)$. The key generation algorithm $KGen$ generates a pair $(pk, sk) \leftarrow_R KGen(1^n)$, where pk is a public key and sk is a secret key. The encryption algorithm Enc takes a public key pk and a plaintext m, and returns a ciphertext $c \leftarrow_R Enc_{pk}(m)$. The decryption algorithm Dec takes a secret key sk and a ciphertext c, and returns m or *reject* (denoted by $\perp$). It is required that for any $(pk, sk) \leftarrow_R KGen(1^n)$, any message m from the plaintext space, it holds that

$$Dec(sk, Enc(pk, m)) = m.$$

The chosen-plaintext attack (IND-CPA) game is defined as follows.

Definition 5 (IND-CPA attack)**.** *A public-key encryption scheme Π = (KGen, Enc, Dec) is IND-CPA secure if for any PPT adversary $A = (A_1, A_2)$ it holds that*

$$\mathbf{Adv}^{\text{IND-CPA}}_{\Pi,A}(n) \stackrel{def}{=} \left|\Pr\left[\mathbf{Expt}^{\text{IND-CPA}}_{\Pi,A}(0) = 1\right] - \Pr\left[\mathbf{Expt}^{\text{IND-CPA}}_{\Pi,A}(1) = 1\right]\right|$$

is negligible in n, where $\mathbf{Expt}^{\text{IND-CPA}}_{\Pi,A}(b)$ is defined as follows:

1. $(pk, sk) \leftarrow KGen(1^n)$.
2. $(m_0, m_1, st_{A_1}) \leftarrow A_1(pk)$ *such that* $|m_0| = |m_1|$.
3. $C \leftarrow Enc_{pk}(m_b)$.
4. $b' \leftarrow A_2(C, st_{A_1})$.
5. *Output* b'.

In the a-priori chosen-ciphertext attack (IND-CCA1) game, the adversary is allowed to adaptively access a decryption oracle $Dec(sk, \cdot)$ before choosing a pair of messages and sending to the challenger. Where, $Dec(sk, \cdot)$ is an algorithm that receives as input a ciphertext and outputs a decryption using the secret key sk.

Definition 6 (IND-CCA1 attack)**.** *A public-key encryption scheme Π = (KGen, Enc, Dec) is IND-CCA1 secure if for any PPT adversary $A = (A_1, A_2)$ it holds that*

$$\mathbf{Adv}^{\text{IND-CCA1}}_{\Pi,A}(n) \stackrel{def}{=} \left|\Pr\left[\mathbf{Expt}^{\text{IND-CCA1}}_{\Pi,A}(0) = 1\right] - \Pr\left[\mathbf{Expt}^{\text{IND-CCA1}}_{\Pi,A}(1) = 1\right]\right|$$

is negligible in n, where $\mathbf{Expt}^{\text{IND-CCA1}}_{\Pi,A}(b)$ is defined as follows:

1. $(pk, sk) \leftarrow KGen(1^n)$.
2. $(m_0, m_1, st_{A_1}) \leftarrow A_1^{Dec(sk,\cdot)}(pk)$ *such that* $|m_0| = |m_1|$.

3. $C \leftarrow Enc_{pk}(m_b)$.

4. $b' \leftarrow A_2(C, st_{A_1})$.

5. *Output* b'.

In the adaptively-chosen-ciphertext attack (IND-CCA2) game, the adversary is allowed to adaptively access a decryption oracle $Dec(sk, \cdot)$ not only before choosing a pair of messages and sending to the challenger, but also after that. We denote by $Dec_{\neq C}(sk, \cdot)$ a decryption oracle that decrypts any ciphertext other than c.

Definition 7 (IND-CCA2 attack)**.** *A public-key encryption scheme* Π *= (KGen, Enc, Dec) is IND-CCA2 secure if for any PPT adversary* $A = (A_1, A_2)$ *it holds that*

$$\mathbf{Adv}^{\text{IND-CCA2}}_{\Pi,A}(n) \stackrel{def}{=} \left|\Pr\left[\mathbf{Expt}^{\text{IND-CCA2}}_{\Pi,A}(0) = 1\right] - \Pr\left[\mathbf{Expt}^{\text{IND-CCA2}}_{\Pi,A}(1) = 1\right]\right|$$

is negligible in n*, where* $\mathbf{Expt}^{\text{IND-CCA2}}_{\Pi,A}(b)$ *is defined as follows:*

1. $(pk, sk) \leftarrow KGen(1^n)$.

2. $(m_0, m_1, st_{A_1}) \leftarrow A_1^{Dec(sk,\cdot)}(pk)$ *such that* $|m_0| = |m_1|$.

3. $C \leftarrow Enc_{pk}(m_b)$.

4. $b' \leftarrow A_2^{Dec_{\neq C}(sk,\cdot)}(C, st_{A_1})$.

5. *Output* b'.

2.4 Symmetric Key Encryption

We review the formal notion of symmetric key encryption and its security definition as follows.

Definition 8 (Symmetric Key Encryption [7])**.** *Let* $\mathcal{K}_D$ *be the key space, and* $\mathcal{M}$ *be the message space. A symmetric key encryption scheme* **SYM** *consists of the following algorithms:*

1. **SYM.Enc:** *Taking a key* $K \in \mathcal{K}_D$ *and a plaintext* $M \in \mathcal{M}$ *as input, this algorithm encrypts* M *into a ciphertext* e*. We write* $e \leftarrow \mathbf{SYM.Enc}(K, M)$.

2. **SYM.Dec:** *Taking* $K \in \mathcal{K}_D$ *and* e *as input, this algorithm decrypts* e *into* M*. Note that* M *can be* $\perp$*. We write* $M \leftarrow \mathbf{SYM.Enc}(K, e)$.

Definition 9 (CCA Security of Symmetric Key Encryption [7])**.** *Let* **SYM** *be a symmetric key encryption scheme as defined in Definition 8. Consider a game played with an attacker* $\mathcal{A}$*:*

Phase 1. *The game chooses* $K \leftarrow_R \mathcal{K}_D$.

Phase 2. $\mathcal{A}$ *issues encryption queries, each of which is denoted by* M*. On receiving this, the game computes* $e \leftarrow \mathbf{SYM.Enc}(K, M)$ *and gives* e *to* $\mathcal{A}$*.* $\mathcal{A}$ *also issues decryption queries, each of which is denoted by* e*. On receiving this, the game computes* $M \leftarrow \mathbf{SYM.Dec}(K, e)$ *and gives* M *to* $\mathcal{A}$*.*

Challenge. *$\mathcal{A}$ issues a challenge query (a pair of plaintexts) (m_0, m_1) such that $|m_0| = |m_1|$. On receiving this, the game picks $b \leftarrow_R \{0,1\}$, computes $e^* \leftarrow \mathbf{SYM.Enc}(K, m_b)$ and gives e^* to $\mathcal{A}$.*

Phase 3. *$\mathcal{A}$ continues to issue encryption and decryption queries as in Phase 2. However, a restriction here is that $\mathcal{A}$ is not allowed to issue e^* as decryption query. The game responds to $\mathcal{A}$'s queries in the same way as it did in Phase 2.*

Guess. *$\mathcal{A}$ outputs its guess $b' \in \{0,1\}$.*

We define A's advantage by $\mathrm{Adv}^{\text{IND-CCA}}_{\text{SYM},\mathcal{A}} = \left|\Pr[b' = b] - \frac{1}{2}\right|$.

2.5 Signature

The following definitions describe the functionality of a signature scheme, and the security notion of strong unforgeability that is used in this thesis.

Definition 10 (Signature Scheme)**.** *A signature scheme is a triplet $(\mathbf{SigGen}, \mathbf{Sig}, \mathbf{Vrf})$ of probabilistic polynomial-time algorithms such that:*

1. *The key generation algorithm $\mathbf{SigGen}$ receives as input a security parameter 1^n and outputs a verification key vk and a signing key sk.*

2. *The signing algorithm $\mathbf{Sig}$ receives as input a signing key sk and a message m (in some implicit message space), and outputs a signature σ.*

3. *The verification algorithm $\mathbf{Vrf}$ receives as input a verification key vk, a message m, and a signature σ, and outputs a bit $b \in \{0,1\}$.*

4. *For any message m it holds that $\mathbf{Vrf}(vk, m, \mathbf{Sig}(sk, m)) = 1$ with overwhelming probability over the internal coin tosses of $\mathbf{SigGen}$, $\mathbf{Sig}$, and $\mathbf{Vrf}$.*

Definition 11 (Strong Unforgeability). *A signature scheme $(\mathbf{SigGen}, \mathbf{Sig}, \mathbf{Vrf})$ is said to be strongly unforgeable if the success probability of any probabilistic polynomial-time adversary $\mathcal{A}$ in the following interaction is negligible in the security parameter:*

1. *$\mathbf{SigGen}(1^n)$ outputs (vk, sk), and $\mathcal{A}$ is given vk.*

2. *$\mathcal{A}$ can make signing queries by outputting messages m_i and is then given in return $\sigma_i = \mathbf{Sig}(sk, m_i)$. If $\mathcal{A}$ chooses not to output any message, we set $(m, \sigma) = (\perp, \perp)$.*

3. *$\mathcal{A}$ outputs a pair (m^*, σ^*).*

We say that $\mathcal{A}$ succeeds if $\mathbf{Vrf}(vk, m^*, \sigma^*) = 1$ and $(m^*, \sigma^*) \neq (m_i, \sigma_i)$ for any i.

2.6 Target-Collision Resistant Hash Function

In this section, we review the formal notion of target-collision resistant (TCR) hash function as follows.

Definition 12 (Target-Collision Resistant Hash Function)**.** *Let $H : X \to Y$ be a hash function. For an algorithm $\mathcal{A}$, define its advantage as*

$$\mathrm{Adv}^{\mathrm{TCR}}_{H,\mathcal{A}} = \Pr[x \leftarrow X, x' \leftarrow \mathcal{A}(x) : x' \neq x \wedge H(x') = H(x)].$$

We say that H is target-collision resistant if for any probabilistic polynomial-time algorithm $\mathcal{A}$, its advantage $\mathrm{Adv}^{\mathrm{TCR}}_{H,\mathcal{A}}$ is negligible.

Chapter 3

Models and Security Notions

Contents

3.1 Bidirectional Multi-Hop Proxy Re-Encryption 19
3.1.1 Model 19
3.1.2 Security Notions 20
3.2 Bidirectional Single-Hop Proxy Re-Encryption 21
3.2.1 Model 21
3.2.2 Security Notions 22
3.3 Unidirectional Single-Hop Proxy Re-Encryption 24
3.3.1 Model 24
3.3.2 Security Notions 25
3.3.3 Discussion in the Comparison with the Previous Security Models 28
3.4 Conditional Proxy Re-Encryption 34
3.4.1 Model 34
3.4.2 Security Notions 35

3.1 Bidirectional Multi-Hop Proxy Re-Encryption

In this section, we review the concept of bidirectional multi-hop PRE.

3.1.1 Model

In this section, we first review the definition of bidirectional multi-hop PRE proposed in [16], and then present its security model.

Definition 13 (Bidirectional Multi-Hop PRE)**.** *A bidirectional multi-hop PRE scheme is a tuple of algorithms* $\Pi = (\mathbf{Setup}, \mathbf{KGen}, \mathbf{RKGen}, \mathbf{Enc}, \mathbf{ReEnc}, \mathbf{Dec})$ *for message space* $\mathcal{M}$*:*

1. $\mathbf{Setup}(1^\lambda) \to PP$. *On input a security parameter* 1^λ*, the setup algorithm outputs a public parameter* PP.

2. $\mathbf{KGen}(PP) \to (pk, sk)$. *On input a public parameters* PP*, the key generation algorithm outputs a public key* pk *and a secret key* sk.

3. $\mathbf{RKGen}(PP, sk_i, sk_j) \to rk_{ij}$. *On input public parameter* PP*, two secret keys* sk_i*, and* sk_j*, the re-encryption key generation algorithm outputs a bidirectional re-encryption key* rk_{ij}.

4. $\mathbf{Enc}(PP, pk, m) \to CT$. *On input public parameter* PP*, a public key* pk*, and a message* $m \in \mathcal{M}$*, the encryption algorithm outputs a ciphertext* CT.

5. $\mathbf{ReEnc}(PP, rk_{ij}, CT_i) \to CT_j$. *On input public parameter* PP*, a re-encryption key* rk_{ij}*, and a ciphertext* CT_i *for* i*, the re-encryption algorithm outputs a ciphertext* CT_j *for* j *or the symbol* $\perp$.

6. $\mathbf{Dec}(PP, sk, CT) \to m$. *On input public parameter* PP*, a secret key* sk*, and a second-level ciphertext* CT*, the decryption algorithm outputs a message* $m \in \mathcal{M}$ *or the symbol* $\perp$.

To lighten notations, from now, we will omit the public parameter PP from the inputs of the algorithms.

For all $m \in \mathcal{M}$ and all pair $(pk_i, sk_i), (pk_j, sk_j)$, these algorithms should satisfy the following conditions of correctness:

$$\mathbf{Dec}(sk_i, \mathbf{Enc}(pk_i, m)) = m;$$
$$\mathbf{Dec}(sk_j, \mathbf{ReEnc}(\mathbf{RKGen}(sk_i, sk_j), \mathbf{Enc}(pk_i, m))) = m.$$

3.1.2 Security Notions

We present the formal definition of CPA security of bidirectional multi-hop PRE, denoted by BM-CPA. The game framework for BM-CPA security is as follows.

Definition 14 (Game Framework of BM-CPA Security)**.**

Setup. *The challenger* $\mathcal{C}$ *takes a security parameter* λ *and executes the setup algorithm to get the system parameter* PP. $\mathcal{C}$ *initializes two empty lists* L_{un} *and* L_{corr}*, then maintains it to record all uncorrupted users and corrupted users, respectively, in the game.*

Phase 1. $\mathcal{A}$ *can adaptively query to the following oracles* $\mathcal{O}_{unKGen}$*,* $\mathcal{O}_{corrKGen}$*,* $\mathcal{O}_{RKGen}$*, and* $\mathcal{O}_{ReEnc}$*:*

- $\mathcal{O}_{unKGen}$ *takes* i *and returns the error symbol* $\perp$ *if* $i \in L_{un} \cup L_{corr}$*; otherwise returns* $(pk_i, sk_i) \leftarrow \mathbf{KGen}(PP)$ *and adds* i *in* L_{un}.
- $\mathcal{O}_{corrKGen}$ *takes* i *and returns the error symbol* $\perp$ *if* $i \in L_{un} \cup L_{corr}$*, otherwise returns* $(pk_i, sk_i) \leftarrow \mathbf{KGen}(PP)$ *and adds* i *in* L_{corr}.

– $\mathcal{O}_{RKGen}$ *takes* i, j *and returns* $rk_{ij} \leftarrow \mathbf{RKGen}(sk_i, sk_j)$ *if* $i \neq j \in L_{un}$ *or* $i \neq j \in L_{corr}$*; otherwise returns the error symbol* $\perp$.

– $\mathcal{O}_{ReEnc}$ *takes* i, j *and a ciphertext* CT_i*. If* $i \neq j \in L_{un}$ *or* $i \neq j \in L_{corr}$*, then it computes* $rk_{ij} \leftarrow \mathbf{RKGen}(sk_i, sk_j)$*, and returns* $CT_j \leftarrow \mathbf{ReEnc}(rk_{ij}, CT_i)$*; otherwise returns the error symbol* $\perp$.

Challenge. *When* $\mathcal{A}$ *decides that Phase 1 is over, it outputs a target public key* pk_{i^*} *(we require* $i^* \in L_{un}$ *for* $\mathcal{A}$ *to win) and two equal-length plaintexts* $m_0, m_1 \in \mathcal{M}$*. The challenger* $\mathcal{C}$ *flips a random coin* $\sigma \in \{0, 1\}$*, and sends to* $\mathcal{A}$ *a challenge ciphertext* $CT^* \leftarrow \mathbf{Enc}(pk_{i^*}, m_\sigma)$.

Phase 2. $\mathcal{A}$ *issues queries as in Phase 1.*

Guess. *Finally,* $\mathcal{A}$ *outputs a guess* $\sigma' \in \{0, 1\}$.

We define $\mathcal{A}$*'s advantage in attacking the PRE scheme as* $\mathrm{Adv}^{\text{BM-CPA}}_{\mathbf{PRE},\mathcal{A}}(\lambda) = \left|\Pr[\sigma' = \sigma] - 1/2\right|$*, where the probability is taken over the random coins consumed by the challenger and the adversary. A bidirectional and multi-hop PRE scheme is defined to be* $(q_{rk}, q_{re}, q_{dec})$*-BM-CPA secure, if for any PPT adversary* $\mathcal{A}$ *who makes at most* q_{rk} *re-encryption key generation queries, at most* q_{re} *re-encryption queries and at most* q_{dec} *decryption queries, we have* $\mathrm{Adv}^{\text{BM-CPA}}_{\mathbf{PRE},\mathcal{A}}(\lambda) \leq negl(\lambda)$.

3.2 Bidirectional Single-Hop Proxy Re-Encryption

In this section, we review the concept of bidirectional single-hop PRE.

3.2.1 Model

In this section, we first review the concept of bidirectional single-hop PRE, and then present its security model.

Definition 15 (Bidirectional Single-Hop PRE [42])**.** *A bidirectional single-hop PRE scheme is a tuple of algorithms* $\Pi = (\mathbf{Setup}, \mathbf{KGen}, \mathbf{RKGen}, \mathbf{Enc}, \mathbf{ReEnc}, \mathbf{Dec}_1, \mathbf{Dec}_2)$ *for message space* $\mathcal{M}$*:*

1. $\mathbf{Setup}(1^\lambda) \rightarrow PP$*. On input a security parameter* 1^λ*, the setup algorithm outputs a public parameters* PP.

2. $\mathbf{KGen}(PP) \rightarrow (pk, sk)$*. On input public parameter* PP*, the key generation algorithm outputs a public key* pk *and a secret key* sk.

3. $\mathbf{RKGen}(PP, sk_i, sk_j) \rightarrow rk_{ij}$*. On input public parameter* PP*, two secret keys* sk_i*, and* sk_j*, the re-encryption key generation algorithm outputs a bidirectional re-encryption key* rk_{ij}.

4. $\mathbf{Enc}(PP, pk, m) \rightarrow CT$. *On input public parameter PP, a public key pk, and a message $m \in \mathcal{M}$, the encryption algorithm outputs a second-level ciphertext CT that can be re-encrypted into a first-level one (intended for a possibly different receiver) using the suitable re-encryption key.*

5. $\mathbf{ReEnc}(PP, pk_i, pk_j, rk_{i \rightarrow j}, CT_i) \rightarrow CT_j$. *On input public parameter PP, two public keys pk_i, pk_j, a re-encryption key from i to j, and a second-level ciphertext for i, the re-encryption algorithm outputs a first-level ciphertext for j or the symbol $\perp$.*

6. $\mathbf{Dec}_1(PP, sk, CT) \rightarrow m$. *On input public parameter PP, a secret key sk, and a first-level ciphertext CT, the decryption algorithm outputs a message $m \in \mathcal{M}$ or the symbol $\perp$.*

7. $\mathbf{Dec}_2(PP, sk, CT) \rightarrow m$. *On input public parameter PP, a secret key sk, and a second-level ciphertext CT, the decryption algorithm outputs a message $m \in \mathcal{M}$ or the symbol $\perp$.*

To lighten notations, from now, we will omit the public parameter PP as the input of the algorithms. We also omit two public keys pk_i, pk_j as the input of the algorithm $\mathbf{ReEnc}$ when the context is obvious.

For all $m \in \mathcal{M}$ and all pair $(pk_i, sk_i), (pk_j, sk_j)$ these algorithms should satisfy the following conditions of correctness:

$$\mathbf{Dec}_2(sk_i, \mathbf{Enc}(pk_i, m)) = m;$$
$$\mathbf{Dec}_1(sk_j, \mathbf{ReEnc}(\mathbf{RKGen}(sk_i, pk_j), \mathbf{Enc}(pk_i, m))) = m.$$

3.2.2 Security Notions

We present a CCA security (BS-CCA, for short) definition for bidirectional single-hop PRE, which extends that of [20] by proposing a CCA security for the first-level ciphertext. In particular, we allow the adversary to make both first and second-level decryption queries. In the challenge phase, the adversary will challenge with either second-level ciphertex or first-level (re-encrypted) ciphertext.

Definition 16 (Game Framework of BS-CCA Security)**.**

Setup. *The challenger $\mathcal{C}$ takes a security parameter λ and executes the setup algorithm to get the system parameter PP. $\mathcal{C}$ initializes two empty lists L_{un} and L_{corr}, then maintains it to record all uncorrupted users and corrupted users, respectively, in the game.*

Phase 1. *$\mathcal{A}$ can adaptively query to the following oracles $\mathcal{O}_{unKGen}$, $\mathcal{O}_{corrKGen}$, $\mathcal{O}_{RKGen}$, $\mathcal{O}_{ReEnc}$, $\mathcal{O}_{Dec_1}$, and $\mathcal{O}_{Dec_2}$:*

- *$\mathcal{O}_{unKGen}$ takes i and returns $\perp$ if $i \in L_{un} \cup L_{corr}$; otherwise returns $(pk_i, sk_i) \leftarrow \mathbf{KGen}(PP)$ and adds i in L_{un}.*
- *$\mathcal{O}_{corrKGen}$ takes i and returns $\perp$ if $i \in L_{un} \cup L_{corr}$, otherwise returns $(pk_i, sk_i) \leftarrow \mathbf{KGen}(PP)$ and adds i in L_{corr}.*

– *$\mathcal{O}_{RKGen}$ takes i, j and returns $rk_{ij} \leftarrow \mathbf{RKGen}(sk_i, pk_j)$ if $i \neq j$; otherwise returns the error symbol $\perp$.*

– *$\mathcal{O}_{ReEnc}$ takes i, j and a ciphertext CT_i. If $i \neq j$, then it computes $rk_{ij} \leftarrow \mathbf{RKGen}(sk_i, pk_j)$, and returns $CT_j \leftarrow \mathbf{ReEnc}(rk_{i\rightarrow j}, CT_i)$; otherwise returns the error symbol $\perp$.*

– *$\mathcal{O}_{Dec_1}$ takes a public key pk and a ciphertext CT, then returns $m \leftarrow \mathbf{Dec}_1(sk, CT)$.*

– *$\mathcal{O}_{Dec_2}$ takes a public key pk and a ciphertext CT, then returns $m \leftarrow \mathbf{Dec}_2(sk, CT)$.*

Challenge. *When $\mathcal{A}$ decides that Phase 1 is over, it also decides which type of ciphertext for the challenge is first-level (re-encrypted) or second-level.*

– In the case that the challenge ciphertext is second-level, $\mathcal{A}$ outputs a target public key pk_{i^*} (we require $i^* \in L_{un}$ for $\mathcal{A}$ to win) and two messages $m_0, m_1 \in \mathcal{M}$. Challenger $\mathcal{C}$ flips a random coin $\sigma \in \{0, 1\}$, and sends $\mathcal{A}$ a challenge ciphertext $CT^* \leftarrow \mathbf{Enc}(pk_{i^*}, m_\sigma)$.

– In the case that the challenge ciphertext is first-level, $\mathcal{A}$ outputs a (corrupted or not) public key $pk_{i'}$, a target public key pk_{i^*}, and two second-level ciphertexts CT_0, CT_1 which can be re-encrypted from $pk_{i'}$ to pk_{i^*}. Challenger $\mathcal{C}$ flips a random coin $\sigma \in \{0, 1\}$, computes $rk_{i'\rightarrow i^*} \leftarrow \mathbf{RKGen}(sk_{i'}, pk_{i^*})$, and sends $\mathcal{A}$ a challenge ciphertext $CT^* \leftarrow \mathbf{ReEnc}(rk_{i'\rightarrow i^*}, CT_\sigma)$.

Phase 2. *$\mathcal{A}$ issues queries as in Phase 1.*

Guess. *Finally, $\mathcal{A}$ outputs a guess $\sigma' \in \{0, 1\}$.*

The precise conditions of the attacks to second and first-level ciphertexts are described separately as follows.

The Security of Second-Level Ciphertexts. Intuitively speaking, in this model the adversary $\mathcal{A}$ challengs with an untransformed ciphertext encrypted by $\mathbf{Enc}$ for a target user i^*. In a PRE scheme, however, $\mathcal{A}$ can ask for the re-encryption of many ciphertexts or even a set of re-encryption keys. These queries are allowed as long as they would not allow $\mathcal{A}$ to decrypt trivially. For examples, $\mathcal{A}$ should not get the re-encryption key from user i^* to user j if the secret key of user j has been compromised; however, $\mathcal{A}$ can certainly get a re-encryption of the challenge ciphertext from user i^* to user j as long as j is an honest user and the decryption oracle of user j has not been queried with the resulting transformed ciphertext. This explains the intuition behind the notion of derivative and the associated restrictions.

Definition 17 (2nd-BS-CCA Security). *For the 2nd-BS-CCA security, the adversary $\mathcal{A}$ plays the CCA game with the challenger $\mathcal{C}$ as in Definition 22, where the challenge ciphertext is formed by $CT^* \leftarrow \mathbf{Enc}(pk_{i^*}, m_\sigma)$, and $\mathcal{A}$ has the following additional constraints:*

1. *$\mathcal{O}_{RKGen}(i, j)$ is only allowed if if $i \neq j \in L_{un}$ or $i \neq j \in L_{corr}$.*
2. *If $\mathcal{A}$ issues $\mathcal{O}_{ReEnc}(i, j, CT_i)$ where $j \in L_{corr}$, (pk_i, CT_i) cannot be (pk_{i^*}, CT^*).*
3. *$\mathcal{O}_{Dec_1}$ is only allowed if (pk, CT) is not a* derivative *of (pk_{i^*}, CT^*) (to be defined later).*

We define $\mathcal{A}$'s advantage in attacking the PRE scheme at level 2 as $\mathrm{Adv}^{\text{2nd-CCA}}_{\text{PRE},\mathcal{A}}(\lambda) = \left|\Pr[\sigma' = \sigma] - 1/2\right|$, where the probability is taken over the random coins consumed by the challenger and the adversary. A unidirectional PRE scheme is defined to be 2nd-BS-CCA secure, if for any PPT adversary $\mathcal{A}$, the advantage $\mathrm{Adv}^{\text{2nd-CCA}}_{\text{PRE},\mathcal{A}}(\lambda)$ is negligible.

Definition 18 (Derivative for Chosen-Ciphertext Security [17]). *Derivatives of* (pk_{i^*}, CT^*) *in the CCA setting is defined as follows:*

1. *Reflexivity:* (pk_{i^*}, CT^*) *is a derivative of itself.*
2. *Derivative by re-encryption: If* $\mathcal{A}$ *has issued a re-encryption query* (i^*, j, CT^*) *and obtained the resulting re-encryption ciphertext* CT_j*, then* (pk_j, CT_j) *is a derivative of* (pk_{i^*}, CT^*).
3. *Derivative by re-encryption key: If* $\mathcal{A}$ *directly obtains the re-encryption key* rk_{ij} *by issuing a re-encryption key generation query* (i^*, j)*, or indirectly obtains the re-encryption key* rk_{i^*j} *from two directly obtained re-keys* rk_{i^*k}, rk_{jk} *in the case that the scheme meets the transitivity between re-encryption keys, and computes* $CT_j \leftarrow \mathbf{ReEnc}(rk_{i^*j}, CT^*)$*, then* (pk_j, CT_j) *is a derivative of* (pk_{i^*}, CT^*).

The Security of First-Level Ciphertexts. The above definition provides adversaries with a second-level ciphertext in the challenge phase. A complementary definition of security captures their inability to distinguish first-level ciphertexts as well. The definition is as follows.

Definition 19 (1st-BS-CCA Security). *For the 1st-BS-CCA security, the adversary* $\mathcal{A}$ *plays the CCA game with the challenger* $\mathcal{C}$ *as in Definition 22, where the challenge ciphertext is formed by* $CT^* = \mathbf{ReEnc}(rk_{i'i^*}, CT_\sigma)$*, and* $\mathcal{A}$ *has the following additional constraints:*

1. $\mathcal{O}_{RKGen}(i, j)$ *is only allowed if* $i \neq j \in L_{un}$ *or* $i \neq j \in L_{corr}$.
2. $\mathcal{O}_{Dec_1}(pk_{i^*}, CT^*)$ *is not allowed.*

We define $\mathcal{A}$*'s advantage in attacking the PRE scheme at level 1 as* $\mathrm{Adv}^{\text{1st-CCA}}_{\mathrm{PRE},\mathcal{A}}(\lambda) = \left|\Pr[\sigma' = \sigma] - 1/2\right|$*, where the probability is taken over the random coins consumed by the challenger and the adversary. A unidirectional PRE scheme is defined to be 1st-BS-CCA secure, if for any PPT adversary* $\mathcal{A}$*, the advantage* $\mathrm{Adv}^{\text{1st-CCA}}_{\mathrm{PRE},\mathcal{A}}(\lambda)$ *is negligible.*

Definition 20 (BS-CCA Security). *We say a PRE scheme is BS-CCA secure if the scheme is 1st-BS-CCA and 2nd-BS-CCA secure.*

3.3 Unidirectional Single-Hop Proxy Re-Encryption

In this section, we first review the concept of unidirectional single-hop PRE. Then, we present a new CCA security definition for PRE, which naturally extends that of [14] by giving an adversary all the possible resources as the observation in [25].

3.3.1 Model

In this section, we review the concept of unidirectional single-hop PRE.

Definition 21 (Unidirectional Single-Hop PRE [42])**.** *A unidirectional single-hop PRE scheme is a tuple of algorithms* $\Pi = (\mathbf{Setup}, \mathbf{KGen}, \mathbf{ReKey}, \mathbf{Enc_1}, \mathbf{Enc_2}, \mathbf{ReEnc}, \mathbf{Dec_1}, \mathbf{Dec_2})$ *for message space* $\mathcal{M}$*:*

1. $\mathbf{Setup}(1^\lambda) \rightarrow PP$*. On input security parameter* 1^λ*, the setup algorithm outputs a public parameter* PP.

2. $\mathbf{KGen}(PP) \rightarrow (pk, sk)$. *On input public parameter PP, the key generation algorithm outputs a public key pk and a secret key sk.*

3. $\mathbf{ReKey}(PP, sk_i, pk_j) \rightarrow rk_{i\rightarrow j}$. *On input public parameter PP, a secret key sk_i, and a public key pk_j, this algorithm outputs a unidirectional re-encryption key $rk_{i\rightarrow j}$.*

4. $\mathbf{Enc}_1(PP, pk, m) \rightarrow CT$. *On input public parameter PP, a public key pk, and a message $m \in \mathcal{M}$, the encryption algorithm outputs a first-level ciphertext CT that cannot be re-encrypted for another party.*

5. $\mathbf{Enc}_2(PP, pk, m) \rightarrow CT$. *On input public parameter PP, a public key pk, and a message $m \in \mathcal{M}$, the encryption algorithm outputs a second-level ciphertext CT that can be re-encrypted into a first level one (intended for a possibly different receiver) using the suitable re-encryption key.*

6. $\mathbf{ReEnc}(PP, pk_i, pk_j, rk_{i\rightarrow j}, CT_i) \rightarrow CT_j$. *On input public parameter PP, two public keys pk_i, pk_j, a re-encryption key from i to j, and a second-level ciphertext for i, the re-encryption algorithm outputs a first-level ciphertext for j or the symbol $\perp$.*

7. $\mathbf{Dec}_1(PP, sk, CT) \rightarrow m$. *On input public parameter PP, a secret key sk, and a first-level ciphertext CT, the decryption algorithm outputs a message $m \in \mathcal{M}$ or the symbol $\perp$.*

8. $\mathbf{Dec}_2(PP, sk, CT) \rightarrow m$. *On input public parameter PP, a secret key sk, and a second-level ciphertext CT, the decryption algorithm outputs a message $m \in \mathcal{M}$ or the symbol $\perp$.*

To lighten notations, from now, we will omit the public parameters PP as the input of the algorithms. We also omit two public keys pk_i, pk_j as the input of the algorithm $\mathbf{ReEnc}$ when the context is obvious.

For all $m \in \mathcal{M}$ and all pair $(pk_i, sk_i), (pk_j, sk_j)$ these algorithms should satisfy the following conditions of correctness:

$$\begin{aligned} \mathbf{Dec}_1(sk_i, \mathbf{Enc}_1(pk_i, m)) &= m; \\ \mathbf{Dec}_2(sk_i, \mathbf{Enc}_2(pk_i, m)) &= m; \\ \mathbf{Dec}_1(sk_j, \mathbf{ReEnc}(\mathbf{ReKey}(sk_i, pk_j), \mathbf{Enc}_2(pk_i, m))) &= m. \end{aligned}$$

3.3.2 Security Notions

In this section, we present a CCA security (US-CCA, for short) definition for unidirectional single-hop PRE, which naturally extends that of [14, 25].

Definition 22 (Game Framework of US-CCA Security)**.** *The game framework of chosen-ciphertext security is as follows.*

Setup. *The challenger $\mathcal{C}$ takes a security parameter 1^λ and executes the setup algorithm to get the public parameters PP. $\mathcal{C}$ executes the key generation algorithm n_{un} times resulting a list L_{un} of public/private keys $\mathcal{PK}_{un}, \mathcal{SK}_{un}$ for uncorrupted users. $\mathcal{C}$ also executes the key generation algorithm n_{corr} times resulting a list L_{corr} of public/private keys $\mathcal{PK}_{corr}, \mathcal{SK}_{corr}$*

for corrupted users. Next, $\mathcal{C}$ picks a challenge key pair $(pk_{i^*}, sk_{i^*}) \leftarrow \mathbf{KGen}(PP)$. $\mathcal{A}$ gets PP, $\mathcal{PK} = (\mathcal{PK}_{un}, \mathcal{PK}_{corr})$, $\mathcal{SK}_{corr}$, and the challenge public key pk_{i^*}.

Phase 1. *$\mathcal{A}$ adaptively queries to oracles $\mathcal{O}_{rk}, \mathcal{O}_{re}$, $\mathcal{O}_{dec_1}$, and $\mathcal{O}_{dec_2}$:*

- *$\mathcal{O}_{rk}$ takes (pk_i, pk_j) and returns a re-encryption key $rk_{i\to j} \leftarrow \mathbf{ReKey}(sk_i, pk_j)$.*
- *$\mathcal{O}_{re}$ takes public keys pk_i, pk_j and a second-level ciphertext CT_i, and returns a re-encryption of CT_i from pk_i to pk_j.*
- *$\mathcal{O}_{dec_1}$ takes a public key pk and a first-level ciphertext CT, and returns the decryption of CT using the private key with respect to pk if $pk \in \mathcal{PK} \cup \{pk_{i^*}\}$; otherwise returns symbol $\perp$.*
- *$\mathcal{O}_{dec_2}$ takes a public key pk and a second-level ciphertext CT, and returns the decryption of CT using the private key with respect to pk if $pk \in \mathcal{PK} \cup \{pk_{i^*}\}$; otherwise returns $\perp$.*

Challenge. *When $\mathcal{A}$ decides that Phase 1 is over, it also decides which type of ciphertext is for the challenge: first level (encrypted by $\mathbf{Enc}_1$ or re-encrypted by $\mathbf{ReEnc}$) or second level (encrypted by $\mathbf{Enc}_2$).*

– In the cases that the challenge ciphertext is the one encrypted by $\mathbf{Enc}_1$ or $\mathbf{Enc}_2$, $\mathcal{A}$ outputs two messages $m_0, m_1 \in \mathcal{M}$. Challenger $\mathcal{C}$ flips a random coin $\sigma \in \{0,1\}$, and sends $\mathcal{A}$ a challenge ciphertext CT^* depending on pk_{i^*} and m_σ.

– In the case that the challenge ciphertext is the one re-encrypted by $\mathbf{ReEnc}$, $\mathcal{A}$ outputs a (corrupted or not) public key $pk_{i'}$, and two second-level ciphertexts CT_0, CT_1 which can be re-encrypted from $pk_{i'}$ to pk_{i^*}. Challenger $\mathcal{C}$ flips a random coin $\sigma \in \{0,1\}$, and sends $\mathcal{A}$ a challenge ciphertext $CT^* \leftarrow \mathbf{ReEnc}(rk_{i'\to i^*}, CT_\sigma)$.

Phase 2. *$\mathcal{A}$ issues queries as in Phase 1.*

Guess. *Finally, $\mathcal{A}$ outputs a guess $\sigma' \in \{0,1\}$.*

The precise conditions of the attacks to second and first-level ciphertexts are described separately as follows.

CCA Security of Second-Level Ciphertext. Intuitively speaking, in this model the adversary $\mathcal{A}$ challenges with an untransformed ciphertext encrypted by $\mathbf{Enc}_2$ for a target user i^*. In the game, $\mathcal{A}$ can ask for the re-encryption of many ciphertexts or even a set of re-encryption keys. These queries are allowed as long as they would not allow $\mathcal{A}$ to decrypt trivially. For examples, $\mathcal{A}$ should not get the re-encryption key from user i^* to user j if the secret key of user j has been compromised; however, $\mathcal{A}$ can certainly get a re-encryption of the challenge ciphertext from user i^* to user j as long as j is an honest user and the decryption oracle of user j has not been queried with the resulting transformed ciphertext. This explains the intuition behind the notion of derivative and the associated restrictions.

Definition 23 (2nd-US-CCA Security [17]). *For 2nd-US-CCA security, the adversary $\mathcal{A}$ plays the CCA game with the challenger $\mathcal{C}$ as in Definition 22, where the challenge ciphertext is formed by $CT^* \leftarrow \mathbf{Enc}_2(pk_{i^*}, m_\sigma)$, and $\mathcal{A}$ has the following additional constraints:*

1. $\mathcal{O}_{rk}(pk_{i^*}, pk_j)$ *is only allowed if* pk_j *is an uncorrupted key.*

2. $\mathcal{O}_{re}(pk_{i^*}, pk_j, CT^*)$ *is only allowed if* pk_j *is an uncorrupted key.*

3. $\mathcal{O}_{dec_1}$ *is only allowed if* (pk, CT) *is not a* derivative *of* (pk_{i^*}, CT^*) *(to be defined later).*

We define $\mathcal{A}$*'s advantage in attacking the scheme at second-level ciphertext as* $\mathrm{Adv}^{\text{second-CCA}}_{\mathrm{PRE},\mathcal{A}}(\lambda) = \left|\Pr[\sigma' = \sigma] - 1/2\right|$*. A unidirectional single-hop scheme is defined to be 2nd-US-CCA secure, if for any probabilistic polynomial-time algorithm adversary* $\mathcal{A}$*, the advantage* $\mathrm{Adv}^{\text{second-CCA}}_{\mathrm{PRE},\mathcal{A}}(\lambda)$ *is negligible.*

Definition 24 (Derivatives for CCA Security [17])**.** *Derivatives of* (pk_{i^*}, CT^*) *in the CCA setting is defined as follows:*

1. *Reflexivity:* (pk_{i^*}, CT^*) *is a derivative of itself.*

2. *Derivative by re-encryption query: If* $\mathcal{A}$ *has issued a re-encryption query* (pk_{i^*}, pk_j, CT^*) *and obtained the resulting re-encryption ciphertext* CT_j*, then* (pk_j, CT_j) *is a derivative of* (pk_{i^*}, CT^*)*.*

3. *Derivative by re-encryption key generation query: If* $\mathcal{A}$ *has issued a re-encryption key generation query* (pk_{i^*}, pk_j) *to obtain the re-encryption key* $rk_{i^*\to j}$*, and computes* $CT_j \leftarrow \mathbf{ReEnc}(rk_{i^*\to j}, CT^*)$*, then* (pk_j, CT_j) *is a derivative of* (pk_{i^*}, CT^*)*.*

CCA Security of First-Level Ciphertext. The above definition provides adversaries with a second-level ciphertext in the challenge phase. A complementary definition of security captures their inability to distinguish first-level ciphertexts as well.

For single-hop schemes, $\mathcal{A}$ is granted access to all the re-encryption keys in this definition. Since first-level ciphertexts cannot be re-encrypted, there is indeed no reason to keep attackers from obtaining all the honest-to-corrupt re-encryption keys. The re-encryption oracle becomes useless since all the re-encryption keys are available to $\mathcal{A}$.

Definition 25 (**1st-US-CCA Security**)**.** *For 1st-US-CCA security, the adversary* $\mathcal{A}$ *plays the CCA game with the challenger* $\mathcal{C}$ *as in Definition 22, where the challenge ciphertext is formed as follows.*

1. *If it is the one encrypted by* $\mathbf{Enc}_1$*:* $CT^* \leftarrow \mathbf{Enc}_1(pk_{i^*}, m_\sigma)$*.*

2. *If it is the one encrypted by* $\mathbf{ReEnc}$*:* $CT^* \leftarrow \mathbf{ReEnc}(rk_{i'\to i^*}, CT_\sigma)$*.*

$\mathcal{A}$ *has the only constraint that:* $\mathcal{O}_{dec_1}(pk_{i^*}, CT^*)$ *is not allowed.*

We define $\mathcal{A}$*'s advantage in attacking the PRE scheme at first-level ciphertext as* $\mathrm{Adv}^{\text{first-CCA}}_{\mathrm{PRE},\mathcal{A}}(\lambda) = \left|\Pr[\sigma' = \sigma] - 1/2\right|$*. A unidirectional single-hop scheme is defined to be 1st-US-CCA secure, if for any probabilistic polynomial-time adversary* $\mathcal{A}$*, the advantage* $\mathrm{Adv}^{\text{first-CCA}}_{\mathrm{PRE},\mathcal{A}}(\lambda)$ *is negligible.*

Definition 26 (**US-CCA Security**)**.** *We say a PRE scheme is US-CCA secure if the scheme is 1st-US-CCA and 2nd-US-CCA secure.*

Master Secret Security Master secret security is considered by Ateniese et al. [4] which captures the intuition that, even if a dishonest proxy colludes with the delegatee, they still cannot derive the delegator's private key in full. The attack mode is quite simple and can be covered by the first-level ciphertext security (see *e.g.* [42]). The reason behind is easy to see that there is no restriction in the re-encryption key generation queries, and that decryption is easy when the adversary can derive the delegator's private key in full.

3.3.3 Discussion in the Comparison with the Previous Security Models

In comparison with the security model of [14], ours is strengthened by giving an adversary all the possible resources as the observation in [25]. In particular, we allow the adversary to make both first and second level decryption queries.

Our security model is stronger than that of [25] by two reasons which are explained in the next subsections, respectively.

3.3.3.1 Discussion on the Security Notion of second-level ciphertext

In this section, we will show why our security notion of second-level ciphertext is stronger than that of [25], which we denote by *w-2nd-CCA* security.

First, it is not difficult to see that our security notion implies that of [25] (i.e., a scheme which is 2nd-US-CCA secure is also w-2nd-CCA secure). Indeed, the difference between these notions is only in the constraints on the first level decryption queries in which our notion is more loosely than that of [25]. In particular, the difference is the response of the challenger in the case that $\mathcal{A}$ has asked a re-encryption key query $(pk_{i^*}, pk_j \in \mathcal{PK})$ previously, then asks a first level decryption query of CT_j with the public key pk_j. In the notion of [25], the challenger returns the symbol "**test**" whenever $\mathbf{Dec}_1(sk_j, CT_j) \in \{m_0, m_1\}$. Whereas, in our notion, the challenger returns the symbol $\perp$ only if CT_j is computed by re-encrypting the challenge ciphertext using the re-encryption key $rk_{i^* \to j}$; otherwise it returns $\mathbf{Dec}_1(sk_j, CT_j)$.

Second, we show that our notion is strictly stronger than that of [25] by giving a scheme which is w-2nd-CCA secure, but is not 2nd-US-CCA secure. In particular, using a secure PRE scheme $\mathbf{\Pi}$ in the sense of the definition of [25], and a CCA-secure public key encryption scheme $\mathbf{PKE} = (\mathbf{P.KGen}, \mathbf{P.Enc}, \mathbf{P.Dec})$ as two building blocks, we construct a scheme $\mathbf{\Pi'}$ as follows.

The Construction of the Scheme $\mathbf{\Pi'}$. Algorithms for parameters generation, re-encryption key generation, second level encryption, and second level decryption are identical to the scheme $\mathbf{\Pi}$. The other algorithms are as follows.

- The key generation algorithm $\mathbf{\Pi'.KGen}$ runs the key generation algorithms $\mathbf{\Pi.KGen}$ and $\mathbf{P.KGen}$, obtaining two pair of keys (pk, sk) and (ek, dk), respectively. Then, it outputs a pair of key (Pk, Sk), where $Pk = (pk, ek)$ and $Sk = (sk, dk)$.

- The re-encryption algorithm $\mathbf{\Pi'.ReEnc}$ with input a second-level ciphertext CT and a re-encryption key first runs the re-encryption algorithm $\mathbf{\Pi.ReEnc}$ with the same input CT, generating a ciphertext rCT. Then, it computes and outputs $re\text{-}CT \leftarrow$

$\mathbf{P}.\mathbf{Enc}(ek, rCT||0)$ (i.e., an encryption of $rCT||0$ under the encryption key ek) as re-encrypted ciphertext.

- The first level decryption algorithm $\mathbf{\Pi}'.\mathbf{Dec}_1$ for $\mathbf{\Pi}'$ with input $re\text{-}CT$ and $Sk = (sk, dk)$ first computes $rCT||a = \mathbf{P}.\mathbf{Dec}(dk, re\text{-}CT)$. Then it runs the first level decryption algorithm $\mathbf{\Pi}.\mathbf{Dec}_1$ with input rCT and sk (i.e., ignore the last bit a), and outputs the obtained output.

Security Analysis. First, we show that the scheme $\mathbf{\Pi}'$ is w-2nd-CCA secure. Then, we show that $\mathbf{\Pi}'$ is not secure in the sense of our notion by giving an adversary which can break its 2nd-US-CCA security.

In more details, the w-2nd-CCA security of $\mathbf{\Pi}'$ is guaranteed by the following theorem.

Theorem 1. *The scheme $\mathbf{\Pi}'$ is w-2nd-CCA secure, assuming that the underlying scheme $\mathbf{\Pi}$ is w-2nd-CCA secure and the PKE scheme $\mathbf{PKE}$ is CCA-secure.*

Proof. Suppose there exists an adversary $\mathcal{A}$ who can break the w-2nd-CCA security of $\mathbf{\Pi}'$ then we show how to construct another algorithm $\mathcal{B}$ that can break the w-2nd-CCA security of $\mathbf{\Pi}$. $\mathcal{B}$ works with interacting with the adversary $\mathcal{A}$ as follows.

Setup: $\mathcal{B}$ initializes an empty lists $\mathbf{L}_{re}$, then maintains it to record all re-encryption ciphertext of the challenge ciphertext in the game. $\mathcal{B}$ gets the public parameter PP and $\mathcal{PK}^* = \mathcal{PK} \cup \{pk_{i^*}\}$ from the challenger in the game of attacking the wCCA-security of $\mathbf{\Pi}$. Next, $\mathcal{B}$ generates the challenge key and the other keys (in the game with $\mathcal{A}$) as follows.

- *The challenge key:* $\mathcal{B}$ generates $(ek_{i^*}, dk_{i^*}) \leftarrow \mathbf{P}.\mathbf{KGen}$, and sets $Pk_{i^*} = (pk_{i^*}, ek_{i^*})$ (meaning that $Sk_{i^*} = (sk_{i^*}, dk_{i^*})$, where $\mathcal{B}$ knows only dk_{i^*}).
- *For the other keys:* $\mathcal{B}$ generates $(ek_i, dk_i) \leftarrow \mathbf{P}.\mathbf{KGen}$, and sets $Pk_i = (pk_i, ek_i)$ (meaning that $Sk_i = (sk_i, dk_i)$, where $\mathcal{B}$ knows only dk_i). $\mathcal{B}$ adds Pk_i to $\mathcal{PK}'$.

$\mathcal{B}$ provides $\mathcal{A}$ with public parameter PP and $\mathcal{PK}^{*\prime} = \mathcal{PK}' \cup \{Pk_{i^*}\}$.

Find stage. $\mathcal{B}$ answers the queries from $\mathcal{A}$ as follows.

Re-Encryption Key Generation Query. For a re-encryption key generation query $(Pk_i \in \mathcal{PK}^{*\prime}, Pk_j)$, where Pk_j is an arbitrary public key chosen by $\mathcal{A}$, $\mathcal{B}$ responds as follows. If $pk_i = pk_{i^*}$ and $pk_j \notin \mathcal{PK}^{*\prime}$, then $\mathcal{B}$ outputs $\perp$. Otherwise, $\mathcal{B}$ queries (pk_i, pk_j) to the re-encryption key generation oracle in the game of attacking the w-2nd-CCA security of $\mathbf{\Pi}$, obtaining a re-encryption key $rk_{i\to j}$. $\mathcal{B}$ returns the obtained re-encryption key to $\mathcal{A}$.

Re-Encryption Query. For a re-encryption query $(Pk_i \in \mathcal{PK}^{*\prime}, Pk_j, CT_i)$, $\mathcal{B}$ responds as follows. If $(Pk_i, CT_i) = (Pk_{i^*}, CT^*)$ and $pk_j \notin \mathcal{PK}^{*\prime}$, then $\mathcal{B}$ outputs $\perp$. Otherwise, $\mathcal{B}$ first queries (pk_i, pk_j, CT_i) to the re-encryption oracle in the game of attacking the wCCA-security of $\mathbf{\Pi}$, obtaining a re-encrypted ciphertext rCT. Then $\mathcal{B}$ computes and outputs $re\text{-}CT \leftarrow \mathbf{P}.\mathbf{Enc}(ek_j, rCT||0)$ as the answer for $\mathcal{A}$. If $(Pk_i, CT_i) = (Pk_{i^*}, CT^*)$ holds, then $\mathcal{B}$ adds (j, rCT) to the list $\mathbf{L}_{re}$.

First Level Decryption Query. For a first level decryption query $(Pk_i \in \mathcal{PK}^{*\prime}, CT_i)$, if $\mathcal{A}$ has asked a re-encryption query (Pk_{i^*}, Pk_i, CT^*) and obtained CT_i previously, then $\mathcal{B}$ returns $\perp$ to $\mathcal{A}$. Otherwise, $\mathcal{B}$ first computes $rCT||a = \mathbf{P}.\mathbf{Dec}(dk_i, CT_i)$. If $\mathcal{A}$ has not asked the re-encryption key query (Pk_{i^*}, Pk_i) previously, and $(i, rCT) \in \mathbf{L}_{re}$ then $\mathcal{B}$ returns $\perp$ and aborts. (We will show after the proof of Theorem 1 that the probability of this event is negligible.) Otherwise, it queries (pk_i, rCT) to the first level decryption oracle in the game of attacking the wCCA-security of $\mathbf{\Pi}$, and outputs the obtained answer.

We have the following claim (the proof of this claim will be shown after that of Theorem 1).

Claim 1. *The probability of the event which $\mathcal{B}$ aborts in the above case is negligible, assuming that $\mathbf{PKE}$ is CCA-secure.*

Second Level Decryption Query. For a second level decryption query $(Pk_i \in \mathcal{PK}^{*\prime}, CT_i)$, $\mathcal{B}$ queries (pk_i, CT_i) to the second decryption oracle in the game of attacking the w-2nd-CCA security of $\mathbf{\Pi}$, and outputs the obtained answer.

Challenge. When $\mathcal{A}$ decides that Phase 1 is over, it outputs two equal-length messages $m_0, m_1 \in \mathbb{G}_T$. At this stage, $\mathcal{B}$ gives these messages as the challenge ones to the challenger. After receiving the output from the challenger, $\mathcal{B}$ returns it to $\mathcal{A}$ as the challenge ciphertext.

Guess stage. The adversary $\mathcal{A}$ continues to issue the rest queries. In this stage, $\mathcal{B}$ can respond these queries for $\mathcal{A}$ as in the find stage.

Output. Eventually, adversary $\mathcal{A}$ returns a guess $\sigma' \in \{0, 1\}$. $\mathcal{B}$ output it (i.e., σ') as the final output in the game of attacking the w-2nd-CCA security of $\mathbf{\Pi}$.

This completes the description of the simulation.

It is easy to see that the simulation is perfect, therefore we have that if $\mathcal{A}$ can break the w-2nd-CCA security of $\mathbf{\Pi}'$, then $\mathcal{B}$ can break that of $\mathbf{\Pi}$.

The theorem follows. □

In order to complete the proof of Theorem 1, we now present the proof of Claim 1.

Proof of Claim 1. Assume towards a contradiction that after the challenge phase is finished, there exist an adversary $\mathcal{A}'$ can output a first-level ciphertext CT_i which (i, rCT) is in the list $\mathbf{L}_{re}$, where $rCT||a = \mathbf{P}.\mathbf{Dec}(dk_i, CT_i)$. Then we show that we can use $\mathcal{A}'$ to construct algorithm $\mathcal{B}'$ which can break the CCA-security of the scheme $\mathbf{PKE}$.

$\mathcal{B}'$ works almost the same as the algorithm $\mathcal{B}$ in the proof of Theorem 1, except that in the setup phase instead of receiving key pairs (pk_i, sk_i) from the challenger, $\mathcal{B}'$ runs the key generation algorithm of $\mathbf{\Pi}$ by himself to generate them. $\mathcal{B}'$ then chooses $t \xleftarrow{\mathrm{R}} \{1, ..., n\}$ and sets the challenge public key in the CCA-security game of $\mathbf{PKE}$ to be the public key ek_t.

Now, since $\mathcal{B}'$ knows $\{sk_i\}_i$ and $\{dk_i\}_{i \neq t}$, $\mathcal{B}'$ can simulate re-encryption key generation, re-encryption, and second level decryption queries perfectly. Also, $\mathcal{B}'$ can simulate first decryption queries using decryption queries on ek_t, and secret keys $\{sk_i\}_i, \{dk_i\}_{i \neq t}$ that $\mathcal{B}'$ knows.

After the challenge phase is finished, when $\mathcal{A}'$ queries $\mathcal{O}_{re}(Pk_{i^*}, pk_t, CT^*)$, $\mathcal{B}'$ first computes $rCT \leftarrow \mathbf{\Pi}.\mathbf{ReEnc}$ and adds (t, rCT) to the list $\mathbf{L}_{re}$. $\mathcal{B}'$ then outputs two massages $m_0 =$

$rCT||0, m_1$, where m_1 is randomly chosen from the message space of $\mathbf{PKE}$, as the challenge plaintext in the CCA game with the scheme $\mathbf{PKE}$ and receives a challenger ciphertext C^*. $\mathcal{B}'$ returns C^* to $\mathcal{A}'$.

In the first time that $\mathcal{A}'$ queries $\mathcal{O}_{dec_1}(pk_i, CT_i)$ such that $i = t$ and (i, rCT) is in the list $\mathbf{L}_{re}$, where $rCT||a = \mathbf{P}.\mathbf{Dec}(dk_i, CT_i)$. $\mathcal{B}'$ outputs 0 to guess $\sigma = 0$ in the CCA game of $\mathbf{PKE}$, and terminates the game with $\mathcal{A}'$.

Since $\mathcal{B}'$ perfectly simulates the wCCA game for $\mathcal{A}'$ and the assumption that $\mathcal{A}'$ can output a first-level ciphertext CT_i as above (i.e., $i = t$ and (i, rCT) is in the list $\mathbf{L}_{re}$, where $rCT||a = \mathbf{P}.\mathbf{Dec}(dk_i, CT_i)$), $\mathcal{B}'$ succeeds in guessing σ with the probability of $1/n$. In another words, $\mathcal{B}'$ can break the CCA-security of the scheme $\mathbf{PKE}$. □

We have shown that the scheme $\mathbf{\Pi}'$ is w-2nd-CCA secure. We now show that $\mathbf{\Pi}'$ is not secure in the sense of our notion by giving an adversary $\mathcal{A}$ which can break its 2nd-US-CCA security. In particular, $\mathcal{A}$ does as follows.

1. After receiving the challenge ciphertext CT_{i^*}, $\mathcal{A}$ queries $\mathcal{O}_{rk}$ to obtain a valid re-encryption key $rk_{i^* \to j}$ from challenger to uncorrupted user j.

2. $\mathcal{A}$ first computes $rCT \leftarrow \mathbf{\Pi}.\mathbf{ReEnc}(rk_{i^* \to j}, CT_{i^*})$. Then, it computes and outputs $re\text{-}CT_j \leftarrow \mathbf{P}.\mathbf{Enc}(ek, rCT||1)$ (i.e., an encryption of $rCT||1$ instead of $rCT||0$ as in the algorithm $\mathbf{\Pi}'.\mathbf{ReEnc}$) as re-encrypted ciphertext.

3. $\mathcal{A}$ issues a decryption oracle query under Pk_j to decrypt the re-encrypted ciphertext $re\text{-}CT_j$, and the result is the message encrypted in CT_{i^*}. Note that, since $re\text{-}CT_j$ is an encryption of $rCT||1$ (under the encryption key ek), $(Pk_j, re\text{-}CT_j)$ is not a derivative of (Pk_{i^*}, CT_{i^*}) and this decryption query is not restricted in our security model.

□

3.3.3.2 Discussion on the Security Notion of first-level ciphertext

In this section, we will show why our security notion of first-level ciphertext is stronger than that of [25], which we denote by *w-1st-CCA* security.

First, it is not difficult to see that our security notion implies that of [25] (i.e., a scheme which is 1st-US-CCA secure is also w-1st-CCA secure). Indeed, the difference between these notions is the action of the adversary and the challenger in the challenge phase. In our model, the adversary give the challenger a public key $pk_{i'}$, and two second-level ciphertexts CT_0, CT_1 which can be re-encrypted from $pk_{i'}$ to pk_{i^*}. The challenger flips a random coin $\sigma \in \{0, 1\}$, and sends $\mathcal{A}$ a challenge ciphertext $CT^* \leftarrow \mathbf{ReEnc}(rk_{i' \to i^*}, CT_\sigma)$. Whereas, in the model of [25], the adversary give the challenger a public key $pk_{i'}$, and two messages m_0, m_1. The challenger flips a random coin $\sigma \in \{0, 1\}$, and sends $\mathcal{A}$ a challenge ciphertext $CT^* \leftarrow \mathbf{ReEnc}(rk_{i' \to i^*}, \mathbf{Enc}(pk_{i'}, m_\sigma))$. Therefore, any adversary that can distinguish the first-level ciphertexts of two challenge messages, can also distinguish the re-encryptions of two challenge second-level ciphertexts.

Second, we show that our notion is strictly stronger than that of [25] by giving a scheme which is w-1st-CCA secure, but is not 1st-US-CCA secure.

Roughly speaking, different with the notion of [25], ours captures the CCA security of first-level ciphertext, especially that of re-encrypted ciphertext. Intuitively, this guarantees that the

re-encrypted ciphertext is still secure even the adversary compromised the delegator and also the proxy. In particular, the compromise of $rk_{i' \to i^*}$ corresponds to the fact that the proxy, which is designated by the delegator $pk_{i'}$ for the delegation to the delegatee pk_{i^*}, is compromised. Ideally, it seems that whether the delegator $pk_{i'}$ is compromised or not in this situation does not affect the security of the transformed ciphertext for pk_{i^*}. However, if the adversary $\mathcal{A}$ compromised the delegator $pk_{i'}$ and also the proxy, $\mathcal{A}$ can simply ask the proxy to surrender the original ciphertext $\mathbf{Enc}(pk_{i'}, m_\sigma)$ before any actual transformation, and uses $sk_{i'}$ to decrypt trivially. So if the re-encrypted ciphertext contains a part of the original ciphertext, then with the original ciphertext $\mathbf{Enc}(pk_{i'}, m_\sigma)$, $sk_{i'}$, and $rk_{i' \to i^*}$ $\mathcal{A}$ can easily determine the message which the challenge ciphertext is decrypted to. It is true that if the proxy was initially honest and erased the original ciphertexts after their transformation, the same attack does not apply; however, a ciphertext is by definition public in nature and the adversary may have captured the ciphertext already and decrypt it when $sk_{i'}$ is obtained. Therefore, to capture the sense of a stronger CCA security, we should consider the security model where the first-level ciphertexts by the two ways satisfy indistinguishability.

Next, following the above intuition, we give a concrete scheme $\mathbf{\Pi}$ which is w-1st-CCA secure, but is not 1st-US-CCA secure. The construction of the scheme $\mathbf{\Pi}$ is as follows.

Setup: $(p, g, \mathbb{G}, \mathbb{G}_T) \xleftarrow{\mathrm{R}} \mathrm{Setup}(1^\lambda)$, $\mathbf{PKE} = (\mathbf{PKGen}, \mathbf{PEnc}, \mathbf{PDec})$ is a PKE scheme of which the message space is $\mathbb{G} \times \mathbb{G}_T^2$. Return $PP = (p, g, \mathbb{G}, \mathbb{G}_T)$.

KGen: On input $PP = (p, g, \mathbb{G}, \mathbb{G}_T)$, choose $x \xleftarrow{\mathrm{R}} \mathbb{Z}_p$, $(ek, dk) \leftarrow \mathbf{PKGen}$ and output (pk, sk), where $pk = (g^x, ek), sk = (x, dk)$.

ReKey: On input a private key $sk_i = (sk_{i,1}, sk_{i,2})$ and a public key $pk_j = (pk_{j,1}, pk_{j,2})$, the algorithm outputs $rk_{i \to j} = pk_{j,1}^{1/sk_{i,1}} (= g^{x_j/x_i})$.

$\mathbf{Enc_1}$: Given $pk = (pk_1, pk_2)$ and a message $m \in \mathbb{G}_T$, choose $r, r' \xleftarrow{\mathrm{R}} \mathbb{Z}_p$ and compute: $C_1 = pk_1^{r'r}$, $C_2 = \mathbf{e}(g, g)^r \cdot m$, $C_3 = \mathbf{e}(pk_1^r, g)$, $CT_i' = C_1 || C_2 || C_3$; $\hat{C} \leftarrow \mathbf{PEnc}(pk_2, CT_i')$. Finally, output $CT = (C_1, \hat{C})$.

$\mathbf{Enc_2}$: Given $pk = (pk_1, pk_2)$ and a message $m \in \mathbb{G}$, choose $r \xleftarrow{\mathrm{R}} \mathbb{Z}_p$ and compute: $C_1 = pk_1^r$, $C_2 = \mathbf{e}(g, g)^r \cdot m$. Finally, output $CT = (C_1, C_2)$.

ReEnc: On input two public keys pk_i, pk_j, a re-encryption key $rk_{i \to j}$, and an original ciphertext CT_i this algorithm does as follows:
Compute $C_3 = \mathbf{e}(C_1, rk_{i \to j})$; $CT_i' = C_1 || C_2 || C_3$; $\hat{C} \leftarrow \mathbf{PEnc}(ek_j, CT_i')$.
Finally, output $CT_j = (C_1, \hat{C})$.

$\mathbf{Dec_2}$: Given $sk = (x, dk)$ and a ciphertext CT, output $m = \frac{C_2}{\mathbf{e}(C_1^{1/x}, g)}$.

$\mathbf{Dec_1}$: Given (pk_j, sk_j), and a ciphertext $CT_j = (C, \hat{C})$, do as follows.

Compute $CT_i' = \mathbf{PDec}(dk_j, \hat{C})$;
Parse $CT_i' = C_1 || C_2 || C_3$, if $C_1 \neq C$, then output $\perp$ and halt.

Else, output $m = \frac{C_2}{C_3^{1/x_j}}$.

Security Analysis. First, we show that the scheme $\mathbf{\Pi}$ is w-1st-CCA secure. Then, we show that $\mathbf{\Pi}$ is not secure in the sense of our notion by giving an adversary which can break its 1st-US-CCA security.

The w-1st-CCA security of $\mathbf{\Pi}$ is guaranteed by the following theorem.

Theorem 2. *The above scheme is w-1st-CCA secure, assuming that the PKE scheme* $\mathbf{PKE}$ *is CCA-secure.*

Proof. Suppose there exists an adversary $\mathcal{A}$ who can break the w-1st-CCA security of $\mathbf{\Pi}$, then we show how to construct another algorithm $\mathcal{B}$ that can break the CCA-security of $\mathbf{PKE}$. $\mathcal{B}$ works with interacting with adversary $\mathcal{A}$ as follows.

Setup: $\mathcal{B}$ gets the public parameter PP and the challenge public key ek_{i^*} from the challenger in the game of attacking the CCA-security of $\mathbf{PKE}$. Next, $\mathcal{B}$ generates the challenge key (in the game with $\mathcal{A}$) as follows. $\mathcal{B}$ chooses $x^* \xleftarrow{\mathrm{R}} \mathbb{Z}_p$, and sets $pk_{i^*} = (g^{x^*}, ek_{i^*})$ (meaning that $sk_{i^*} = (x^*, dk_{i^*})$, where $\mathcal{B}$ knows only x^*). $\mathcal{B}$ provides $\mathcal{A}$ with the public parameter PP and pk_{i^*}.

Find stage. $\mathcal{B}$ answers the queries from $\mathcal{A}$ as follows.

Re-Encryption Key Generation Query. For a re-encryption key generation query $pk = (pk_1, pk_2)$, $\mathcal{B}$ computes and returns pk_1^{1/x^*} to $\mathcal{A}$.

First Level Decryption Query. For a first level decryption query $CT = (C, \hat{C})$, $\mathcal{B}$ responds as follows. $\mathcal{B}$ first queries $\hat{C}$ to the decryption oracle in the game of attacking the CCA-security of $\mathbf{PKE}$, and receives the answer $C_1||C_2||C_3$. If $C_1 \neq C$, then $\mathcal{B}$ outputs $\perp$. Otherwise, $\mathcal{B}$ outputs $m = \frac{C_2}{C_3^{1/x^*}}$.

Second Level Decryption Query. For a second level decryption query CT, $\mathcal{B}$ queries CT to the decryption oracle in the game of attacking the CCA-security of $\mathbf{PKE}$, and outputs the obtained answer.

Challenge. At this stage, $\mathcal{A}$ outputs a public key $pk_{\mathcal{A}} = (pk_1, pk_2)$ and two equal-length messages $m_0, m_1 \in \mathbb{G}_T$. $\mathcal{B}$ first chooses $r \xleftarrow{\mathrm{R}} \mathbb{Z}_p$, and computes $C_1 = pk_1^r, C_2 = \mathbf{e}(g,g)^r \cdot m_0, C_2' = \mathbf{e}(g,g)^r \cdot m_1, C_3 = \mathbf{e}(g,g)^{x^* r}$. Then, $\mathcal{B}$ sets $CT_0 = C_1||C_2||C_3, CT_1 = C_1||C_2'||C_3$ and gives them as two challenge plaintexts to the challenger. After receiving the output $C^* \leftarrow \mathbf{PEnc}(ek^*, CT_\sigma)$ from the challenger, $\mathcal{B}$ returns $CT^* = (C_1, C^*)$ to $\mathcal{A}$ as the challenge ciphertext.

Guess stage. The adversary $\mathcal{A}$ continues to issue the rest queries. At this stage, $\mathcal{B}$ can respond these queries for $\mathcal{A}$ as in the find stage.

Output. Eventually, $\mathcal{A}$ returns a guess $\sigma' \in \{0,1\}$. $\mathcal{B}$ outputs it (i.e., σ') as the final output in the game of attacking the CCA-security of $\mathbf{PKE}$.

This completes the description of the simulation.

Next, we analyze the simulation.

In the challenge phase, since $C_1||C_2$ and $C_1||C_2'$ are the encryptions of m_0 and m_1 under the public key $pk_{\mathcal{A}}$ (with the same randomness r), respectively, $CT^* = (C_1, C^*)$ is really a re-encrypted ciphertext of the message m_σ. Therefore the simulation of this phase is perfect.

It is easy to see that the other simulations are also perfect, therefore we have that if $\mathcal{A}$ can break the first level security (in the sense of definition in [25]) of $\mathbf{\Pi}$, then $\mathcal{B}$ can break the CCA-security of $\mathbf{PKE}$.

The theorem follows. □

We have shown that the scheme $\mathbf{\Pi}$ is w-1st-CCA secure. We now show that $\mathbf{\Pi}$ is not secure in the sense of our notion by giving an adversary $\hat{\mathcal{A}}$ which can break its 1st-US-CCA security. In particular, $\hat{\mathcal{A}}$ does as follows.

1. $\hat{\mathcal{A}}$ gives the challenger a public key pk_i, and two messages CT_0, CT_1 which can be re-encrypted from $pk_{i'}$ to pk_{i^*}, where $CT_0 = (C_1, C_2), CT_1 = (C_1', C_2')$, and $C_1 \neq C_1'$.

2. After receiving the challenge ciphertext $CT_{i^*} = (C, \hat{C})$, $\hat{\mathcal{A}}$ outputs 0 if $C = C_1$, else returns 1 if $C = C_1'$. Note that, since $CT_{i^*} \leftarrow \mathbf{ReEnc}(rk_{i \to i^*}, CT_\sigma)$, we have $C = C_1$ if $\sigma = 0$ and $C = C_1'$ if $\sigma = 1$. Therefore, $\hat{\mathcal{A}}$ succeeds in breaking the 1st-US-CCA security of the scheme.

3.4 Conditional Proxy Re-Encryption

In this section, we first review the model of conditional proxy re-encryption. Then, we present security models for CPRE, which naturally extends that of [68] as the observation in Section 3.3. In particular, we allow the adversary to make both first and second level decryption queries in the second level security game. Following the observation in [25], we can also show that in general by giving the adversary the second level decryption oracle the condition CCA security will be strictly stronger.

3.4.1 Model

In this section, we review the model of conditional PRE.

Definition 27 (Conditional Proxy Re-Encryption [68])**.** *A CPRE scheme is a tuple of algorithms* $\Pi = (\mathbf{Setup}, \mathbf{KGen}, \mathbf{RKGen}, \mathbf{Enc}_1, \mathbf{Enc}_2, \mathbf{Dec}_1, \mathbf{Dec}_2)$ *for message space* $\mathcal{M}$*:*

- $\mathbf{Setup}(1^n) \to PP$*. On input security parameter* 1^n*, the setup algorithm outputs a public parameter* PP*.*

- $\mathbf{KGen}(PP) \to (pk, sk)$*. On input parameter* PP*, the key generation algorithm outputs a public key* pk *and a secret key* sk*.*

- $\mathbf{RKGen}(PP, sk_i, pk_j, \omega) \rightarrow rk_{i \rightarrow j|\omega}$. *On input parameter PP, a secret key sk_i, a public key pk_j, and a condition ω, this algorithm outputs a unidirectional re-encryption key $rk_{i \rightarrow j|\omega}$.*

- $\mathbf{Enc}_1(PP, pk, m) \rightarrow CT$. *On input parameter PP, a public key pk, and a message $m \in \mathcal{M}$, the encryption algorithm outputs a first-level ciphertext CT that cannot be re-encrypted for another party.*

- $\mathbf{Enc}_2(PP, pk, m, \omega) \rightarrow CT$. *On input parameter PP, a public key pk, a message $m \in \mathcal{M}$, and a condition ω, the encryption algorithm outputs a second-level ciphertext CT that can be re-encrypted into a first level one (intended for a possibly different receiver) using the suitable re-encryption key.*

- $\mathbf{ReEnc}(PP, pk_i, pk_j, rk_{i \rightarrow j|\omega}, CT_i) \rightarrow CT_j$. *On input parameter PP, two public keys pk_i, pk_j, a re-encryption key $rk_{i \rightarrow j|\omega}$, and an original ciphertext CT_i associated with ω under public key pk_i, the re-encryption algorithm outputs a first-level ciphertext for j or the symbol $\perp$.*

- $\mathbf{Dec}_1(PP, sk, CT) \rightarrow m$. *On input parameter PP, a secret key sk, and a first-level ciphertext CT, the decryption algorithm outputs a message $m \in \mathcal{M}$ or the symbol $\perp$.*

- $\mathbf{Dec}_2(PP, sk, CT) \rightarrow m$. *On input parameter PP, a secret key sk, and a second-level ciphertext CT, the decryption algorithm outputs a message $m \in \mathcal{M}$ or the symbol $\perp$.*

To lighten notations, from now, we will omit the public parameter PP as the input of the algorithms. We also omit two public keys pk_i, pk_j as the input of the algorithm $\mathbf{ReEnc}$ when the context is obvious.

For all $m \in \mathcal{M}$ and all pair $(pk_i, sk_i), (pk_j, sk_j)$ these algorithms should satisfy the following conditions of correctness:

$$\begin{aligned} \mathbf{Dec}_1(sk_i, \mathbf{Enc}_1(pk_i, m)) &= m; \\ \mathbf{Dec}_2(sk_i, \mathbf{Enc}_2(pk_i, m, \omega)) &= m; \\ \mathbf{Dec}_1(sk_j, \mathbf{ReEnc}(\mathbf{RKGen}(sk_i, pk_j, \omega), \mathbf{Enc}_2(pk_i, m, \omega))) &= m. \end{aligned}$$

3.4.2 Security Notions

In this section, we present a game framework of selective and adaptive condition chosen-ciphertext security, respectively.

The purpose of this thesis is to solve the open problem left by Weng et al. [68]. It is enough to present a scheme which is secure in the CCA-security model proposed in [68] without random oracles. However, recently, there are many researchs in which the CCA security model of PRE is strengthened, refer to [14, 25, 33] for some examples. Therefore, in this thesis, we consider how to construct conditional PRE schemes which are secure in a stronger model extended from that of [68]. In particular, we allow the adversary to make both first and second level decryption queries.

3.4.2.1 Selective Condition CCA Security

In this section, we first present a game framework of selective condition chosen-ciphertext security. We then define second/first level CCA security based on this game framework. Roughly speaking, these security models are extensions of those proposed by Weng et al. [68] as the observation in [33].

Definition 28 (Game Framework of Selective Condition CCA Security)**.** *The game framework of selective condition chosen-ciphertext security is as follows.*

Initiation. *The game begins with $\mathcal{A}$ first outputting a condition ω^* that it intends to attack.*

Setup. *The challenger $\mathcal{C}$ takes a security parameter 1^n and executes the setup algorithm to get the system parameter PP. $\mathcal{C}$ executes the key generation algorithm n_{un} times resulting a list L_{un} of public/private keys $\mathcal{PK}_{un}, \mathcal{SK}_{un}$ and executes the key generation algorithm n_{corr} times resulting a list L_{corr} of public/private keys $\mathcal{PK}_{corr}, \mathcal{SK}_{corr}$. Next, $\mathcal{C}$ picks a challenge user's key pair $(pk_{i^*}, sk_{i^*}) \leftarrow \mathbf{KGen}(PP)$. $\mathcal{A}$ gets PP, $\mathcal{PK} = (\mathcal{PK}_{un}, \mathcal{PK}_{corr})$, $\mathcal{SK}_{corr}$, and the challenge public key pk_{i^*}.*

Phase 1. *$\mathcal{A}$ adaptively queries to oracles $\mathcal{O}_{rk}, \mathcal{O}_{re}$, $\mathcal{O}_{dec_1}$, and $\mathcal{O}_{dec_2}$:*

- *$\mathcal{O}_{rk}$ takes (pk_i, pk_j, ω) and returns a re-encryption key $rk_{i \to j|\omega} \leftarrow \mathbf{RKGen}(sk_i, pk_j, \omega)$.*
- *$\mathcal{O}_{re}$ takes public keys pk_i, pk_j, a second-level ciphertext CT_i, and a condition ω, then returns a re-encryption of CT_i from pk_i to pk_j.*
- *$\mathcal{O}_{dec_1}$ takes a public key pk and a first-level ciphertext CT, then returns the decryption of CT using the private key with respect to pk if $pk \in \mathcal{PK} \cup \{pk_{i^*}\}$; otherwise returns symbol $\perp$.*
- *$\mathcal{O}_{dec_2}$ takes a public key pk, a condition ω, and a second-level ciphertext CT, then returns the decryption of CT using the private key with respect to pk if $pk \in \mathcal{PK} \cup \{pk_{i^*}\}$; otherwise returns $\perp$.*

Challenge. *When $\mathcal{A}$ decides that Phase 1 is over, it also decides which type of ciphertext is for the challenge: first level (encrypted by $\mathbf{Enc}_1$ or re-encrypted by $\mathbf{ReEnc}$) or second level (encrypted by $\mathbf{Enc}_2$).*

– In the cases that the challenge ciphertext is the one encrypted by $\mathbf{Enc}_1$ or $\mathbf{Enc}_2$, $\mathcal{A}$ outputs two messages $m_0, m_1 \in \mathcal{M}$. Challenger $\mathcal{C}$ flips a random coin $\sigma \in \{0,1\}$, and sends $\mathcal{A}$ a challenge ciphertext CT^* depending on $pk_{i^*}, \omega^*, m_\sigma$.

– In the case that the challenge ciphertext is the one re-encrypted by $\mathbf{ReEnc}$, $\mathcal{A}$ outputs a (corrupted or not) public key $pk_{i'}$, and two second-level ciphertexts CT_0, CT_1 which can be re-encrypted from $pk_{i'}$ to pk_{i^*}. Challenger $\mathcal{C}$ flips a random coin $\sigma \in \{0,1\}$, and sends $\mathcal{A}$ a challenge ciphertext $CT^* \leftarrow \mathbf{ReEnc}(rk_{i' \to i^*|\omega^*}, CT_\sigma)$.

Phase 2. *$\mathcal{A}$ issues queries as in Phase 1.*

Guess. *Finally, $\mathcal{A}$ outputs a guess $\sigma' \in \{0,1\}$.*

The precise conditions of the attacks to second and first-level ciphertexts are described separately as follows.

Selective Condition CCA Security of Second-Level Ciphertext. Intuitively speaking, in this model the adversary $\mathcal{A}$ challenges with an untransformed ciphertext CT^* encrypted under a target public key pk_{i^*} and a target condition ω^*. In a PRE scheme, however, $\mathcal{A}$ can ask for the re-encryption of many ciphertexts or even a set of re-encryption keys. These queries are allowed as long as they would not allow $\mathcal{A}$ to decrypt trivially. For examples, $\mathcal{A}$ should not get the re-encryption key from user i^* to user j under the condition ω^* if the secret key of user j has been compromised.

Definition 29 (2nd-sCond-CCA Security)**.** *For 2nd-sCond-CCA security, the adversary $\mathcal{A}$ plays the CCA game with the challenger $\mathcal{C}$ as in Definition 28, where the challenge ciphertext is formed by $CT^* \leftarrow \mathbf{Enc}_2(pk_{i^*}, m_\sigma, \omega^*)$, and $\mathcal{A}$ has the following additional constraints:*

1. $\mathcal{O}_{rk}(pk_{i^*}, pk_j, \omega^*)$ is only allowed if pk_j is an uncorrupt key.

2. If $\mathcal{A}$ issues $\mathcal{O}_{re}(pk_i, pk_j, CT_i, \omega)$ where pk_j is a corrupted key, (pk_i, CT_i, ω) cannot be a derivative *of $(pk_{i^*}, CT_{i^*}, \omega^*)$ (to be defined later).*

3. $\mathcal{O}_{dec_1}$ is only allowed if (pk, CT, ω) is not a derivative *of $(pk_{i^*}, CT_{i^*}, \omega^*)$.*

We define $\mathcal{A}$'s advantage in attacking the CPRE scheme at second-level ciphertext as $\mathrm{Adv}^{\text{2nd-sCond}}_{\text{CPRE},\mathcal{A}}(\lambda) = \left|\Pr[\sigma' = \sigma] - 1/2\right|$. A unidirectional CPRE scheme is defined to be 2nd-sCond-CCA secure, if for any PPT adversary $\mathcal{A}$, the advantage $\mathrm{Adv}^{\text{2nd-sCond}}_{\text{CPRE},\mathcal{A}}(\lambda)$ is negligible.

Definition 30 (Derivative for Chosen-Ciphertext Security)**.** *Derivatives of $(pk_{i^*}, CT_{i^*}, \omega^*)$ in the CCA setting is defined as follows:*

1. Reflexivity: $(pk_{i^*}, CT_{i^*}, \omega^*)$ is a derivative of itself.

2. Derivative by re-encryption: If $\mathcal{A}$ has issued a re-encryption query (pk, pk', CT, ω) and obtained the resulting re-encryption ciphertext CT', then (pk', CT', ω) is a derivative of (pk, CT, ω).

3. Derivative by re-encryption key: If $\mathcal{A}$ has issued a re-encryption key generation query (pk, pk', ω) to obtain the re-encryption key rk, and $CT' \leftarrow \mathbf{ReEnc}(rk, CT)$, then (pk', CT', ω) is a derivative of (pk, CT, ω).

The above derivative definition is taken from Definition 34, adapted for the game framework of condition chosen-ciphertext security.

Selective Condition CCA Security of First-Level Ciphertext. For single-hop schemes, $\mathcal{A}$ is granted access to all the re-encryption keys in this definition. Since first-level ciphertexts cannot be re-encrypted, there is indeed no reason to keep attackers from obtaining all the honest-to-corrupt re-encryption keys. The re-encryption oracle becomes useless since all the re-encryption keys are available to $\mathcal{A}$.

Definition 31 (1st-sCond-CCA Security)**.** *For 1st-sCond-CCA security, the adversary $\mathcal{A}$ plays the CCA game with the challenger $\mathcal{C}$ as in Definition 28, where the challenge ciphertext is formed as follows.*

1. In the case of original ciphertext, $CT^* \leftarrow \mathbf{Enc}_1(pk_{i^*}, m_\sigma)$.

2. In the case of re-encrypted ciphertext, $CT^* \leftarrow \mathbf{ReEnc}(rk_{i' \to i^* | \omega^*}, CT_\sigma)$.

$\mathcal{A}$ has the only constraint that: $\mathcal{O}_{dec_1}(pk_{i^*}, CT^*)$ is not allowed.

We define $\mathcal{A}$'s advantage in attacking the CPRE scheme at first-level ciphertext as $\mathrm{Adv}^{\text{1st-sCond}}_{\text{CPRE},\mathcal{A}}(\lambda) = \left|\Pr[\sigma' = \sigma] - 1/2\right|$. *A unidirectional CPRE scheme is defined to be 1st-sCond-CCA secure, if for any PPT adversary $\mathcal{A}$, the advantage* $\mathrm{Adv}^{\text{1st-sCond}}_{\text{CPRE},\mathcal{A}}(\lambda)$ *is negligible.*

Remark 1. The above definition (for first-level ciphertext security) is a strict extension of that of [68], since it additionally includes the CCA security for the re-encrypted ciphertext. Intuitively, this guarantees that the re-encrypted ciphertext is still secure even the adversary compromised the delegator and also the proxy.

3.4.2.2 Adaptive Condition CCA Security

In this section, we first present a game framework of adaptive condition chosen-ciphertext security. We then define second/first level CCA security based on this game framework. Roughly speaking, these security models are extensions of those proposed by Weng et al. [68] as the observation in [33].

Intuitively, the definitions of the adaptive condition CCA security for second/first-level ciphertext (2nd-aCond-CCA and 1st-aCond-CCA, respectively) are almost the same as those of 2nd-sCond-CCA and 1st-sCond-CCA, respectively, except that the adversary $\mathcal{A}$ outputs its chosen challenge condition ω^* in the challenge phase instead of the initialization phase. The details are as follows.

Definition 32 (Game Framework of Adaptive Condition CCA Security)**.** *The game framework of adaptive condition chosen-ciphertext security is as follows.*

Setup. *The challenger $\mathcal{C}$ takes a security parameter 1^n and executes the setup algorithm to get the system parameter PP. $\mathcal{C}$ executes the key generation algorithm n_{un} times resulting a list L_{un} of public/private keys $\mathcal{PK}_{un}, \mathcal{SK}_{un}$ and executes the key generation algorithm n_{corr} times resulting a list L_{corr} of public/private keys $\mathcal{PK}_{corr}, \mathcal{SK}_{corr}$. Next, $\mathcal{C}$ picks a challenge user's key pair $(pk_{i^*}, sk_{i^*}) \leftarrow \mathbf{KGen}(PP)$. $\mathcal{A}$ gets PP, $\mathcal{PK} = (\mathcal{PK}_{un}, \mathcal{PK}_{corr})$, $\mathcal{SK}_{corr}$, and the challenge public key pk_{i^*}.*

Phase 1. *$\mathcal{A}$ adaptively queries to oracles $\mathcal{O}_{rk}, \mathcal{O}_{re}$, $\mathcal{O}_{dec_1}$, and $\mathcal{O}_{dec_2}$:*

- *$\mathcal{O}_{rk}$ takes (pk_i, pk_j, ω) and returns a re-encryption key $rk_{i\to j|\omega} \leftarrow \mathbf{RKGen}(sk_i, pk_j, \omega)$.*
- *$\mathcal{O}_{re}$ takes public keys pk_i, pk_j, a second-level ciphertext CT_i, and a condition ω, then returns a re-encryption of CT_i from pk_i to pk_j.*
- *$\mathcal{O}_{dec_1}$ takes a public key pk and a first-level ciphertext CT, then returns the decryption of CT using the private key with respect to pk if $pk \in \mathcal{PK} \cup \{pk_{i^*}\}$; otherwise returns symbol $\perp$.*
- *$\mathcal{O}_{dec_2}$ takes a public key pk, a condition ω, and a second-level ciphertext CT, then returns the decryption of CT using the private key with respect to pk if $pk \in \mathcal{PK} \cup \{pk_{i^*}\}$; otherwise returns $\perp$.*

Challenge. *When $\mathcal{A}$ decides that Phase 1 is over, it also decides which type of ciphertext is for the challenge: first level (encrypted by $\mathbf{Enc}_1$ or re-encrypted by $\mathbf{ReEnc}$) or second level (encrypted by $\mathbf{Enc}_2$).*

– In the cases that the challenge ciphertext is the one encrypted by $\mathbf{Enc}_1$ or $\mathbf{Enc}_2$, $\mathcal{A}$ outputs two messages $m_0, m_1 \in \mathcal{M}$, and a target condition ω^*. Challenger $\mathcal{C}$ flips a random coin $\sigma \in \{0,1\}$, and sends $\mathcal{A}$ a challenge ciphertext CT^* depending on pk_{i^*}, ω^*, m_σ.

– In the case that the challenge ciphertext is the one re-encrypted by $\mathbf{ReEnc}$, $\mathcal{A}$ outputs a (corrupted or not) public key $pk_{i'}$, a target condition ω^*, and two second-level ciphertexts CT_0, CT_1 which can be re-encrypted from $pk_{i'}$ to pk_{i^*}. Challenger $\mathcal{C}$ flips a random coin $\sigma \in \{0,1\}$, and sends $\mathcal{A}$ a challenge ciphertext $CT^* \leftarrow \mathbf{ReEnc}(rk_{i' \to i^* | \omega^*}, CT_\sigma)$.

Phase 2. *$\mathcal{A}$ issues queries as in Phase 1.*

Guess. *Finally, $\mathcal{A}$ outputs a guess $\sigma' \in \{0,1\}$.*

The precise conditions of the attacks to second and first-level ciphertexts are described separately as follows.

Adaptive Condition CCA Security of Second-Level Ciphertext. Intuitively speaking, in this model the adversary $\mathcal{A}$ challenges with an untransformed ciphertext CT^* encrypted under a target public key pk_{i^*} and a target condition ω^*. In a PRE scheme, however, $\mathcal{A}$ can ask for the re-encryption of many ciphertexts or even a set of re-encryption keys. These queries are allowed as long as they would not allow $\mathcal{A}$ to decrypt trivially. For examples, $\mathcal{A}$ should not get the re-encryption key from user i^* to user j under the condition ω^* if the secret key of user j has been compromised.

Definition 33 (2nd-aCond-CCA Security)**.** *For 2nd-aCond-CCA security, the adversary $\mathcal{A}$ plays the CCA game with the challenger $\mathcal{C}$ as in Definition 32, where the challenge ciphertext is formed by $CT^* \leftarrow \mathbf{Enc}_2(pk_{i^*}, m_\sigma, \omega^*)$, and $\mathcal{A}$ has the following additional constraints:*

1. $\mathcal{O}_{rk}(pk_{i^*}, pk_j, \omega^*)$ is only allowed if pk_j is an uncorrupt key.

2. If $\mathcal{A}$ issues $\mathcal{O}_{re}(pk_i, pk_j, CT_i, \omega)$ where pk_j is a corrupted key, (pk_i, CT_i, ω) cannot be a derivative *of $(pk_{i^*}, CT_{i^*}, \omega^*)$ (to be defined later).*

3. $\mathcal{O}_{dec_1}$ is only allowed if (pk, CT, ω) is not a derivative *of $(pk_{i^*}, CT_{i^*}, \omega^*)$.*

We define $\mathcal{A}$'s advantage in attacking the CPRE scheme at second-level ciphertext as $\mathrm{Adv}^{\text{2nd-aCond}}_{\text{CPRE},\mathcal{A}}(\lambda) = \left|\Pr[\sigma' = \sigma] - 1/2\right|$. A unidirectional CPRE scheme is defined to be 2nd-aCond-CCA secure, if for any PPT adversary $\mathcal{A}$, the advantage $\mathrm{Adv}^{\text{2nd-aCond}}_{\text{CPRE},\mathcal{A}}(\lambda)$ is negligible.

Definition 34 (Derivative for Chosen-Ciphertext Security)**.** *Derivatives of $(pk_{i^*}, CT_{i^*}, \omega^*)$ in the CCA setting is defined as follows:*

1. Reflexivity: $(pk_{i^*}, CT_{i^*}, \omega^*)$ is a derivative of itself.

2. Derivative by re-encryption: If $\mathcal{A}$ has issued a re-encryption query (pk, pk', CT, ω) and obtained the resulting re-encryption ciphertext CT', then (pk', CT', ω) is a derivative of (pk, CT, ω).

3. Derivative by re-encryption key: If $\mathcal{A}$ has issued a re-encryption key generation query (pk, pk', ω) to obtain the re-encryption key rk, and $CT' \leftarrow \mathbf{ReEnc}(rk, CT)$, then (pk', CT', ω) is a derivative of (pk, CT, ω).

The above derivative definition is taken from Definition 34, adapted for the game framework of condition chosen-ciphertext security.

Apdaptive Condition CCA Security of First-Level Ciphertext. For single-hop schemes, $\mathcal{A}$ is granted access to all the re-encryption keys in this definition. Since first-level ciphertexts cannot be re-encrypted, there is indeed no reason to keep attackers from obtaining all the honest-to-corrupt re-encryption keys. The re-encryption oracle becomes useless since all the re-encryption keys are available to $\mathcal{A}$.

Definition 35 (1st-aCond-CCA Security)**.** *For 1st-aCond-CCA security, the adversary $\mathcal{A}$ plays the CCA game with the challenger $\mathcal{C}$ as in Definition 32, where the challenge ciphertext is formed as follows.*

1. *In the case of original ciphertext, $CT^* \leftarrow \mathbf{Enc}_1(pk_{i^*}, m_\sigma)$.*
2. *In the case of re-encrypted ciphertext, $CT^* \leftarrow \mathbf{ReEnc}(rk_{i' \to i^*|\omega^*}, CT_\sigma)$.*

$\mathcal{A}$ has the only constraint that: $\mathcal{O}_{dec_1}(pk_{i^*}, CT^*)$ is not allowed.
We define $\mathcal{A}$'s advantage in attacking the CPRE scheme at first-level ciphertext as $\mathrm{Adv}^{\text{1st-aCond}}_{\text{CPRE},\mathcal{A}}(\lambda) = \left|\Pr[\sigma' = \sigma] - 1/2\right|$. A unidirectional CPRE scheme is defined to be 1st-aCond-CCA secure, if for any PPT adversary $\mathcal{A}$, the advantage $\mathrm{Adv}^{\text{1st-aCond}}_{\text{CPRE},\mathcal{A}}(\lambda)$ is negligible.

Remark 2. The above definition (for first-level ciphertext security) is a strict extension of that of [68], since it additionally includes the CCA security for the re-encrypted ciphertext. Intuitively, this guarantees that the re-encrypted ciphertext is still secure even the adversary compromised the delegator and also the proxy.

Chapter 4

Schemes Based on Factoring

Contents

4.1 Introduction 41
4.2 A Strongly Unforgeable Signature Scheme from Factoring 42
4.2.1 Construction 42
4.2.2 Security Analysis 43
4.3 The Proposed Schemes 44
4.3.1 The BM-CPA-Secure Scheme 44
4.3.2 The BS-CCA-Secure Scheme 45
4.3.3 Extension to a Unidirectional Single-Hop Scheme 47
4.4 Proofs 49
4.4.1 Proof of Theorem 4 49
4.4.2 Proof of Theorem 5 50
4.4.3 Proof of Theorem 6 60
4.5 Summary 71

4.1 Introduction

Proxy re-encryption introduced by Blaze, Bleumer, and Strauss [8], is a cryptosystem which allows the proxy to re-encrypt a cirphertext without accessing the underlying message. This process allows the proxy to manage distributed encryption by using a digital certificate or a re-encryption key from the sender.

In [8], Blaze et al. proposed the first bidirectional PRE scheme. Ateniese, Fu, Green, and Hohenberger [3, 4] presented unidirectional PRE schemes from bilinear maps in 2005. All of these schemes are only secure against chosen-plaintext attack (CPA). Canetti and Hohenberger [16] presented the first bidirectional multi-hop PRE scheme that is secure against replayable chosen-ciphertext attack (RCCA) in the standard model. Libert and Vergnaud [42] proposed a unidirectional single-hop PRE scheme, which is also RCCA-secure

in the standard model. The above schemes all rely on bilinear maps. In 2012, Hanaoka, Kawai, Kunihiro, Matsuda, Weng, Zhang, and Zhao [25] proposed the first generic construction of CCA-secure unidirectional single-hop PRE. However, since they only gave a concrete example which also uses bilinear maps, it is unknown whether there is an instantiation without bilinear maps.

All of the above PRE schemes inherently rely on decisional assumptions, e.g., the decisional Diffie-Hellman (DDH) or the decisional bilinear Diffie-Hellman (DBDH) assumption. In general, decisional assumptions are a much stronger class of assumptions than computational assumptions based on search problems, such as factoring, finding shortest vectors in lattices, or even the computational Diffie-Hellman (CDH) problem. Indeed, there are groups, such as certain elliptic curve groups with bilinear pairing map, where the DDH assumption does not hold, but the CDH problem appears to be hard. As such, schemes based on search problems are generally preferred to those based on decisional assumptions. However, such schemes seem to be very hard to obtain.

In 2010, Deng, Weng, Liu, and Chen [20] proposed the first PRE scheme based on a computational assumption, namely the CDH assumption (in the random oracle model). Since then, there are several works that show how to base CCA-secure PRE on the same assumption [17, 14].

In this chapter, we propose the first PRE schemes based on the hardness of the factoring problem. In particular, we present three PRE schemes. The first is a CPA-secure bidirectional multi-hop PRE scheme. The second is a CCA-secure bidirectional single-hop PRE scheme. The last is a CCA-secure unidirectional single-hop PRE scheme. The former is in the standard model, and the others in the random oracle model. In order to construct the second and the third schemes, we propose a new factoring-based strong signature scheme which is a variant of the Schnorr signature scheme.

4.2 A Strongly Unforgeable Signature Scheme from Factoring

In this section, we present a new factoring-based strongly unforgeable signature scheme.

4.2.1 Construction

The proposed signature scheme is a variant of the Schnorr signature [50]. Its strong unforgeability is based on the hardness of the factoring problem. We call the scheme factoring-based Schnorr (FB-Schnorr, for short) signature.

Let $\mathsf{param} = (1^\lambda, N, \mathbb{QR}_N^+, g, H)$ be the public parameter, where $N \leftarrow \mathbf{BlumGen}(1^\lambda), g \leftarrow_R \mathbb{QR}_N^+$, and $H : \{0,1\}^* \to \mathbb{Z}_{2^\lambda}^*$ is a hash function.

The FB-Schnorr signature scheme is as follows:

$\mathbf{SigGen}(\mathsf{param})$**:** Choose $x \leftarrow_R [(N-1)/4]$. Set $sk := g^x$, $vk := g^{2^\lambda x}$.
Output (vk, sk).

$\mathbf{Sign}(sk, m)$**:** Choose $r \leftarrow_R [(N-1)/4]$. Compute $c = g^{2^\lambda r}, t = H(m, c)$, and $s = g^r \cdot sk^t$.
Output $\sigma = (c, s)$.

$\mathbf{Vrf}(vk, \sigma, m)$**:** Parse $\sigma = (c, s)$. Compute $t = H(m, c)$.
If $s^{2^\lambda} = c \cdot vk^t$ holds, then return 1, else return 0.

Correctness of the scheme is straight-forward.

4.2.2 Security Analysis

The strong unforgeability of the above scheme is guaranteed by the following theorem.

Theorem 3. *The FB-Schnorr signature scheme is strongly unforgeable in the random oracle model, if the factoring assumption holds.*

Before starting to prove, we state the following well-known lemma (a.k.a. the forking lemma), which explicitly appeared in [48].

Lemma 3 (the forking lemma [48])**.** *Let $\mathcal{A}$ be a probabilistic polynomial time Turing machine, given only the public data as input. If $\mathcal{A}$ can find, with non-negligible probability, a valid signature (m, c, t, s), then, with non-negligible probability, a replay of this machine, with the same random tape and a different oracle, outputs two valid signatures (m, c, t, s) and (m, c, t', s') such that $t \neq t'$.*

Remark. Probabilities are taken over random tapes, random oracles, and in some cases, over messages and keys.

We now return to the theorem.

Proof. We will prove that, if there exists an algorithm $\mathcal{A}$ that breaks the strong unforgeability of the scheme with non-negligible probability ε, then there is an algorithm $\mathcal{B}$ that breaks the one-wayness of the permutation ISQ (i.e., the permutation: $x \mapsto x^{2^\lambda}$ acting on the groups $\mathbb{QR}_N^+$) with non-negligible probability.

To construct $\mathcal{B}$, we make use of $\mathcal{A}$ as an internal algorithm we are using, in which we are able to dive into the code of the algorithm and take snapshots of its state after every step. This way, we can backtrack to any state that the attacker was in at any point during its execution.

Without the loss of generality we assume that:

– $\mathcal{A}$ makes at most q_H random oracle queries.

– $\mathcal{A}$ never makes the same random oracle query twice.

– If $\mathcal{A}$ outputs (m, c, s) then it had previously queried $H(m, c)$.

We build an algorithm $\mathcal{B}$ which is, given an instance (N, h), computing an iterated square root of h (i.e. computing $h^{2^{-\lambda}}$), using the adversary $\mathcal{A}$. Algorithm $\mathcal{B}$ is defined as:

1. Pick $i^* \leftarrow_R \{1, 2, \ldots, q_H\}$.
2. Choose a generator $g \leftarrow_R \mathbb{QR}_N^+$, and set $vk := h$. $\mathcal{B}$ provides $\mathcal{A}$ the public parameter $PP := (N, g, H)$ and the verify key vk, where H is a random oracle simulated by $\mathcal{B}$ as follows.

   Random Oracle Simulation. $\mathcal{B}$ maintains a hash list H^{list} which is initially empty, and simulates H as follow:
   – $H(m, c)$: If there is a tuple (m, c, t) in H^{list} then return t, otherwise choose $t \leftarrow_R \mathbb{Z}_{2^\lambda}$, add the tuple (m, c, t) to H^{list} and return t.
3. Simulate $\mathcal{A}$:

- When $\mathcal{A}$ makes its ith oracle query then response by using the above simulation of H. In addition, if $i = i^*$, then $\mathcal{B}$ records this state as $st_1 = (m_{i^*}, c_{i^*}, t_1)$.
- When $\mathcal{A}$ requests to sign a message m, choose $s \leftarrow_R \mathbb{QR}_N^+, t \leftarrow_R \mathbb{Z}_{2^\lambda}$, compute $c = s \cdot vk^{-t}$, add the tuple (m, c, t) to H^{list} and return a signature (c, s).

4. After $\mathcal{A}$ finishes, it outputs (m^*, c^*, s_1). If $(m^*, c^*) \neq (m_{i^*}, c_{i^*})$, then $\mathcal{B}$ outputs $\perp$ and halts. Otherwise, it rewinds $\mathcal{A}$ from the point that $\mathcal{A}$ queries the i^*th oracle query. $\mathcal{B}$ now chooses $t_2 \leftarrow_R \mathbb{Z}_{2^\lambda}$, add the tuple (m, c, t_2) to H^{list} and returns t_2. Then $\mathcal{B}$ records this state as $st_2 = (m_{i^*}, c_{i^*}, t_2)$. $\mathcal{B}$ proceeds exactly the same as above to simulate $\mathcal{A}$. From the forking lemma (Lemma 3), after a polynomial replay of the attacker $\mathcal{A}$, $\mathcal{B}$ obtains one more valid signature (m^*, c^*, s_2) such that $t_1 \neq t_2$. Now, since $s_1^{2^\lambda} \cdot vk^{-t_1} = c^* = s_2^{2^\lambda} \cdot vk^{-t_2}$ and $vk = h$, $\mathcal{B}$ obtains

$$\left(h^{t_2 - t_1}\right)^{2^{-\lambda}} = s_1 \cdot s_2^{-1}.$$

Using Shamir's GCD in the exponent algorithm [52], $\mathcal{B}$ recovers $z = h^{2^{-\lambda}}$ as follows:

(a) Let $\delta = t_2 - t_1$. Since $|t_2 - t_1| < 2^\lambda$ we have that $\mathsf{GCD}(\delta, 2^\lambda) = 1$. By Euclid's algorithm, $\mathcal{B}$ computes integers a, b such that $a\delta + b2^\lambda = 1$.

(b) $\mathcal{B}$ obtains

$$z = h^{2^{-\lambda}} = \left(h^{a\delta + b2^\lambda}\right)^{2^{-\lambda}} = h^{a\delta 2^{-\lambda}} \cdot h^b = (s_1 \cdot s_2^{-1})^a \cdot h^b$$

Finally, $\mathcal{B}$ outputs z as the answer.

This completes the description of the simulation.

Analysis. The index i^* chosen by $\mathcal{B}$ in the first step represents a guess that $\mathcal{A}$ will forge on its i^*th oracle query (i.e. (m_{i^*}, c_{i^*})). Since, $\mathcal{A}$ queries at most q_H random oracle queries, the probability that $\mathcal{B}$ success in breaking the strong unforgeability of the scheme is at least ε/q_H. This is non-negligible if ε is non-negligible. The theorem follows. □

4.3 The Proposed Schemes

In this section, we first propose a bidirectional multi-hop PRE scheme and show that it meets the BM-CPA security under the factoring assumption, in the standard model (i.e., without using random oracles). We then present our main scheme which is BS-CCA secure (in the random oracle model) under the factoring assumption. Finally, we show how to extend our main scheme to achieve a unidirectional and single-hop PRE scheme which is secure in the sense of chosen-ciphertext attack.

4.3.1 The BM-CPA-Secure Scheme

In this section, we first review the IND-CPA secure PKE scheme proposed by Wee [65]. We then show the construction of our BM-CPA secure PRE scheme.

Wee Encryption Scheme [65]. We make use of a public parameter $PP = (N, g)$ for the scheme, where N is a random 2λ-bit Blum integer and g is chosen uniformly from $\mathbb{QR}_N^+$. The Wee encryption scheme for λ-bit message is as follows.

$\mathbf{KGen}(PP)$**:** Choose $x \leftarrow_R [(N-1)/4]$. Set $sk := x, pk := g^{2^\lambda x}$. Output (pk, sk).

$\mathbf{Enc}(PP, pk, m)$**:** Choose $r \leftarrow_R [(N-1)/4]$. Output $(g^{2^\lambda r}, (pk \cdot g)^r, BBS_N(g^r) \oplus m)$.

$\mathbf{Dec}(PP, sk, CT)$**:** Parse $CT = (c_1, c_2, c_3)$. Output $BBS_N(c_2 \cdot c_1^{-sk}) \oplus c_3$.

As shown in [65], the above scheme is IND-CPA secure under the factoring assumption in the standard model.

Construction. Our PRE scheme is an extension of the Wee encryption, in which the algorithms $\mathbf{KGen}$, $\mathbf{Enc}$, and $\mathbf{Dec}$ are exactly the same as that of the Wee encryption, the other are as follows:

$\mathbf{Setup}(1^\lambda)$**:** Output a public parameter $PP = (N, g, h)$, where $N \leftarrow \mathbf{BlumGen}(1^\lambda), g \leftarrow_R \mathbb{QR}_N^+$, and $h := g^{2^\lambda}$.

$\mathbf{RKGen}(PP, sk_i, sk_j)$**:** Output $rk_{ij} := sk_i - sk_j$.

$\mathbf{ReEnc}(PP, rk_{ij}, CT_i)$**:** Parse $CT_i = (c_1, c_2, c_3)$. Set $c_2' := c_2 \cdot c_1^{rk_{ij}}$. Output (c_1, c_2', c_3).

Security Analysis. The security of the scheme is guaranteed by the following theorem.

Theorem 4. *The above PRE scheme is BM-CPA secure under the factoring assumption.*

The detailed proof is given in Section 4.4.1. The proof is quite simple, and follows that of the IND-CPA security of the underlying encryption (i.e., the Wee encryption).

4.3.2 The BS-CCA-Secure Scheme

Idea of the Construction. We first observe that the Wee encryption scheme can be made IND-CCA secure, in the random oracle, thanks to Fujisaki-Okamoto's technique [23] (we call it here the modified Wee encryption scheme). We then follow the idea of constructing bidirectional single-hop PRE proposed by Deng et al. [20]. In particular, we propose a new (and the first) CCA-secure factoring-based PRE scheme by using the FB-Schnorr signature scheme as an one-time strong signature and integrating it with the modified Wee encryption scheme. Since both the underlying signature and encryption schemes are secure under the factoring assumption (See Section 4.2), our scheme achieves the highest security level (i.e. the BS-CCA security) under the same assumption in the random oracle model.

Description of the Construction. The proposed scheme $\mathbf{PRE} = (\mathbf{Setup}, \mathbf{KGen}, \mathbf{RKGen}, \mathbf{Enc}, \mathbf{ReEnc}, \mathbf{Dec}_1, \mathbf{Dec}_2)$ is as follows:

$\mathbf{Setup}(1^\lambda)$**:** Generate $N \leftarrow \mathbf{BlumGen}(1^\lambda), g \leftarrow_R \mathbb{QR}_N^+$, and set $h := g^{2^\lambda}$. Choose four hash function $H_1 : \{0,1\}^* \times \{0,1\}^{\ell_1} \rightarrow [(N-1)/4], H_2 : \mathbb{QR}_N^+ \rightarrow \{0,1\}^{\ell_0+\ell_1}, H_3 : \{0,1\}^* \rightarrow \mathbb{Z}_{2^\lambda}^*, H_4 : \{0,1\}^* \rightarrow \mathbb{QR}_N^+ \times \mathbb{QR}_N^+ \times \{0,1\}^{\ell_0+2\ell_1}$. Output a public parameter $PP = (N, g, h, H_1, H_2, H_3, H_4)$.

$\mathbf{KGen}(PP)$: Choose $x \leftarrow_R [(N-1)/4]$. Set $sk := x, pk := h^x$. Output (pk, sk).

$\mathbf{RKGen}(PP, sk_i, sk_j)$: Output the re-encryption key $rk_{ij} = sk_j - sk_i$.

$\mathbf{Enc}(PP, pk, m)$: this algorithm works as follows:

1. Pick $v \leftarrow_R [(N-1)/4], \omega \leftarrow_R \{0,1\}^{\ell_1}$, and compute $u = H_1(m, \omega)$.
2. Compute $c_0 = h^v, c_1 = h^u, c_2 = (pk \cdot g)^u, c_3 = H_2(g^u) \oplus (m||\omega)$, and $s = g^{v+ut}$, where $t = H_3(c_0, c_1, c_2, c_3)$.
3. Output the ciphertext $CT = (c_0, c_1, c_2, c_3, s)$.

$\mathbf{ReEnc}(PP, rk_{ij}, CT_i)$: this algorithm works as follows:

1. Parse CT_i as $CT_i = (c_0, c_1, c_2, c_3, s)$.
2. Compute $t = H_3(c_0, c_1, c_2, c_3)$ and check whether $s^{2^\lambda} = c_0 \cdot c_1^t$ holds. If not, output $\perp$.
3. Otherwise, compute $c_2' = c_2 \cdot c_1^{rk_{ij}}$. Let $\bar{m} := (c_1, c_2', c_3)$.
4. Pick $\omega_2 \leftarrow_R \{0,1\}^{\ell_1}$, and compute $r = H_1(\bar{m}, \omega_2)$.
5. Compute $A = h^r, B = (pk_j \cdot g)^r$, and $C = H_4(g^r) \oplus (\bar{m}||\omega_2)$, and output the first-level ciphertext $CT_j = (A, B, C)$.

$\mathbf{Dec}_2(PP, sk, CT)$: this algorithm works as follows:

1. Parse CT as $CT = (c_0, c_1, c_2, c_3, s)$.
2. Compute $t = H_3(c_0, c_1, c_2, c_3)$ and check whether $s^{2^\lambda} = c_0 \cdot c_1^t$ holds. If not, output $\perp$.
3. Compute $m||\omega = c_3 \oplus H_2(c_2 \cdot c_1^{-sk})$, and return m if $c_1 = h^{H_1(m,\omega)}$ holds, and $\perp$ otherwise.

$\mathbf{Dec}_1(PP, sk, CT)$: this algorithm works as follows:

1. Parse CT as $CT_=(A, B, C)$, and compute $\bar{m}||\omega_2 = C \oplus H_4(B \cdot A^{-sk})$. Check whether $A = h^{H_1(\bar{m},\omega_2)}$ holds. If not, output $\perp$. Otherwise, parse $\bar{m}$ as $\bar{m} := (c_1, c_2', c_3)$.
2. Compute $m||\omega = c_3 \oplus H_2(c_2' \cdot c_1^{sk})$, and return m if $c_1 = h^{H_1(m,\omega)}$ holds, and $\perp$ otherwise.

Security Analysis. The intuition of CCA security of the proposed scheme can be seen from the following properties.

1. The validity of the original ciphertexts can be publicly verifiable by everyone including the proxy; otherwise, it will suffer from an attack as illustrated in [20, 45]. For our scheme, we integrate the FB-Schnorr signature scheme with the CCA-secure PKE, that is how we get public verifiability. More precisely, the ciphertext component c_1, s in the original ciphertext $CT = (c_0, c_1, c_2, c_3, s)$ can be viewed as a signature signing the message (c_0, c_2, c_3).

2. It should be impossible for the adversary to transform the second-level ciphertext to the first-level one without knowledge of the delegator's secret key or re-encryption key; otherwise, it only yields the RCCA security. In our scheme, the component $c_2' = c_2 \cdot c_1^{rk_{ij}}$ is computed using re-encryption key and completely hidden in C, so the adversary cannot transform the second-level ciphertext to a ciphertext re-encrypted by $\mathbf{ReEnc}$ if he has no knowledge of the re-encryption key.

3. It should be impossible for the adversary to compute the re-encryption key from the target user i^* to itself (i.e., $rk_{i^* \to i^*}$), otherwise it will suffer from an attack as illustrated in [33]. In our scheme, it is clear since the re-encryption key $rk_{i \to i} = sk_i - sk_i = 0$ for all i.

4. For a first-level ciphertext CT_j re-encrypted from a second-level ciphertext CT_i, CT_j should not exhibit any component of CT_i; otherwise, it will fail in achieving CCA-security of $\mathbf{ReEnc}$ (i.e., 1st-level-CCA security). In our scheme, all of the components from the original ciphertext are hidden in C. Furthermore, we use a CCA-secure PKE in the re-encryption algorithm to guarantee the CCA security of re-encrypted ciphertext.

5. In our scheme, $\mathbf{ReEnc}$ and $\mathbf{Dec_2}$ use the same algorithm of checking the validity of the second-level ciphertext CT_i. So in the security game, providing the adversary with a second-level decryption oracle is useless. Indeed, ciphertexts encrypted under public keys from L_{un} can be re-encrypted for corrupted users by using re-encryption oracle. Besides, second-level ciphertext under pk_{i^*} can be translated for other honest users by using rk_{i^*j} (where $j \in L_{un}$) and the resulting ciphertext can be queried for decryption at the first-level by using $\mathcal{O}_{Dec_1}$. This does not contradict the observation of Hanaoka et al. [25].

Theorem 5. *The above PRE scheme is BS-CCA secure in the random oracle model, if the factoring assumption holds.*

The proof of the above theorem is given in Section 4.4.2.

4.3.3 Extension to a Unidirectional Single-Hop Scheme

In this section, we show that our BS-CCA-secure PRE scheme can be extended to achieve a unidirectional and single-hop PRE (US-PRE, for short) scheme which is secure in the sense of chosen-ciphertext attack. Following [17], by using the "token-controlled encryption" technique, we obtain the first factoring-based unidirectional and single-hop PRE scheme. In particular, the scheme is as follows:

$\mathbf{Setup}(1^\lambda)$**:** This algorithm is almost the same at that of the original PRE scheme, except that the domain of the hash function H_4 is modified to agree with the new re-encryption algorithm, and that it has an additional hash function H_5. Concretely, the hash function H_4 now becomes $H_4 : \{0,1\}^* \to \mathbb{QR}_N^+ \times \mathbb{QR}_N^+ \times \{0,1\}^{\ell_0+\ell_1} \times \mathbb{QR}_N^+ \times \mathbb{QR}_N^+ \times \{0,1\}^{\ell_1}$, and the additional hash function $H_5 : \{0,1\}^* \to \mathbb{Z}_{2^\lambda}^*$.

$\mathbf{KGen}(PP)$**:** Choose $x_1, x_2 \leftarrow_R [(N-1)/4]$. Set $sk := (x_1, x_2), pk := (h^{x_1}, h^{x_2})$. Output (pk, sk).

$\mathbf{RKGen}(PP, sk_i, pk_j)$: Output the re-encryption key $rk_{i\to j} = (R_1, R_2, R_3, R_4)$, where R_1, R_2, R_3, and R_4 are computed as follows:

1. Pick $\alpha \leftarrow_R [(N-1)/4], \omega_1 \leftarrow_R \{0,1\}^{\ell_1}$, and compute $r_1 = H_1(\alpha, \omega_1)$.
2. Compute $R_1 = \alpha - (sk_{i,1} H_5(pk_{i,2}) + sk_{i,2})$.
3. Compute $R_2 = h^{r_1}, R_3 = (pk_{j,1} \cdot g)^{r_1}$, and $R_4 = H_2(g^{r_1}) \oplus (\alpha || \omega_1)$.

$\mathbf{Enc}_1(PP, pk, m)$: this algorithm works as follows:

1. Pick $\alpha, u \leftarrow_R [(N-1)/4], \omega, \omega_1 \leftarrow_R \{0,1\}^{\ell_1}$, and compute $r = H_1(m, \omega), r_1 = H_1(\alpha, \omega_1)$.
2. Compute $c_1 = h^u, c_2' = g^u h^{u\alpha}, c_3 = H_2(g^u) \oplus (m||\omega), R_2 = h^{r_1}, R_3 = (pk_1 \cdot g)^{r_1}$, and $R_4 = H_2(g^{r_1}) \oplus (\alpha||\omega_1)$. Let $\bar{m} := (c_1, c_2', c_3, R_2, R_3, R_4)$.
3. Pick $\omega_2 \leftarrow_R \{0,1\}^{\ell_1}$, and compute $r = H_1(\bar{m}, \omega_2)$.
4. Compute $A = h^r, B = (pk_2 \cdot g)^r$, and $C = H_4(g^r) \oplus (\bar{m}||\omega_2)$, and output the first-level ciphertext $CT_j = (A, B, C)$.

$\mathbf{Enc}_2(PP, pk, m)$: this algorithm works as follows:

1. Pick $v \leftarrow_R [(N-1)/4], \omega \leftarrow_R \{0,1\}^{\ell_1}$, and compute $u = H_1(m, \omega)$.
2. Compute $c_0 = h^v, c_1 = h^u, c_2 = (pk_1^{H_5(pk_2)} pk_2 \cdot g)^u, c_3 = H_2(g^u) \oplus (m||\omega)$, and $s = g^{v+ut}$, where $t = H_3(c_0, c_1, c_2, c_3)$.
3. Output the ciphertext $CT = (c_0, c_1, c_2, c_3, s)$.

$\mathbf{ReEnc}(PP, rk_{i\to j}, CT_i)$: this algorithm works as follows:

1. Parse CT_i as $CT_i = (c_0, c_1, c_2, c_3, s)$ and $rk_{i\to j}$ as $rk_{i\to j} = (R_1, R_2, R_3, R_4)$.
2. Compute $t = H_3(c_0, c_1, c_2, c_3)$ and check whether $s^{2^\lambda} = c_0 \cdot c_1^t$ holds. If not, output $\perp$.
3. Otherwise, compute $c_2' = c_2 \cdot c_1^{R_1}$. Let $\bar{m} := (c_1, c_2', c_3, R_2, R_3, R_4)$.
4. Pick $\omega_2 \leftarrow_R \{0,1\}^{\ell_1}$, and compute $r = H_1(\bar{m}, \omega_2)$.
5. Compute $A = h^r, B = (pk_{j,2} \cdot g)^r$, and $C = H_4(g^r) \oplus (\bar{m}||\omega_2)$, and output the first-level ciphertext $CT_j = (A, B, C)$.

$\mathbf{Dec}_2(PP, sk, CT)$: this algorithm works as follows:

1. Parse CT as $CT = (c_0, c_1, c_2, c_3, s)$.
2. Compute $t = H_3(c_0, c_1, c_2, c_3)$ and check whether $s^{2^\lambda} = c_0 \cdot c_1^t$ holds. If not, output $\perp$.
3. Compute $m||\omega = c_3 \oplus H_2(c_2 \cdot c_1^{-(sk_1 H_5(pk_2) + sk_2)})$, and return m if $c_1 = h^{H_1(m,\omega)}$ holds, and $\perp$ otherwise.

$\mathbf{Dec}_1(PP, sk, CT)$: this algorithm works as follows:

1. Parse CT as $CT_{=}(A, B, C)$, and compute $\bar{m}||\omega_2 = C \oplus H_4(B \cdot A^{-sk_2})$. Check whether $A = h^{H_1(\bar{m},\omega_2)}$ holds. If not, output $\perp$. Otherwise, parse $\bar{m}$ as $\bar{m} := (c_1, c_2', c_3, R_2, R_3, R_4)$.
2. Compute $\alpha||\omega_1 = R_4 \oplus H_2(R_3 \cdot R_2^{-sk_1})$. Check whether $R_2 = h^{H_1(\alpha,\omega_1)}$ If not, output $\perp$.
3. Compute $m||\omega = c_3 \oplus H_2(c_2' \cdot c_1^{-\alpha})$, and return m if $c_1 = h^{H_1(m,\omega)}$ holds, and $\perp$ otherwise.

We have the following theorem.

Theorem 6. *The above PRE scheme is US-CCA secure in the random oracle model, if the factoring assumption holds.*

The proof of the above theorem is given in Section 4.4.3.

4.4 Proofs

4.4.1 Proof of Theorem 4

We build an algorithm $\mathcal{B}$ which is, given a factoring instance $(N, g, g^{2^\lambda a}, Q)$, deciding whether $Q = BBS_N(g^a)$, using the BM-CPA adversary $\mathcal{A}$. $\mathcal{B}$ runs the adversary $\mathcal{A}$ simulating its view as in the BM-CPA security game as follows:

Setup. $\mathcal{B}$ sets $h := g^{2^\lambda}$, then provides $\mathcal{A}$ the public parameter $PP := (N, g, h)$. $\mathcal{B}$ initializes two empty lists L_{un} and L_{corr}, then maintains it to record the indices of all uncorrupted and corrupted users, respectively, in the game.

Phase 1. $\mathcal{B}$ simulates the oracles $\mathcal{O}_{unKGen}$, $\mathcal{O}_{corrKGen}$, $\mathcal{O}_{RKGen}$, and $\mathcal{O}_{ReEnc}$ to answer the questions issued by $\mathcal{A}$ as follows:

- $\mathcal{O}_{unKGen}(i)$: $\mathcal{B}$ returns the symbol $\perp$ if $i \in L_{un} \cup L_{corr}$; otherwise $\mathcal{B}$ chooses randomly $x_i \leftarrow_R [(N-1)/4]$, sets and returns $pk_i := h^{x_i} \cdot g^{-1}$ (meaning that $sk_i = x_i - \frac{1}{2^\lambda} \bmod |\mathbb{QR}_N^+|$). $\mathcal{B}$ adds i to L_{un}. Note that, $\mathcal{B}$ does not need to know sk_i.
- $\mathcal{O}_{corrKGen}(i)$: $\mathcal{B}$ returns the symbol $\perp$ if $i \in L_{un} \cup L_{corr}$; otherwise $\mathcal{B}$ chooses randomly $x_i \leftarrow_R [(N-1)/4]$, and returns (pk_i, sk_i), where $pk_i := h^{x_i}$, $sk_i := x_i$. $\mathcal{B}$ adds i to L_{corr}.
- $\mathcal{O}_{RKGen}(i, j)$: if $i \neq j \in L_{un}$ or $i \neq j \in L_{corr}$ $\mathcal{B}$ returns $rk_{ij} := x_j - x_i$; otherwise returns the symbol $\perp$.
- $\mathcal{O}_{ReEnc}(i, j, CT_i)$: $\mathcal{B}$ parses $CT_i = (c_1, c_2, c_3)$. If $i \neq j \in L_{un}$ or $i \neq j \in L_{corr}$, then it computes $rk_{ij} = x_j - x_i$, and returns $CT_j \leftarrow \mathbf{ReEnc}(rk_{ij}, CT_i)$ (i.e. $CT_j = (c_1, c_2', c_3)$, where $c_2' := c_2 \cdot c_1^{rk_{ij}}$). Otherwise, it returns the symbol $\perp$.

Challenge. When $\mathcal{A}$ decides that Phase 1 is over, it outputs a target public key pk_{i^*} and two equal-length plaintexts $m_0, m_1 \in \{0,1\}^\lambda$. At this stage, $\mathcal{B}$ flips a random coin $\sigma \in \{0,1\}$

and defines: $c_1 := g^{2^\lambda a}, c_2 := \left(g^{2^\lambda a}\right)^{x_{i^*}}, c_3 := Q \oplus m_\sigma$. Finally, $\mathcal{B}$ returns $CT^* = (c_1, c_2, c_3)$ as the challenge ciphertext to $\mathcal{A}$.

Observe the following to see that, CT^* is indeed a valid challenge ciphertext under the public key pk_{i^*} with the random exponent $r = a$ if $Q = BBS_N(g^a)$.

$$c_1 = g^{2^\lambda a} = h^r, c_2 = \left(g^{2^\lambda a}\right)^{x_{i^*}} = \left(g^{2^\lambda a}\right)^{sk_{i^*} - \frac{1}{2^\lambda}} = \left(g^{2^\lambda sk_{i^*}} \cdot g\right)^a = (pk_{i^*} \cdot g)^r,$$
$$c_3 = Q \oplus m_\sigma.$$

In contrast, if Q is random in $\{0,1\}^\lambda$, CT^* perfectly hides m_σ and $\mathcal{A}$ cannot guess σ with better probability than $1/2$.

Phase 2. $\mathcal{B}$ does exactly as in **Phase 1**.

Guess. Finally, $\mathcal{A}$ outputs a guess $\sigma' \in \{0,1\}$.

Output. If $\sigma' = \sigma$, $\mathcal{B}$ outputs 1 to guess $Q = BBS_N(g^a)$; else $\mathcal{B}$ outputs 0 to guess that Q is a random element in $\{0,1\}^\lambda$.

This completes the description of the simulation.

Next, we analyze the simulation. It is obviously that the simulation of $\mathcal{B}$ is perfect from the view of $\mathcal{A}$. Therefore $\mathcal{B}$ can deciding whether $Q = BBS_N(g^a)$ with probability at least that of $\mathcal{A}$ in breaking the BM-CPA security of the proposed scheme.

This concludes the proof of Theorem 4.

4.4.2 Proof of Theorem 5

We split up the proof of Theorem 5 into two parts: we prove that our scheme is 2nd-BS-CCA secure and 1st-BS-CCA secure in Section 4.4.2.1 and Section 4.4.2.2, respectively. Combining both parts yields Theorem 5.

4.4.2.1 2nd-BS-CCA Security

Since the permutation ISQ (i.e., the permutation : $x \mapsto x^{2^\lambda}$acting on the groups $\mathbb{QR}_N^+$) is one-way function if and only if the factoring problem is hard (Theorem 2), it is sufficient to prove that if there exists an adversary $\mathcal{A}$ that breaks the 2nd-BS-CCA security of the scheme with non-negligible probability, then there exists an adversary $\mathcal{B}$ that breaks the one-wayness of ISQ with overwhelming probability. Let $\mathrm{Adv}_{\mathcal{B}}^{\mathsf{ISQ}}(\lambda)$ denote the advantage of $\mathcal{B}$ in breaking the one-wayness of ISQ.

Since the fact that the FB-Schnorr scheme is one-time strongly unforgeable under the factoring assumption, we can assume that the signature scheme used to sign the challenge ciphertext CT_i^* (i.e. the scheme with the verify/sign keys pair are (c_1^*, g^{u^*})) is one-time strongly unforgeable.

We build an algorithm $\mathcal{B}$ which is, given a factoring instance $(N, g, g^{2^\lambda a})$, computing (g^a), using the 2nd-BS-CCA adversary $\mathcal{A}$. $\mathcal{B}$ runs the adversary $\mathcal{A}$ simulating its view as in the 2nd-BS-CCA security game as follows:

Setup. $\mathcal{B}$ initializes two empty lists L_{un} and L_{corr}, then maintains it to record the indices of all uncorrupted and corrupted users, respectively, in the game. $\mathcal{B}$ sets $h := g^{2^\lambda}$, then provides $\mathcal{A}$ the public parameter $PP := (N, g, h, H_1, H_2, H_3, H_4)$, where $H_1, H_2, H_3,$ and H_4 are the random oracles simulated by $\mathcal{B}$ as follows.

Hash Oracles Simulation. At any time $\mathcal{A}$ can issue the random oracle queries $H_1, H_2, H_3,$ and H_4. $\mathcal{B}$ maintains four hash lists $H_1^{\text{list}}, H_2^{\text{list}}, H_3^{\text{list}},$ and H_4^{list} which are initially empty, and respond as follow:

- $H_1(m, \omega)$: If there is a tuple (m, ω, r) in H_1^{list} then return r, otherwise choose $r \leftarrow_R [(N-1)/4]$, add the tuple (m, ω, r) to H_1^{list} and return r.
- $H_2(X)$: If there is a tuple (X, β) in H_2^{list} then return β, otherwise choose $\beta \leftarrow_R \{0,1\}^{\ell_0+\ell_1}$, add the tuple (X, β) to H_2^{list} and return β.
- $H_3(c_0, c_1, c_2, c_3)$: If there is a tuple (c_0, c_1, c_2, c_3, t) in H_3^{list} then return t, otherwise choose $t \leftarrow_R \mathbb{Z}_{2^\lambda}^*$, add the tuple (c_0, c_1, c_2, c_3, t) to H_3^{list}, and return t.
- $H_4(f)$: If there is a tuple (f, γ) in H_4^{list} then return γ, otherwise choose $\gamma \leftarrow_R \mathbb{QR}_N^+ \times \mathbb{QR}_N^+ \times \{0,1\}^{\ell_0+2\ell_1}$, add the tuple (f, γ) to H_4^{list} and return γ.

Find stage (i.e. Phases 1 and 2). $\mathcal{B}$ simulates the oracles $\mathcal{O}_{unKGen}$, $\mathcal{O}_{corrKGen}$, $\mathcal{O}_{RKGen}$, $\mathcal{O}_{ReEnc}$, $\mathcal{O}_{Dec_1}$, and $\mathcal{O}_{Dec_2}$ to answer the questions issued by $\mathcal{A}$ as follows:

- $\mathcal{O}_{unKGen}(i)$: $\mathcal{B}$ returns $\perp$ if $i \in L_{un} \cup L_{corr}$; otherwise $\mathcal{B}$ chooses randomly $x_i \leftarrow_R [(N-1)/4]$, sets and returns $pk_i := h^{x_i} \cdot g^{-1}$ (meaning that $sk_i = x_i - \frac{1}{2^\lambda} \bmod |\mathbb{QR}_N^+|$). $\mathcal{B}$ adds i to L_{un}. Note that, $\mathcal{B}$ does not need to know sk_i.
- $\mathcal{O}_{corrKGen}(i)$: $\mathcal{B}$ returns $\perp$ if $i \in L_{un} \cup L_{corr}$; otherwise $\mathcal{B}$ chooses randomly $x_i \leftarrow_R [(N-1)/4]$, and returns (pk_i, sk_i), where $pk_i := h^{x_i}$, $sk_i := x_i$. $\mathcal{B}$ adds i to L_{corr}.
- $\mathcal{O}_{RKGen}(i, j)$: if $i \neq j \in L_{un}$ or $i \neq j \in L_{corr}$ $\mathcal{B}$ returns $rk_{ij} := x_j - x_i$; otherwise returns the symbol $\perp$.
- $\mathcal{O}_{ReEnc}(i, j, CT_i)$: $\mathcal{B}$ parses $CT_i = (c_0, c_1, c_2, c_3, s)$, and does as follows:
  1. If $i = j$, then return $\perp$.
  2. if $i \neq j \in L_{un}$ or $i \neq j \in L_{corr}$ $\mathcal{B}$ computes the re-encryption key $rk_{ij} := x_j - x_i$. Next, $\mathcal{B}$ follows the algorithm $\mathbf{ReEnc}$ to compute the ciphtertext CT_j by using rk_{ij} and simulating the hash oracles H_1, H_3, H_4 as follows:
     (a) Search in the list H_3^{list} to see whether there exists $(a_0, a_1, a_2, a_3, t) \in H_3^{\text{list}}$ such that $(a_0 = c_0) \wedge (a_1 = c_1) \wedge (a_2 = c_2) \wedge (a_3 = c_3)$. If there exists no such tuple, then return $\perp$. (This corresponds to the event $\texttt{ReEncErr}_1$ to be explained).
     (b) Check whether $s^{2^\lambda} = c_0 \cdot c_1^t$ holds. If not, then return $\perp$.
     (c) Search in the list H_1^{list} to see whether there exists $(m, \omega, r) \in H_1^{\text{list}}$ such that $h^r = c_1, (pk_i \cdot g)^r = c_2$. If there exists no such tuple, then return $\perp$. (This corresponds to the event $\texttt{ReEncErr}_2$ to be explained).
     (d) Compute $c_2' = c_2 \cdot c_1^{rk_{ij}}$. Let $\bar{m} := (c_1, c_2', c_3)$.

(e) Pick $\omega_2 \leftarrow_R \{0,1\}^{\ell_1}, r_1 \leftarrow_R [(N-1)/4]$, and add $(\bar{m}, \omega_2, r_1)$ to the list H_1^{list}.

(f) Compute $A = h^{r_1}, B = (pk_j \cdot g)^{r_1}$,

(g) Search in the list H_4^{list} to see whether there exists (f, γ) such that $f = g^{r_1}$. If there exists no such tuple, then pick $\gamma \leftarrow_R \mathbb{QR}_N^+ \times \mathbb{QR}_N^+ \times \{0,1\}^{\ell_0+2\ell_1}$, add the tuple (f, γ) to H_4^{list}.

(h) Compute $C = \gamma \oplus (\bar{m}||\omega_2)$, and output the first-level ciphertext $CT_j = (A, B, C)$.

3. If $(i = i^*) \wedge (j \in L_{corr}) \wedge (c_1 = c_1^*)$, then return $\perp$. Since c_1^* is the verify key of the one-time strong signature scheme **SIG** used to sign the challenge ciphertext CT^*, CT_i is indeed the challenge ciphertext CT^*. Therefore, $\mathcal{B}$ should return $\perp$.

4. If $i \in L_{corr}, j \in L_{un}$ or $i \in L_{un}, j \in L_{corr}$, then $\mathcal{B}$ does as follows:

(a) Search in the list H_3^{list} to see whether there exists $(a_0, a_1, a_2, a_3, t) \in H_3^{\text{list}}$ such that $(a_0 = c_0) \wedge (a_1 = c_1) \wedge (a_2 = c_2) \wedge (a_3 = c_3)$. If there exists no such tuple, then return $\perp$.

(b) Check whether $s^{2^\lambda} = c_0 \cdot c_1^t$ holds. If not, then return $\perp$.

(c) Search whether there exists a tuple $(m, \omega, r) \in H_1^{\text{list}}$ such that $h^r = c_1, (pk_i \cdot g)^r$. If there exists no such tuple, then return $\perp$.

(d) Compute $c_2' = (pk_j \cdot g)^r)$. Let $\bar{m} := (c_1, c_2', c_3)$.

(e) Pick $\omega_2 \leftarrow_R \{0,1\}^{\ell_1}, r_1 \leftarrow_R [(N-1)/4]$, and add $(\bar{m}, \omega_2, r_1)$ to the list H_1^{list}.

(f) Compute $A = h^{r_1}, B = (pk_j \cdot g)^{r_1}$,

(g) Search in the list H_4^{list} to see whether there exists (f, γ) such that $f = g^{r_1}$. If there exists no such tuple, then pick $\gamma \leftarrow_R \mathbb{QR}_N^+ \times \mathbb{QR}_N^+ \times \{0,1\}^{\ell_0+2\ell_1}$, add the tuple (f, γ) to H_4^{list}.

(h) Compute $C = \gamma \oplus (\bar{m}||\omega_2)$, and output the first-level ciphertext $CT_j = (A, B, C)$.

– $\mathcal{O}_{Dec_1}(i, CT_i)$: If $i \in L_{corr}$, then $\mathcal{B}$ does as the algorithm $\mathbf{Dec}_1$ to decrypt the ciphertext CT_i by using $sk_i = x_i$ and hash lists $H_1^{\text{list}}, H_2^{\text{list}}, H_3^{\text{list}}$, and H_4^{list}. Otherwise $\mathcal{B}$ parses CT_i as $CT_i = (A, B, C)$, and does as follows:

1. Search in the list H_1^{list} and H_4^{list} to see whether there exist $(m, \omega, r) \in H_1^{\text{list}}$ and $(f, \gamma) \in H_4^{\text{list}}$ such that

$$h^r = A, (pk_i \cdot g)^r = B, \gamma \oplus (m||\omega) = C, \text{and } f = g^r. \tag{4.1}$$

If there exists no such tuple, then return $\perp$. (This corresponds to the event DErr_1 to be explained).

2. Parse $\bar{m}$ as $\bar{m} := (c_1, c_2', c_3)$.

3. Search in the list H_1^{list} to see whether there exists $(m, \omega, r) \in H_1^{\text{list}}$ such that $h^r = c_1, (pk_j \cdot g)^r = c_2'$. If there exists no such tuple, then return $\perp$. (This corresponds to the event DErr_2 to be explained).

4. Compute $m||\omega = c_3 \oplus H_2(g^r)$, and returns m if $c_1 = h^{H_1(m,\omega)}$ holds, and $\perp$ otherwise.

– $\mathcal{O}_{Dec_2}(i, CT_i)$: second-level ciphertexts encrypted under public keys from L_{un} can be re-encrypted for corrupted users by using re-encryption oracle $\mathcal{O}_{ReEnc}$. Besides, second-level ciphertexts under pk_{i^*} can be translated for other honest users by using rk_{i^*j} (where $j \in L_{un}$) which can be computed as in simulating of the re-encryption key oracle $\mathcal{O}_{RKGen}$, and the resulting ciphertexts are decrypted by using $\mathcal{O}_{Dec_1}$.

Challenge. When $\mathcal{A}$ decides that Phase 1 is over, it outputs a target public key pk_{i^*} and two equal-length plaintexts $m_0, m_1 \in \{0,1\}^\lambda$. If $i^* \in L_{corr}$, then $\mathcal{B}$ outputs $\perp$ and halts, since we require $i^* \in L_{un}$ for $\mathcal{A}$ to win the game. Otherwise, $\mathcal{B}$ flips a random coin $\sigma \in \{0,1\}$ and does as follows:

1. Pick $s^* \leftarrow_R \mathbb{QR}_N^+, t^* \leftarrow_R \mathbb{Z}_{2^\lambda}$, and compute $c_0^* = (s^*)^{2^\lambda} \cdot (g^{2^\lambda a})^{-t^*}$.
2. Define $c_1^* = g^{2^\lambda a}$ (meaning that $u^* = a$).
3. Compute $c_2^* = (g^{2^\lambda a})^{x_{i^*}}$.
4. Pick $c_3^* \leftarrow_R \{0,1\}^{\ell_0+\ell_1}$ and define $H_3(c_0^*, c_1^*, c_2^*, c_3^*) = t^*$.
5. Pick $\omega^* \leftarrow_R \{0,1\}^{\ell_1}$, and implicitly define $H_2(g^a) = c_3^* \oplus (m_\sigma||\omega^*)$ and $H_1(m_\sigma||\omega^*) = a$ (note that $\mathcal{B}$ does not know a and g^a).
6. Return $CT^* = (c_0^*, c_1^*, c_2^*, c_3^*, s^*)$ as the challenge ciphertext to $\mathcal{A}$.

Observe the following to see that, CT^* is identically distributed as the real one from the construction. Let $g^{v^*} := s^* \cdot g^{-at^*}$ and $u^* := a$. We have

1. $c_0^* = (s^*)^{2\lambda} \cdot (g^{2^\lambda a})^{-t^*} = (s^* \cdot g^{-at^*})^{2^\lambda} = g^{2^\lambda v^*} = h^{v^*}$.
2. $c_1^* = g^{2^\lambda a} = g^{2^\lambda u^*} = h^{u^*}$.
3. $c_2^* = (g^{2^\lambda a})^{x_{i^*}} = (g^{2^\lambda x_{i^*}})^a = (g^{2^\lambda sk_{i^*}+1})^{u^*} = (pk_{i^*} \cdot g)^{u^*}$. (Note that $sk_{i^*} = x_{i^*} - \frac{1}{2^\lambda} \bmod |\mathbb{QR}_N^+|$ since $i^* \in L_{un}$.)
4. $c_3^* = H_2(g^a) \oplus (m_\sigma||\omega^*) = H_2(g^{u^*}) \oplus (m_\sigma||\omega^*)$.
5. $s^* = s^* \cdot g^{-at^*} \cdot g^{at^*} = g^{v^*+u^*t^*} = g^{v^*+u^*H_3(c_0^*,c_1^*,c_2^*,c_3^*)}$.

Guess. Finally, $\mathcal{A}$ outputs a guess $\sigma' \in \{0,1\}$.

Output. If $\sigma' = \sigma$, then $\mathcal{B}$ searches (X, β) from the list H_2^{list} such that $X^{2^\lambda} = g^{2^\lambda a}$, and output X as the solution; otherwise $\mathcal{B}$ outputs $\perp$ and halts.

This completes the description of the simulation.

Analysis. Before analyzing the simulation we set $q := |\mathbb{QR}_N^+|$, and assume that $\mathcal{A}$ issues at most $q_{H_1}, q_{H_2}, q_{H_3}, q_{H_4}, q_{re}$, and q_d queries to $H_1, H_2, H_3, H_4, \mathcal{O}_{ReEnc}$, and $\mathcal{O}_{Dec_1}$, respectively.

The main idea of the analysis is borrowed from [20]. We first evaluate the simulations of the random oracles. It is clear that the simulation of H_4 is perfect. Let $\texttt{AskH}_1^*$ be the event that $\mathcal{A}$ queried (m_σ, ω^*) to H_1, where ω^* is chosen by $\mathcal{B}$ in the challenge phase. It is obvious that the

simulation of H_1 is perfect if $\texttt{AskH}_1^*$ did not occur. Since $\mathcal{B}$ picks $\omega^* \leftarrow_R \{0,1\}^{\ell_1}$ which is hidden by the "one-time pad" given by H_2, we have

$$\Pr[\texttt{AskH}_1^*] \leq \frac{q_{H_1}}{2^{\ell_1}}.$$

Let $\texttt{AskH}_2^*$, and $\texttt{AskH}_3^*$ be the event that $\mathcal{A}$ queried g^a to H_2 and $(c_0^*, c_1^*, c_2^*, c_3^*)$ to H_3, respectively. The simulation of H_2 and H_3 are also perfect, as long as $\texttt{AskH}_2^*$ and $\texttt{AskH}_3^*$ did not occur. Similarly, since $\mathcal{B}$ picks $c_3^* \leftarrow_R \{0,1\}^{\ell_0+\ell_1}$ which is hidden by the "one-time pad" given by H_2, we have

$$\Pr[\texttt{AskH}_3^*] \leq \frac{q_{H_3}}{2^{\ell_0+\ell_1}}.$$

As argued before, the challenged ciphertext provided for $\mathcal{A}$ is identically distributed as the real one from the construction. From the description of the simulation, it can be seen that the responses to $\mathcal{A}$'s re-encryption key queries are also perfect.

Next, we evaluate the simulation of the re-encryption oracle. The responses to $\mathcal{A}$'s re-encryption queries are perfect, unless $\mathcal{A}$ can submit valid second-level ciphertexts without querying hash functions H_3 and H_1 (i.e. the events $\texttt{ReEncErr}_1$ and $\texttt{ReEncErr}_2$, respectively). Let $\texttt{ReEncErr}$ be the event that $\texttt{ReEncErr}_1 \vee \texttt{ReEncErr}_2$ happens during the entire simulation. Since H_1 and H_3 act as random oracles and $\mathcal{A}$ issues at most q_{re} re-encryption queries, we have

$$\begin{aligned}\Pr[\texttt{ReEncErr}] &= \Pr[\texttt{ReEncErr}_1 \vee \texttt{ReEncErr}_2] \\ &\leq \Pr[\texttt{ReEncErr}_1] + \Pr[\texttt{ReEncErr}_2] \leq \frac{q_{re}}{q}.\end{aligned}$$

Now, we analyze the simulation of the decryption oracle. The simulation of the decryption oracle is perfect, with the exception that simulation errors may occur in rejecting some valid ciphertexts. However, these errors are not significant as shown as follows: Suppose that (pk_i, CT_i) has been issued as a valid first-level ciphertext. Even CT_i is valid, there are the three following cases that CT_i can be produced without querying to the random oracles:

- $\mathcal{A}$ did not query (m, ω) to H_1 or f to H_4 such that (4.4) holds (i.e. the event DErr_1 occurs). Let $\texttt{Valid}$ be the event that CT_1 is valid. Let $\texttt{AskH}_1$and $\texttt{AskH}_4$ respectively be events that (m, ω) has been queried to H_1 and f has been queried to H_4 in the event DecErr_1. We have

$$\begin{aligned}\Pr[\texttt{Valid}|\mathsf{DErr}_1] &= \Pr[\texttt{Valid}|(\neg\texttt{AskH}_1 \vee \neg\texttt{AskH}_4)] \\ &\leq \Pr[\texttt{Valid}|\neg\texttt{AskH}_1] + \Pr[\texttt{Valid}|\neg\texttt{AskH}_4].\end{aligned}$$

  On the other hand, observe that

$$\begin{aligned}\Pr[\texttt{Valid}|\neg\texttt{AskH}_1] &= \Pr[\texttt{Valid} \wedge \texttt{AskH}_4|\neg\texttt{AskH}_1] + \Pr[\texttt{Valid} \wedge \neg\texttt{AskH}_4|\neg\texttt{AskH}_1] \\ &\leq \Pr[\texttt{AskH}_4|\neg\texttt{AskH}_1] + \Pr[\texttt{Valid}|(\neg\texttt{AskH}_1 \wedge \neg\texttt{AskH}_4)] \\ &\leq \frac{q_{H_4}}{q^2 \cdot 2^{\ell_0+2\ell_1}} + \frac{1}{q}.\end{aligned}$$

  Similarly, we have

$$\Pr[\texttt{Valid}|\neg\texttt{AskH}_4] \leq \frac{q_{H_1}}{q} + \frac{1}{q^2 \cdot 2^{\ell_0+2\ell_1}}.$$

Therefore, we have

$$\begin{aligned}\Pr[\mathtt{Valid}|\mathsf{DErr}_1] &\leq \Pr[\mathtt{Valid}|\neg\mathtt{AskH}_1] + \Pr[\mathtt{Valid}|\neg\mathtt{AskH}_4] \\ &\leq \frac{q_{H_4}+1}{q^2 \cdot 2^{\ell_0+2\ell_1}} + \frac{q_{H_1}+1}{q}.\end{aligned}$$

Let DecErr_1 be the event that $\mathtt{Valid}|\mathsf{DErr}_1$ happens during the entire simulation. Then, since at most q_d decryption oracles are issued, we have

$$\Pr[\mathsf{DecErr}_1] \leq q_d \left(\frac{q_{H_4}+1}{q^2 \cdot 2^{\ell_0+2\ell_1}} + \frac{q_{H_1}+1}{q} \right).$$

- $\mathcal{A}$ did not query (m, ω) to H_1 such that $h^{H_1(m,\omega)} = c_1$ and $(pk_j \cdot g)^{H_1(m,\omega)} = c_2'$ holds (i.e. the event DErr_2 occurs). It is clear that $\Pr[\mathtt{Valid}|\mathsf{DErr}_2] = \frac{1}{q}$. Let DecErr_2 be the event that $\mathtt{Valid}|\mathsf{DErr}_2$ happens during the entire simulation. We have

$$\Pr[\mathsf{DecErr}_2] \leq \frac{q_d}{q}.$$

Let DecErr be the event that $\mathtt{Valid}|\mathsf{DErr}_1 \vee \mathsf{DErr}_2$ happens during the entire simulation. Then, we have

$$\begin{aligned}\Pr[\mathsf{DecErr}] &= \Pr[\mathtt{Valid}|\mathsf{DErr}_1 \vee \mathsf{DErr}_2] \\ &\leq \Pr[\mathtt{Valid}|\mathsf{DErr}_1] + \Pr[\mathtt{Valid}|\mathsf{DErr}_2] \\ &= \Pr[\mathsf{DecErr}_1] + \Pr[\mathsf{DecErr}_2] \\ &\leq \frac{q_d(q_{H_4}+1)}{q^2 \cdot 2^{\ell_0+2\ell_1}} + \frac{q_d(q_{H_1}+2)}{q}.\end{aligned}$$

Now, let $\mathtt{Error}$ denote the event $\mathtt{AskH}_1^* \vee \mathtt{AskH}_2^* \vee \mathtt{AskH}_3^* \vee \mathtt{ReEncErr} \vee \mathsf{DecErr}$. If the event $\mathtt{Error}$ does not happen, it is clear that $\mathcal{A}$ cannot gain any advantage greater than $\frac{1}{2}$ in guessing σ due to the randomness of the output of the random oracle H_2. Namely, we have $\Pr[\sigma' = \sigma|\neg\mathtt{Error}] = \frac{1}{2}$. Hence, by splitting $\Pr[\sigma' = \sigma]$, we have

$$\begin{aligned}\Pr[\sigma' = \sigma] &= \Pr[\sigma' = \sigma|\neg\mathtt{Error}] \Pr[\neg\mathtt{Error}] + \Pr[\sigma' = \sigma|\mathtt{Error}] \Pr[\mathtt{Error}] \\ &\leq \frac{1}{2} \Pr[\neg\mathtt{Error}] + \Pr[\mathtt{Error}] \\ &= \frac{1}{2} + \frac{1}{2} \Pr[\mathtt{Error}],\end{aligned}$$

and

$$\Pr[\sigma' = \sigma] \geq \Pr[\sigma' = \sigma|\neg\mathtt{Error}] \Pr[\neg\mathtt{Error}] = \frac{1}{2} - \frac{1}{2} \Pr[\mathtt{Error}].$$

Then we have

$$\left| \Pr[\sigma' = \sigma] - \frac{1}{2} \right| \leq \frac{1}{2} \Pr[\mathtt{Error}].$$

By definition of the advantage for the 2nd-BS-CCA adversary, then we have

$$\begin{aligned}
\varepsilon := \mathrm{Adv}^{\text{2nd-CCA}}_{\mathrm{PRE},\mathcal{A}}(\lambda) &= \left|\Pr[\sigma' = \sigma] - \frac{1}{2}\right| \\
&\leq \frac{1}{2}\Pr[\mathtt{Error}] \\
&= \frac{1}{2}\Pr[\mathtt{AskH}_1^* \vee \mathtt{AskH}_2^* \vee \mathtt{AskH}_3^* \vee \mathtt{ReEncErr} \vee \mathsf{DecErr}] \\
&\leq \frac{1}{2}\left(\Pr[\mathtt{AskH}_1^*] + \Pr[\mathtt{AskH}_2^*] + \Pr[\mathtt{AskH}_3^*] + \Pr[\mathtt{ReEncErr}] + \Pr[\mathsf{DecErr}]\right).
\end{aligned}$$

Since $\Pr[\mathtt{AskH}_1^*] \leq \frac{q_{H_1}}{2^{\ell_1}}, \Pr[\mathtt{AskH}_3^*] \leq \frac{q_{H_3}}{2^{\ell_0+\ell_1}}, \Pr[\mathtt{ReEncErr}] \leq \frac{q_{re}}{q}$, and $\Pr[\mathsf{DecErr}] \leq \frac{q_d(q_{H_4}+1)}{q^2 \cdot 2^{\ell_0+2\ell_1}} + \frac{q_d(q_{H_1}+2)}{q}$, we obtain

$$\begin{aligned}
\Pr[\mathtt{AskH}_2^*] &\geq 2\varepsilon - \left(\Pr[\mathtt{AskH}_1^*] + \Pr[\mathtt{AskH}_3^*] + \Pr[\mathtt{ReEncErr}] + \Pr[\mathsf{DecErr}]\right) \\
&\geq 2\varepsilon - \frac{q_{H_1}}{2^{\ell_1}} - \frac{q_{H_3}}{2^{\ell_0+\ell_1}} - \frac{q_{re}}{q} - \frac{q_d(q_{H_4}+1)}{q^2 \cdot 2^{\ell_0+2\ell_1}} - \frac{q_d(q_{H_1}+2)}{q} \\
&= 2\varepsilon - \frac{q_{re} + q_d(q_{H_1}+2)}{q} - \frac{q_{H_1}}{2^{\ell_1}} - \frac{q_{H_3}}{2^{\ell_0+\ell_1}} - \frac{q_d(q_{H_4}+1)}{q^2 \cdot 2^{\ell_0+2\ell_1}}.
\end{aligned}$$

Meanwhile, if event $\mathtt{AskH}_2^*$ happens, the algorithm $\mathcal{B}$ will be able to compute g^a, and consequently, we obtain

$$\mathrm{Adv}^{\mathsf{ISQ}}_{\mathcal{B}}(\lambda) = \Pr[\mathtt{AskH}_2^*] \geq 2\varepsilon - \frac{q_{re} + q_d(q_{H_1}+2)}{q} - \frac{q_{H_1}}{2^{\ell_1}} - \frac{q_{H_3}}{2^{\ell_0+\ell_1}} - \frac{q_d(q_{H_4}+1)}{q^2 \cdot 2^{\ell_0+2\ell_1}}.$$

From the description of the simulation, the running time of the algorithm $\mathcal{B}$ can be bounded by

$$\begin{aligned}
t' \leq t &+ (q_{H_1} + q_{H_2} + q_{H_3} + q_u + q_c + q_{rk} + q_{re} + q_d)\mathcal{O}(1) \\
&+ (q_u + q_c + (2q_{H_1} + q_{H_4})q_{re} + 5q_{H_1}q_d)t_{exp},
\end{aligned}$$

where t_{exp} denotes the running time of an exponentiation in group $\mathbb{QR}_N^+$.

This completes the proof of the 2nd-BS-CCA security of the scheme.

4.4.2.2 1st-BS-CCA Security

We build an algorithm $\mathcal{B}$ which is, given a factoring instance $(N, g, g^{2^\lambda a})$, computing (g^a), using the 1st-BS-CCA adversary $\mathcal{A}$. $\mathcal{B}$ runs the adversary $\mathcal{A}$ simulating its view as in the 1st-BS-CCA security game as follows:

Setup. $\mathcal{B}$ initializes two empty lists L_{un} and L_{corr}, then maintains it to record all uncorrupted users and corrupted users, respectively, in the game. $\mathcal{B}$ sets $h := g^{2^\lambda}$, then provides $\mathcal{A}$ the public parameter $PP := (N, g, h, H_1, H_2, H_3, H_4)$, where $H_1, H_2, H_3,$ and H_4 are the random oracles simulated by $\mathcal{B}$ as shown for the proof of the 2nd-BS-CCA security (See 4.4.2.1).

Find stage (i.e. Phases 1 and 2). $\mathcal{B}$ simulates the oracles $\mathcal{O}_{unKGen}$, $\mathcal{O}_{corrKGen}$, $\mathcal{O}_{RKGen}$, $\mathcal{O}_{ReEnc}$, $\mathcal{O}_{Dec_1}$, and $\mathcal{O}_{Dec_2}$ to answer the questions issued by $\mathcal{A}$ as follows:

– $\mathcal{O}_{unKGen}(i)$: $\mathcal{B}$ returns $\perp$ if $i \in L_{un} \cup L_{corr}$; otherwise $\mathcal{B}$ chooses randomly $x_i \leftarrow_R [(N-1)/4]$, sets and returns $pk_i := h^{x_i} \cdot g^{-1}$ (meaning that $sk_i = x_i - \frac{1}{2^\lambda} \bmod |\mathbb{QR}_N^+|$). $\mathcal{B}$ adds i to L_{un}. Note that, $\mathcal{B}$ does not need to know sk_i.

– $\mathcal{O}_{corrKGen}(i)$: $\mathcal{B}$ returns $\perp$ if $i \in L_{un} \cup L_{corr}$; otherwise $\mathcal{B}$ chooses randomly $x_i \leftarrow_R [(N-1)/4]$, and returns (pk_i, sk_i), where $pk_i := h^{x_i}$, $sk_i := x_i$. $\mathcal{B}$ adds i to L_{corr}.

– $\mathcal{O}_{RKGen}(i,j)$: if $i \neq j \in L_{un}$ or $i \neq j \in L_{corr}$ $\mathcal{B}$ returns $rk_{ij} := x_j - x_i$; otherwise returns the symbol $\perp$.

– $\mathcal{O}_{ReEnc}(i,j,CT_i)$: $\mathcal{B}$ parses $CT_i = (c_0, c_1, c_2, c_3, s)$, and does as follows:

1. If $i = j$, then return $\perp$.
2. if $i \neq j \in L_{un}$ or $i \neq j \in L_{corr}$ $\mathcal{B}$ computes the re-encryption key $rk_{ij} := x_j - x_i$. Next, $\mathcal{B}$ follows the algorithm $\mathbf{ReEnc}$ to compute the ciphtertext CT_j by using rk_{ij} and simulating the hash oracles H_1, H_3, H_4 as follows:
   (a) Search in the list H_3^{list} to see whether there exists $(a_0, a_1, a_2, a_3, t) \in H_3^{\text{list}}$ such that $(a_0 = c_0) \wedge (a_1 = c_1) \wedge (a_2 = c_2) \wedge (a_3 = c_3)$. If there exists no such tuple, then return $\perp$. (This corresponds to the event $\texttt{ReEncErr}_1$ to be explained).
   (b) Check whether $s^{2^\lambda} = c_0 \cdot c_1^t$ holds. If not, then return $\perp$.
   (c) Search in the list H_1^{list} to see whether there exists $(m, \omega, r) \in H_1^{\text{list}}$ such that $h^r = c_1, (pk_i \cdot g)^r = c_2$. If there exists no such tuple, then return $\perp$. (This corresponds to the event $\texttt{ReEncErr}_2$ to be explained).
   (d) Compute $c_2' = c_2 \cdot c_1^{rk_{ij}}$. Let $\bar{m} := (c_1, c_2', c_3)$.
   (e) Pick $\omega_2 \leftarrow_R \{0,1\}^{\ell_1}, r_1 \leftarrow_R [(N-1)/4]$, and add $(\bar{m}, \omega_2, r_1)$ to the list H_1^{list}.
   (f) Compute $A = h^{r_1}, B = (pk_j \cdot g)^{r_1}$,
   (g) Search in the list H_4^{list} to see whether there exists (f, γ) such that $f = g^{r_1}$. If there exists no such tuple, then pick $\gamma \leftarrow_R \mathbb{QR}_N^+ \times \mathbb{QR}_N^+ \times \{0,1\}^{\ell_0 + 2\ell_1}$, add the tuple (f, γ) to H_4^{list}.
   (h) Compute $C = \gamma \oplus (\bar{m} || \omega_2)$, and output the first-level ciphertext $CT_j = (A, B, C)$.
3. If $i \in L_{corr}, j \in L_{un}$ or $i \in L_{un}, j \in L_{corr}$, then $\mathcal{B}$ does as follows:
   (a) Search in the list H_3^{list} to see whether there exists $(a_0, a_1, a_2, a_3, t) \in H_3^{\text{list}}$ such that $(a_0 = c_0) \wedge (a_1 = c_1) \wedge (a_2 = c_2) \wedge (a_3 = c_3)$. If there exists no such tuple, then return $\perp$.
   (b) Check whether $s^{2^\lambda} = c_0 \cdot c_1^t$ holds. If not, then return $\perp$.
   (c) Search whether there exists a tuple $(m, \omega, r) \in H_1^{\text{list}}$ such that $h^r = c_1, (pk_i \cdot g)^r = c_2$. If there no such tuple, then return $\perp$.
   (d) Compute $c_2' = (pk_j \cdot g)^r)$. Let $\bar{m} := (c_1, c_2', c_3)$.
   (e) Pick $\omega_2 \leftarrow_R \{0,1\}^{\ell_1}, r_1 \leftarrow_R [(N-1)/4]$, and add $(\bar{m}, \omega_2, r_1)$ to the list H_1^{list}.
   (f) Compute $A = h^{r_1}, B = (pk_j \cdot g)^{r_1}$.

(g) Search in the list H_4^{list} to see whether there exists (f, γ) such that $f = g^{r_1}$. If there exists no such tuple, then pick $\gamma \leftarrow_R \mathbb{QR}_N^+ \times \mathbb{QR}_N^+ \times \{0,1\}^{\ell_0+2\ell_1}$, add the tuple (f, γ) to H_4^{list}.

(h) Compute $C = \gamma \oplus (\bar{m}||\omega_2)$, and output the first-level ciphertext $CT_j = (A, B, C)$.

– $\mathcal{O}_{Dec_1}(i, CT_i)$: If $i \in L_{corr}$, then $\mathcal{B}$ does as the algorithm $\mathbf{Dec}_1$ to decrypt the ciphertext CT_i by using $sk_i = x_i$ and hash lists $H_1^{\text{list}}, H_2^{\text{list}}, H_3^{\text{list}}$, and H_4^{list}. Otherwise $\mathcal{B}$ parses CT_i as $CT_i = (A, B, C)$, and does as follows:

1. Search in the list H_1^{list} and H_4^{list} to see whether there exist $(m, \omega, r) \in H_1^{\text{list}}$ and $(f, \gamma) \in H_4^{\text{list}}$ such that

$$h^r = A, (pk_i \cdot g)^r = B, \gamma \oplus (m||\omega) = C, \text{and } f = g^r. \tag{4.2}$$

   If there exists no such tuple, then return $\perp$. (This corresponds to the event DErr_1 to be explained).
2. Parse $\bar{m}$ as $\bar{m} := (c_1, c_2', c_3)$.
3. Search in the list H_1^{list} to see whether there exists $(m, \omega, r) \in H_1^{\text{list}}$ such that $h^r = c_1, (pk_j \cdot g)^r = c_2'$. If there exists no such tuple, then return $\perp$. (This corresponds to the event DErr_2 to be explained).
4. Compute $m||\omega = c_3 \oplus H_2(g^r)$, and returns m if $c_1 = h^{H_1(m,\omega)}$ holds, and $\perp$ otherwise.

– $\mathcal{O}_{Dec_2}(i, CT_i)$: second-level ciphertexts encrypted under public keys from L_{un} can be re-encrypted for corrupted users by using re-encryption oracle $\mathcal{O}_{ReEnc}$. Besides, second-level ciphertexts under pk_{i^*} can be translated for other honest users by using rk_{i^*j} (where $j \in L_{un}$) which can be computed as in simulating of the re-encryption key oracle $\mathcal{O}_{RKGen}$, and the resulting ciphertexts are decrypted by using $\mathcal{O}_{Dec_1}$.

Challenge. When $\mathcal{A}$ decides that Phase 1 is over, a (corrupted or not) public key $pk_{i'}$, a target public key pk_{i^*}, and two "good messages" CT_0, CT_1. If $i^* \in L_{corr}$, then $\mathcal{B}$ outputs $\perp$ and halts, since we require $i^* \in L_{un}$ for $\mathcal{A}$ to win the game. Otherwise, $\mathcal{B}$ flips a random coin $\sigma \in \{0,1\}$ and does as follows:

1. $\mathcal{B}$ parses $CT_\sigma = (c_0, c_1, c_2, c_3, s)$.
2. Search in the list H_3^{list} to see whether there exists $(a_0, a_1, a_2, a_3, t) \in H_3^{\text{list}}$ such that $(a_0 = c_0) \wedge (a_1 = c_1) \wedge (a_2 = c_2) \wedge (a_3 = c_3)$. If there exists no such tuple, then return $\perp$.
3. Check whether $s^{2^\lambda} = c_0 \cdot c_1^t$ holds. If not, then return $\perp$.
4. Search whether there exists a tuple $(m, \omega, r) \in H_1^{\text{list}}$ such that $h^r = c_1, (pk_i \cdot g)^r = c_2$. If there no such tuple, then return $\perp$.
5. Compute $c_2' = (pk_j \cdot g)^r)$. Let $\bar{m} := (c_1, c_2', c_3)$.
6. Pick $C^* \leftarrow_R \{0,1\}^{\ell_0+\ell_1}$.
7. Pick $\omega^* \leftarrow_R \{0,1\}^{\ell_1}$, and implicitly define $H_4(g^a) = C^* \oplus (CT_\sigma||\omega^*)$ and $H_1(CT_\sigma||\omega^*) = a$ (note that $\mathcal{B}$ does not know a and g^a).

8. Define $A^* = g^{2^\lambda a}$ (meaning that $r^* = a$).
9. Compute $B^* = (g^{2^\lambda a})^{x_{i^*}}$.
10. Return $CT^* = (A^*, B^*, C^*)$ as the challenge ciphertext to $\mathcal{A}$.

Observe the following to see that, CT^* is identically distributed as the real one from the construction. Let $r^* := a$. We have

1. $A^* = g^{2^\lambda a} = g^{2^\lambda r^*} = h^{r^*}$.
2. $B^* = (g^{2^\lambda a})^{x_{i^*}} = (g^{2^\lambda x_{i^*}})^a = (g^{2^\lambda sk_{i^*}+1})^{r^*} = (pk_{i^*} \cdot g)^{r^*}$. (Note that $sk_{i^*} = x_{i^*} - \frac{1}{2^\lambda} \bmod |\mathbb{QR}_N^+|$ since $i^* \in L_{un}$.)
3. $C^* = H_4(g^a) \oplus (CT_\sigma || \omega^*) = H_4(g^{r^*}) \oplus (CT_\sigma || \omega^*)$.

Guess. Finally, $\mathcal{A}$ outputs a guess $\sigma' \in \{0, 1\}$.

Output. If $\sigma' = \sigma$, then $\mathcal{B}$ searches (X, β) from the list H_2^{list} such that $X^{2^\lambda} = g^{2^\lambda a}$, and output X as the solution; otherwise $\mathcal{B}$ outputs $\perp$ and halts.

This completes the description of the simulation.

Analysis. Before analyzing the simulation we set $q := |\mathbb{QR}_N^+|$, and assume that $\mathcal{A}$ issues at most $q_{H_1}, q_{H_2}, q_{H_3}, q_{H_4}, q_{re}$, and q_d queries to $H_1, H_2, H_3, H_4, \mathcal{O}_{ReEnc}$, and $\mathcal{O}_{Dec_1}$, respectively.

Let $\mathtt{AskH}_1^*$ be the event that $\mathcal{A}$ queried (CT_σ, ω^*) to H_1, where ω^* is chosen by $\mathcal{B}$ in the challenge phase. It is obvious that the simulation of H_1 is perfect if $\mathtt{AskH}_1^*$ did not occur. Since $\mathcal{B}$ picks $\omega^* \leftarrow_R \{0,1\}^{\ell_1}$ which is hidden by the "one-time pad" given by H_2, we have

$$\Pr[\mathtt{AskH}_1^*] \leq \frac{q_{H_1}}{2^{\ell_1}}.$$

Let $\mathtt{AskH}_4^*$ be the event that $\mathcal{A}$ queried g^a to H_4. The simulation of H_4 is also perfect, as long as $\mathtt{AskH}_4^*$ did not occur. It is clear that the simulations of H_2 and H_3 are perfect.

Next, we evaluate the simulation of the re-encryption oracle. The responses to $\mathcal{A}$'s re-encryption queries are perfect, unless $\mathcal{A}$ can submit valid second-level ciphertexts without querying hash functions H_3 and H_1 (i.e. the events $\mathtt{ReEncErr}_1$ and $\mathtt{ReEncErr}_2$, respectively). Let $\mathtt{ReEncErr}$ be the event that $\mathtt{ReEncErr}_1 \vee \mathtt{ReEncErr}_2$ happens during the entire simulation. Since H_1 and H_3 act as random oracles and $\mathcal{A}$ issues at most q_{re} re-encryption queries, we have

$$\begin{aligned} \Pr[\mathtt{ReEncErr}] &= \Pr[\mathtt{ReEncErr}_1 \vee \mathtt{ReEncErr}_2] \\ &\leq \Pr[\mathtt{ReEncErr}_1] + \Pr[\mathtt{ReEncErr}_2] \leq \frac{q_{re}}{q}. \end{aligned}$$

Now, we analyze the simulation of the decryption oracle. The simulation of the decryption oracle is perfect, with the exception that simulation errors may occur in rejecting some valid ciphertexts. In the similar way of analysis in the proof of the 2nd-BS-CCA security, we obtain

$$\Pr[\mathsf{DecErr}] \leq \frac{q_d(q_{H_4}+1)}{q^2 \cdot 2^{\ell_0+2\ell_1}} + \frac{q_d(q_{H_1}+2)}{q},$$

where DecErr denotes the event that the decryption oracle rejects some valid ciphertexts.

Now, let $\texttt{Error}$ denote the event $\texttt{AskH}_1^* \vee \texttt{AskH}_4^* \vee \texttt{ReEncErr} \vee \mathsf{DecErr}$. If the event $\texttt{Error}$ does not happen, it is clear that $\mathcal{A}$ cannot gain any advantage greater than $\frac{1}{2}$ in guessing σ due to the randomness of the output of the random oracle H_2. Namely, we have $\Pr[\sigma' = \sigma | \neg \texttt{Error}] = \frac{1}{2}$. Following the proof in Apdendix 4.4.2.1, we have

$$\left| \Pr[\sigma' = \sigma] - \frac{1}{2} \right| \leq \frac{1}{2} \Pr[\texttt{Error}].$$

By definition of the advantage for the 1st-BS-CCA adversary, then we have

$$\begin{aligned}
\varepsilon := \mathrm{Adv}^{\text{1st-CCA}}_{\mathrm{PRE},\mathcal{A}}(\lambda) &= \left| \Pr[\sigma' = \sigma] - \frac{1}{2} \right| \\
&\leq \frac{1}{2} \Pr[\texttt{Error}] \\
&= \frac{1}{2} \Pr[\texttt{AskH}_1^* \vee \texttt{AskH}_4^* \vee \texttt{ReEncErr} \vee \mathsf{DecErr}] \\
&\leq \frac{1}{2} \left(\Pr[\texttt{AskH}_1^*] + \Pr[\texttt{AskH}_4^*] + \Pr[\texttt{ReEncErr}] + \Pr[\mathsf{DecErr}] \right).
\end{aligned}$$

Since $\Pr[\texttt{AskH}_1^*] \leq \frac{q_{H_1}}{2^{\ell_1}}, \Pr[\texttt{ReEncErr}] \leq \frac{q_{re}}{q}$, and $\Pr[\mathsf{DecErr}] \leq \frac{q_d(q_{H_4}+1)}{q^2 \cdot 2^{\ell_0+2\ell_1}} + \frac{q_d(q_{H_1}+2)}{q}$, we obtain

$$\begin{aligned}
\Pr[\texttt{AskH}_4^*] &\geq 2\varepsilon - (\Pr[\texttt{AskH}_1^*] + \Pr[\texttt{ReEncErr}] + \Pr[\mathsf{DecErr}]) \\
&\geq 2\varepsilon - \frac{q_{H_1}}{2^{\ell_1}} - \frac{q_{re}}{q} - \frac{q_d(q_{H_4}+1)}{q^2 \cdot 2^{\ell_0+2\ell_1}} - \frac{q_d(q_{H_1}+2)}{q} \\
&= 2\varepsilon - \frac{q_{re} + q_d(q_{H_1}+2)}{q} - \frac{q_{H_1}}{2^{\ell_1}} - \frac{q_d(q_{H_4}+1)}{q^2 \cdot 2^{\ell_0+2\ell_1}}.
\end{aligned}$$

Meanwhile, if event $\texttt{AskH}_4^*$ happens, the algorithm $\mathcal{B}$ will be able to compute g^a, and consequently, we obtain

$$\mathrm{Adv}^{\mathsf{ISQ}}_{\mathcal{B}}(\lambda) = \Pr[\texttt{AskH}_4^*] \geq 2\varepsilon - \frac{q_{re} + q_d(q_{H_1}+2)}{q} - \frac{q_{H_1}}{2^{\ell_1}} - \frac{q_d(q_{H_4}+1)}{q^2 \cdot 2^{\ell_0+2\ell_1}}.$$

From the description of the simulation, the running time of the algorithm $\mathcal{B}$ can be bounded by

$$\begin{aligned}
t' \leq t &+ (q_{H_1} + q_{H_2} + q_{H_3} + q_u + q_c + q_{rk} + q_{re} + q_d)\mathcal{O}(1) \\
&+ (q_u + q_c + (2q_{H_1} + q_{H_4})q_{re} + 5q_{H_1}q_d)t_{exp},
\end{aligned}$$

where t_{exp} denotes the running time of an exponentiation in group $\mathbb{QR}_N^+$.

This completes the proof of the 1st-BS-CCA security of the scheme.

4.4.3 Proof of Theorem 6

We split up the proof of Theorem 5 into two parts: we prove that our scheme is 2nd-US-CCA secure and 1st-US-CCA secure in Section 4.4.3.1 and Section 4.4.3.2, respectively. Combining both parts yields Theorem 6.

4.4.3.1 2nd-US-CCA Security

Since the permutation ISQ (i.e., the permutation : $x \mapsto x^{2^\lambda}$acting on the groups $\mathbb{QR}_N^+$) is one-way function if and only if the factoring problem is hard (Theorem 2), it is sufficient to prove that if there exists an adversary $\mathcal{A}$ that breaks the 2nd-BS-CCA security of the scheme with non-negligible probability, then there exists an adversary $\mathcal{B}$ that breaks the one-wayness of ISQ with overwhelming probability. Let $\mathrm{Adv}_{\mathcal{B}}^{\mathsf{ISQ}}(\lambda)$ denote the advantage of $\mathcal{B}$ in breaking the one-wayness of ISQ.

Since the fact that the FB-Schnorr scheme is one-time strongly unforgeable under the factoring assumption, we can assume that the signature scheme used to sign the challenge ciphertext CT_i^* (i.e. the scheme with the verify/sign keys pair are (c_1^*, g^{u^*})) is one-time strongly unforgeable.

We build an algorithm $\mathcal{B}$ which is, given a factoring instance $(N, g, g^{2^\lambda a})$, computing (g^a), using the 2nd-US-CCA adversary $\mathcal{A}$. $\mathcal{B}$ runs the adversary $\mathcal{A}$ simulating its view as in the 2nd-US-CCA security game as follows:

Setup. $\mathcal{B}$ initializes two empty lists L_{un} and L_{corr}, then maintains it to record all uncorrupted users and corrupted users, respectively, in the game. $\mathcal{B}$ sets $h := g^{2^\lambda}$, then provides $\mathcal{A}$ the public parameter $PP := (N, g, h, H_1, H_2, H_3, H_4)$, where $H_1, H_2, H_3,$ and H_4 are the random oracles simulated by $\mathcal{B}$ as follows.

Hash Oracles Simulation. At any time $\mathcal{A}$ can issue the random oracle queries $H_1, H_2, H_3,$ and H_4. $\mathcal{B}$ maintains four hash lists $H_1^{\text{list}}, H_2^{\text{list}}, H_3^{\text{list}},$ and H_4^{list} which are initially empty, and respond as follow:

- $H_1(m, \omega)$: If there is a tuple (m, ω, r) in H_1^{list} then return r, otherwise choose $r \leftarrow_R [(N-1)/4]$, add the tuple (m, ω, r) to H_1^{list} and return r.
- $H_2(X)$: If there is a tuple (X, β) in H_2^{list} then return β, otherwise choose $\beta \leftarrow_R \{0,1\}^{\ell_0+\ell_1}$, add the tuple (X, β) to H_2^{list} and return β.
- $H_3(c_0, c_1, c_2, c_3)$: If there is a tuple (c_0, c_1, c_2, c_3, t) in H_3^{list} then return t, otherwise choose $t \leftarrow_R \mathbb{Z}_{2^\lambda}^*$, add the tuple (c_0, c_1, c_2, c_3, t) to H_3^{list}, and return t.
- $H_4(f)$: If there is a tuple (f, γ) in H_4^{list} then return γ, otherwise choose $\gamma \leftarrow_R \mathbb{QR}_N^+ \times \mathbb{QR}_N^+ \times \{0,1\}^{\ell_0+\ell_1} \times \mathbb{QR}_N^+ \times \mathbb{QR}_N^+ \times \{0,1\}^{\ell_1}$, add the tuple (f, γ) to H_4^{list} and return γ.

Find stage (i.e. Phases 1 and 2). $\mathcal{B}$ simulates the oracles $\mathcal{O}_{unKGen}$, $\mathcal{O}_{corrKGen}$, $\mathcal{O}_{RKGen}$, $\mathcal{O}_{ReEnc}$, $\mathcal{O}_{Dec_1}$, and $\mathcal{O}_{Dec_2}$ to answer the questions issued by $\mathcal{A}$ as follows:

- $\mathcal{O}_{unKGen}(i)$: $\mathcal{B}$ returns $\perp$ if $i \in L_{un} \cup L_{corr}$; otherwise $\mathcal{B}$ chooses randomly $x_{i,1}, x_{i,2} \leftarrow_R [(N-1)/4]$, sets and returns $pk_i := (h^{x_{i,1}}, h^{x_{i,2}} \cdot g^{-1})$ (meaning that $sk_i = (x_{i,1}, x_{i,2} - \frac{1}{2^\lambda} \bmod |\mathbb{QR}_N^+|)$). $\mathcal{B}$ adds i to L_{un}. Note that, $\mathcal{B}$ does not need to know sk_i.
- $\mathcal{O}_{corrKGen}(i)$: $\mathcal{B}$ returns $\perp$ if $i \in L_{un} \cup L_{corr}$; otherwise $\mathcal{B}$ chooses randomly $x_{i,1}, x_{i,2} \leftarrow_R [(N-1)/4]$, and returns (pk_i, sk_i), where $pk_i := (h^{x_{i,1}}, h^{x_{i,2}})$, $sk_i := (x_{i,1}, x_{i,2})$. $\mathcal{B}$ adds i to L_{corr}.

– $\mathcal{O}_{RKGen}(i,j)$: $\mathcal{B}$ does as follows:
  - If $i = i^*, j \in L_{corr}$: return $\perp$.
  - If $i \in L_{corr}$: $\mathcal{B}$ does exactly the same as the $\mathbf{RKGen}$ algorithm.
    1. Pick $\alpha, r_1 \leftarrow_R [(N-1)/4], \omega_1 \leftarrow_R \{0,1\}^{\ell_1}$, define $r_1 = H_1(\alpha, \omega_1)$, and add (α, ω_1, r_1) to the list H_1^{list}.
    2. Compute $R_1 = \alpha - (sk_{i,1} H_5(pk_{i,2}) + sk_{i,2})$.
    3. Compute $R_2 = h^{r_1}, R_3 = (pk_{j,1} \cdot g)^{r_1}$, and $R_4 = H_2(g^{r_1}) \oplus (\alpha || \omega_1)$.
    4. Return $rk_{i \to j} = (R_1, R_2, R_3, R_4)$.
  - $i \neq i^* \in L_{un}$:
    1. Pick $r_1 \leftarrow_R [(N-1)/4], \omega_1 \leftarrow_R \{0,1\}^{\ell_1}$, and add $(\text{“}-\text{”}, \omega_1, r_1)$ to the list H_1^{list}.
    2. Pick $R_1 \leftarrow_R \{x_{i,1} H_5 pk_{i,2} + x_{i,2}, \ldots, x_{i,1} H_5 pk_{i,2} + x_{i,2} + (N-1)/4\}$ (meaning that $\alpha = R_1 + (sk_{i,1} H_5(pk_{i,2}) + sk_{i,2}) \in [(N-1)/4]$). Note that, $\mathcal{B}$ does not need to know α.
    3. Compute $R_2 = h^{r_1}, R_3 = (pk_{j,1} \cdot g)^{r_1}$, and $R_4 \leftarrow_R \{0,1\}^{\ell_0 + \ell_1}$ (meaning that $H_2(g^{r_1} = R_4 \oplus (\alpha || \omega_1)$, where α is defined as in Step 2).
    4. Return $rk_{i \to j} = (R_1, R_2, R_3, R_4)$.

– $\mathcal{O}_{ReEnc}(i, j, CT_i)$: $\mathcal{B}$ parses $CT_i = (c_0, c_1, c_2, c_3, s)$, and does as follows:
  1. If $i = j$, then return $\perp$.
  2. If $(i = i^*) \wedge (j \in L_{corr}) \wedge (c_1 = c_1^*)$, then return $\perp$. Since c_1^* is the verify key of the one-time strong signature scheme $\mathbf{SIG}$ used to sign the challenge ciphertext CT^*, CT_i is indeed the challenge ciphertext CT^*. Therefore, $\mathcal{B}$ should return $\perp$.
  3. Otherwise $\mathcal{B}$ computes the re-encryption key rk_{ij} as in the oracle $\mathcal{O}_{RKGen}$. Next, $\mathcal{B}$ follows the algorithm $\mathbf{ReEnc}$ to compute the ciphtertext CT_j by using rk_{ij} and simulating the hash oracles H_1, H_3, H_4 as follows:
     (a) Search in the list H_3^{list} to see whether there exists $(a_0, a_1, a_2, a_3, t) \in H_3^{\text{list}}$ such that $(a_0 = c_0) \wedge (a_1 = c_1) \wedge (a_2 = c_2) \wedge (a_3 = c_3)$. If there exists no such tuple, then return $\perp$. (This corresponds to the event $\texttt{ReEncErr}_1$ to be explained).
     (b) Check whether $s^{2^\lambda} = c_0 \cdot c_1^t$ holds. If not, then return $\perp$.
     (c) Search in the list H_1^{list} to see whether there exists $(m, \omega, r) \in H_1^{\text{list}}$ such that $h^r = c_1, (pk_i \cdot g)^r = c_2$. If there exists no such tuple, then return $\perp$. (This corresponds to the event $\texttt{ReEncErr}_2$ to be explained).
     (d) Compute $c_2' = c_2 \cdot c_1^{R_1}$. Let $\bar{m} := (c_1, c_2', c_3, R_2, R_3, R_4)$.
     (e) Pick $\omega_2 \leftarrow_R \{0,1\}^{\ell_1}, r_1 \leftarrow_R [(N-1)/4]$, and add $(\bar{m}, \omega_2, r_1)$ to the list H_1^{list}.
     (f) Compute $A = h^{r_1}, B = (pk_{j,2} \cdot g)^{r_1}$,
     (g) Search in the list H_4^{list} to see whether there exists (f, γ) such that $f = g^{r_1}$. If there exists no such tuple, then pick $\gamma \leftarrow_R \mathbb{QR}_N^+ \times \mathbb{QR}_N^+ \times \{0,1\}^{\ell_0 + \ell_1} \times \mathbb{QR}_N^+ \times \mathbb{QR}_N^+ \times \{0,1\}^{\ell_1}$, add the tuple (f, γ) to H_4^{list}.

(h) Compute $C = \gamma \oplus (\bar{m}||\omega_2)$, and output the first-level ciphertext $CT_j = (A, B, C)$.

– $\mathcal{O}_{Dec_1}(i, CT_i)$: If $i \in L_{corr}$, then $\mathcal{B}$ does as the algorithm $\mathbf{Dec}_1$ to decrypt the ciphertext CT_i by using $sk_i = (x_{i,1}, x_{i,2})$ and hash lists $H_1^{\text{list}}, H_2^{\text{list}}, H_3^{\text{list}}$, and H_4^{list}. Otherwise $\mathcal{B}$ parses CT_i as $CT_i = (A, B, C)$, and does as follows:

1. Search in the list H_1^{list} and H_4^{list} to see whether there exist $(m, \omega, r) \in H_1^{\text{list}}$ and $(f, \gamma) \in H_4^{\text{list}}$ such that

$$h^r = A, (pk_{i,2} \cdot g)^r = B, \gamma \oplus (m||\omega) = C, \text{and } f = g^r. \tag{4.3}$$

   If there exists no such tuple, then return $\perp$. (This corresponds to the event DErr_1 to be explained).
2. Parse $\bar{m}$ as $\bar{m} := (c_1, c_2', c_3, R_2, R_3, R_4)$.
3. Search in the list H_1^{list} to see whether there exists $(m, \omega, r) \in H_1^{\text{list}}$ such that $h^r = c_1, (pk_{j,1} \cdot g)^r = c_2'$. If there exists no such tuple, then return $\perp$. (This corresponds to the event DErr_2 to be explained).
4. Compute $m||\omega = c_3 \oplus H_2(g^r)$, and return m if $c_1 = h^{H_1(m,\omega)}$ holds, and $\perp$ otherwise.

– $\mathcal{O}_{Dec_2}(i, CT_i)$: second-level ciphertexts encrypted under public keys from L_{un} can be re-encrypted for corrupted users by using re-encryption oracle $\mathcal{O}_{ReEnc}$. Besides, second-level ciphertexts under pk_{i^*} can be translated for other honest users by using rk_{i^*j} (where $j \in L_{un}$) which can be computed as in simulating of the re-encryption key oracle $\mathcal{O}_{RKGen}$, and the resulting ciphertexts are decrypted by using $\mathcal{O}_{Dec_1}$.

Challenge. When $\mathcal{A}$ decides that Phase 1 is over, it outputs a target public key pk_{i^*} and two equal-length plaintexts $m_0, m_1 \in \{0,1\}^\lambda$. If $i^* \in L_{corr}$, then $\mathcal{B}$ outputs $\perp$ and halts, since we require $i^* \in L_{un}$ for $\mathcal{A}$ to win the game. Otherwise, $\mathcal{B}$ flips a random coin $\sigma \in \{0,1\}$ and does as follows:

1. Pick $s^* \leftarrow_R \mathbb{QR}_N^+, t^* \leftarrow_R \mathbb{Z}_{2^\lambda}$, and compute $c_0^* = (s^*)^{2^\lambda} \cdot (g^{2^\lambda a})^{-t^*}$.
2. Define $c_1^* = g^{2^\lambda a}$ (meaning that $u^* = a$).
3. Compute $c_2^* = (g^{2^\lambda a})^{x_{i^*,1} H_5(pk_{i^*,2}) + x_{i^*,2}}$.
4. Pick $c_3^* \leftarrow_R \{0,1\}^{\ell_0+\ell_1}$ and define $H_3(c_0^*, c_1^*, c_2^*, c_3^*) = t^*$.
5. Pick $\omega^* \leftarrow_R \{0,1\}^{\ell_1}$, and implicitly define $H_2(g^a) = c_3^* \oplus (m_\sigma||\omega^*)$ and $H_1(m_\sigma||\omega^*) = a$ (note that $\mathcal{B}$ does not know a and g^a).
6. Return $CT^* = (c_0^*, c_1^*, c_2^*, c_3^*, s^*)$ as the challenge ciphertext to $\mathcal{A}$.

Observe the following to see that, CT^* is identically distributed as the real one from the construction. Let $g^{v^*} := s^* \cdot g^{-at^*}$ and $u^* := a$. We have

1. $c_0^* = (s^*)^{2^\lambda} \cdot (g^{2^\lambda a})^{-t^*} = (s^* \cdot g^{-at^*})^{2^\lambda} = g^{2^\lambda v^*} = h^{v^*}$.
2. $c_1^* = g^{2^\lambda a} = g^{2^\lambda u^*} = h^{u^*}$.

3. $c_2^* = (g^{2^\lambda a})^{x_{i^*,1}H_5(pk_{i^*,2})+x_{i^*,2}} = (g^{2^\lambda sk_{i^*}+1})^{u^*} = (pk_{i^*,1}^{H_5(pk_{i^*,2})} pk_{i^*,2} \cdot g)^{u^*}$. (Note that $sk_{i^*,1}H_5(pk_{i^*,2}) + sk_{i^*,2} = x_{i^*,1}H_5(pk_{i^*,2}) + x_{i^*,2} - \frac{1}{2^\lambda} \bmod |\mathbb{QR}_N^+|$ since $i^* \in L_{un}$.)
4. $c_3^* = H_2(g^a) \oplus (m_\sigma || \omega^*) = H_2(g^{u^*}) \oplus (m_\sigma || \omega^*)$.
5. $s^* = s^* \cdot g^{-at^*} \cdot g^{at^*} = g^{v^*+u^*t^*} = g^{v^*+u^*H_3(c_0^*,c_1^*,c_2^*,c_3^*)}$.

Guess. Finally, $\mathcal{A}$ outputs a guess $\sigma' \in \{0,1\}$.

Output. If $\sigma' = \sigma$, then $\mathcal{B}$ searches (X, β) from the list H_2^{list} such that $X^{2^\lambda} = g^{2^\lambda a}$, and output X as the solution; otherwise $\mathcal{B}$ outputs $\perp$ and halts.

This completes the description of the simulation.

Analysis. Before analyzing the simulation we set $q := |\mathbb{QR}_N^+|$, and assume that $\mathcal{A}$ issues at most $q_{H_1}, q_{H_2}, q_{H_3}, q_{H_4}, q_{re}$, and q_d queries to $H_1, H_2, H_3, H_4, \mathcal{O}_{ReEnc}$, and $\mathcal{O}_{Dec_1}$, respectively.

The main idea of the analysis is borrowed from [20]. We first evaluate the simulations of the random oracles. It is clear that the simulation of H_4 is perfect. Let $\texttt{AskH}_1^*$ be the event that $\mathcal{A}$ queried (m_σ, ω^*) to H_1, where ω^* is chosen by $\mathcal{B}$ in the challenge phase. It is obvious that the simulation of H_1 is perfect if $\texttt{AskH}_1^*$ did not occur. Since $\mathcal{B}$ picks $\omega^* \leftarrow_R \{0,1\}^{\ell_1}$ which is hidden by the "one-time pad" given by H_2, we have

$$\Pr[\texttt{AskH}_1^*] \leq \frac{q_{H_1}}{2^{\ell_1}}.$$

Let $\texttt{AskH}_2^*$, and $\texttt{AskH}_3^*$ be the event that $\mathcal{A}$ queried g^a to H_2 and $(c_0^*, c_1^*, c_2^*, c_3^*)$ to H_3, respectively. The simulation of H_2 and H_3 are also perfect, as long as $\texttt{AskH}_2^*$ and $\texttt{AskH}_3^*$ did not occur. Similarly, since $\mathcal{B}$ picks $c_3^* \leftarrow_R \{0,1\}^{\ell_0+\ell_1}$ which is hidden by the "one-time pad" given by H_2, we have

$$\Pr[\texttt{AskH}_3^*] \leq \frac{q_{H_3}}{2^{\ell_0+\ell_1}}.$$

As argued before, the challenged ciphertext provided for $\mathcal{A}$ is identically distributed as the real one from the construction. From the description of the simulation, it can be seen that the responses to $\mathcal{A}$'s re-encryption key queries are also perfect.

Next, we evaluate the simulation of the re-encryption oracle. The responses to $\mathcal{A}$'s re-encryption queries are perfect, unless $\mathcal{A}$ can submit valid second-level ciphertexts without querying hash functions H_3 and H_1 (i.e. the events $\texttt{ReEncErr}_1$ and $\texttt{ReEncErr}_2$, respectively). Let $\texttt{ReEncErr}$ be the event that $\texttt{ReEncErr}_1 \vee \texttt{ReEncErr}_2$ happens during the entire simulation. Since H_1 and H_3 act as random oracles and $\mathcal{A}$ issues at most q_{re} re-encryption queries, we have

$$\begin{aligned} \Pr[\texttt{ReEncErr}] &= \Pr[\texttt{ReEncErr}_1 \vee \texttt{ReEncErr}_2] \\ &\leq \Pr[\texttt{ReEncErr}_1] + \Pr[\texttt{ReEncErr}_2] \leq \frac{q_{re}}{q}. \end{aligned}$$

Now, we analyze the simulation of the decryption oracle. The simulation of the decryption oracle is perfect, with the exception that simulation errors may occur in rejecting some valid ciphertexts. However, these errors are not significant as shown as follows: Suppose that (pk_i, CT_i) has been issued as a valid first-level ciphertext. Even CT_i is valid, there are the three following cases that CT_i can be produced without querying to the random oracles:

- $\mathcal{A}$ did not query (m, ω) to H_1 or f to H_4 such that the equation 4.4 holds (i.e. the event DErr_1 occurs). Let $\mathtt{Valid}$ be the event that CT_1 is valid. Let $\mathtt{AskH}_1$and $\mathtt{AskH}_4$ respectively be events that (m, ω) has been queried to H_1 and f has been queried to H_4 in the event DecErr_1. We have

$$\begin{aligned}\Pr[\mathtt{Valid}|\mathsf{DErr}_1] &= \Pr[\mathtt{Valid}|(\neg\mathtt{AskH}_1 \vee \neg\mathtt{AskH}_4)] \\ &\leq \Pr[\mathtt{Valid}|\neg\mathtt{AskH}_1] + \Pr[\mathtt{Valid}|\neg\mathtt{AskH}_4].\end{aligned}$$

  On the other hand, observe that

$$\begin{aligned}\Pr[\mathtt{Valid}|\neg\mathtt{AskH}_1] &= \Pr[\mathtt{Valid} \wedge \mathtt{AskH}_4|\neg\mathtt{AskH}_1] + \Pr[\mathtt{Valid} \wedge \neg\mathtt{AskH}_4|\neg\mathtt{AskH}_1] \\ &\leq \Pr[\mathtt{AskH}_4|\neg\mathtt{AskH}_1] + \Pr[\mathtt{Valid}|(\neg\mathtt{AskH}_1 \wedge \neg\mathtt{AskH}_4)] \\ &\leq \frac{q_{H_4}}{q^4 \cdot 2^{\ell_0+2\ell_1}} + \frac{1}{q}.\end{aligned}$$

  Similarly, we have

$$\Pr[\mathtt{Valid}|\neg\mathtt{AskH}_4] \leq \frac{q_{H_1}}{q} + \frac{1}{q^4 \cdot 2^{\ell_0+2\ell_1}}.$$

  Therefore, we have

$$\begin{aligned}\Pr[\mathtt{Valid}|\mathsf{DErr}_1] &\leq \Pr[\mathtt{Valid}|\neg\mathtt{AskH}_1] + \Pr[\mathtt{Valid}|\neg\mathtt{AskH}_4] \\ &\leq \frac{q_{H_4}+1}{q^4 \cdot 2^{\ell_0+2\ell_1}} + \frac{q_{H_1}+1}{q}.\end{aligned}$$

  Let DecErr_1 be the event that $\mathtt{Valid}|\mathsf{DErr}_1$ happens during the entire simulation. Then, since at most q_d decryption oracles are issued, we have

$$\Pr[\mathsf{DecErr}_1] \leq q_d \left(\frac{q_{H_4}+1}{q^4 \cdot 2^{\ell_0+2\ell_1}} + \frac{q_{H_1}+1}{q} \right).$$

- $\mathcal{A}$ did not query (m, ω) to H_1 such that $h^{H_1(m,\omega)} = c_1$ and $(pk_j \cdot g)^{H_1(m,\omega)} = c_2'$ hold (i.e. the event DErr_2 occurs). It is clear that $\Pr[\mathtt{Valid}|\mathsf{DErr}_2] = \frac{1}{q}$. Let DecErr_2 be the event that $\mathtt{Valid}|\mathsf{DErr}_2$ happens during the entire simulation. We have

$$\Pr[\mathsf{DecErr}_2] \leq \frac{q_d}{q}.$$

Let DecErr be the event that $\mathtt{Valid}|\mathsf{DErr}_1 \vee \mathsf{DErr}_2$ happens during the entire simulation. Then, we have

$$\begin{aligned}\Pr[\mathsf{DecErr}] &= \Pr[\mathtt{Valid}|\mathsf{DErr}_1 \vee \mathsf{DErr}_2] \\ &\leq \Pr[\mathtt{Valid}|\mathsf{DErr}_1] + \Pr[\mathtt{Valid}|\mathsf{DErr}_2] \\ &= \Pr[\mathsf{DecErr}_1] + \Pr[\mathsf{DecErr}_2] \\ &\leq \frac{q_d(q_{H_4}+1)}{q^4 \cdot 2^{\ell_0+2\ell_1}} + \frac{q_d(q_{H_1}+2)}{q}.\end{aligned}$$

Now, let $\texttt{Error}$ denote the event $\texttt{AskH}_1^* \vee \texttt{AskH}_2^* \vee \texttt{AskH}_3^* \vee \texttt{ReEncErr} \vee \mathsf{DecErr}$. If the event $\texttt{Error}$ does not happen, it is clear that $\mathcal{A}$ can not gain any advantage greater than $\frac{1}{2}$ in guessing σ due to the randomness of the output of the random oracle H_2. Namely, we have $\Pr[\sigma' = \sigma | \neg\texttt{Error}] = \frac{1}{2}$. Hence, by splitting $\Pr[\sigma' = \sigma]$, we have

$$\begin{aligned}\Pr[\sigma' = \sigma] &= \Pr[\sigma' = \sigma | \neg\texttt{Error}] \Pr[\neg\texttt{Error}] + \Pr[\sigma' = \sigma | \texttt{Error}] \Pr[\texttt{Error}] \\ &\leq \frac{1}{2} \Pr[\neg\texttt{Error}] + \Pr[\texttt{Error}] \\ &= \frac{1}{2} + \frac{1}{2} \Pr[\texttt{Error}],\end{aligned}$$

and

$$\Pr[\sigma' = \sigma] \geq \Pr[\sigma' = \sigma | \neg\texttt{Error}] \Pr[\neg\texttt{Error}] = \frac{1}{2} - \frac{1}{2} \Pr[\texttt{Error}].$$

Then we have

$$\left| \Pr[\sigma' = \sigma] - \frac{1}{2} \right| \leq \frac{1}{2} \Pr[\texttt{Error}].$$

By definition of the advantage for the 2nd-BS-CCA adversary, we then have

$$\begin{aligned}\varepsilon := \mathrm{Adv}^{\text{2nd-CCA}}_{\mathrm{PRE},\mathcal{A}}(\lambda) &= \left| \Pr[\sigma' = \sigma] - \frac{1}{2} \right| \\ &\leq \frac{1}{2} \Pr[\texttt{Error}] \\ &= \frac{1}{2} \Pr[\texttt{AskH}_1^* \vee \texttt{AskH}_2^* \vee \texttt{AskH}_3^* \vee \texttt{ReEncErr} \vee \mathsf{DecErr}] \\ &\leq \frac{1}{2} \left(\Pr[\texttt{AskH}_1^*] + \Pr[\texttt{AskH}_2^*] + \Pr[\texttt{AskH}_3^*] + \Pr[\texttt{ReEncErr}] + \Pr[\mathsf{DecErr}] \right).\end{aligned}$$

Since $\Pr[\texttt{AskH}_1^*] \leq \frac{q_{H_1}}{2^{\ell_1}}, \Pr[\texttt{AskH}_3^*] \leq \frac{q_{H_3}}{2^{\ell_0+\ell_1}}, \Pr[\texttt{ReEncErr}] \leq \frac{q_{re}}{q}$, and $\Pr[\mathsf{DecErr}] \leq \frac{q_d(q_{H_4}+1)}{q^4 \cdot 2^{\ell_0+2\ell_1}} + \frac{q_d(q_{H_1}+2)}{q}$, we obtain

$$\begin{aligned}\Pr[\texttt{AskH}_2^*] &\geq 2\varepsilon - \left(\Pr[\texttt{AskH}_1^*] + \Pr[\texttt{AskH}_3^*] + \Pr[\texttt{ReEncErr}] + \Pr[\mathsf{DecErr}] \right) \\ &\geq 2\varepsilon - \frac{q_{H_1}}{2^{\ell_1}} - \frac{q_{H_3}}{2^{\ell_0+\ell_1}} - \frac{q_{re}}{q} - \frac{q_d(q_{H_4}+1)}{q^2 \cdot 2^{\ell_0+2\ell_1}} - \frac{q_d(q_{H_1}+2)}{q} \\ &= 2\varepsilon - \frac{q_{re} + q_d(q_{H_1}+2)}{q} - \frac{q_{H_1}}{2^{\ell_1}} - \frac{q_{H_3}}{2^{\ell_0+\ell_1}} - \frac{q_d(q_{H_4}+1)}{q^4 \cdot 2^{\ell_0+2\ell_1}}.\end{aligned}$$

Meanwhile, if event $\texttt{AskH}_2^*$ happens, the algorithm $\mathcal{B}$ will be able to compute g^a, and consequently, we obtain

$$\mathrm{Adv}^{\mathsf{ISQ}}_{\mathcal{B}}(\lambda) = \Pr[\texttt{AskH}_2^*] \geq 2\varepsilon - \frac{q_{re} + q_d(q_{H_1}+2)}{q} - \frac{q_{H_1}}{2^{\ell_1}} - \frac{q_{H_3}}{2^{\ell_0+\ell_1}} - \frac{q_d(q_{H_4}+1)}{q^4 \cdot 2^{\ell_0+2\ell_1}}.$$

From the description of the simulation, the running time of the algorithm $\mathcal{B}$ can be bounded by

$$\begin{aligned}t' \leq t &+ (q_{H_1} + q_{H_2} + q_{H_3} + q_u + q_c + q_{rk} + q_{re} + q_d)\mathcal{O}(1) \\ &+ (q_u + q_c + (2q_{H_1} + q_{H_4})q_{re} + 5q_{H_1}q_d)t_{exp},\end{aligned}$$

where t_{exp} denotes the running time of an exponentiation in group $\mathbb{QR}_N^+$.

This completes the proof of the 2nd-US-CCA security of the scheme.

4.4.3.2 1st-US-CCA Security

We build an algorithm $\mathcal{B}$ which is, given a factoring instance $(N, g, g^{2^\lambda a})$, computing (g^a), using the 1st-US-CCA adversary $\mathcal{A}$. $\mathcal{B}$ runs the adversary $\mathcal{A}$ simulating its view as in the 1st-US-CCA security game as follows:

Setup. $\mathcal{B}$ initializes two empty lists L_{un} and L_{corr}, then maintains it to record all uncorrupted users and corrupted users, respectively, in the game. $\mathcal{B}$ sets $h := g^{2^\lambda}$, then provides $\mathcal{A}$ the public parameter $PP := (N, g, h, H_1, H_2, H_3, H_4)$, where $H_1, H_2, H_3,$ and H_4 are the random oracles simulated by $\mathcal{B}$ as shown in the proof of the 2nd-US-CCA security (See 4.4.3.1).

Find stage (i.e. Phases 1 and 2). $\mathcal{B}$ simulates the oracles $\mathcal{O}_{unKGen}$, $\mathcal{O}_{corrKGen}$, $\mathcal{O}_{RKGen}$, $\mathcal{O}_{ReEnc}$, $\mathcal{O}_{Dec_1}$, and $\mathcal{O}_{Dec_2}$ to answer the questions issued by $\mathcal{A}$ as follows:

- $\mathcal{O}_{unKGen}(i)$: $\mathcal{B}$ returns $\perp$ if $i \in L_{un} \cup L_{corr}$; otherwise $\mathcal{B}$ chooses randomly $x_{i,1}, x_{i,2} \leftarrow_R [(N-1)/4]$, sets and returns $pk_i := (h^{x_{i,1}}, h^{x_{i,2}} \cdot g^{-1})$ (meaning that $sk_i = (x_{i,1}, x_{i,2} - \frac{1}{2^\lambda} \bmod |\mathbb{QR}_N^+|)$). $\mathcal{B}$ adds i to L_{un}. Note that, $\mathcal{B}$ does not need to know sk_i.
- $\mathcal{O}_{corrKGen}(i)$: $\mathcal{B}$ returns $\perp$ if $i \in L_{un} \cup L_{corr}$; otherwise $\mathcal{B}$ chooses randomly $x_{i,1}, x_{i,2} \leftarrow_R [(N-1)/4]$, and returns (pk_i, sk_i), where $pk_i := (h^{x_{i,1}}, h^{x_{i,2}})$, $sk_i := (x_{i,1}, x_{i,2})$. $\mathcal{B}$ adds i to L_{corr}.
- $\mathcal{O}_{RKGen}(i, j)$: $\mathcal{B}$ does as follows:
  - If $i \in L_{corr}$: $\mathcal{B}$ does exactly the same as the $\mathbf{RKGen}$ algorithm.
    1. Pick $\alpha, r_1 \leftarrow_R [(N-1)/4], \omega_1 \leftarrow_R \{0,1\}^{\ell_1}$, define $r_1 = H_1(\alpha, \omega_1)$, and add (α, ω_1, r_1) to the list H_1^{list}.
    2. Compute $R_1 = \alpha - (sk_{i,1} H_5(pk_{i,2}) + sk_{i,2})$.
    3. Compute $R_2 = h^{r_1}, R_3 = (pk_{j,1} \cdot g)^{r_1}$, and $R_4 = H_2(g^{r_1}) \oplus (\alpha || \omega_1)$.
    4. Return $rk_{i \to j} = (R_1, R_2, R_3, R_4)$.
  - $i \in L_{un}$:
    1. Pick $r_1 \leftarrow_R [(N-1)/4], \omega_1 \leftarrow_R \{0,1\}^{\ell_1}$, and add $(\text{“}-\text{”}, \omega_1, r_1)$ to the list H_1^{list}.
    2. Pick $R_1 \leftarrow_R \{x_{i,1} H_5 pk_{i,2} + x_{i,2}, \ldots, x_{i,1} H_5 pk_{i,2} + x_{i,2} + (N-1)/4\}$ (meaning that $\alpha = R_1 + (sk_{i,1} H_5(pk_{i,2}) + sk_{i,2}) \in [(N-1)/4]$). Note that, $\mathcal{B}$ does not need to know α.
    3. Compute $R_2 = h^{r_1}, R_3 = (pk_{j,1} \cdot g)^{r_1}$, and $R_4 \leftarrow_R \{0,1\}^{\ell_0 + \ell_1}$ (meaning that $H_2(g^{r_1} = R_4 \oplus (\alpha || \omega_1)$, where α is defined as in Step 2).
    4. Return $rk_{i \to j} = (R_1, R_2, R_3, R_4)$.
- $\mathcal{O}_{ReEnc}(i, j, CT_i)$: $\mathcal{B}$ parses $CT_i = (c_0, c_1, c_2, c_3, s)$, and does as follows:
  1. If $i = j$, then return $\perp$.
  2. If $(i = i^*) \wedge (j \in L_{corr}) \wedge (c_1 = c_1^*)$, then return $\perp$. Since c_1^* is the verify key of the one-time strong signature scheme $\mathbf{SIG}$ used to sign the challenge ciphertext CT^*, CT_i is indeed the challenge ciphertext CT^*. Therefore, $\mathcal{B}$ should return $\perp$.

3. Otherwise $\mathcal{B}$ computes the re-encryption key rk_{ij} as in the oracle $\mathcal{O}_{RKGen}$. Next, $\mathcal{B}$ follows the algorithm $\mathbf{ReEnc}$ to compute the ciphtertext CT_j by using rk_{ij} and simulating the hash oracles H_1, H_3, H_4 as follows:
   (a) Search in the list H_3^{list} to see whether there exists $(a_0, a_1, a_2, a_3, t) \in H_3^{\text{list}}$ such that $(a_0 = c_0) \wedge (a_1 = c_1) \wedge (a_2 = c_2) \wedge (a_3 = c_3)$. If there exists no such tuple, then return $\perp$. (This corresponds to the event $\texttt{ReEncErr}_1$ to be explained).
   (b) Check whether $s^{2^\lambda} = c_0 \cdot c_1^t$ holds. If not, then return $\perp$.
   (c) Search in the list H_1^{list} to see whether there exists $(m, \omega, r) \in H_1^{\text{list}}$ such that $h^r = c_1, (pk_i \cdot g)^r = c_2$. If there exists no such tuple, then return $\perp$. (This corresponds to the event $\texttt{ReEncErr}_2$ to be explained).
   (d) Compute $c_2' = c_2 \cdot c_1^{R_1}$. Let $\bar{m} := (c_1, c_2', c_3, R_2, R_3, R_4)$.
   (e) Pick $\omega_2 \leftarrow_R \{0,1\}^{\ell_1}, r_1 \leftarrow_R [(N-1)/4]$, and add $(\bar{m}, \omega_2, r_1)$ to the list H_1^{list}.
   (f) Compute $A = h^{r_1}, B = (pk_{j,2} \cdot g)^{r_1}$,
   (g) Search in the list H_4^{list} to see whether there exists (f, γ) such that $f = g^{r_1}$. If there exists no such tuple, then pick $\gamma \leftarrow_R \mathbb{QR}_N^+ \times \mathbb{QR}_N^+ \times \{0,1\}^{\ell_0+\ell_1} \times \mathbb{QR}_N^+ \times \mathbb{QR}_N^+ \times \{0,1\}^{\ell_1}$, add the tuple (f, γ) to H_4^{list}.
   (h) Compute $C = \gamma \oplus (\bar{m}||\omega_2)$, and output the first-level ciphertext $CT_j = (A, B, C)$.

– $\mathcal{O}_{Dec_1}(i, CT_i)$: If $i \in L_{corr}$, then $\mathcal{B}$ does as the algorithm $\mathbf{Dec}_1$ to decrypt the ciphertext CT_i by using $sk_i = (x_{i,1}, x_{i,2})$ and hash lists $H_1^{\text{list}}, H_2^{\text{list}}, H_3^{\text{list}}$, and H_4^{list}. Otherwise $\mathcal{B}$ parses CT_i as $CT_i = (A, B, C)$, and does as follows:
   1. Search in the list H_1^{list} and H_4^{list} to see whether there exist $(m, \omega, r) \in H_1^{\text{list}}$ and $(f, \gamma) \in H_4^{\text{list}}$ such that
   $$h^r = A, (pk_{i,2} \cdot g)^r = B, \gamma \oplus (m||\omega) = C, \text{and } f = g^r. \tag{4.4}$$
   If there exists no such tuple, then return $\perp$. (This corresponds to the event DErr_1 to be explained).
   2. Parse $\bar{m}$ as $\bar{m} := (c_1, c_2', c_3, R_2, R_3, R_4)$.
   3. Search in the list H_1^{list} to see whether there exists $(m, \omega, r) \in H_1^{\text{list}}$ such that $h^r = c_1, (pk_{j,1} \cdot g)^r = c_2'$. If there exists no such tuple, then return $\perp$. (This corresponds to the event DErr_2 to be explained).
   4. Compute $m||\omega = c_3 \oplus H_2(g^r)$, and returns m if $c_1 = h^{H_1(m,\omega)}$ holds, and $\perp$ otherwise.

– $\mathcal{O}_{Dec_2}(i, CT_i)$: second-level ciphertexts encrypted under public keys from L_{un} can be re-encrypted for corrupted users by using re-encryption oracle $\mathcal{O}_{ReEnc}$. Besides, second-level ciphertexts under pk_{i^*} can be translated for other honest users by using rk_{i^*j} (where $j \in L_{un}$) which can be computed as in simulating of the re-encryption key oracle $\mathcal{O}_{RKGen}$, and the resulting ciphertexts are decrypted by using $\mathcal{O}_{Dec_1}$.

Challenge. When $\mathcal{A}$ decides that Phase 1 is over, a (corrupted or not) public key $pk_{i'}$, a target public key pk_{i^*}, and two "good messages" CT_0, CT_1. If $i^* \in L_{corr}$, then $\mathcal{B}$ outputs $\perp$ and halts, since we require $i^* \in L_{un}$ for $\mathcal{A}$ to win the game. Otherwise, $\mathcal{B}$ flips a random coin $\sigma \in \{0,1\}$ and does as follows:

1. $\mathcal{B}$ computes re-encryption key $rk_{i' \to i^*} = (R_1, R_2, R_3, R_4)$ as in the $\mathcal{O}_{RKGen}$ algorithm.
2. $\mathcal{B}$ parses $CT_\sigma = (c_0, c_1, c_2, c_3, s)$.
3. Search in the list H_3^{list} to see whether there exists $(a_0, a_1, a_2, a_3, t) \in H_3^{\text{list}}$ such that $(a_0 = c_0) \wedge (a_1 = c_1) \wedge (a_2 = c_2) \wedge (a_3 = c_3)$. If there exists no such tuple, then return $\perp$.
4. Check whether $s^{2^\lambda} = c_0 \cdot c_1^t$ holds. If not, then return $\perp$.
5. Search whether there exists a tuple $(m, \omega, r) \in H_1^{\text{list}}$ such that $h^r = c_1, (pk_{i',1} \cdot g)^r = c_2$. If there no such tuple, then return $\perp$.
6. Otherwise, compute $c_2' = c_2 \cdot c_1^{R_1}$. Let $\bar{m} := (c_1, c_2', c_3, R_2, R_3, R_4)$.
7. Pick $C^* \leftarrow_R \{0,1\}^{\ell_0 + \ell_1}$.
8. Pick $\omega^* \leftarrow_R \{0,1\}^{\ell_1}$, and implicitly define $H_4(g^a) = C^* \oplus (CT_\sigma || \omega^*)$ and $H_1(CT_\sigma || \omega^*) = a$ (note that $\mathcal{B}$ does not know a and g^a).
9. Define $A^* = g^{2^\lambda a}$ (meaning that $r^* = a$).
10. Compute $B^* = (g^{2^\lambda a})^{x_{i^*,2}}$.
11. Return $CT^* = (A^*, B^*, C^*)$ as the challenge ciphertext to $\mathcal{A}$.

Observe the following to see that, CT^* is identically distributed as the real one from the construction. Let $r^* := a$. We have

1. $A^* = g^{2^\lambda a} = g^{2^\lambda r^*} = h^{r^*}$.
2. $B^* = (g^{2^\lambda a})^{x_{i^*,2}} = (g^{2^\lambda x_{i^*,2}})^a = (g^{2^\lambda sk_{i^*,2} + 1})^{r^*} = (pk_{i^*,2} \cdot g)^{r^*}$. (Note that $sk_{i^*,2} = x_{i^*,2} - \frac{1}{2^\lambda} \bmod |\mathbb{QR}_N^+|$ since $i^* \in L_{un}$.)
3. $C^* = H_4(g^a) \oplus (CT_\sigma || \omega^*) = H_4(g^{r^*}) \oplus (CT_\sigma || \omega^*)$.

Guess. Finally, $\mathcal{A}$ outputs a guess $\sigma' \in \{0,1\}$.

Output. If $\sigma' = \sigma$, then $\mathcal{B}$ searches (X, β) from the list H_2^{list} such that $X^{2^\lambda} = g^{2^\lambda a}$, and output X as the solution; otherwise $\mathcal{B}$ outputs $\perp$ and halts.

This completes the description of the simulation.

Analysis. Before analyzing the simulation we set $q := |\mathbb{QR}_N^+|$, and assume that $\mathcal{A}$ issues at most $q_{H_1}, q_{H_2}, q_{H_3}, q_{H_4}, q_{re}$, and q_d queries to $H_1, H_2, H_3, H_4, \mathcal{O}_{ReEnc}$, and $\mathcal{O}_{Dec_1}$, respectively.

Let $\texttt{AskH}_1^*$ be the event that $\mathcal{A}$ queried (CT_σ, ω^*) to H_1, where ω^* is chosen by $\mathcal{B}$ in the challenge phase. It is obvious that the simulation of H_1 is perfect if $\texttt{AskH}_1^*$ did not occur. Since $\mathcal{B}$ picks $\omega^* \leftarrow_R \{0,1\}^{\ell_1}$ which is hidden by the "one-time pad" given by H_2, we have

$$\Pr[\texttt{AskH}_1^*] \leq \frac{q_{H_1}}{2^{\ell_1}}.$$

Let $\mathtt{AskH}_4^*$ be the event that $\mathcal{A}$ queried g^a to H_4. The simulation of H_4 is also perfect, as long as $\mathtt{AskH}_4^*$ did not occur. It is clear that the simulations of H_2 and H_3 are perfect.

Next, we evaluate the simulation of the re-encryption oracle. The responses to $\mathcal{A}$'s re-encryption queries are perfect, unless $\mathcal{A}$ can submit valid second-level ciphertexts without querying hash functions H_3 and H_1 (i.e. the events $\mathtt{ReEncErr}_1$ and $\mathtt{ReEncErr}_2$, respectively). Let $\mathtt{ReEncErr}$ be the event that $\mathtt{ReEncErr}_1 \vee \mathtt{ReEncErr}_2$ happens during the entire simulation. Since H_1 and H_3 act as random oracles and $\mathcal{A}$ issues at most q_{re} re-encryption queries, we have

$$\begin{aligned}\Pr[\mathtt{ReEncErr}] &= \Pr[\mathtt{ReEncErr}_1 \vee \mathtt{ReEncErr}_2] \\ &\leq \Pr[\mathtt{ReEncErr}_1] + \Pr[\mathtt{ReEncErr}_2] \leq \frac{q_{re}}{q}.\end{aligned}$$

Now, we analyze the simulation of the decryption oracle. The simulation of the decryption oracle is perfect, with the exception that simulation errors may occur in rejecting some valid ciphertexts. In the similar way of analysis in the proof of the 2nd-US-CCA security, we obtain

$$\Pr[\mathsf{DecErr}] \leq \frac{q_d(q_{H_4}+1)}{q^4 \cdot 2^{\ell_0+2\ell_1}} + \frac{q_d(q_{H_1}+2)}{q},$$

where DecErr denotes the event that the decryption oracle rejects some valid ciphertexts.

Now, let $\mathtt{Error}$ denote the event $\mathtt{AskH}_1^* \vee \mathtt{AskH}_4^* \vee \mathtt{ReEncErr} \vee \mathsf{DecErr}$. If the event $\mathtt{Error}$ does not happen, it is clear that $\mathcal{A}$ can not gain any advantage greater than $\frac{1}{2}$ in guessing σ due to the randomness of the output of the random oracle H_2. Namely, we have $\Pr[\sigma' = \sigma | \neg \mathtt{Error}] = \frac{1}{2}$. Following the proof in Section 4.4.2.1, we have

$$\left| \Pr[\sigma' = \sigma] - \frac{1}{2} \right| \leq \frac{1}{2} \Pr[\mathtt{Error}].$$

By definition of the advantage for the 1st-BS-CCA adversary, we then have

$$\begin{aligned}\varepsilon := \mathrm{Adv}^{\text{1st-CCA}}_{\mathrm{PRE},\mathcal{A}}(\lambda) &= \left| \Pr[\sigma' = \sigma] - \frac{1}{2} \right| \\ &\leq \frac{1}{2} \Pr[\mathtt{Error}] \\ &= \frac{1}{2} \Pr[\mathtt{AskH}_1^* \vee \mathtt{AskH}_4^* \vee \mathtt{ReEncErr} \vee \mathsf{DecErr}] \\ &\leq \frac{1}{2} \left(\Pr[\mathtt{AskH}_1^*] + \Pr[\mathtt{AskH}_4^*] + \Pr[\mathtt{ReEncErr}] + \Pr[\mathsf{DecErr}] \right).\end{aligned}$$

Since $\Pr[\mathtt{AskH}_1^*] \leq \frac{q_{H_1}}{2^{\ell_1}}, \Pr[\mathtt{ReEncErr}] \leq \frac{q_{re}}{q}$, and $\Pr[\mathsf{DecErr}] \leq \frac{q_d(q_{H_4}+1)}{q^4 \cdot 2^{\ell_0+2\ell_1}} + \frac{q_d(q_{H_1}+2)}{q}$, we obtain

$$\begin{aligned}\Pr[\mathtt{AskH}_4^*] &\geq 2\varepsilon - \left(\Pr[\mathtt{AskH}_1^*] + \Pr[\mathtt{ReEncErr}] + \Pr[\mathsf{DecErr}] \right) \\ &\geq 2\varepsilon - \frac{q_{H_1}}{2^{\ell_1}} - \frac{q_{re}}{q} - \frac{q_d(q_{H_4}+1)}{q^4 \cdot 2^{\ell_0+2\ell_1}} - \frac{q_d(q_{H_1}+2)}{q} \\ &= 2\varepsilon - \frac{q_{re} + q_d(q_{H_1}+2)}{q} - \frac{q_{H_1}}{2^{\ell_1}} - \frac{q_d(q_{H_4}+1)}{q^4 \cdot 2^{\ell_0+2\ell_1}}.\end{aligned}$$

Meanwhile, if event $\mathtt{AskH}_4^*$ happens, the algorithm $\mathcal{B}$ will be able to compute g^a, and consequently, we obtain

$$\mathrm{Adv}_{\mathcal{B}}^{\mathsf{ISQ}}(\lambda) = \Pr[\mathtt{AskH}_4^*] \;\geq\; 2\varepsilon - \frac{q_{re} + q_d(q_{H_1}+2)}{q} - \frac{q_{H_1}}{2^{\ell_1}} - \frac{q_d(q_{H_4}+1)}{q^4 \cdot 2^{\ell_0+2\ell_1}}.$$

From the description of the simulation, the running time of the algorithm $\mathcal{B}$ can be bounded by

$$\begin{aligned} t' \leq t \;&+\; (q_{H_1} + q_{H_2} + q_{H_3} + q_u + q_c + q_{rk} + q_{re} + q_d)\mathcal{O}(1) \\ &+\; (q_u + q_c + (2q_{H_1} + q_{H_4})q_{re} + 5q_{H_1}q_d)t_{exp}, \end{aligned}$$

where t_{exp} denotes the running time of an exponentiation in group $\mathbb{QR}_N^+$.

This completes the proof of the 1st-US-CCA security of the scheme.

4.5 Summary

In this chapter, we have proposed the first factoring-based PRE schemes. In particular, we have presented three PRE schemes. The first is a CPA-secure bidirectional and multi-hop PRE scheme. The second is a CCA-secure bidirectional and single-hop PRE scheme. The last is a CCA-secure unidirectional and single-hop PRE scheme extended from the second by using the "token-controlled encryption" technique. The former is in the standard model, and the others are in the random oracle model. In order to construct the second and the third schemes, we have proposed a new factoring-based strong signature scheme which is a variant of the Schnorr signature scheme.

Chapter 5

A Scheme with Bilinear Maps

Contents

5.1 Introduction . . . 73
5.2 Analysis of the Shao-Liu-Zhou Scheme . . . 74
5.2.1 Review of the Shao-Liu-Zhou Scheme . . . 74
5.2.2 Our Attack . . . 76
5.3 The Proposed Scheme . . . 77
5.3.1 Construction . . . 77
5.3.2 Security Analysis . . . 78
5.4 On the Hardness of the 6-AmDBDH Problem . . . 82
5.4.1 The DBDH Problem and Its Variant . . . 82
5.4.2 Equivalent Definitions of the 6-AmDBDH Problem . . . 82
5.4.3 The General Diffie-Hellman Problem . . . 83
5.4.4 Complexity Lower Bounds in Generic Bilinear Groups . . . 83
5.5 Proofs . . . 84
5.5.1 Proof of Theorem 7 . . . 84
5.5.2 Proof of Theorem 8 . . . 88
5.6 Summary . . . 91

5.1 Introduction

After the first PRE scheme proposed by Blaze et al. [8] which is secure against chosen-plaintext attack, there have been a lot of works trying to achieve chosen-ciphertext (CCA) security for PRE. We mention [20, 17, 14, 32] achieving CCA security in the random oracle model. As we have seen in the previous sections, it seems very difficult to achieve PRE schemes with the CCA security in the standard model (i.e. without the random oracle idealization). In 2011, Shao, Liu, and Zhou [55] proposed a construction of PRE and claimed that their scheme achieves key privacy without losing the CCA security in the standard model, which is an open problem left by Ateniese

et al. [2]. In this chapter, however, we will show that their scheme is vulnerable to the chosen-ciphertext attack, and present a concrete attack to break the CCA security of it (see Section 5.2 for more details).

In 2012, Hanaoka, Kawai, Kunihiro, Matsuda, Weng, Zhang, and Zhao [25] proposed a CCA security definition for PRE, and claimed that it is stronger than all the previous works. They also presented a unidirectional single-hop scheme which is secure in their security model. As we have shown in Section 3.3.3, their model can be further strengthened by presenting a new CCA security definition which is extended from theirs. In this chapter, we propose the first scheme with this security in the standard model. Our scheme is efficient and relies on mild complexity assumptions in bilinear groups.

5.2 Analysis of the Shao-Liu-Zhou Scheme

Shao, Liu, and Zhou [55] proposed a construction of PRE and claimed that their scheme achieves key privacy without losing CCA security, which is an open problem left by Ateniese et al. [2]. Unfortunately, this scheme is actually not CCA secure. In this section, we present a concrete attack to break its CCA security, where the CCA security is in the sense of our definition. This attack can also be applied in their CCA security model or that of other previous works as [54, 17, 14].

5.2.1 Review of the Shao-Liu-Zhou Scheme

Now we describe the Shao-Liu-Zhou scheme in [55].

PRE.Setup: The system parameters are: $(g, h, h', g_2, g_3, P_1, P_2, p, e, \mathbb{G}, \mathbb{G}_T, SIG, SKE, F(\cdot), \dot{F}(\cdot), \times\bar{F}(\cdot), H_i(\cdot)(i = 1, 2, 3), \bar{H}_j(\cdot)(j = 1, 2))$, where $(g, p, \mathbb{G}, \mathbb{G}_T, e) \leftarrow BSetup(1^\lambda)$, (h, h', g_2, g_3) are random numbers from $\mathbb{G}$, P_1 and P_2 are random numbers from $\mathbb{G}_T$, $SIG = (G, S, V)$ is a strongly unforgeable one-time signature scheme, SKE is a CCA-secure one-time symmetric key encryption scheme, $F(\cdot), \dot{F}(\cdot), \bar{F}(\cdot)$ are pseudo-random bit generators (PRBG), and $H_i(\cdot)(i = 1, 2, 3), \bar{H}_i(\cdot)(i = 1, 2)$ are hash functions. We denote l_1 and l_2 as the bit lengths of the private key of SKE and the verifying key of SIG, respectively. C is the ciphertext space of SKE.

PRE.KeyGen: The user sets his public key as $pk = (X_1, X_2, Z_1, Z_2, Z_3) = (g^x, h^{1/x}, P^{x_1}P^{x_2}, P^{x_3}P^{x_4}, P^{x_5})$ for random $sk = (x, x_1, x_2, x_3, x_4, x_5) \leftarrow_R \mathbb{Z}_p$. If the values (X_2, Z_1, Z_2, Z_3) are presented, which signifies that the user is willing to accept delegations.

PRE.ReKeyGen: On input a public key $pk' = (X'_1, X'_2, Z'_1, Z'_2, Z'_3) = (g^y, h^{1/y}, P^{y_1}P^{y_2}, P^{y_3}P^{y_4}, P^{y_5})$, and a private key $sk = (x, x_1, x_2, x_3, x_4, x_5)$, it outputs a unidirectional re-encryption key $rk_{pk,pk'} = (rk^{(1)}_{pk,pk'}, rk^{(2)}_{pk,pk'}, rk^{(3)}_{pk,pk'})$, which is computed as follows.

1. Choose random numbers $r, \bar{r}, \bar{k} \leftarrow_R \mathbb{Z}_p$.
2. Set $rk^{(1)}_{pk,pk'} = (X'_2)^{x \cdot r} = h^{(x/y) \cdot r}$.

3. Set $rk^{(2)}_{pk,pk'} = (Q_1, Q_2, T_1, T_2, T_3) = (P_1^{\bar{r}}, P_2^{\bar{r}}, (Z'_1)^{\bar{H}_1(r)\bar{r}}, (Z'_2)^{\bar{H}_2(r)\bar{r}}, (Z'_3)^{\bar{r}})$.
4. Compute $rk^{(3)}_{pk,pk'} = (\bar{A}, \bar{B}, \bar{C}, \bar{D})$ as follows.
$\bar{A} = P_1^{\bar{k}}, \bar{B} = P_2^{\bar{k}}, \bar{C} = \bar{F}((Z'_3)^{\bar{k}}) \oplus r, \bar{\alpha} = H_1(\bar{A}||\bar{B}||\bar{C}), \bar{D} = (Z'_1)^{\bar{k}}(Z'_2)^{\bar{k}\cdot\bar{\alpha}}$.

PRE.Enc: On input a public key $pk = (X_1, X_2, Z_1, Z_2, Z_3) = (g^x, h^{1/x}, P_1^{x_1}P_2^{x_2}, P_1^{x_3}P_2^{x_4}, P_1^{x_5})$ and a message $m \in \{0,1\}^n$, the encryptor does the following steps:

1. Choose a random number $k \leftarrow_R \mathbb{Z}_p$.
2. Run the key generation algorithm $SIG.\mathcal{G}(1^\lambda)$ to get a signing and verifying key pair (ssk, svk).
3. Set $A = svk, B = g^k, C = (g_2^{H_2(A)} \cdot g_3)^k, D = h'^k, E = SKE.Enc(F(A||C||D||e(X_1,h)^k), m), S = SIG.\mathcal{S}(ssk, (B,C,D,E))$.
4. Output the ciphertext $K = (A, B, C, D, E, S)$.

PRE.ReEnc: On input a re-encryption key $rk_{pk,pk'} = (rk^{(1)}_{pk,pk'}, rk^{(2)}_{pk,pk'}, rk^{(3)}_{pk,pk'})$ and a ciphertext $K = (A, B, C, D, E, S)$ for public key pk_1. The proxy does the following steps.

- If $e(B, g_2^{H_2(A)} \cdot g_3) = e(g, C), e(B, h') = e(g, D)$, and $SIG.\mathcal{V}(A, (B,C,D,E), S) = 1$, all hold, go to the next step; otherwise, output $\perp$ and halt.
- Choose a random number $\dot{k} \leftarrow_R \mathbb{Z}_p$.
- Set $\mathfrak{C} = A||C||D||E||\bar{A}||\bar{B}||\bar{C}||\bar{D}$.
- Compute $B' = e(B, rk^{(1)}_{pk,pk'}), \dot{A} = Q_1^{\dot{k}}, \dot{B} = Q_2^{\dot{k}}, \dot{C} = \dot{F}(T_3^{\dot{k}}) \oplus (B'||\mathfrak{C}), \dot{\alpha} = H_3(\dot{A}||\dot{B}||\dot{C}), \dot{D} = T_1^{\dot{k}}T_2^{\dot{k}\cdot\alpha}$.
- Output the re-encrypted ciphertext $\dot{K} = (\dot{A}, \dot{B}, \dot{C}, \dot{D})$.

PRE.Dec: We have two cases in this algorithm.

- If K is an original ciphertext, such that $K = (A, B, C, D, E, S)$, the decryptor does the following steps. In this case, the private key is $(x, x_1, x_2, x_3, x_4, x_5)$.
  1. If $e(B, g_2^{H_2(A)} \cdot g_3) = e(g, C), e(B, h') = e(g, D)$, and $SIG.\mathcal{V}(A, (B,C,D,E), S) = 1$, all hold, go to the next step; otherwise, output $\perp$ and halt.
  2. Output $m = SKE.Dec(F(A||C||D||e(B,h)^x), E)$. Note that m may be $\perp$.
- If $\dot{K}$ is a re-encrypted ciphertext, such that $\dot{K} = (\dot{A}, \dot{B}, \dot{C}, \dot{D})$. The decryptor does the following steps. In this case, the private key is $(y, y_1, y_2, y_3, y_4, y_5)$.
  1. Compute $m' = \dot{F}(\cdot A^{y_5}) \oplus \dot{C}$, if $m' = (B'||A||C||D||E||\bar{A}||\bar{B}||\bar{C}||\bar{D})$ does not satisfy the format $\mathbb{G}_T \times \{0,1\}^{l_2} \times \mathbb{G}^2 \times \mathcal{C} \times \mathbb{G}_T^2 \times \mathbb{Z}_p \times \mathbb{G}$, then output $\perp$ and halt; otherwise, do the following steps.
  2. If $\bar{A}^{y_1+y_3\cdot\alpha}B^{y_2+y_4\cdot\bar{\alpha}} = \bar{D}$ does not hold, where $\bar{\alpha} = H_1(\bar{A}||\bar{B}||\bar{C})$, then output $\perp$ and halt; otherwise, compute $r = \bar{F}(\bar{A}^{y_5}) \oplus \bar{C}$.

3. If $\dot{A}^{y_1\bar{H}_1(r)+y_3\bar{H}_2(r)\cdot\dot{\alpha}}\dot{B}^{y_2\bar{H}_1(r)+y_4\bar{H}_2(r)\cdot\dot{\alpha}} = \dot{D}$ does not hold, where $\dot{\alpha} = H_1(\dot{A}||\dot{B}||\dot{C})$, then output $\perp$ and halt; otherwise, compute $m = SKE.Dec(F(A||C||D||B'^{y/r}), E)$. Note that m may be $\perp$.

5.2.2 Our Attack

Before describing our attack, we briefly explain how the re-encryption key is generated in the Shao-Liu-Zhou scheme. There are three components in the re-encryption key, where only the first component ($rk^{(1)}_{pk,pk'}$) is computed using the secret key of the delegator (in particular $x \in sk$). The second component ($rk^{(2)}_{pk,pk'}$) is computed using the parameter, the public key of the delegatee, and the randomness chosen by the proxy. The third component ($rk^{(3)}_{pk,pk'}$) is computed almost the same as that in the Cramer-Shoup scheme with the public key $(P_1, P_2, Z'_1, Z'_2, Z'_3)$, except that $\bar{C}$ is computed as $\bar{F}((Z'_3)^{\bar{k}}) \oplus m$ instead of $(Z'_3)^{\bar{k}} \cdot m$.

It is easy to see that, everyone can compute the re-encryption key from delegator to itself (i.e. $rk_{pk,pk}$) without any knowledge of delegator's secret key. In particular, the adversary first chooses randomness $r, \bar{r}, \bar{k} \leftarrow_R \mathbb{Z}_p$ then computes the first component as h^r (because $h^r = h^{(x/x)\cdot r}$). The two latter do not depend on delegator's secret key, so it is easily computed as in $\mathbf{PRE.ReKeyGen}$. Using the computed re-encryption key, the adversary can re-encrypt the challenge ciphertext, then invokes the re-encrypted cihertext to decryption oracle $\mathcal{O}_{dec_1}$ to obtain the plaintext (note that, this is not restricted in the CCA security model).

From the above observation, we build the following efficient attacker $\mathcal{A}$ which aims to decrypt challenge ciphertext $K^* = (A^*, B^*, C^*, D^*, E^*, S^*)$ encrypted for $pk^* = (X^*_1, X^*_2, Z^*_1, Z^*_2, Z^*_3)$.

1. After obtaining the challenge ciphertext K^* from the challenger, $\mathcal{A}$ chooses random numbers $r, \bar{r}, \bar{k} \leftarrow_R \mathbb{Z}_p$. Then $\mathcal{A}$ computes re-encryption key rk_{pk^*,pk^*} as follows:
   - Set $rk^{(1)}_{pk^*,pk^*} = (X'_2)^r (= h^{(x^*/x^*)\cdot r})$.
   - Set $rk^{(2)}_{pk^*,pk^*} = (Q_1, Q_2, T_1, T_2, T_3) = (P_1^{\bar{r}}, P_2^{\bar{r}}, (Z^*_1)^{\bar{H}_1(r)\bar{r}}, (Z^*_2)^{\bar{H}_2(r)\bar{r}}, (Z^*_3)^{\bar{r}})$.
   - Compute $rk^{(3)}_{pk^*,pk^*} = (\bar{A}, \bar{B}, \bar{C}, \bar{D})$ as follows.
     $\bar{A} = P_1^{\bar{k}}, \bar{B} = P_2^{\bar{k}}, \bar{C} = \bar{F}((Z^*_3)^{\bar{k}}) \oplus r, \bar{\alpha} = H_1(\bar{A}||\bar{B}||\bar{C}), \bar{D} = (Z^*_1)^{\bar{k}}(Z^*_2)^{\bar{k}\cdot\bar{\alpha}}$.
2. $\mathcal{A}$ himself re-encrypts the challenge ciphertext CT^* using $\mathbf{PRE.ReEnc}$ with the above re-encryption key as the following:
   - Choose a random number $\dot{k} \leftarrow_R \mathbb{Z}_p$.
   - Set $\mathfrak{C} = A^*||C^*||D^*||E^*||\bar{A}||\bar{B}||\bar{C}||\bar{D}$.
   - Compute $B' = e(B^*, rk^{(1)}_{pk^*,pk^*}), \dot{A} = Q_1^{\dot{k}}, \dot{B} = Q_2^{\dot{k}}, \dot{C} = \dot{F}(T_3^{\dot{k}}) \oplus (B'||\mathfrak{C}), \dot{\alpha} = H_3(\dot{A}||\dot{B}||\dot{C}), \dot{D} = T_1^{\dot{k}} T_2^{\dot{k}\cdot\alpha}$.
   - Output the re-encrypted ciphertext $\dot{K} = (\dot{A}, \dot{B}, \dot{C}, \dot{D})$.
3. $\mathcal{A}$ issues a decryption oracle query under pk^* to decrypt the re-encrypted ciphertext $\dot{K}$ (this is not restricted in the CCA security model), and the result is the message encrypted in K^*.

5.3 The Proposed Scheme

In this section, we first describe a new scheme for PRE, and then show that it meets the 1st-US-CCA and the 2nd-US-CCA security. Our scheme is obtained by extending the PKE scheme proposed by Lai, Deng, Liu, and Kou [37].

5.3.1 Construction

To achieve our CCA security given in Section 3.3, we start from the following observations which are important and necessary principles for designing CCA-secure schemes:

1. The validity of the original ciphertexts can be publicly verifiable by everyone including the proxy; otherwise, it will suffer from the attack as illustrated in [20].

2. It should also be impossible for the adversary to transform the second-level ciphertext to the first level one without knowledge of the delegator's secret key or re-encryption key; otherwise, it will suffer from an attack as applied to the Shao-Liu-Zhou scheme in Section 5.2.

3. For the first-level ciphertext CT_j re-encrypted from the second-level ciphertext CT_i, it should not exhibit any component of CT_i; otherwise, it will fail in achieving the 1st-US-CCA security.

We now describe our scheme. In Section 5.3.2, we will explain how our scheme follows the above principles in the following description of our scheme.

Setup: On input a security parameter 1^λ, run $\mathsf{Gen}(1^\lambda)$ in Section 2.2.3 to obtain $(\mathbb{G}, g, p, H)$. Let $\mathbb{G}_T$ be a multiplicative group of prime order p. Let $\mathbf{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ be a bilinear map. Choose $g_1, h, u, v, d \in \mathbb{G}$, and two collision-resistant hash functions $\mathsf{TCR} : \mathbb{G} \times \mathbb{G}_T \to \mathbb{Z}_p$, $\mathsf{TCR}' : \mathbb{G} \to \mathbb{Z}_p$. $\mathbf{SYM}$ is a symmetric encryption scheme of which the key space is $\{0,1\}^{\ell(n)}$, where $\ell(n)$ is the bit-length of the output of the hash function H. Output a public parameter $PP = (p, \mathbb{G}, \mathbb{G}_T, g, h, g_1, u, v, d, \mathbf{e}, H, \mathsf{TCR}, \mathsf{TCR}')$.

KGen: On input a public parameter $PP = (p, \mathbb{G}, \mathbb{G}_T, g, h, g_1, u, v, d, \mathbf{e}, H, \mathsf{TCR}, \mathsf{TCR}')$, choose $x, y \leftarrow_R \mathbb{Z}_p$, and output a pair (pk, sk) of a public key and a private key, where $pk = (g^x, g_1^{x^2}, g^y), sk = (x, y)$.

ReKey: On input a private key $sk_i = (sk_{i,1}, sk_{i,2})$ and a public key $pk_j = (pk_{j,1}, pk_{j,2}, pk_{j,3})$, output a re-encryption key $rk_{i\to j} = pk_{j,2}^{1/sk_{i,1}}$.

$\mathbf{Enc_1}$: On input a public key $pk_i = (pk_{i,1}, pk_{i,2}, pk_{i,3})$ and a message $m \in \mathbb{G}_\mathrm{T}$, choose $r, R, r', s \leftarrow_R \mathbb{Z}_p$, and compute $C_2 = h^r$, $C_3 = \mathbf{e}(g, g_1)^r \cdot m$, $t = \mathsf{TCR}(C_2, C_3)$, $C_4 = (u^t v^s d)^r$, $C_5 = s$, $C_6 = pk_{i,2}^{rR}$, $C_7 = pk_{i,2}^{R}$, $C_8 = g^{1/R}$, $CT_i' = C_2||C_3||...||C_8$, $A = g^{r'}$, $t' = \mathsf{TCR}'(A)$, $B = (pk_{i,3}^{t'} \cdot h)^{r'}$, $C \leftarrow \mathbf{SYM.Enc}(H(pk_{i,3}^{r'}), CT_i')$, $D = pk_{i,3}$. Finally, output a ciphertext $CT_i = (A, B, C, D)$.

$\mathbf{Enc_2}$: On input a public key $pk_i = (pk_{i,1}, pk_{i,2}, pk_{i,3})$ and a message $m \in \mathbb{G}_{\mathrm{T}}$, choose $r, s \leftarrow_R \mathbb{Z}_p$, and compute $C_1 = pk_{i,1}^r$, $C_2 = h^r$, $C_3 = \mathbf{e}(g, g_1)^r \cdot m$, $t = \mathsf{TCR}(C_2, C_3)$, $C_4 = (u^t v^s d)^r$, $C_5 = s$. Finally, output a ciphertext $CT = (C_1, C_2, C_3, C_4, C_5)$.

ReEnc: On input two public keys pk_i, pk_j a re-encryption key $rk_{i \to j}$, a second-level ciphertext $CT_i = (C_1, C_2, C_3, C_4, C_5)$ do as follows:
Compute $t = \mathsf{TCR}(C_2, C_3)$, and check the validity of the ciphertext CT_i by testing whether the following equalities hold (if both hold, we write $Check(CT_i) = 1$):

$$\mathbf{e}(h, C_1) = \mathbf{e}(C_2, pk_{i,1}) \tag{5.1}$$
$$\mathbf{e}(h, C_4) = \mathbf{e}(C_2, u^t v^s d). \tag{5.2}$$

If one of the above equations does not hold, output $\perp$ indicating an invalid ciphertext. Otherwise, choose $R, r' \leftarrow_R \mathbb{Z}_p$, and compute $C_6 = C_1^R$; $C_7 = pk_{i,1}^R$; $C_8 = rk_{i \to j}^{1/R}$; $CT_i' = C_2 || C_3 || ... || C_8$; $A = g^{r'}$, $t' = \mathsf{TCR}'(A)$, $B = (pk_{j,3}^{t'} \cdot h)^{r'}$, $C \leftarrow \mathbf{SYM.Enc}(H(pk_{j,3}^{r'}), CT_i')$, $D = pk_{j,3}$. Finally, output a ciphertext $CT_j = (A, B, C, D)$.

$\mathbf{Dec_2}$: On input a private key sk_i and a ciphertext $CT = (C_1, C_2, C_3, C_4, C_5)$, do as follows:
Compute $t = \mathsf{TCR}(C_2, C_3)$, and check the validity of the ciphertext CT by testing whether $Check(CT) = 1$ holds. If not, output $\perp$ and halt. Otherwise, output $m = \frac{C_3}{\mathbf{e}(C_1, g_1)^{1/sk_{i,1}}}$.

$\mathbf{Dec_1}$: On input a private key sk_j and a ciphertext $CT_j = (A, B, C, D)$, do as follows:
Compute $t' = \mathsf{TCR}(A)$, if $\mathbf{e}(A, D^{t'} \cdot h) \neq \mathbf{e}(g, B)$ then output $\perp$ indicating an invalid ciphertext. Otherwise, do the following.
Compute $CT_i' = \mathbf{SYM.Dec}(H(A^{sk_{j,2}}), C)$. Parse $CT_i' = C_2 || C_3 || ... || C_8$, if this is not the case, output $\perp$ and halt; else, compute $t = \mathsf{TCR}(C_2, C_3)$ and check the validity of the ciphertext CT_i' by testing whether the following equalities hold:

$$\mathbf{e}(h, C_4) = \mathbf{e}(C_2, u^t v^{C_5} d) \tag{5.3}$$
$$\mathbf{e}(C_7, C_8) = \mathbf{e}(pk_{j,2}, g) \tag{5.4}$$
$$\mathbf{e}(h, C_6) = \mathbf{e}(C_2, C_7). \tag{5.5}$$

If one of the above equations does not hold, output $\perp$ indicating an invalid ciphertext. Otherwise, output $m = \frac{C_3}{\mathbf{e}(C_6, C_8)^{1/sk_{j,1}^2}}$.

5.3.2 Security Analysis

The intuition of the CCA security of our scheme can be seen from the following properties.

1. The validity of the original ciphertexts can be publicly verifiable by everyone including the proxy; otherwise, it will suffer from the attack as illustrated in [20]. For our scheme, the ciphertext components C_4, C_5 in the original ciphertext $CT = (C_1, C_2, C_3, C_4, C_5)$ can be viewed as a signature signing the "message" C_1, C_2, C_3, that is how we get public verifiability.

2. It should be impossible for the adversary to transform the second-level ciphertext to the first level one without knowledge of the delegator's secret key or re-encryption key; otherwise, it only yields the RCCA security. In our scheme, the component $C_8 = rk_{i\to j}^{1/R}$ is computed using the re-encryption key and completely hidden in C, so the adversary cannot transform the second-level ciphertext to a ciphertext re-encrypted by $\mathbf{ReEnc}$ if he has no knowledge of the re-encryption key. The adversary also cannot transform the second-level ciphertext to a ciphertext encrypted by $\mathbf{Enc}_1$ without any knowledge of random component r used in the original ciphertext, because the component $C_6 = pk_{i,2}^{rR}$ is independent of a ciphertext encrypted by $\mathbf{Enc}_2$, and completely hidden in C.

3. It should be impossible for the adversary to compute the re-encryption key from the target user i^* to itself (i.e., $rk_{i^*\to i^*}$), otherwise it will suffer from an attack as applied to the Shao-Liu-Zhou scheme [33]. This follows from Lemma 4.

4. For the first-level ciphertext CT_j re-encrypted from a second-level ciphertext CT_i, it should not exhibit any component of CT_i; otherwise, it will fail in achieving the CCA-security of $\mathbf{ReEnc}$. In our scheme, all of the components from the original ciphertext are hidden in C. Furthermore, we use the KEM/DEM scheme of Kiltz [36] in the re-encryption algorithm to guarantee the CCA security of the re-encrypted ciphertext.

Lemma 4. *$\mathcal{A}$ cannot make $rk_{i^*\to i^*} = g_1^{sk_{i^*,1}}$ without knowledge of the secret key $sk_{i^*,1}$, assuming the 2-AmCDH problem is hard.*

Proof. Suppose there exists an adversary $\mathcal{A}$ who can compute $rk_{i^*\to i^*} = g_1^{sk_{i^*,1}}$ then we show how to construct another algorithm $\mathcal{B}$ that can break the 2-AmCDH assumption in $\mathbb{G}$. Given a 2-AmCDH instance $(g, g^a, g^b, g^{ba^2}, g^{\frac{b}{a}}) \in \mathbb{G}^5$ with unknown $a, b \in \mathbb{Z}_p$, $\mathcal{B}$'s goal is to compute g^{ab}. $\mathcal{B}$ works with interacting with the adversary $\mathcal{A}$ as follows:

Setup: $\mathcal{B}$ chooses $\omega, u, v, d \leftarrow_R \mathbb{Z}_p$, sets $g_1 := g^b$, and computes $h = g^\omega$. The other parameters are chosen as in the algorithm **Setup**. The public parameter is $PP = (p, \mathbb{G}, \mathbb{G}_T, g, h, g_1, u, v, d, \mathsf{e}, H, \mathsf{TCR}, \mathsf{TCR}')$. Next, $\mathcal{B}$ generates the challenge key, the uncorrupted-keys, and the corrupted-keys as follows.

- *The challenge key.* $\mathcal{B}$ chooses $x_{i^*}, y_{i^*} \leftarrow_R \mathbb{Z}_p$, and sets $pk_{i^*} = ((g^a)^{x_{i^*}}, (g^{ba^2})^{x_{i^*}^2}, g^{y_{i^*}})$ (meaning that $sk_{i^*} = (ax_{i^*}, y_{i^*})$).
- *Uncorrupted-key generation.* $\mathcal{B}$ chooses $x_i, y_i \leftarrow_R \mathbb{Z}_p$, and defines $pk_i = (g^{x_i}, (g^b)^{x_i^2}, g^{y_i})$, (meaning that $sk_i = (x_i, y_i)$). $\mathcal{B}$ adds pk_i to $\mathcal{PK}_{un}$.
- *Corrupted-key generation.* $\mathcal{B}$ chooses $x_i, y_i \leftarrow_R \mathbb{Z}_p$, and defines $pk_i = (g^{x_i}, (g^b)^{x_i^2}, g^{y_i}), sk_i = (x_i, y_i)$. $\mathcal{B}$ adds pk_i to $\mathcal{PK}_{corr}$ and sk_i to $\mathcal{SK}_{corr}$.

$\mathcal{B}$ provides $\mathcal{A}$ with the public parameter PP, two sets of key $\mathcal{PK} = (\mathcal{PK}_{un}, \mathcal{PK}_{corr})$, $\mathcal{SK}_{corr}$, and the challenge public key pk_{i^*}.

Find stage. The adversary $\mathcal{A}$ issues a series of questions as in the 2nd-US-CCA game, which is the CCA game as in Definition 22 with the additional constraints from Definition 23. $\mathcal{B}$ answers these queries for $\mathcal{A}$ by simulating the oracles as follows.

Re-Encryption Key Generation Oracle $\mathcal{O}_{rk}(pk_i, pk_j)$**:** $\mathcal{B}$ does as follows.

1. If $(pk_i, pk_j \in \mathcal{PK}_{corr})$ or $(pk_i, pk_j \in \mathcal{PK}_{un})$, then return $rk_{i\to j} = g_1^{x_j^2/x_i}$.
2. If $pk_i \in \mathcal{PK}_{un}$ and $pk_j \in \mathcal{PK}_{corr}$, then return $rk_{i\to j} = g_1^{x_j^2/x_i}$.
3. If $pk_i \in \mathcal{PK}_{corr}$ and $pk_j \in \mathcal{PK}_{un} \cup \{pk_{i^*}\}$, then return $rk_{i\to j} = pk_{j,2}^{1/x_i}$.
4. If $pk_i = pk_{i^*}$ and $pk_j \in \mathcal{PK}_{un}$, then return $rk_{i^*\to j} = (g^{\frac{b}{a}})^{x_j^2/x_{i^*}} (= g_1^{sk_{j,1}^2/sk_{i^*,1}})$.
5. If $pk_i \in \mathcal{PK}_{un}$ and $pk_j = pk_{i^*}$, then return $rk_{i\to i^*} = (g^{ba^2})^{x_{i^*}^2/x_i} (= g_1^{sk_{i^*,1}^2/sk_{i,1}})$.
6. If $pk_i = pk_{i^*}$ and $pk_j \in \mathcal{PK}_{corr}$, then return $\perp$.

First Level Decryption Oracle $\mathcal{O}_{dec_1}(pk_i, CT_i)$**:** If $i \neq i^*$, $\mathcal{B}$ does as the algorithm $\mathbf{Dec}_1$ to decrypt the ciphertext CT_i using the secret key $sk_i = (x_i, y_i)$. Otherwise, $\mathcal{B}$ does as follows. $\mathcal{B}$ parses $CT_i = (A, B, C, D)$, computes $t' = \mathsf{TCR}'(A)$, and checks whether the following equality holds: $\mathbf{e}(A, D^{t'} \cdot h) = \mathbf{e}(g, B)$. If not, output $\perp$ indicating an invalid ciphertext. Otherwise, it does as follows.

Compute $CT_i' = \mathbf{SYM.Dec}(H(A^{y_i}), C)$;

Parse $CT_i' = C_2||C_3||...||C_8$, if this is not the case, output $\perp$ and halt.

Else, $\mathcal{B}$ computes $t = \mathsf{TCR}(C_2, C_3)$ and does as follows:

1. Check the validity of the ciphertext CT_i as the equations 5.3 – 5.5 in the algorithm $\mathbf{Dec}_1$.
2. Check whether $(C_2, C_3) = (C_2^*, C_3^*)$ and $t \neq t^*$ hold. If so, $\mathcal{B}$ aborts and outputs a random bit.

If all of these verifications pass, then there exists $r \in \mathbb{Z}_p$ such that $C_2 = h^r, C_4 = (u^t v^{C_5} d)^r$. If $(C_2, C_3, C_4, C_5) = (C_2^*, C_3^*, C_4^*, C_5^*)$, $\mathcal{B}$ outputs $\perp$ which deems CT_i as a derivative of the challenge pair (pk_{i^*}, CT^*); otherwise, since $C_2 = h^r = g^{\omega r}$, we have $C_2^{1/\omega} = g^r$. Therefore $\mathcal{B}$ can compute $m = \frac{C_3}{\mathbf{e}(g^r, g_1)}$, then returns it to $\mathcal{A}$.

Second Level Decryption Oracle $\mathcal{O}_{dec_2}(pk_i, CT_i)$**:** $\mathcal{B}$ does as follows.

1. If $i \neq i^*$, then do as the algorithm $\mathbf{Dec}_2$ to decrypt the ciphertext CT_i using the secret key $sk_i = (x_i, y_i)$, and return the obtained result to $\mathcal{A}$.
2. If $i = i^*$ and $CT_i \neq CT^*$, then $\mathcal{B}$ first computes a re-encryption key $rk_{i^*\to j} = (g^{\frac{b}{a}})^{x_j^2/x_{i^*}}$, where $pk_j \in \mathcal{PK}_{un}$. $\mathcal{B}$ then re-encrypts the ciphertext CT_i by using $rk_{i^*\to j}$ to obtain $CT_j \leftarrow \mathbf{ReEnc}(rk_{i^*\to j}, CT_i)$. Finally, $\mathcal{B}$ does as in **First Level Decryption Oracle** $\mathcal{O}_{dec_1}$ to decrypt the ciphertext CT_j, and returns the obtained result to $\mathcal{A}$.

Re-Encryption Oracle $\mathcal{O}_{re}(pk_i, pk_j, CT_i)$**:** $\mathcal{B}$ parses $CT_i = (C_1, C_2, C_3, C_4, C_5)$. If this is not the case, then $\mathcal{B}$ outputs $\perp$ and halts. Otherwise, $\mathcal{B}$ computes $t = \mathsf{TCR}(C_2, C_3)$ and checks whether $Check(CT_i) = 1$ holds. If not, then $\mathcal{B}$ outputs $\perp$ and halts; else $\mathcal{B}$ works according to the following cases:

1. If $pk_i \neq pk_{i^*}$ or $pk_j \notin \mathcal{PK}_{corr}$: $\mathcal{B}$ executes the algorithm $\mathbf{ReEnc}$ with the re-encryption key computed as in $\mathcal{O}_{rk}(pk_i, pk_j)$ and ciphertext CT_i, then returns $CT_j \leftarrow \mathbf{ReEnc}(rk_{i\to j}, CT_i)$ to $\mathcal{A}$.
2. If $pk_i = pk_{i^*}, pk_j \in \mathcal{PK}_{corr}$ (meaning that $sk_{i^*} = ax_{i^*}$ and $sk_j = x_j$): $\mathcal{B}$ chooses $R', r' \leftarrow_R \mathbb{Z}_p$ and computes g^r as in **First Decryption Oracle** $\mathcal{O}_{dec_1}$. Using g^r, $\mathcal{B}$ computes:
   $C_6 = (g^r)^{x_{i^*}R'} (= (g^{ax_{i^*}r})^{R'/a} = C_1^R)$, where $R = R'/a$;
   $C_7 = g^{x_{i^*}R'} (= (g^{ax_{i^*}})^{R'/a} = pk_{i^*,1}^R)$;
   $C_8 = (g_1^{x_j^2/x_{i^*}})^{1/R'} (= (g_1^{(sk_{j,1}^2/sk_{i^*,1})^{a/R'}} = rk_{i^*\to j}^{1/R})$;
   $CT'_{i^*} = C_2||C_3||...||C_8$; $A = g^{r'}, t' = \mathsf{TCR}'(A), B = (pk_{j,3}^{t'} \cdot h)^{r'}, C \leftarrow \mathbf{SYM}.\mathbf{Enc}(H(pk_{j,3}^{r'}), CT'_{i^*})$, $D = pk_{j,3}$. Then, $\mathcal{B}$ outputs $CT_j = (A, B, C, D)$.

Challenge. At this stage, $\mathcal{A}$ outputs two messages $m_0, m_1 \in \mathbb{G}_T$. $\mathcal{B}$ flips a coin $\sigma \leftarrow_R \{0, 1\}$ and encrypts m_σ using the algorithm $\mathbf{Enc_2}$ under the public key pk_{i^*}.

Guess stage. The adversary $\mathcal{A}$ continues to issue the rest queries. In this stage, $\mathcal{B}$ can respond these queries for $\mathcal{A}$ as in the find stage.

Whenever, $\mathcal{A}$ outputs $rk_{i^*\to i^*} = g_1^{sk_{i^*,1}}$ (meaning that $rk_{i^*\to i^*} = g^{abx_{i^*}}$), then $\mathcal{B}$ outputs $g^{ab} = k_{i^*\to i^*}^{\frac{1}{x_{i^*}}}$ as the answer to the 2-AmCDH problem.

This completes the description of the simulation.

It is easy to see that the simulation is perfect, therefore we have the probability that $\mathcal{A}$ can compute $rk_{i^*\to i^*}$ is bound by $\mathrm{Adv}_{\mathcal{B}}^{\text{2-AmCDH}}(\lambda)$.

The lemma follows. □

Theorem 7. *The scheme meets the 2nd-US-CCA security, assuming the hash function* TCR *is target collision resistant, the 6-AmDBDH assumption holds in groups* $(\mathbb{G}, \mathbb{G}_T)$*, and the 2-AmCDH problem is hard.*

The proof of Theorem 7 is given in Section 5.5.1.

Theorem 8. *The scheme meets the 1st-US-CCA security, assuming* TCR' *is a target collision resistant hash function, the GHDH problem is hard, and* $\mathbf{SYM}$ *is CCA-secure.*

Remark. In the algorithms $\mathbf{Enc}_1$ and $\mathbf{ReEnc}$, we make use of the encryption algorithm of Kiltz's KEM/DEM scheme [36] to mask all of the computed components including the second-level ciphertext. Therefore, the 1st-US-CCA security of our scheme is implied by the CCA security of Kiltz's KEM/DEM scheme.

The proof of Theorem 8 is given in Section 5.5.2.

We have the following theorem which is immediately implied from Theorems 7 and 8.

Theorem 9. *The scheme meets the US-CCA security, assuming the hash functions* TCR *and* TCR' *are target collision resistant, the 6-AmDBDH assumption holds in groups* $(\mathbb{G}, \mathbb{G}_T)$*, the 2-AmCDH problem and the GHDH problem are hard, and* $\mathbf{SYM}$ *is CCA-secure.*

5.4 On the Hardness of the 6-AmDBDH Problem

In this section, we review Boneh et al.'s results [10] on the general Diffie-Hellman problem in the generic bilinear group model, and analyze the relative hardness of the 6-AmDBDH problem in the generic bilinear group model.

5.4.1 The DBDH Problem and Its Variant

We shall assume the intractability of a variant, introduced for the first time in [10], of the decisional bilinear Diffie-Hellman (DBDH) problem which is, given (g^a, g^b, g^c) with $a, b, c \leftarrow_R \mathbb{Z}_p$, to distinguish $\mathbf{e}(g,g)^{abc}$ from random elements of $\mathbb{G}_T$.

Definition 36 (mDBDH Problem) [16, 49])**.** *The modified decisional bilinear Diffie-Hellman (*mDBDH*) problem in groups $(\mathbb{G}, \mathbb{G}_T)$ is, given $(g, g^a, g^b, g^c, Q) \in \mathbb{G}^4 \times \mathbb{G}_T$, where $a, b, c \leftarrow_R \mathbb{Z}_p$, decide whether $Q = \mathbf{e}(g,g)^{ab/c}$ holds.*

5.4.2 Equivalent Definitions of the 6-AmDBDH Problem

For an easy analysis of the relative hardness of the 6-AmDBDH problem in the generic bilinear group model, we here first describe a variant of the 6-AmDBDH problem named 7-AmDBDH, which is almost identical to the 6-AmDBDH problem except introducing the additional term $g^{1/c}$.

Definition 37 (7-AmDBDH Problem)**.** *The 7-augmented modified decisional bilinear Diffie-Hellman (7-*AmDBDH*) problem in groups $(\mathbb{G}, \mathbb{G}_T)$ is, given $(g, g^{1/c}, g^a, g^b, g^c, g^{a/c}, g^{c^2}, g^{ac}, g^{ac^2}, g^{ac^3}, g^{ac^4}, Q) \in \mathbb{G}^{11} \times \mathbb{G}_T$, where $a, b, c \leftarrow_R \mathbb{Z}_p$, decide whether $Q = \mathbf{e}(g,g)^{ab/c^2}$ holds. For an adversary $\mathcal{A}$, we define its advantage in solving the* 7-AmDBDH *problem in groups $(\mathbb{G}, \mathbb{G}_T)$ as*

$$\begin{aligned}
&\mathrm{Adv}^{\text{7-AmDBDH}}_{\mathcal{A}} \stackrel{def}{=} \\
&\left|\Pr[\mathcal{A}(g, g^{1/c}, g^a, g^b, g^c, g^{a/c}, g^{c^2}, g^{ac}, g^{ac^2}, g^{ac^3}, g^{ac^4}, \mathbf{e}(g,g)^{ab/c^2}) = 1 | a, b, c \leftarrow_R \mathbb{Z}_p]\right| \\
- &\left|\Pr[\mathcal{A}(g, g^{1/c}, g^a, g^b, g^c, g^{a/c}, g^{c^2}, g^{ac}, g^{ac^2}, g^{ac^3}, g^{ac^4}, \mathbf{e}(g,g)^{z}) = 1 | a, b, c, z \leftarrow_R \mathbb{Z}_p]\right|.
\end{aligned}$$

Since the 7-AmDBDH problem is almost identical to the 6-AmDBDH problem except introducing the additional term $g^{1/c}$, we have the following trivial lemma.

Lemma 5. *For any probabilistic polynomial-time adversary $\mathcal{A}$ we have*

$$\mathrm{Adv}^{\text{6-AmDBDH}}_{\mathcal{A}} \leqq \mathrm{Adv}^{\text{7-AmDBDH}}_{\mathcal{A}}.$$

Next we describe an equivalent variant of the 7-AmDBDH problem.

Proposition 1. *The 7-AmDBDH problem in groups $(\mathbb{G}, \mathbb{G}_T)$ is equivalent to given $(g, g^c, g^a, g^{ac}, g^{bc}, g^{c^2}, g^{ac^2}, g^{ac^3}, g^{ac^4}, g^{ac^5} Q) \in \mathbb{G}^{11} \times \mathbb{G}_T$ with unknown $a, b, c \leftarrow_R \mathbb{Z}_p$ decide whether $Q = \mathbf{e}(g,g)^{ab}$ holds.*

Proof. Given input $(g, g^c, g^a, g^{ac}, g^{bc}, g^{c^2}, g^{ac^2}, g^{ac^3}, g^{ac^4}, g^{ac^5} Q)$ we can construct a 7-AmDBDH instance by setting $(h = g^c, h^{1/c} = g, h^a = g^{ac}, h^b = g^{bc}, h^c = g^{c^2}, h^{ac} = g^{ac^2}, h^{ac^2} = g^{ac^3}, h^{ac^3} = g^{ac^4}, h^{ac^4} = g^{ac^5})$. Then, we have $\mathbf{e}(h,h)^{ab/c^2} = \mathbf{e}(g^c, g^c)^{ab/c^2} = \mathbf{e}(g,g)^{ab}$. The converse implication is easily established and demonstrates the equivalence between both problems. □

5.4.3 The General Diffie-Hellman Problem

Before giving the definition of the general Diffie-Hellman problem, we review some notations used in [10]. Let p be an integer and let s, n be positive integers. Let $P, Q \in \mathbb{F}_p[X_1, ..., X_n]^s$ be two s-tuples of n-variate polynomials over $\mathbb{F}_p$ and let $f \in \mathbb{F}_p[X_1, ..., X_n]$. We write $P = (p_1, p_2, ..., p_s)$ and $Q = (q_1, q_2, ..., q_s)$. It is required that both p_1 and q_1 are constant polynomial 1. For a set Ω, a function $h : \mathbb{F}_p \to \Omega$, and a vector $(x_1, ..., x_n) \in \mathbb{F}_p^n$, we write

$$\begin{aligned} h(P(x_1, ..., x_n)) &= (h(p_1(x_1, ..., x_n)), ..., h(p_s(x_1, ..., x_n))) \in \Omega^s, \\ h(Q(x_1, ..., x_n)) &= (h(q_1(x_1, ..., x_n)), ..., h(q_s(x_1, ..., x_n))) \in \Omega^s. \end{aligned}$$

For a polynomial $f \in \mathbb{F}_p[X_1, ..., X_n]$, we let d_f denote the total degree of f. For a set $P \subseteq \mathbb{F}_p[X_1, ..., X_n]^s$, we let $d_P = \max\{d_f | f \in P\}$.

Let $\mathbb{G}_0, \mathbb{G}_1$ be groups of order p and let $e : \mathbb{G}_0 \times \mathbb{G}_0 \to \mathbb{G}_1$ be a non-degenerate bilinear map. Let g be a generator of G_0 and set $g_1 = e(g, g) \in \mathbb{G}_1$.

Now the (P, Q, f)-Diffie-Hellman problem in $(\mathbb{G}_0, \mathbb{G}_1)$ is depicted as below: given the vector

$$H(x_1, ..., x_n) = (g^{P(x_1,...,x_n)}, g_1^{Q(x_1,...,x_n)}) \in \mathbb{G}_0^s \times \mathbb{G}_1^s,$$

compute $g_1^{f(x_1,...,x_n)} \in \mathbb{G}_1$.

Example 1. Many problems, including the DBDH problem and the computational version of the 7-AmDBDH problem, fall in the (P, Q, f)-Diffie-Hellman problem.

- The DBDH problem: Set $P = (1, x, y, z), Q = (1), f = xyz$. Here $s = 4, n = 3, d_P = 1, d_Q = 0$, and $d_f = 3$.
- The 7-AmDBDH problem: Set $P = (1, z, xz, yz, z^2, xz^2, xz^3, xz^4, xz^5)$, $Q = (1), f = xy$. Here $s = 9, n = 3, d_P = 6, d_Q = 0$, and $d_f = 2$.

To obtain the most general result, Boneh et al. [10] considered the decisional version of (P, Q, f)- Diffie-Hellman problem. It is said that an algorithm $\mathcal{B}$ that outputs $b \in \{0, 1\}$ has advantage ε in solving the decisional the (P, Q, f)-Diffie-Hellman problem if

$$\Pr\big[\mathcal{B}(H(x_1, ..., x_n), g_1^{f(x_1,...,x_n)}) = 0\big] - \Pr\big[\mathcal{B}(H(x_1, ..., x_n), T) = 0\big] > \varepsilon,$$

where the probability is over the random choice of generator $g \in \mathbb{G}_0$, the random choice of $x_1, ..., x_n$ in $\mathbb{F}_p$, the random choice of $T \in \mathbb{G}_1$, and the random bits consumed by $\mathcal{B}$.

5.4.4 Complexity Lower Bounds in Generic Bilinear Groups

We first review what is a generic bilinear group. Consider two random encodings ξ_0, ξ_1 of the additive group $\mathbb{Z}_p$, i.e. injective maps $\xi_0, \xi_1 : \mathbb{Z}_p \to \{0, 1\}^m$. For $i = 0, 1$, denote $\mathbb{G}_i = \{\xi_i(x) | x \in \mathbb{Z}_p\}$. We are given oracles to compute the induced group action on $\mathbb{G}_0, \mathbb{G}_1$, and an oracle to compute a non-degenerate bilinear map $e : \mathbb{G}_0 \times \mathbb{G}_0 \to \mathbb{G}_1$. Group $\mathbb{G}_0$ is called a generic bilinear group.

Before giving the complexity lower bound for the decisional (P, Q, f)-Diffie-Hellman problem in generic bilinear groups, we review the following useful definitions, which are used in [10].

Let $P, Q \in \mathbb{F}_p[X_1, ..., X_n]^s$ be two s-tuples of n-variate polynomials over $\mathbb{F}_p$. We write $P = (p_1, p_2, ..., p_s)$ and $Q = (q_1, q_2, ..., q_s)$ where $p_1 = q_1 = 1$. We say that a polynomial $f \in \mathbb{F}_p[X_1, ..., X_n]$ is dependent on the sets (P, Q) if there exists $s^2 + s$ constants $\{a_{i,j}\}_{i,j=1}^s, \{b_k\}_{k=1}^s$ such that $f = \sum_{i,j=1}^s a_{i,j} p_i p_j + \sum_{k=1}^s b_k q_k$. We say that f is independent of (P, Q) if f is not dependent on (P, Q). It can be verified that, as to the assumption listed in Example 1, polynomial f is indeed independent of (P, Q).

Boneh et al. [10] gave the following theorem, which gives a lower bound on the advantage of a generic algorithm in solving the decisional (P, Q, f)-Diffie-Hellman problem.

Theorem 10 ([10]). *Let $P, Q \in \mathbb{F}_p[X_1, ..., X_n]^s$ be two s-tuples of n-variate polynomials over $\mathbb{F}_p$ and let $f \in \mathbb{F}_p[X_1, ..., X_n]$. Let $d = max(2d_P, d_Q, d_f)$. Let ξ_0, ξ_1 and $\mathbb{G}_0, \mathbb{G}_1$ be defined as above. If f is independent of (P, Q), then for any $\mathcal{A}$ that makes a total of at most q queries to the oracles computing the group operation in $\mathbb{G}_0, \mathbb{G}_1$ and the bilinear pairing $e : \mathbb{G}_0 \times \mathbb{G}_0 \to \mathbb{G}_1$, we have*

$$\left| \Pr\left[\mathcal{A}\begin{pmatrix} p, \xi_0(P(x_1, ..., x_n)), \\ \xi_1(Q(x_1, ..., x_n)), \\ \xi_1(t_0), \xi_1(t_1), \end{pmatrix} = b : \begin{array}{c} x_1, ..., x_n, y \leftarrow_R \mathbb{F}_p, \\ b \leftarrow_R \{0, 1\}, \\ t_b \leftarrow f(x_1, ..., x_n), \\ t_{1-b} \leftarrow y \end{array} \right] - \tfrac{1}{2} \right| \leq \frac{(q + 2s + 2)^2 \cdot d}{2p}.$$

Table 3 gives the complexity lower bound for the decisional version of the problems listed in **Example 1** (i.e., the DBDH and the 7-AmDBDH problems).

Assumptions	DBDH	7-AmDBDH
Complexity lower bound	$\frac{3(q+10)^2}{2p}$	$\frac{12(q+20)^2}{2p}$

Table 3: Complexity lower bounds of the DBDH and the 7-AmDBDH problems in the generic bilinear group model

From the results in Table 3 and Lemma 5 we obtain the complexity lower bound of the 6-AmDBDH problem, on which our CCA-secure PRE scheme is based, in the generic bilinear group is $\frac{12(q+20)^2}{2p}$. Hence, a generic attacker's advantage in solving the 6-AmDBDH problem is less than 16 times of that in solving the DBDH problem.

5.5 Proofs

5.5.1 Proof of Theorem 7

We prove that the proposed scheme is 2nd-US-CCA secure under the 6-AmDBDH assumption. We build an algorithm $\mathcal{B}$ which is, given a 6-AmDBDH instance $(g, g^a, g^b, g^c, g^{a/c}, g^{c^2}, g^{ac}, g^{ac^2}, g^{ac^3}, g^{ac^4}, Q) \in \mathbb{G}^{10} \times \mathbb{G}_T$, deciding whether $Q = \mathbf{e}(g, g)^{ab/c^2}$ holds, using the 2nd-US-CCA adversary $\mathcal{A}$. The algorithm $\mathcal{B}$ runs the adversary $\mathcal{A}$ simulating its view as in the 2nd-US-CCA security experiment as follows:

Setup: $\mathcal{B}$ chooses $\omega, x_v, x_d, y_u, y_v, y_d \leftarrow_R \mathbb{Z}_p$, sets $g_1 := g^a$, and computes $h = (g^{c^2})^{\omega}, u = g \cdot (g^{c^2})^{y_u}, v = g^{x_v} \cdot (g^{c^2})^{y_v}, d = g^{x_d} \cdot (g^{c^2})^{y_d}$. The other parameters are chosen as in the algorithm **Setup**. The public parameter is $PP = (p, \mathbb{G}, \mathbb{G}_T, g, h, g_1, u, v, d, \mathbf{e}, H, \mathsf{TCR}, \mathsf{TCR}')$.

Next, $\mathcal{B}$ generates the challenge key, uncorrupted-keys, and corrupted-keys as follows.

- *Challenge-key generation.* $\mathcal{B}$ chooses $x_{i^*}, y_{i^*} \xleftarrow{\mathrm{R}} \mathbb{Z}_p$, and defines $pk_{i^*} = ((g^{c^2})^{x_{i^*}}, (g^{ac^4})^{x_{i^*}^2}, g^{y_{i^*}})$ (meaning that $sk_{i^*} = (c^2 x_{i^*}, y_{i^*})$).
- *Uncorrupted-key generation.* $\mathcal{B}$ chooses $x_i, y_i \xleftarrow{\mathrm{R}} \mathbb{Z}_p$, and defines $pk_i = ((g^c)^{x_i}, (g^{ac^2})^{x_i^2}, g^{y_i})$ (meaning that $sk_i = (cx_i, y_i)$). $\mathcal{B}$ adds pk_i to $\mathcal{PK}_{un}$.
- *Corrupted-key generation.* $\mathcal{B}$ chooses $x_i, y_i \xleftarrow{\mathrm{R}} \mathbb{Z}_p$, and defines $pk_i = (g^{x_i}, (g^a)^{x_i^2}, g^{y_i}), sk_i = (x_i, y_i)$. $\mathcal{B}$ adds pk_i to $\mathcal{PK}_{corr}$ and sk_i to $\mathcal{SK}_{corr}$.

$\mathcal{B}$ provides $\mathcal{A}$ with the public parameter PP, two sets of key $\mathcal{PK} = (\mathcal{PK}_{un}, \mathcal{PK}_{corr})$, $\mathcal{SK}_{corr}$, and the challenge public key pk_{i^*}.

Find stage. Adversary $\mathcal{A}$ issues a series of questions as in the 2nd-US-CCA game. $\mathcal{B}$ answers these queries for $\mathcal{A}$ by simulating the oracles as follows.

Re-Encryption Key Generation Oracle $\mathcal{O}_{rk}(pk_i, pk_j)$: $\mathcal{B}$ does as follows.

1. If $pk_i, pk_j \in \mathcal{PK}_{corr}$, then return $rk_{i \to j} = g_1^{x_j^2/x_i}$.
2. If $pk_i, pk_j \in \mathcal{PK}_{un}$, then return $rk_{i \to j} = (g^{ac})^{x_j^2/x_i}$.
3. If $pk_i \in \mathcal{PK}_{un}$ and $pk_j \in \mathcal{PK}_{corr}$, then return $rk_{i \to j} = (g^{a/c})^{x_j^2/x_i}$.
4. If $pk_i \in \mathcal{PK}_{corr}$ and $pk_j \in \mathcal{PK}_{un} \cup \{pk_{i^*}\}$, then return $rk_{i \to j} = pk_{j,2}^{1/x_i}$.
5. If $pk_i = pk_{i^*}$ and $pk_j \in \mathcal{PK}_{un}$, then return $rk_{i^* \to j} = (g^{ac})^{x_j^2/x_{i^*}}$.
6. If $pk_i \in \mathcal{PK}_{un}$ and $pk_j = pk_{i^*}$, then return $rk_{i \to i^*} = (g^{ac^3})^{x_{i^*}^2/x_i}$.
7. If $pk_i = pk_{i^*}$ and $pk_j \in \mathcal{PK}_{corr}$, then return $\perp$.

First Level Decryption Oracle $\mathcal{O}_{dec_1}(pk_i, CT_i)$: If $pk_i \in \mathcal{PK}_{corr}$, $\mathcal{B}$ does as $\mathbf{Dec}_1$ to decrypt ciphertext CT_i using secret key sk_i. Otherwise, $\mathcal{B}$ does as follows. $\mathcal{B}$ parses $CT_i = (A, B, C, D)$, computes $t' = \mathsf{TCR}'(A)$, and checks whether the following equality holds: $\mathbf{e}(A, D^{t'} \cdot h) = \mathbf{e}(g, B)$. If not, output $\perp$ indicating an invalid ciphertext. Otherwise, it does the following.

Compute $CT_i' = \mathbf{SYM.Dec}(H(A^{y_i}), C)$;

Parse $CT_i' = C_2 || C_3 || ... || C_8$, if this is not the case, output $\perp$ and halt.

Else, $\mathcal{B}$ computes $t = \mathsf{TCR}(C_2, C_3)$ and does as follows:

1. Check the validity of the ciphertext CT_i' as the equations 5.3 – 5.5 in $\mathbf{Dec}_1$.
2. Check whether $(C_2, C_3) = (C_2^*, C_3^*)$ and $t \neq t^*$. If so, $\mathcal{B}$ aborts and outputs a random bit.

If all of these verifications pass, then there exists $r \in \mathbb{Z}_p$ such that $C_2 = h^r, C_4 = (u^t v^{C_5} d)^r$. If $(C_2, C_3, C_4, C_5) = (C_2^*, C_3^*, C_4^*, C_5^*)$, $\mathcal{B}$ outputs $\perp$ which deems CT_i as a derivative of the challenge pair (pk_{i^*}, CT^*) by the following claim (the proof of this claim will be shown after that of Theorem 7).

Claim 2. *If $(C_2, C_3, C_4, C_5) = (C_2^*, C_3^*, C_4^*, C_5^*)$ and CT_i' is valid (i.e., it passes the verifications the equations 5.3 – 5.5), then CT_i is a derivative of the challenge pair (pk_{i^*}, CT^*).*

We now assume $(C_2, C_3, C_4, C_5) \neq (C_2^*, C_3^*, C_4^*, C_5^*)$. Using the technique used in [37, 47], we can decrypt the above ciphertext as follows. $\mathcal{B}$ checks whether $t + sx_v + x_d = 0$. If so, $\mathcal{B}$ aborts and outputs a random bit. Otherwise, since $C_2 = h^r = g^{c^2 \omega r}$, we have $C_2^{1/\omega} = g^{c^2 r}$. Therefore $\mathcal{B}$ can compute $\alpha := C_2^{(ty_u + sy_v + y_d)/\omega}$ by computing $g^{c^2 r(ty_u + sy_v + y_d)}$. Since $C_4 = (u^t v^s d)^r = g^{r(t + sx_v + x_d)} \cdot g^{c^2 r(ty_u + sy_v + y_d)}$ we have $g^r = (C_4/\alpha)^{1/(t + sx_v + x_d)}$. Therefore, $\mathcal{B}$ can compute g^r by computing $(C_4/\alpha)^{1/(t + sx_v + x_d)}$. $\mathcal{B}$ computes $m = \frac{C_3}{\mathbf{e}(g^r, g_1)}$, then returns it to $\mathcal{A}$.

Second Level Decryption Oracle $\mathcal{O}_{dec_2}(pk_i, CT_i)$: $\mathcal{B}$ does as follows.

1. If $pk_i \in \mathcal{PK}_{corr}$, then do as the algorithm $\mathbf{Dec}_2$ to decrypt the ciphertext CT_i using the secret key $sk_i = (x_i, y_i)$, and return the obtained result to $\mathcal{A}$.
2. If $pk_i \in \mathcal{PK}_{un}$, then $\mathcal{B}$ first computes a re-encryption key $rk_{i \to j} = (g^{a/c})^{x_j^2/x_i}$, where $pk_j \in \mathcal{PK}_{corr}$. $\mathcal{B}$ then re-encrypts the ciphertext CT_i by using $rk_{i \to j}$ to obtain $CT_j \leftarrow \mathbf{ReEnc}(rk_{i \to j}, CT_i)$. Finally, $\mathcal{B}$ does as the algorithm $\mathbf{Dec}_1$ to decrypt the ciphertext CT_j using the secret key $sk_j = (x_j, y_j)$, and returns the obtained result to $\mathcal{A}$.
3. If $i = i^*$ and $CT_i \neq CT^*$, then $\mathcal{B}$ first computes a re-encryption key $rk_{i^* \to j} = (g^{ac})^{x_j^2/x_{i^*}}$, where $pk_j \in \mathcal{PK}_{un}$. $\mathcal{B}$ then re-encrypts the ciphertext CT_i by using $rk_{i^* \to j}$ to obtain $CT_j \leftarrow \mathbf{ReEnc}(rk_{i^* \to j}, CT_i)$. Finally, $\mathcal{B}$ does as in **First Decryption Oracle** $\mathcal{O}_{dec_1}$ to decrypt the ciphertext CT_j, and returns the obtained result to $\mathcal{A}$.

Re-Encryption Oracle $\mathcal{O}_{re}(pk_i, pk_j, CT_i)$: $\mathcal{B}$ parses $CT_i = (C_1, C_2, C_3, C_4, C_5)$. If this is not the case, output $\perp$ and halt. Otherwise, $\mathcal{B}$ computes $t = \mathsf{TCR}(C_2, C_3)$ and check whether $Check(CT_i) = 1$ holds. If not, output $\perp$ and halt; else $\mathcal{B}$ works according to the following cases:

1. If $pk_i \neq pk_{i^*}$ or $pk_j \notin \mathcal{PK}_{corr}$: $\mathcal{B}$ executes the algorithm $\mathbf{ReEnc}$ with a re-encryption key computed as in $\mathcal{O}_{rk}(pk_i, pk_j)$ and the ciphertext CT_i, then returns $CT_j \leftarrow \mathbf{ReEnc}(rk_{i \to j}, CT_i)$ to $\mathcal{A}$.
2. If $pk_i = pk_{i^*}, pk_j \in \mathcal{PK}_{corr}$ (meaning that $sk_{i^*} = c^2 x_{i^*}$ and $sk_j = x_j$): $\mathcal{B}$ chooses $R', r' \xleftarrow{\mathrm{R}} \mathbb{Z}_p$ and computes g^r as in the simulation of **First Level Decryption Oracle** $\mathcal{O}_{dec_1}$. Using g^r, $\mathcal{B}$ computes:
   $C_6 = (g^r)^{x_{i^*} R'} (= (g^{c^2 x_{i^*} r})^{R'/c^2} = C_1^R)$, where $R = R'/c^2$;
   $C_7 = g^{x_{i^*} R'} (= (g^{c^2 x_{i^*}})^{R'/c^2} = pk_{i^*,1}^R)$;

$C_8 = (g_1^{x_j^2/x_{i^*}})^{1/R'} (= (g_1^{(sk_{j,1}^2/sk_{i^*,1})^{c^2/R'}} = rk_{i^* \to j}^{1/R})$;
$CT'_{i^*} = C_2||C_3||...||C_8$; $A = g^{r'}, t' = \mathsf{TCR}'(A), B = (pk_{j3}^{t'} \cdot h)^{r'}, C \leftarrow \mathbf{SYM.Enc}(H(pk_{j,3}^{r'}), CT'_{i^*})$, $D = pk_{j,3}$. Then, $\mathcal{B}$ outputs $CT_j = (A, B, C, D)$.

Challenge. At this stage, $\mathcal{A}$ outputs two messages $m_0, m_1 \in \mathbb{G}_T$. $\mathcal{B}$ flips a coin $\sigma \xleftarrow{\mathrm{R}} \{0,1\}$ and defines: $C_1^* = (g^b)^{x_{i^*}}, C_2^* = (g^b)^{\omega}, C_3^* = Q \cdot m_\sigma, C_5^* = s^* = -(t^* + x_d)/x_v$ where $t^* = \mathsf{TCR}(C_2^*, C_3^*)$, and $C_4^* = (g^b)^{t^* y_u + s^* y_v + y_d}$. Return $CT^* = (C_1^*, C_2^*, C_3^*, C_4^*, C_5^*)$ as the challenge ciphertext to $\mathcal{A}$.

Observe the following to see that, if $Q = \mathbf{e}(g,g)^{ab/c^2}$, then CT^* is indeed a valid challenge ciphertext under the public key pk_{i^*} with the random exponent $r = b/c^2$.

$$C_1^* = (g^b)^{x_{i^*}} = (g^{c^2 x_{i^*}})^{b/c^2} = pk_{i^*,1}^r; C_2^* = (g^b)^{\omega} = (g^{c^2\omega})^{b/c^2} = h^r;$$

$$C_3^* = Q \cdot m_\sigma; t^* = \mathsf{TCR}(C_2^*, C_3^*); C_5^* = s^* = -(t^* + x_d)/x_v;$$

$$C_4^* = (g^b)^{t^* y_u + s^* y_v + y_d} = (g^{c^2(t^* y_u + s^* y_v + y_d)})^{b/c^2} \cdot (g^{t^* + s^* x_v + x_d})^{b/c^2} = (u^{t^*} v^{s^*} d)^r.$$

In contrast, if Q is random in $\mathbb{G}_T$, then CT^* perfectly hides m_σ and $\mathcal{A}$ cannot guess σ with better probability than $1/2$.

Guess stage. The adversary $\mathcal{A}$ continues to issue the rest queries. In this stage, $\mathcal{B}$ can respond these queries for $\mathcal{A}$ as in the find stage.

Output. Eventually, the adversary $\mathcal{A}$ returns a guess $\sigma' \in \{0,1\}$. If $\sigma' = \sigma$, $\mathcal{B}$ outputs 1 to guess $Q = \mathbf{e}(g,g)^{ab/c^2}$; else $\mathcal{B}$ outputs 0 to guess that Q is a random element in $\mathbb{G}_T$.

This completes the description of the simulation.

Next, we analyze the simulation.

In the simulation of the first level decryption oracle $\mathcal{O}_{dec_1}$, $\mathcal{B}$ aborts in two cases. In the first case (i.e., checking whether $(C_2, C_3) = (C_2^*, C_3^*)$ and $t \neq t^*$) the probability that $\mathcal{B}$ aborts is at most $\mathrm{Adv}^{\mathrm{TCR}}_{\mathsf{TCR},\mathcal{A}}$. In the second case (i.e., checking whether $t + sx_v + x_d = 0$), the probability that $\mathcal{B}$ aborts is at most q_{dec}/p by the following reason. For the challenge ciphertext, the adversary could obtain the information that $t^* + s^* x_v + x_d = 0$. However, there are exactly p possible (x_v, x_d) pairs that satisfy this equation and each of them are equally likely. Thus, information-theoretically, the probability that $t + sx_v + x_d = 0$ is at most $1/p$.

Therefore, the probability that $\mathcal{B}$ aborts during the simulation of the oracle $\mathcal{O}_{dec_1}$ is at most $\mathrm{Adv}^{\mathrm{TCR}}_{\mathsf{TCR},\mathcal{A}} + q_{dec}/p$.

Observe that, if $\mathcal{B}$ does not abort during the simulation and $Q = \mathbf{e}(g,g)^{ab/c^2}$, then the simulation of $\mathcal{B}$ is perfect from the view of $\mathcal{A}$. On the other hand, if $\mathcal{B}$ does not abort during the simulation and Q is distributed uniformly and independently over $\mathbb{G}_T$ then the advantage that $\mathcal{A}$ wins the attack game is $1/2$. Since the simulations for $\mathcal{O}_{re}, \mathcal{O}_{rk}$ are perfect, we have $\mathrm{Adv}^{\text{6-AmDBDH}}_{\mathcal{B}}(\lambda) \geq \frac{1}{2} \cdot \mathrm{Adv}^{\text{second-CCA}}_{\mathrm{PRE},\mathcal{A}}(\lambda) - \mathrm{Adv}^{\mathrm{TCR}}_{\mathsf{TCR},\mathcal{A}} - q_{dec}/p$.

This concludes the proof of Theorem 7.

We now present the proof of Claim 2.

Proof of Claim 2. Assume towards a contradiction that after the challenge phase is finished, there exists an adversary $\mathcal{A}'$ can compute a valid ciphertext $CT'_t = C^*_2||C^*_3||C^*_4||C^*_5||C_6||C_7||C_8$ (i.e., it passes the verifications the equations 5.3 – 5.5) for an index $t \in \{1, ..., n_{un}\}$ without using the re-encryption key $rk_{i^* \to t}$ or querying the re-encryption oracle. Then we show that we can use $\mathcal{A}'$ to construct an algorithm $\mathcal{B}'$ which is, given a 6-AmDBDH instance $(g, g^a, g^b, g^c, g^{a/c}, g^{c^2}, g^{ac}, g^{ac^2}, g^{ac^3}, g^{ac^4}, Q) \in \mathbb{G}^{10} \times \mathbb{G}_T$, deciding whether $Q = \mathbf{e}(g,g)^{ab/c^2}$.

$\mathcal{B}'$ works almost the same as the algorithm $\mathcal{B}$ in the proof of Theorem 7, except that in the uncorrupted-key generation phase, $\mathcal{B}'$ chooses $t \xleftarrow{\mathrm{R}} \{1, ..., n_{un}\}$ and generates a pair of key (pk_t, sk_t) as in the corrupted-key generation phase.

After the challenge phase is finished, whenever $\mathcal{A}'$ queries $\mathcal{O}_{dec_1}(pk_i, CT_i)$ such that $i = t$ and $CT'_i = C^*_2||C^*_3||C^*_4||C^*_5||C_6||C_7||C_8$ is valid (i.e., it passes the verifications the equations 5.3 – 5.5), where CT'_i is computed by $\mathcal{B}'$ as in the simulation of the first decryption oracle in the proof of Theorem 7, then $\mathcal{B}'$ computes $Q' = \mathbf{e}(C_6, C_8)^{1/x_t^2} = \mathbf{e}(g,g)^{ab/c^2}$ and succeeds in deciding whether $Q = \mathbf{e}(g,g)^{ab/c^2}$ by checking if $Q = Q'$ holds. It is easy to see that $\mathcal{B}'$ succeeds with the probability of $1/n_{un}$.

In order to conclude the proof of Claim 2, we show that $\mathbf{e}(C_6, C_8)^{1/x_t^2} = \mathbf{e}(g,g)^{ab/c^2}$ holds. Since CT'_t passes the verifications the equation 5.5, i.e., $\mathbf{e}(h, C_6) = \mathbf{e}(C^*_2, C_7)$, and $C^*_2 = h^{b/c^2}$, we have $C_6 = C_7^{b/c^2}$. Therefore, $\mathbf{e}(C_6, C_8)^{1/x_t^2} = \mathbf{e}(C_7^{b/c^2}, C_8)^{1/x_t^2} = (\mathbf{e}(pk_{t,2}, g)^{b/c^2})^{1/x_t^2} = \mathbf{e}(g,g)^{ab/c^2}$. □

5.5.2 Proof of Theorem 8

We will prove the theorem using a sequences of game transitions.

Game G0: This is the original 1st-US-CCA game, which is the CCA game as in Definition 22 with the additional constraints from Definition 25. The adversary $\mathcal{A}$ has access to the re-encryption key oracle $\mathcal{O}_{rk}$ and the decryption oracle $\mathcal{O}_{dec_1}$, gets a target ciphertext $CT^* = (A^*, B^*, C^*, D^*)$, where CT^* is computed as follows:

1. If the challenge ciphertext is the original one: $CT^* \leftarrow \mathbf{Enc}_1(pk_{i^*}, m_\sigma)$.
2. If the challenge ciphertext is the re-encrypted one: $CT^* \leftarrow \mathbf{ReEnc}(rk_{i' \to i^*}, CT_\sigma)$, where $CT_\sigma = (C_1, ..., C_5)$ is a second-level ciphertext which can be re-encrypted from $pk_{i'}$ to pk_{i^*}.

Let $\sigma' \in \{0,1\}$ denote the output of $\mathcal{A}$, and let T_0 be the event that $\sigma' = \sigma$ in **G0**, so that

$$\mathrm{Adv}^{\text{first-CCA}}_{\mathrm{PRE},\mathcal{A}}(\lambda) = |\Pr[T_0] - 1/2|. \tag{5.6}$$

Game G1: Now we define a modified game **G1**. This game behaves just like game **G0**, except that we use a completely random symmetric key $K^\dagger$ in place of the key $K^* = H(pk^{r'}_{i^*,1})$ in both the encryption and decryption oracles. Let T_1 be the event that $\sigma' = \sigma$ in game **G1**.

Now, we show that there is an oracle query machine $\mathcal{B}$, whose running time is essentially the same as that of $\mathcal{A}$, such that

$$|\Pr[T_1] - \Pr[T_0]| \leq \mathrm{Adv}^{\mathrm{ghdh}}_{\mathrm{Gen},\mathcal{B}}(\lambda) + \mathrm{Adv}^{\text{hash-TCR}}_{\mathsf{TCR}',\mathcal{B}}(\lambda) \tag{5.7}$$

$\mathcal{B}$ runs the adversary $\mathcal{A}$ as a subroutine to solve a random instance of the GHDH problem.

We now give the description of the adversary $\mathcal{B}$. The adversary $\mathcal{B}$ inputs an instance of the GHDH problem, *i.e.* $\mathcal{B}$ inputs the values $(1^k, g, g^a, g^b, Q)$. $\mathcal{B}$' goal is to determine whether $Q = H(g^{ab})$ or $Q \in \{0,1\}^\ell$ is a random bit string. The adversary $\mathcal{B}$ runs the adversary $\mathcal{A}$ simulating its view as in the original 1st-US-CCA security experiment as follows:

Setup: $\mathcal{B}$ chooses collision-resistant hash function $\mathsf{TCR} : \mathbb{G} \times \mathbb{G}_T \to \mathbb{Z}_p$ and $\mathsf{TCR}' : \mathbb{G} \to \mathbb{Z}_p$. To generate the challenge key, $\mathcal{B}$ chooses randomly $x_{i^*} \xleftarrow{\mathrm{R}} \mathbb{Z}_p, g_1 \xleftarrow{\mathrm{R}} \mathbb{G}$, and defines $pk_{i^*} = (g^{x_{i^*}}, g_1^{x_{i^*}^2}, g^a), A^* = g^b$ (meaning that $sk_{i^*} = (x_{i^*}, a), r^* = b$). $\mathcal{B}$ computes $t^* = \mathsf{TCR}'(A^*)$. $\mathcal{B}$ chooses $\alpha, u, v, d \xleftarrow{\mathrm{R}} \mathbb{Z}_p$, sets $h := pk_{i^*,3}^{-t^*} \cdot g^\alpha$. $\mathcal{B}$ sets public parameter $PP = (p, \mathbb{G}, \mathbb{G}_T, g, h, g_1, u, v, d, \mathbf{e}, H, \mathsf{TCR}, \mathsf{TCR}')$. Next, $\mathcal{B}$ generates the uncorrupted-keys and corrupted-keys as follows.

- *Uncorrupted-key generation.* $\mathcal{B}$ chooses randomly $x_i, y_i \xleftarrow{\mathrm{R}} \mathbb{Z}_p$, and defines $pk_i = (g^{x_i}, g_1^{x_i^2}, g^{y_i})$ (meaning that $sk_i = (x_i, y_i)$). $\mathcal{B}$ adds pk_i to $\mathcal{PK}_{un}$.
- *Corrupted-key generation.* $\mathcal{B}$ chooses randomly $x_i, y_i \xleftarrow{\mathrm{R}} \mathbb{Z}_p$, and defines $pk_i = (g^{x_i}, g_1^{x_i^2}, g^{y_i}), sk_i = (x_i, y_i)$. $\mathcal{B}$ adds pk_i to $\mathcal{PK}_{corr}$ and sk_i to $\mathcal{SK}_{corr}$.

$\mathcal{B}$ provides $\mathcal{A}$ with the public parameter PP, two sets of public key $\mathcal{PK} = (\mathcal{PK}_{un}, \mathcal{PK}_{corr})$, $\mathcal{SK}_{corr}$, and the challenge public key pk_{i^*}.

Find stage The adversary $\mathcal{A}$ issues a series of questions as in 1st-US-CCA game. Note that, since $\mathcal{B}$ knows all of the secret keys except $sk_{i^*,2}(= a)$, he can exactly answer every queries of $\mathcal{A}$. In particular, $\mathcal{B}$ answers these queries as follows:

ReKey Oracle $\mathcal{O}_{rk}(pk_i, pk_j)$: $\mathcal{B}$ runs the algorithm **ReKey** using $sk_{i,1}, pk_j$.

First Decryption Oracle $\mathcal{O}_{dec_1}(pk_i, CT_i)$: If $i \neq i^*$ $\mathcal{B}$ does as $\mathbf{Dec}_1$ to decrypt the ciphertext CT_i using the secret key sk_i. Otherwise, $\mathcal{B}$ does as follows.

Let $CT = (A, B, C, D)$ be an arbitrary ciphertext submitted to the decapsulation oracle $\mathcal{O}_{dec_1}$. $\mathcal{B}$ computes $t = \mathsf{TCR}(A)$, if $\mathbf{e}(A, D^t \cdot h) \neq \mathbf{e}(g, B)$ then output $\perp$ indicating an invalid ciphertext. Otherwise, do the following.

Case 1: $t = t^*$ and $A = A^*$: $\mathcal{B}$ rejects the query. In this case $B = (pk_{i^*,3}^t \cdot h)^r = (pk_{i^*,3}^{t-t^*} \cdot g^\alpha)^r = A^\alpha = (A^*)^\alpha = B^*$ and hence $C = C^*$ and the query made by $\mathcal{A}$ is illegal. Therefore it may be rejected by $\mathcal{B}$.

Case 2: $t = t^*$ and $A \neq A^*$: $\mathcal{B}$ finds a collision $A \neq A^*$ in TCR' with $\mathsf{TCR}'(A) = \mathsf{TCR}'(A^*)$. In that case $\mathcal{B}$ returns the collision and aborts.

Case 3: $t \neq t^*$: adversary $\mathcal{B}$ computes the correct session key by $K \leftarrow H((B/(A^\alpha)^{(t-t^*)^{-1}})$. $\mathcal{B}$ using this session key computes $CT_i = \mathbf{SYM.Dec}(K, CT)$, then using $sk_{i^*,1} = x_{i^*}$ computes m as the decryption algorithm.

We have shown that unless $\mathcal{B}$ finds a collision in TCR' (**Case 2**) the simulation of the decryption oracle is always perfect, i.e. the output of the simulated oracle $\mathcal{O}_{dec_1}$ is identically distributed as the output of $\mathbf{Dec}_1$.

Second Decryption Oracle $\mathcal{O}_{dec_2}(pk_i, CT_i)$: $\mathcal{B}$ does as the algorithm $\mathbf{Dec}_2$ to decrypt the ciphertext CT_i using the secret key $sk_{i,1} = x_i$, and returns the obtained result to $\mathcal{A}$.

Challenge

1. If $\mathcal{A}$ decides that the challenge ciphertext is the original one: $\mathcal{A}$ outputs two equal-length messages $m_0, m_1 \in \mathbb{G}_T$. $\mathcal{B}$ flips a coin $\sigma \xleftarrow{\mathrm{R}} \{0,1\}$, randomly chooses $r, R, s \xleftarrow{\mathrm{R}} \mathbb{Z}_p$ and does as follows:

   $C_2 = h^r$; $C_3 = \mathbf{e}(g, g_1)^r \cdot m_\sigma$; $t = \mathsf{TCR}(C_2, C_3)$; $C_4 = (u^t v^s d)^r$; $C_5 = s$; $C_6 = pk_{i^*,2}^{rR}$; $C_7 = pk_{i^*,2}^{R}$; $C_8 = g^{1/R}$; $CT'_{i^*} = C_2||C_3||...||C_8$; $A^* = g^b (= g^{r^*})$, $t^* = \mathsf{TCR}'(A^*)$, $B^* = (g^b)^\alpha (= (pk_{i^*,3}^{t^*} \cdot h)^{r^*})$, $C^* \leftarrow \mathbf{SYM.Enc}(Q, CT'_{i'})$, $D^* = pk_{i^*,3}$ Return $CT^* = (A^*, B^*, C^*, D^*)$.

2. If $\mathcal{A}$ decides that the challenge ciphertext is the re-encrypted one:

   $\mathcal{A}$ outputs two CT_0, CT_1 which can be re-encrypted from $pk_{i'}$ to pk_{i^*}. $\mathcal{B}$ flips a coin $\sigma \xleftarrow{\mathrm{R}} \{0,1\}$, chooses $R \xleftarrow{\mathrm{R}} \mathbb{Z}_p$, and does as follows:

   Parse $CT_\sigma = (C_1, ..., C_5)$, and compute: $rk_{i' \to i^*} \leftarrow \mathbf{ReKey}(sk_{i'}, pk_{i^*})$, $C_6 = C_1^R$, $C_7 = pk_{i',1}^R$; $C_8 = rk_{i' \to i^*}^{1/R}$, $CT'_{i'} = C_2||C_3||...||C_8$, $A^* = g^b (= g^{r^*})$, $t^* = \mathsf{TCR}'(A^*)$, $B^* = (g^b)^\alpha (= (pk_{i^*,3}^{t^*} \cdot h)^{r^*})$, $C^* \leftarrow \mathbf{SYM.Enc}(Q, CT'_{i'})$, $D^* = pk_{i^*,3}$. Return $CT^* = (A^*, B^*, C^*, D^*)$.

Guess stage Eventually, $\mathcal{A}$ outputs a guess $\sigma' \in \{0,1\}$, where $\sigma' = 1$ means that K^* is the correct key. Algorithm $\mathcal{B}$ concludes its own game by outputting $\gamma' = \sigma'$ where $\gamma' = 1$ means that $Q = H(g^{ab})$ and $\gamma' = 0$ means that Q is random.

ANALYSIS. We have shown that as long as there is no hash collision in TCR' found by $\mathcal{B}$, $\mathcal{A}$'s view in the simulation is identically distributed to its view in the real attack game.

Note that A^* is a random element from $\mathbb{G}$ (provided from outside of $\mathcal{B}$'s view), therefore finding a value $A \neq A^*$ with $\mathsf{TCR}'(A) = \mathsf{TCR}'(A^*)$ really contradicts to the security property of the target collision resistant hash function. The probability that $\mathcal{B}$ finds a collision in the hash function TCR' is bounded by $\mathrm{Adv}^{\text{hash-TCR}}_{\mathsf{TCR}',\mathcal{B}}(\lambda)$.

Assume there was no hash collision found by $\mathcal{B}$. On the one hand, if Q is uniform and independent in $\{0,1\}^\ell$ then the game is exactly the same as the game **G1**. On the other hand, if $Q = H(g^{ab})$, then the game is exactly the same as the game **G0**. Therefore, $\mathcal{B}$'s advantage in the GHDH game is $\mathrm{Adv}^{\text{ghdh}}_{\text{Gen},\mathcal{B}}(\lambda) \geq |\Pr[T_1] - \Pr[T_0]| - \mathrm{Adv}^{\text{hash-TCR}}_{\mathsf{TCR}',\mathcal{B}}(\lambda)$, which proves the equation 5.7.

Lastly, we observe that there is an oracle query machine $\mathcal{A}_1$, whose running time is essentially the same as that of $\mathcal{A}$, such that

$$|\Pr[T_1] - 1/2| = \mathrm{Adv}^{\text{CCA}}_{\mathbf{SYM},\mathcal{A}_1}(\lambda) \tag{5.8}$$

To see this, note that in game **G1**, the ciphertext C^* is produced using the random symmetric encryption key $K^\dagger$, and also that some other ciphertexts $C \neq C^*$ are decrypted using $K^\dagger$, but that the key $K^\dagger$ plays no other role in game **G1**. Thus, in game **G1**, the adversary $\mathcal{A}$ essentially just carries out an adaptive chosen ciphertext attack against $\mathbf{SYM}$.

The theorem now follows immediately from the equations 5.6 – 5.8.

5.6 Summary

In this chapter, we first have shown that the scheme proposed by Shao, Liu, and Zhou [55] is vulnerable to chosen-ciphertext attack. We then have presented the first unidirectional single-hop PRE scheme which is secure in the full CCA security model. The scheme is obtained by extending the PKE scheme proposed by Lai, Deng, Liu, and Kou [37]. Our scheme relies on mild complexity assumptions in bilinear groups without random oracles.

Chapter 6

CCA-Secure Conditional Schemes without Random Oracles

Contents

6.1 Introduction 93
6.2 Analysis of the Liang-Liu-Tan-Wong-Tang Scheme 94
6.2.1 Review of the Liang-Liu-Tan-Wong-Tang Scheme 94
6.2.2 Our Attack 96
6.3 The Proposed Schemes 97
6.3.1 The 2nd-sCond-CCA-Secure Scheme 97
6.3.2 The 2nd-aCond-CCA-Secure Scheme 100
6.3.3 Comparisons 101
6.4 Proofs 101
6.4.1 Proof of Theorem 11 101
6.4.2 Proof of Theorem 12 105
6.4.3 Proof of Theorem 13 108
6.4.4 Proof of Theorem 14 113
6.5 Summary 116

6.1 Introduction

Conditional proxy re-encryption (CPRE) is a variant of proxy re-encryption, where only the ciphertext satisfying one condition set by the delegator can be transformed by the proxy and then decrypted by the delegatee.

In 2009, Weng, Deng, Ding, Chu, and Lai [66] proposed an efficient CPRE scheme which is chosen-ciphertext (CCA) secure in the random oracle model, and left an open question of constructing a CCA-secure CPRE scheme without random oracles. Recently, employing CPRE in the identity-based cryptographic setting, Liang, Liu, Tan, Wong, and Tang [40] defined the notion

of identity-based conditional proxy re-encryption (IBCPRE) which can be seen as an extension of identity-based proxy re-encryption. They then proposed a construction for this primitive and claimed that their scheme achieves the security against adaptive condition and adaptive identity chosen-ciphertext attack in the standard model. One might consider that this is an answer to the open problem left by Weng et al. [68]. However, it is not known that IBCPRE, in general, implies CPRE. This is because these primitives are in different settings: in the model of IBCPRE it needs a trusted private key generator to generate all the secret keys for identities, but in that of CPRE each user can generate a secret key by himself. In addition, in this chapter, we show that the scheme proposed by Liang et al. [40] is vulnerable even to selective condition chosen-ciphertext attack (see Section 6.2 for details).

As the main goal of this chapter, we propose two CPRE schemes which are secure against the selective and adaptive condition chosen-ciphertext attack (sCond-CCA and aCond-CCA, for short), respectively. These schemes are extended from the CCA-secure PRE scheme proposed in Section 5.3. (See also [33]). Specifically, the first scheme is constructed by using the technique of constructing the selective-identity secure identity-based encryption proposed by Boneh and Boyen [13]. The other one is presented by using the technique of constructing the adaptive-identity secure identity-based encryption proposed by Waters [64]. Both of these two schemes can be proved secure without random oracles, which answers the problem left by Weng et al. [68]

6.2 Analysis of the Liang-Liu-Tan-Wong-Tang Scheme

Liang, Liu, Tan, Wong, and Tang [40] proposed a construction of identity-based CPRE and claimed that their scheme achieves secure against adaptive condition and adaptive identity chosen-ciphertext attack in the standard model. Unfortunately, this scheme is not secure even in the sense of selective condition CCA. In this section, we describe our attack to break its selective condition CCA security. See [40] for more details of the models in which the scheme is proven to be secure (we do not show them here). Note that, the difference between the selective and the adaptive condition CCA security is in the timing of choosing the challenge condition. Specifically, in the model of the selective condition CCA security the adversary has to submit the challenge condition in the beginning of the security game, but in that of the adaptive condition CCA security the adversary can submit it in the challenge phase. It is obvious that the selective condition CCA security is weaker than the adaptive condition CCA security.

6.2.1 Review of the Liang-Liu-Tan-Wong-Tang Scheme

Now we describe the Liang-Liu-Tan-Wong-Tang scheme [40].

Let $BSetup(1^\lambda)$ be an algorithm that on input the security parameter 1^λ, outputs the parameters of a bilinear map as $(q, g, \mathbb{G}_1, \mathbb{G}_2, e)$, where $\mathbb{G}_1$ and $\mathbb{G}_2$ are multiplicative cyclic groups of prime order $q, |q| = \lambda$, and g is a random generator of $\mathbb{G}_1$.

Setup(1^λ)**:** On input a security parameter 1^λ, $(q, g, \mathbb{G}_1, \mathbb{G}_2, e) \leftarrow BSetup(1^\lambda)$. Let $\omega \in \{0,1\}^n$ be an n-bit condition string. Choose $\alpha \in \mathbb{Z}_q^*, g_2, u'_1, u'_2, u'_3, u_{3,0} \in_R \mathbb{G}_1$ three random n-length sets $U_1 = \{u_{1,i} | 1 \leq i \leq n\}, U_2 = \{u_{2,i} | 1 \leq i \leq n\}, U_3 = \{u_{3,i} | 1 \leq i \leq n\}, u_{1,i}, u_{2,i}, u_{3,i} \in_R \mathbb{G}_1$, a pseudorandom function

$PRF : \mathbb{G}_2 \times \mathbb{G}_1 \to \{0,1\}^{n_1}$, and a TCR hash function $H_1 : \mathbb{G}_2 \to \mathbb{G}_1$, where n_1 is a security parameter. The master secret key is $msk = g_2^{\alpha}$, the master public key is $mpk = (n, n_1, g, g_1, g_2, u_1', u_2', u_3', u_{3,0}, U_1, U_2, U_3, PRF, H_1, (\mathbf{Sign.KeyGen}, \mathbf{Sign}, \mathbf{Verify}))$, where $g_1 = g^{\alpha}$.

$\mathbf{KeyGen}(msk, ID)$: On input a master secret key and an identity ID, the algorithm outputs $sk_{ID} = (sk_{ID_1}, sk_{ID_2}) = (g_2^{\alpha} \cdot (u_1' \prod_{i \in \mathcal{V}_{ID}})^r, g^r)$, where $r \in_R \mathbb{Z}_q^*, ID \in \{0,1\}^n$, and let $\mathcal{V}_{ID}$ be the set of all i for which the i-bit of ID is set to 1.

$\mathbf{Enc}(ID_i, \omega, m)$: On input an identity ID_i, a condition ω, and a message m, the algorithm runs $(K_S, K_V) \leftarrow \mathbf{Sign.KeyGen}(1^{\lambda})$, chooses $t \in_R \mathbb{Z}_q^*, \sigma \in_R \mathbb{G}_2$, generates the ciphertext: $C_0 = [PRF(\sigma, C_2)]^{\lambda_1 - \lambda} || [PRF(\sigma, C_2)]_{\lambda} \oplus m, C_1 = e(g_1, g_2)^t \cdot \sigma, C_2 = g^t, C_3 = (u_1' \prod_{i \in \mathcal{V}_{ID_i}})^t, C_4 = (u_2' \prod_{i \in \xi_{\omega}} u_{2,i})^t, C_5 = (u_3' u_{3,0} \prod_{i \in \mathcal{X}_{ID_i}} u_{3,i})^t, C_6 = \mathbf{Sign}(K_S, (C_0, C_2, C_3, C_4, C_5))$, and outputs $C^{(2)}_{ID_i,\omega} = (K_V, C_0, C_1, C_2, C_3, C_4, C_5)$, where $ID_i \in \{0,1\}^n, m \in 0,1^n$, let $\xi_{\omega}, \mathcal{X}_{K_V}, \mathcal{V}_{ID_i}$ be the sets of all i for which the i-bit of ω, K_V, ID_i is set to 1, respectively.

$\mathbf{ReKeyGen}(sk_{ID_i}, ID_j, \omega)$: On input a secret key sk_{ID_i}, an identity ID_j, and a condition ω, the algorithm chooses $\rho, t' \in_R \mathbb{Z}_q^*, \theta \in_R \mathbb{G}_2$, computes $rk_0 = sk_{ID_{i_1}} \cdot (u_2' \prod_{i \in \xi_{\omega}} u_{2,i})^{\rho}, rk_1 = g^{\rho}, rk_2 = sk_{ID_{i_2}}, rk_3 = e(g_1, g_2)^{t'} \cdot \theta, rk_4 = g^{t'}, rk_5 = (u_1' \prod_{i \in \mathcal{V}_{ID_i}})^{t'}, rk_6 = (u_3' u_{3,0} \prod_{i \in \mathcal{X}_{ID_i}} u_{3,i})^{t'}, rk_7 = \mathbf{Sign}(K_S', (rk_3, rk_4, rk_5, rk_6))$, and outputs $rk_{\omega | ID_i \to ID_j} = (K_V', rk_0, rk_1, rk_2, rk_3, rk_4, rk_5, rk_6, rk_7)$, where $ID_j \in \{0,1\}^n, (K_S', K_V') \leftarrow \mathbf{Sign.KeyGen}(1^{\lambda})$.

$\mathbf{ReEnc}(rk_{\omega | ID_i \to ID_j}, ID_i, \omega, C^{(2)}_{ID_i,\omega})$: On input a re-encrytption key $rk_{\omega | ID_i \to ID_j}$, an identity ID_i, a condition ω, and a ciphertext $C^{(2)}_{ID_i,\omega}$, the algorithm does as follows.

1. Verify

$$e(g, C_3) = e(C_2, u_1' \prod_{i \in \mathcal{V}_{ID_i}} u_{1,i}), e(g, C_4) = e(C_2, u_2' \prod_{i \in \xi_{\omega}} u_{2,i}),$$
$$e(g, C_5) = e(C_2, u_3' u_{3,0} \prod_{i \in \mathcal{X}_{ID_i}} u_{3,i}),$$
$$\mathbf{Verify}(K_V, C_6, (C_0, C_2, C_3, C_4, C_5)) = 1.$$

   If all of the above equations do not hold, output $\perp$. Otherwise, proceed.

2. Compute $C_1' = \frac{C_1 \cdot e(rk_2, C_3)}{e(rk_0, C_2)/e(rk_1, C_4)}$, and output the first level ciphertext $C^{(1)}_{ID_j,\omega} = (K_V, C_0, C_1, C_2, C_3, C_4, C_5, C_6, K_V', rk_3, rk_4, rk_5, rk_6, rk_7)$.

$\mathbf{Dec}_2(ID_i, sk_{ID_i}, \omega, C^{(2)}_{ID_i,\omega})$: On input an identity ID_i, a secret key sk_{ID_i}, a condition ω, and a ciphertext $C^{(2)}_{ID_i,\omega}$, the algorithm does as follows.

1. Verify the equation in the algorithm $\mathbf{ReEnc}$. If it does not hold, output $\perp$. Otherwise, proceed.

2. Compute $\sigma = C_1 \frac{e(sk_{ID_{i_2}}, C_3)}{e(sk_{ID_{i_1}}, C_2)}$, and output $m = [C_0]_\lambda \oplus [PRF(\sigma, C_2)]_\lambda$ if $[PRF(\sigma, C_2)]^{\lambda_1-\lambda} = [C_0]^{\lambda_1-\lambda}$ holds. Otherwise, output $\perp$.

$\mathbf{Dec}_1(ID_i, ID_j, sk_{ID_j}, \omega, C^{(1)}_{ID_i,\omega})$**:** On input identities ID_i, ID_j, a secret key sk_{ID_i}, a condition ω, and a ciphertext $C^{(1)}_{ID_i,\omega}$, the algorithm does as follows.

1. Verify

$$e(g, rk_5) = e(rk_4, u'_1 \prod_{i \in \mathcal{V}_{ID_i}} u_{1,i}), e(g, rk_6) = e(rk_4, u'_3 u_{3,0} \prod_{i \in \mathcal{X}_{ID_i}} u_{3,i}),$$
$$\mathbf{Verify}(K'_V, rk_7, (rk_3, rk_4, rk_5, rk_6)) = 1.$$

   If all of the above equations do not hold, output $\perp$. Otherwise, proceed.

2. Compute $\theta = rk_3 \frac{e(sk_{ID_{j_2}}, rk_5)}{e(sk_{ID_{j_1}}, rk_4)}$.
3. Verify the equation in the algorithm $\mathbf{ReEnc}$. If it does not hold, output $\perp$. Otherwise, proceed.
4. Compute $\sigma = C'_1 / e(H_1(\theta), C_3)$, and output $m = [C_0]_\lambda \oplus [PRF(\sigma, C_2)]_\lambda$ if $[PRF(\sigma, C_2)]^{\lambda_1-\lambda} = [C_0]^{\lambda_1-\lambda}$ holds. Otherwise, output $\perp$.

6.2.2 Our Attack

Before describing our attack, we first identify the potential weakness in the above scheme: for a ciphertext $CT_i = (K_V, C_0, C_1, C_2, C_3, C_4, C_5, C_6)$, where (K_S, K_V) is a pair of sign key and verify key which is generated by the algorithm $\mathbf{Sign.KeyGen}(1^\lambda)$ and $C_6 = \mathbf{Sign}(K_S, (C_0, C_2, C_3, C_4, C_5))$, their $\mathbf{ReEnc}$ algorithm transforms ciphertext component C_1 into C'_1 without verifying the validity of C_1 (it only verifies the validity of the other components, i.e. $K_V, C_0, C_2, C_3, C_4, C_5, C_6$). Then there exists an adversary who can break the CCA-security of their scheme as follows: Given the challenge ciphertext $CT^* = (K_V, C_0, C^*_1, C_2, C_3, C_4, C_5, C_6)$, the adversary can first modify the ciphertext component C^*_1 to obtain a new ciphertext CT', and then asks the re-encryption oracle to re-encrypt CT' into another ciphertext CT'_j for a corrupted user j; next, the adversary can modify CT'_j to obtain the right re-encrypted ciphertext CT_j of the challenge ciphertext, and thus he can derive the underlying plaintext by decrypting CT_j with the secret key of user j.

Attack Details. The adversary works as follows to break the selective condition CCA-security of the Liang-Liu-Tan-Wong-Tang scheme.

1. The adversary submits a condition ω^* to the challenger.
2. The adversary obtains a master public key mpk.
3. The adversary obtains a secret key sk_{ID_j} of user j from the extract oracle.
4. The adversary submits (ID^*, m_0, m_1) to the challenger, and then is given the challenge ciphertext $CT^* = (K^*_V, C^*_0, C^*_1, C^*_2, C^*_3, C^*_4, C^*_5, C^*_6)$.

5. The adversary computes $\hat{C}_1^* = C_1^* \cdot \beta$, where $\beta \leftarrow \mathbb{G}_2$, and modifies the challenge ciphertext CT^* to obtain a new ciphertext $CT' = (K_V^*, C_0^*, \hat{C}_1^*, C_2^*, C_3^*, C_4^*, C_5^*, C_6^*)$. Then the adversary submits $(ID^*, ID_j, \omega^*, CT')$ to the re-encryption oracle. Note that, it is legal for the adversary to issue this query, since $(ID^*, ID_j, \omega^*, CT') \neq (ID^*, ID_j, \omega^*, CT^*)$. Note that the re-encryption algorithm $\mathbf{ReEnc}$ cannot check the validity of the ciphertext component C_1'. So, it will return the re-encrypted ciphertext $CT_j' = \mathbf{ReEnc}(rk_{\omega^*|ID^* \to ID_j}, ID^*, \omega^*, CT')$.

   According to the re-encryption algorithm, we get $CT_j' = (K_V^*, C_0^*, \hat{C_1}', C_2^*, ..., rk_7)$, where $\hat{C_1}' = \frac{\hat{C}_1^* \cdot \mathbf{e}(rk_2, C_3^*)}{\mathbf{e}(rk_0, C_2^*)/\mathbf{e}(rk_1, C_4^*)}$. Now, from $\hat{C_1}'$, the adversary can compute $C_1' = \hat{C_1}'/\beta = \frac{C_1^* \cdot \mathbf{e}(rk_2, C_3^*)}{\mathbf{e}(rk_0, C_2^*)/\mathbf{e}(rk_1, C_4^*)}$, since $\hat{C}_1^* = C_1^* \cdot \beta$. It is easy to see that $CT_j = (K_V^*, C_0^*, C_1', C_2^*, ..., rk_7)$ is indeed the result of $\mathbf{ReEnc}(rk_{\omega^*|ID^* \to ID_j}, ID^*, \omega^*, CT^*)$, which is an encryption of m_b. Now, the adversary can obtain the underlying plaintext m_b by decrypting the re-encrypted ciphertext CT_j using the secret key sk_{ID_j}, and can break the CCA-security of the Liang-Liu-Tan-Wong-Tang scheme.

6.3 The Proposed Schemes

In this section, we propose two CPRE scheme which are 2nd-sCond-CCA-secure and 2nd-aCond-CCA-secure, respectively. Both of these two schemes meet the 1st-aCond-CCA security. Finally, we show the comparison of our results with the scheme in Section 5.3.

6.3.1 The 2nd-sCond-CCA-Secure Scheme

In this section, we first propose a CPRE scheme, and then show that it meets the 2nd-sCond-CCA and the 1st-aCond-CCA security.

Construction. Following [33], in order to achieve the CCA security given in Section 3.4.2.1, we start from the following observations which are important and necessary principles for designing CCA-secure schemes:

1. The validity of the original ciphertexts can be publicly verifiable by everyone including the proxy; otherwise, it will suffer from an attack as illustrated in [20].

2. It should be impossible for the adversary to compute the re-encryption key from the target user i^* to itself (i.e., $rk_{i^* \to i^*|\omega^*}$), otherwise it will suffer from an attack as applied to the Shao-Liu-Zhou scheme in Section 5.2.

3. For the first level ciphertext CT_j re-encrypted from the second level ciphertext CT_i, it should not exhibit any component of CT_i; otherwise, it will fail in achieving the 1st-level-CCA security.

We now describe our scheme. Our scheme is extended from the Isshiki-Nguyen-Tanaka scheme [33] by using the technique of constructing the selective-identity secure identity-based encryption proposed by Boneh and Boyen [13]. The construction is as follows.

Setup: On input a security parameter 1^n, run $\mathsf{Gen}(1^n)$ in Section 2.2.3 to obtain $(\mathbb{G}, g, p, H)$. Let $\mathbb{G}_T$ be a multiplicative group of prime order p. Let $\mathbf{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ be a bilinear map. Choose $g_1, h, u, v, d \in \mathbb{G}$, and two collision-resistant hash functions $\mathsf{TCR} : \mathbb{G} \times \mathbb{G}_T \to \mathbb{Z}_p$, $\mathsf{TCR}' : \mathbb{G} \to \mathbb{Z}_p$. Define $F(\omega) = h_1^{\omega} h_2$. $\mathbf{SYM}$ is a symmetric key encryption scheme of which the key space is $\{0,1\}^{\ell(n)}$. Output a public parameters $PP = (g, g_1, h_1, h_2, u, v, d)$.

KGen: On input a public parameter $PP = (g, g_1, h_1, h_2, u, v, d)$, choose $x, y \xleftarrow{\mathrm{R}} \mathbb{Z}_p$, and output a pair (pk, sk) of a public key and a secret key, where $pk = (g^x, g_1^{x^2}, g^y)$, $sk = (x, y)$.

RKGen: On input a secret key $sk_i = (sk_{i,1}, sk_{i,2})$, a public key $pk_j = (pk_{j,1}, pk_{j,2}, pk_{j,3})$, and a condition ω, choose $\alpha \xleftarrow{\mathrm{R}} \mathbb{Z}_p$ and output a re-encryption key $rk_{i \to j|\omega} = (pk_{j,2}^{1/sk_{i,1}} \cdot F(\omega)^{\alpha}, pk_{i,1}^{\alpha})$.

$\mathbf{Enc}_1$: On input a public key $pk_i = (pk_{i,1}, pk_{i,2}, pk_{i,3})$, and a message $M \in \mathbb{G}_{\mathrm{T}}$ choose $r, R, r', s, \alpha \xleftarrow{\mathrm{R}} \mathbb{Z}_p$ and compute $C_2 = F(0)^r$, $C_3 = \mathbf{e}(g, g_1)^r \cdot M$, $t = \mathsf{TCR}(C_2, C_3)$, $C_4 = (u^t v^s d)^r$, $C_5 = s$, $C_6 = pk_{i,2}^{rR}$, $C_7 = pk_{i,2}^{R}$, $C_8 = (g \cdot F(\omega)^{\alpha})^{1/R}$, $C_9 = pk_{i,1}^{\alpha}$, $CT_i' = C_2||C_3||...||C_8||C_9||0$, $A = g^{r'}$, $t' = \mathsf{TCR}'(A)$, $B = (pk_{i,3}^{t'} \cdot g_1)^{r'}$, $C \leftarrow \mathbf{SYM}.\mathbf{Enc}(H(pk_{i,3}^{r'}), CT_i')$. Finally, output a ciphertext $CT_i = (A, B, C)$.

$\mathbf{Enc}_2$: On input a public key $pk_i = (pk_{i,1}, pk_{i,2}, pk_{i,3})$, a message $M \in \mathbb{G}_{\mathrm{T}}$, and a condition ω choose $r, s \xleftarrow{\mathrm{R}} \mathbb{Z}_p$, and compute: $C_1 = pk_{i,1}^r$, $C_2 = F(\omega)^r$, $C_3 = \mathbf{e}(g, g_1)^r \cdot M$, $t = \mathsf{TCR}(C_2, C_3)$, $C_4 = (u^t v^s d)^r$, $C_5 = s$. Finally, output a ciphertext $CT = (C_1, C_2, C_3, C_4, C_5)$.

ReEnc: On input two public keys pk_i, pk_j, a re-encryption key $rk_{i \to j|\omega} = (rk_1, rk_2)$, an original ciphertext CT_i, and a condition ω, this algorithm does as follows:
Compute $t = \mathsf{TCR}(C_2, C_3)$, and check the validity of the ciphertext CT_i by testing whether the following equalities hold (if both hold, we write $\mathbf{Check}(CT_i) = 1$):

$$\mathbf{e}(F(\omega), C_1) = \mathbf{e}(C_2, pk_{i,1}) \tag{6.1}$$

$$\mathbf{e}(F(\omega), C_4) = \mathbf{e}(C_2, u^t v^s d). \tag{6.2}$$

If one of the above equations does not hold, output $\perp$ indicating an invalid ciphertext. Otherwise, choose $R, r' \xleftarrow{\mathrm{R}} \mathbb{Z}_p$, and compute $C_6 = C_1^R$; $C_7 = pk_{i,1}^R$; $C_8 = rk_1^{1/R}$; $CT_i' = C_2||C_3||...||C_8||rk_2||\omega$; $A = g^{r'}$, $t' = \mathsf{TCR}'(A)$, $B = (pk_{j,3}^{t'} \cdot g_1)^{r'}$, $C \leftarrow \mathbf{SYM}.\mathbf{Enc}(H(pk_{j,3}^{r'}), CT_i')$, $D = pk_{j,3}$. Finally, output a ciphertext $CT_j = (A, B, C, D)$.

$\mathbf{Dec}_2$: On input a secret key sk_i and a ciphertext $CT = (C_1, C_2, C_3, C_4, C_5)$, do as follows:
Compute $t = \mathsf{TCR}(C_2, C_3)$, and check the validity of the ciphertext CT by testing whether $\mathbf{Check}(CT) = 1$. If the verifications fail, output $\perp$ and halt. Otherwise, output $M = \frac{C_3}{\mathbf{e}(C_1, g_1)^{1/sk_{i,1}}}$.

$\mathbf{Dec}_1$: On input a secret key sk_j and a ciphertext $CT_j = (A, B, C, D)$, do as follows:
Compute $t' = \mathsf{TCR}(A)$, if $\mathbf{e}(A, D^{t'} \cdot g_1) \neq \mathbf{e}(g, B)$ then output $\perp$ indicating an invalid ciphertext. Otherwise, do the following.

Compute $CT'_i = \mathbf{SYM.Dec}(H(A^{sk_{j,2}}), C)$;

Parse $CT'_i = C_2||C_3||...||C_8||rk_2||\omega$, if this is not the case, output $\perp$ and halt.

Else, compute $t = \mathsf{TCR}(C_2, C_3)$ and check the validity of the ciphertext CT_i by testing whether the following equalities hold:

$$\mathbf{e}(F(\omega), C_4) = \mathbf{e}(C_2, u^t v^{C_5} d) \quad (6.3)$$

$$\mathbf{e}(F(\omega), C_6) = \mathbf{e}(C_2, C_7) \quad (6.4)$$

$$\mathbf{e}(C_7, C_8) = \mathbf{e}(g, pk_{j,2})\mathbf{e}(rk_2, F(\omega)). \quad (6.5)$$

If one of the above equations does not hold, output $\perp$ indicating an invalid ciphertext. Otherwise, output

$$M = \frac{C_3}{\left[\frac{\mathbf{e}(C_6,C_8)}{\mathbf{e}(C_2,rk_2)}\right]^{1/sk_{j,1}^2}}. \quad (6.6)$$

Next, we see the correctness of the scheme.

Correctness. For any condition ω and any message m, we can see that

– for $\mathbf{Dec}_2$:

$$\frac{C_3}{\mathbf{e}(C_1, g_1)^{1/sk_{i,1}}} = \frac{C_3}{\mathbf{e}(pk_{i,1}^r, g_1)^{1/sk_{i,1}}} = \frac{C_3}{\mathbf{e}(g^{r\cdot sk_{i,1}}, g_1)^{1/sk_{i,1}}} = \frac{C_3}{\mathbf{e}(g, g_1)^r} = M,$$

– for $\mathbf{Dec}_1$: if we consider $X = \frac{\mathbf{e}(C_6,C_8)}{\mathbf{e}(C_2,rk_2)}$, then equation (6.6) becomes to $M = C_3/X^{1/sk_{j,1}^2}$.

Now we can see that,

$$\begin{aligned} X &= \frac{\mathbf{e}(C_6, C_8)}{\mathbf{e}(C_2, rk_2)} = \frac{\mathbf{e}(C_1^R, rk_1^{1/R})}{\mathbf{e}(C_2, rk_2)} = \frac{\mathbf{e}(pk_{i,1}^r, pk_{j,2}^{1/sk_{i,1}} \cdot F(\omega)^\alpha)}{\mathbf{e}(F(\omega)^r, pk_{i,1}^\alpha)} \\ &= \frac{\mathbf{e}(pk_{i,1}^r, pk_{j,2}^{1/sk_{i,1}}) \cdot \mathbf{e}(pk_{i,1}^r, F(\omega)^\alpha)}{\mathbf{e}(F(\omega)^r, pk_{i,1}^\alpha)} = \mathbf{e}(pk_{i,1}^r, pk_{j,2}^{1/sk_{i,1}}) = \mathbf{e}(g, g_1)^{r\cdot sk_{j,1}^2}. \end{aligned}$$

Thus, we have $\frac{C_3}{X^{1/sk_{j,1}^2}} = \frac{C_3}{\mathbf{e}(g,g_1)^r} = M$.

Security Analysis. The intuition of the CCA security of our scheme can be seen from the following properties.

1. The validity of the original ciphertexts can be publicly verifiable by everyone including the proxy; otherwise, it will suffer from an attack as illustrated in [20]. For our scheme, the ciphertext components C_4, C_5 in the original ciphertext $CT = (C_1, C_2, C_3, C_4, C_5)$ can be viewed as a signature signing the messageC_1, C_2, C_3, that is how we get public verifiability.

2. It should be impossible for the adversary to compute the re-encryption key from the target user i^* to itself (i.e., $rk_{i^*\to i^*|\omega^*}$), otherwise it will suffer from an attack as applied to the Shao-Liu-Zhou scheme in Section 5.2. This follows from Lemma 4.

3. For the first level ciphertext CT_j re-encrypted from a second level ciphertext CT_i, it should not exhibit any component of CT_i; otherwise, it will fail in achieving the CCA-security of $\mathbf{ReEnc}$. In our scheme, all of the components from the original ciphertext are hidden in C.

The security of the above scheme is guaranteed by the following theorems.

Theorem 11. *The above scheme meets the 2nd-sCond-CCA security, assuming the hash function* TCR *is target-collision resistant and the 6-AmDBDH assumption holds in groups* $(\mathbb{G}, \mathbb{G}_T)$.

The proof of Theorem 11 is given in Section 6.4.1.

Theorem 12. *Our scheme meets the 1st-aCond-CCA security, assuming* TCR' *is a target-collision resistant hash function, the GHDH problem is hard, and* $\mathbf{SYM}$ *is CCA-secure.*

Remark 3. In the algorithms $\mathbf{Enc}_1$ and $\mathbf{ReEnc}$, we make use of the encryption algorithm of Kiltz's KEM/DEM scheme [36] to mask all of computed components including the second-level ciphertext. Therefore, the 1st-aCond-CCA security of our scheme is implied by the CCA security of Kiltz's KEM/DEM scheme.

The proof of Theorem 12 is given in Section 6.4.2.

6.3.2 The 2nd-aCond-CCA-Secure Scheme

In this section, we present a CPRE scheme which is adaptive condition CCA-secure. This scheme is almost the same as the scheme in the previous section, except that the function $F(\omega)$ is replaced by the Waters hash [64]. In particular, all algorithms of the scheme except the setup algorithm are exactly the same as those of the CPRE scheme in Section 6.3.1. The setup algorithm is as follows.

Setup: $(p, g, \mathbb{G}, \mathbb{G}_T, \mathbf{e}, H) \leftarrow_R \mathrm{Setup}(1^n)$, where $(\mathbb{G}, g, p, H)$ is random parameters obtained by running the parameter algorithm $\mathsf{Gen}(1^\lambda)$, and $H : \mathbb{G} \to \{0,1\}^{\ell(n)}$ is a random instance of a hash function such that the GHDH assumptions holds relative to Gen. Choose $g_1, u, v, d \in_{\mathrm{R}} \mathbb{G}$, and collision-resistant hash functions $\mathsf{TCR} : \mathbb{G} \times \mathbb{G}_T \to \mathbb{Z}_p$, $\mathsf{TCR}' : \mathbb{G} \to \mathbb{Z}_p$. Define $F(\omega) = u' \prod_{i \in \Omega} u_i$, where $u', u_1, \ldots, u_n \in_{\mathrm{R}} \mathbb{G}$, and $\Omega \subseteq \{1, 2, \ldots, n\}$ be the set of all i for which the ith bit of ω is equal 1. $\mathbf{SYM}$ is a symmetric encryption scheme of which the key space is $\{0,1\}^{\ell(n)}$. Return $PP = (g, g_1, u, v, d, u', u_1, \ldots, u_n)$.

Security Analysis. The security of the above scheme is guaranteed by the following theorems.

Theorem 13. *The above scheme meets the 2nd-aCond-CCA security, assuming the hash function* TCR *is target-collision resistant and the 6-AmDBDH assumption holds in groups* $(\mathbb{G}, \mathbb{G}_T)$.

The proof of Theorem 13 is given in Section 6.4.3.

Theorem 14. *Our scheme meets the 1st-aCond-CCA security, assuming* TCR' *is a target-collision resistant hash function, the GHDH problem is hard, and* $\mathbf{SYM}$ *is CCA-secure.*

The proof of Theorem 14 is almost the same as that of Theorem 12, where the 1st-aCond-CCA security of the scheme is implied by the CCA security of Kiltz's KEM/DEM scheme. The details are given in Section 6.4.3.

6.3.3 Comparisons

In Table 4, we compare our schemes with the scheme in Section 5.3. We first explain some notations used in the table. By n we denote the security parameter. $|\mathbb{G}|, |\mathbb{G}_T|$, and $|\mathbf{SYM}|$ denote the bit-length of an element in groups $\mathbb{G}, \mathbb{G}_T$ and a ciphertext output by $\mathbf{SYM.Enc}$, respectively. We use $t_p, t_e, t_{SYM.Enc}, t_{SYM.Dec}$ to represent the computational cost of a bilinear pairing, an exponentiation, an encryption and a decryption in the symmetric key encryption $\mathbf{SYM}$, respectively. "—" means that the condition CCA security of the corresponding scheme is not considered.

The comparison results indicate that our schemes have almost the same computational cost as the scheme in Section 5.3. It means that our schemes are efficient as the underlying scheme. Our schemes have also the same length of a public key, a secret key, and a ciphertext as the scheme in Section 5.3. In comparison on the length of the parameter, that of the first scheme (i.e. the one in Section 6.3.1) is little bit longer than that of the underlying PRE scheme; whereas the other one is quite longer than that of the underlying PRE scheme, since we make use of the Waters hash in the construction.

Schemes		Ours (Sec. 6.3.1)	Ours (Sec. 6.3.2)	Ours (Sec. 5.3)
Length	parameter	$7\|\mathbb{G}\|$	$(6+n)\|\mathbb{G}\|$	$6\|\mathbb{G}\|$
	public key	$3\|\mathbb{G}\|$	$3\|\mathbb{G}\|$	$3\|\mathbb{G}\|$
	secret key	$2\|\mathbb{Z}_p\|$	$2\|\mathbb{Z}_p\|$	$2\|\mathbb{Z}_p\|$
	re-encryption key	$2\|\mathbb{G}\|$	$2\|\mathbb{G}\|$	$1\|\mathbb{G}\|$
	2nd-level ciphertext	$3\|\mathbb{G}\| + 1\|\mathbb{G}_T\| + 1\|\mathbb{Z}_p\|$	$3\|\mathbb{G}\| + 1\|\mathbb{G}_T\| + 1\|\mathbb{Z}_p\|$	$3\|\mathbb{G}\| + 1\|\mathbb{G}_T\| + 1\|\mathbb{Z}_p\|$
	1st-level ciphertext	$2\|\mathbb{G}\| + 1\|\mathbf{SYM}\|$	$2\|\mathbb{G}\| + 1\|\mathbf{SYM}\|$	$2\|\mathbb{G}\| + 1\|\mathbf{SYM}\|$
Cost	$\mathbf{Enc}_2$	$1t_p + 7t_e$	$1t_p + 7t_e$	$1t_p + 6t_e$
	$\mathbf{Enc}_1$	$1t_p + 12t_e + t_{SYM.Enc}$	$1t_p + 12t_e + t_{SYM.Enc}$	$1t_p + 11t_e + t_{SYM.Enc}$
	$\mathbf{ReEnc}$	$4t_p + 9t_e + t_{SYM.Enc}$	$4t_p + 9t_e + t_{SYM.Enc}$	$4t_p + 8t_e + t_{SYM.Enc}$
	$\mathbf{Dec}_2$	$1t_p + 1t_e$	$1t_p + 1t_e$	$1t_p + 1t_e$
	$\mathbf{Dec}_1$	$9t_p + 5t_e + t_{SYM.Dec}$	$9t_p + 5t_e + t_{SYM.Dec}$	$9t_p + 5t_e + t_{SYM.Dec}$
Condition CCA Security		selective	adaptive	—

Table 4: Comparison of our CPRE scheme with the PRE scheme in Section 5.3.

6.4 Proofs

6.4.1 Proof of Theorem 11

We prove that our proposed scheme is 2nd-sCond-CCA secure under the 6-AmDBDH assumption. We build an algorithm $\mathcal{B}$ which is, given a 6-AmDBDH instance $(g, g^a, g^b, g^c, g^{a/c}, g^{c^2}, g^{ac}, g^{ac^2}, g^{ac^3}, g^{ac^4}, Q) \in \mathbb{G}^{10} \times \mathbb{G}_T$, deciding whether $Q = \mathbf{e}(g,g)^{ab/c^2}$, using a 2nd-sCond-CCA adversary $\mathcal{A}$. $\mathcal{B}$ runs the adversary $\mathcal{A}$ simulating its view as in the 2nd-sCond-CCA security experiment as follows:

Initiation: The game begins with $\mathcal{A}$ first outputting a condition ω^* that it intends to attack.

Setup: $\mathcal{B}$ chooses $x_v, x_d, y_u, y_v, y_d \xleftarrow{\mathrm{R}} \mathbb{Z}_p$, sets $g_1 := g^a, h_1 = g^{a/c}, h_2 = h_1^{-\omega^*} g^{c^2}$, and computes $u = g \cdot (g^{c^2})^{y_u}, v = g^{x_v} \cdot (g^{c^2})^{y_v}, d = g^{x_d} \cdot (g^{c^2})^{y_d}$. The other parameters are

chosen as in the algorithm $\mathbf{Setup}$. The public parameter is $PP = (g, g_1, h_1, h_2, u, v, d)$. As before, we define $F(\omega) = h_1^{\omega} h_2 = h_1^{\omega-\omega^*} g^{c^2}$. Next, $\mathcal{B}$ generates the challenge key, uncorrupted-keys, and corrupted-keys as follows.

– *Challenge-key generation.* $\mathcal{B}$ chooses $x_{i^*}, y_{i^*} \xleftarrow{\mathrm{R}} \mathbb{Z}_p$, and defines $pk_{i^*} = ((g^{c^2})^{x_{i^*}}, (g^{ac^4})^{x_{i^*}^2}, g^{y_{i^*}})$ (meaning that $sk_{i^*} = (c^2 x_{i^*}, y_{i^*})$).

– *Uncorrupted-key generation.* $\mathcal{B}$ chooses $x_i, y_i \xleftarrow{\mathrm{R}} \mathbb{Z}_p$, and defines $pk_i = ((g^c)^{x_i}, (g^{ac^2})^{x_i^2}, g^{y_i})$ (meaning that $sk_i = (cx_i, y_i)$). $\mathcal{B}$ adds pk_i to $\mathcal{PK}_{un}$.

– *Corrupted-key generation.* $\mathcal{B}$ chooses $x_i, y_i \xleftarrow{\mathrm{R}} \mathbb{Z}_p$, and defines $pk_i = (g^{x_i}, (g^a)^{x_i^2}, g^{y_i})$, $sk_i = (x_i, y_i)$. $\mathcal{B}$ adds pk_i to $\mathcal{PK}_{corr}$ and sk_i to $\mathcal{SK}_{corr}$.

$\mathcal{B}$ provides $\mathcal{A}$ with the public parameter PP, two sets of public key $\mathcal{PK} = (\mathcal{PK}_{un}, \mathcal{PK}_{corr})$, $\mathcal{SK}_{corr}$, and the challenge public key pk_{i^*}.

Find stage. The adversary $\mathcal{A}$ issues a series of questions as in the 2nd-sCond-CCA game. $\mathcal{B}$ answers these queries for $\mathcal{A}$ by simulating the oracles as follows.

Re-Encryption Key Generation Oracle $\mathcal{O}_{rk}(pk_i, pk_j, \omega)$: $\mathcal{B}$ chooses $\alpha \xleftarrow{\mathrm{R}} \mathbb{Z}_p$ and does as follows.

- If $\omega = \omega^*$ (meaning that $F(\omega) = g^{c^2}$):
  1. If $pk_i = pk_{i^*}$ and $pk_j \in \mathcal{PK}_{corr}$, then return $\perp$.
  2. If $pk_i, pk_j \in \mathcal{PK}_{corr}$, then return $rk_{i\to j|\omega} = (g_1^{x_j^2/x_i} \cdot F(\omega)^{\alpha}, pk_{i,1}^{\alpha})$.
  3. If $pk_i, pk_j \in \mathcal{PK}_{un}$, then return $rk_{i\to j|\omega} = ((g^{ac})^{x_j^2/x_i} \cdot F(\omega)^{\alpha}, pk_{i,1}^{\alpha})$.
  4. If $pk_i \in \mathcal{PK}_{un}$ and $pk_j \in \mathcal{PK}_{corr}$, then return $rk_{i\to j|\omega} = ((g^{a/c})^{x_j^2/x_i} \cdot F(\omega)^{\alpha}, pk_{i,1}^{\alpha})$.
  5. If $pk_i \in \mathcal{PK}_{corr}$ and $pk_j \in \mathcal{PK}_{un} \cup \{pk_{i^*}\}$, then return $rk_{i\to j|\omega} = (pk_{j,2}^{1/x_i} \cdot F(\omega)^{\alpha}, pk_{i,1}^{\alpha})$.
  6. If $pk_i = pk_{i^*}$ and $pk_j \in \mathcal{PK}_{un}$, then return $rk_{i^*\to j|\omega} = ((g^{ac})^{x_j^2/x_{i^*}} \cdot F(\omega)^{\alpha}, pk_{i,1}^{\alpha})$.
  7. If $pk_i \in \mathcal{PK}_{un}$ and $pk_j = pk_{i^*}$, then return $rk_{i\to i^*|\omega} = ((g^{ac^3})^{x_{i^*}^2/x_i} \cdot F(\omega)^{\alpha}, pk_{i,1}^{\alpha})$.
- If $\omega \neq \omega^*$:
  1. If $pk_i = pk_{i^*}$ and $pk_j \in \mathcal{PK}_{corr}$, then return the re-encryption key as

$$rk_{i^*\to j|\omega} = (R_1, R_2) = \left(\left((g^c)^{\frac{-1}{\omega-\omega^*}} F(\omega)^{\alpha} \right)^{\frac{x_j^2}{x_{i^*}}}, \left((g^c)^{\frac{-1}{\omega-\omega^*}} g^{c^2\alpha} \right)^{x_j^2} \right).$$

Let $\tilde{\alpha} = \left(\alpha - \frac{1}{c(\omega-\omega^*)}\right) \cdot \frac{x_j^2}{x_{i^*}}$, and note that $F(\omega) = h_1^{\omega-\omega^*} g^{c^2}$. Then we have

$$\begin{aligned} R_1 &= \left((g^c)^{\frac{-1}{\omega-\omega^*}} F(\omega)^{\alpha}\right)^{\frac{x_j^2}{x_{i^*}}} \\ &= \left(h_1^{\frac{1}{c}} \left(h_1^{\omega-\omega^*} g^{c^2}\right)^{-\frac{1}{c(\omega-\omega^*)}} F(\omega)^{\alpha}\right)^{\frac{x_j^2}{x_{i^*}}} \\ &= \left(h_1^{\frac{1}{c}} F(\omega)^{\alpha-\frac{1}{c(\omega-\omega^*)}}\right)^{\frac{x_j^2}{x_{i^*}}} \\ &= g^{a\frac{x_j^2}{c^2 x_{i^*}}} F(\omega)^{\tilde{\alpha}} \\ &= pk_{j,2}^{1/sk_{i^*,1}} \cdot F(\omega)^{\tilde{\alpha}}. \end{aligned}$$

Additionally, we have

$$R_2 = \left((g^c)^{\frac{-1}{\omega-\omega^*}} g^{c^2\alpha}\right)^{x_j^2} = \left((g^{c^2 x_{i^*}})^{\alpha-\frac{1}{c(\omega-\omega^*)}}\right)^{\frac{x_j^2}{x_{i^*}}} = pk_{i^*,1}^{\tilde{\alpha}}.$$

Thus, we get $rk_{i^*\to j|\omega} = \left(pk_{j,2}^{1/sk_{i^*,1}} \cdot F(\omega)^{\tilde{\alpha}}, pk_{i^*,1}^{\tilde{\alpha}}\right)$. Note that, $\mathcal{B}$ will be able to generate this re-encryption key iff $F(\omega) \neq 0 \bmod p$.

2. Otherwise, $\mathcal{B}$ does as the above cases in which $\omega = \omega^*$.

First Decryption Oracle $\mathcal{O}_{dec_1}(pk_i, CT_i)$: If $pk_i \in \mathcal{PK}_{corr}$, $\mathcal{B}$ does as the $\mathbf{Dec}_1$ to decrypt the ciphertext CT_i using secret key sk_i. Otherwise, $\mathcal{B}$ does as follows. $\mathcal{B}$ parses $CT_i = (A, B, C, D)$, $\mathcal{B}$ computes $t' = \mathsf{TCR}(A)$, and checks whether the following equality hold: $\mathbf{e}(A, D^{t'} \cdot g_1) = \mathbf{e}(g, B)$. If this verification fails, output $\perp$ indicating an invalid ciphertext. Otherwise, it does the following.

Compute $CT_i' = \mathbf{SYM.Dec}(H(A^{y_i}), C)$;

Parse $CT_i' = C_2||C_3||...||C_8||rk_2||\omega$, if this is not the case, output $\perp$ and halt.

Else, $\mathcal{B}$ computes $t = \mathsf{TCR}(C_2, C_3)$ and does as follows:

1. Check the validity of the ciphertext CT_i as the equations 6.3 – 6.5 in $\mathbf{Dec}_1$.
2. Check whether $(C_2, C_3) = (C_2^*, C_3^*)$ and $t \neq t^*$. If so, abort and output a random bit.
3. Check whether $t + sx_v + x_d = 0$. If so, abort and output a random bit.

If all of these verifications pass, then there exists $r \in \mathbb{Z}_p$ such that $C_1 = pk_{i,1}^r, C_4 = (u^t v^s d)^r, s = C_5$. If $(C_2, C_3, C_4, C_5) = (C_2^*, C_3^*, C_4^*, C_5^*)$, $\mathcal{B}$ outputs $\perp$ which deems CT_i as a derivative of the challenge pair (pk_{i^*}, CT^*). We now assume $C_3 \neq C_3^*$. If $\omega \neq \omega^*$, then $\mathcal{B}$ computes re-encryption key $rk_{j\to i'|\omega}$ as in **Re-Encryption Key Generation Oracle** $\mathcal{O}_{rk}$, where $i' \in \mathcal{PK}_{corr}$. Using this re-encryption key, $\mathcal{B}$ computes C_6, C_7, C_8 as in the re-encryption algorithm, then computes m as the equation 6.6. Otherwise ($\omega = \omega^*$), since $F(\omega^*) = g^{c^2}$, we have

$C_2 = g^{c^2 r}$. Therefore $\mathcal{B}$ can compute $X := C_2^{(ty_u + sy_v + y_d)} = g^{c^2 r(ty_u + sy_v + y_d)}$. Since $C_4 = (u^t v^s d)^r = g^{r(t+sx_v+x_d)} \cdot g^{c^2 r(ty_u+sy_v+y_d)}$ we have $g^r = (C_4/X)^{1/(t+sx_v+x_d)}$. Therefore, $\mathcal{B}$ can compute g^r by computing $(C_4/X)^{1/(t+sx_v+x_d)}$. Finally, $\mathcal{B}$ computes $M = \frac{C_3}{\mathbf{e}(g^r, g_1)}$, returns it to $\mathcal{A}$.

Second Level Decryption Oracle $\mathcal{O}_{dec_2}(pk_i, \omega, CT_i)$**:** $\mathcal{B}$ does as follows.

- If $\omega = \omega^*$ (meaning that $F(\omega) = g^{c^2}$):
  1. If $pk_i \in \mathcal{PK}_{corr}$, then do as the algorithm $\mathbf{Dec}_2$ to decrypt the ciphertext CT_i using the secret key $sk_i = (x_i, y_i)$, and return the obtained result to $\mathcal{A}$.
  2. If $pk_i \in \mathcal{PK}_{un}$, then $\mathcal{B}$ first computes a re-encryption key $rk_{i \to j|\omega} = ((g^{a/c})^{x_j^2/x_i} \cdot F(\omega)^{\alpha}, pk_{i,1}^{\alpha})$, where $pk_j \in \mathcal{PK}_{corr}$. $\mathcal{B}$ then re-encrypts the ciphertext CT_i by using $rk_{i \to j|\omega}$ to obtain $CT_j \leftarrow \mathbf{ReEnc}(rk_{i \to j|\omega}, CT_i)$. Finally, $\mathcal{B}$ does as the algorithm $\mathbf{Dec}_1$ to decrypt the ciphertext CT_j using the secret key $sk_j = (x_j, y_j)$, and returns the obtained result to $\mathcal{A}$.
  3. If $i = i^*$ and $CT_i \neq CT^*$, then $\mathcal{B}$ first computes a re-encryption key $rk_{i^* \to j|\omega} = ((g^{ac})^{x_j^2/x_{i^*}} \cdot F(\omega)^{\alpha}, pk_{i,1}^{\alpha})$, where $pk_j \in \mathcal{PK}_{un}$. $\mathcal{B}$ then re-encrypts the ciphertext CT_i by using $rk_{i^* \to j}$ to obtain $CT_j \leftarrow \mathbf{ReEnc}(rk_{i^* \to j}, CT_i)$. Finally, $\mathcal{B}$ does as in **First Decryption Oracle** $\mathcal{O}_{dec_1}$ to decrypt the ciphertext CT_j, and returns the obtained result to $\mathcal{A}$.
- If $\omega \neq \omega^*$: $\mathcal{B}$ first computes $rk_{i \to j|\omega}$ as in **Re-Encryption Key Generation Oracle** $\mathcal{O}_{rk}$, where $pk_j \in \mathcal{PK}_{corr}$. $\mathcal{B}$ then re-encrypts the ciphertext CT_i by using $rk_{i \to j|\omega}$ to obtain $CT_j \leftarrow \mathbf{ReEnc}(rk_{i \to j|\omega}, CT_i)$. Finally, $\mathcal{B}$ does as the algorithm $\mathbf{Dec}_1$ to decrypt the ciphertext CT_j using the secret key $sk_j = (x_j, y_j)$, and returns the obtained result to $\mathcal{A}$.

Re-Encryption Oracle $\mathcal{O}_{re}(pk_i, pk_j, CT_i)$**:** $\mathcal{B}$ parses $CT_i = (C_1, C_2, C_3, C_4, C_5)$. If this is not the case, output $\perp$ and halt. Otherwise, $\mathcal{B}$ computes $t = \mathsf{TCR}(C_2, C_3)$ and check if $Check(CT_i) = 1$. If this is not the case, output $\perp$ and halt; else $\mathcal{B}$ works according to the following cases:

1. If $pk_i = pk_{i^*}, pk_j \in \mathcal{PK}_{corr}, \omega = \omega^*$ (meaning that $sk_{i^*} = c^2 x_{i^*}$ and $sk_j = x_j$): $\mathcal{B}$ chooses $R', r', \alpha' \xleftarrow{\mathrm{R}} \mathbb{Z}_p$ and computes g^r as in **First Decryption Oracle** $\mathcal{O}_{dec_1}$. Using g^r, $\mathcal{B}$ computes: $C_6 = (g^r)^{x_{i^*} R'} (= (g^{c^2 x_{i^*} r})^{R'/c^2} = C_1^R)$, where $R = R'/c^2$; $C_7 = g^{x_{i^*} R'} (= (g^{c^2 x_{i^*}})^{R'/c^2} = pk_{i^*,1}^R)$; $C_8 = (g_1^{x_j^2/x_{i^*}})^{1/R'} F(\omega^*)^{\alpha'} (= (g_1^{sk_{j,1}^2/sk_{i^*,1}} F(\omega^*)^{\alpha})^{c^2/R'} = rk_1^{1/R})$, where $\alpha = \alpha' R'/c^2$ and $rk_{i^* \to j|\omega^*} = (rk_1, rk_2)$; $C_9 = g^{x_{i^*} \alpha' R'} (= pk_{i^*,1}^{\alpha' R'/c^2} = pk_{i^*,1}^{\alpha} = rk_2)$; $CT'_{i^*} = C_2 || C_3 || ... || C_8 || C_9 || \omega^*$; $A = g^{r'}$, $t' = \mathsf{TCR}'(A)$, $B = (pk_{i^*,3}^{t'} \cdot g_1)^{r'}$, $C \leftarrow \mathbf{SYM.Enc}(H(pk_{i^*,3}^{r'}), CT'_{i^*})$, $D = pk_{i^*,3}$. Output $CT_j = (A, B, C, D)$.
2. Otherwise, $\mathcal{B}$ executes the algorithm $\mathbf{ReEnc}$ with the re-encryption key computed as in $\mathcal{O}_{rk}(pk_i, pk_j, \omega)$, then returns $CT_j \leftarrow \mathbf{ReEnc}(rk_{i \to j|\omega}, CT_i)$ to $\mathcal{A}$.

Challenge. At this stage, $\mathcal{A}$ outputs two messages $M_0, M_1 \in \mathbb{G}_T$ and a condition ω^*. $\mathcal{B}$ flips a coin $\sigma \xleftarrow{\mathrm{R}} \{0,1\}$ and defines: $C_1^* = (g^b)^{x_{i^*}}, C_2^* = g^b, C_3^* = Q \cdot M_\sigma, C_5^* = s^* = -(t^* + x_d)/x_v$ where $t^* = \mathsf{TCR}(C_2^*, C_3^*)$, and $C_4^* = (g^b)^{t^* y_u + s^* y_v + y_d}$. Return $CT^* = (C_1^*, C_2^*, C_3^*, C_4^*, C_5^*)$ as the challenge ciphertext to $\mathcal{A}$.

Observe the following to see that, if $Q = \mathbf{e}(g,g)^{ab/c^2}$, then CT^* is indeed a valid challenge ciphertext under public key pk_{i^*} with the random exponent $r = b/c^2$.
$C_1^* = (g^b)^{x_{i^*}} = (g^{c^2 x_{i^*}})^{b/c^2} = pk_{i^*,1}^r$; $C_2^* = g^b = (g^{c^2})^{b/c^2} = F(\omega^*)^r$; $C_3^* = Q \cdot M_\sigma$; $t^* = \mathsf{TCR}(C_2^*, C_3^*)$; $C_5^* = s^* = -(t^* + x_d)/x_v$; $C_4^* = (g^b)^{t^* y_u + s^* y_v + y_d} = (g^{c^2(t^* y_u + s^* y_v + y_d)})^{b/c^2} \cdot (g^{t^* + s^* x_v + x_d})^{b/c^2} = (u^{t^*} v^{s^*} d)^r$.

In contrast, if Q is random in $\mathbb{G}_T$, then CT^* perfectly hides m_σ and $\mathcal{A}$ cannot guess σ with better probability than $1/2$.

Guess stage. Adversary $\mathcal{A}$ continues to issue the rest queries. In this stage, $\mathcal{B}$ can respond these queries for $\mathcal{A}$ as in the find stage.

Output. Eventually, adversary $\mathcal{A}$ returns a guess $\sigma' \in \{0,1\}$. If $\sigma' = \sigma$, $\mathcal{B}$ outputs 1 to guess $Q = \mathbf{e}(g,g)^{ab/c^2}$; else $\mathcal{B}$ outputs 0 to guess that Q is a random element in $\mathbb{G}_T$.

This completes the description of the simulation.

Next, we analyze the simulation.

In the simulation of the first level decryption oracle $\mathcal{O}_{dec_1}$, $\mathcal{B}$ aborts in two cases, those are in the second and third conditions. The probability that $\mathcal{B}$ aborts in the second condition of $\mathcal{O}_{dec_1}$ is at most $\mathrm{Adv}^{\mathrm{TCR}}_{\mathsf{TCR},\mathcal{A}}$. For the challenge ciphertext, the adversary could obtain the information that $t^* + s^* x_v + x_d = 0$. However, there are exactly p possible (x_v, x_d) pairs that satisfy this equation and each of them are equally likely. Thus, information-theoretically, the probability that $t + s x_v + x_d = 0$ is at most $1/p$. Hence, the probability that $\mathcal{B}$ aborts in the third condition of $\mathcal{O}_{dec_1}$ is at most q_{dec}/p. Therefore, the probability that $\mathcal{B}$ aborts during the simulation of oracle $\mathcal{O}_{dec_1}$ is at most $\mathrm{Adv}^{\mathrm{TCR}}_{\mathsf{TCR},\mathcal{A}} + q_{dec}/p$.

Observe that, if $\mathcal{B}$ does not abort during the simulation and $Q = \mathbf{e}(g,g)^{ab/c^2}$, then the simulation of $\mathcal{B}$ is perfect from the view of $\mathcal{A}$. On the other hand, if $\mathcal{B}$ does not abort during the simulation and Q is distributed uniformly and independently over $\mathbb{G}_T$ then the advantage that $\mathcal{A}$ wins the game is $1/2$. Since the simulation for $\mathcal{O}_{re}, \mathcal{O}_{rk}$, we have $\mathrm{Adv}^{\text{6-AmDBDH}}_{\mathcal{B}}(n) \geq \frac{1}{2} \cdot \mathrm{Adv}^{\text{2nd-sCond}}_{\mathrm{CPRE},\mathcal{A}}(n) - \mathrm{Adv}^{\mathrm{TCR}}_{\mathsf{TCR},\mathcal{A}} - q_{dec}/p$.

This concludes the proof of Theorem 11.

6.4.2 Proof of Theorem 12

We will prove the theorem using a sequences of game transitions.

Game G0: This is the original 1st-aCond-CCA game, which is the CCA game as in Definition 22 with the additional constraints from Definition 31. The adversary $\mathcal{A}$ has access to the re-encryption key oracle $\mathcal{O}_{rk}$ and the decryption oracle $\mathcal{O}_{dec_1}$, gets a target ciphertext $CT^* = (A^*, B^*, C^*)$, where CT^* is computed as follows:

1. If the challenge ciphertext is the original one: $CT^* \leftarrow \mathbf{Enc}_1(pk_{i^*}, m_\sigma)$.

2. If the challenge ciphertext is the re-encrypted one: $CT^* \leftarrow \mathbf{ReEnc}(rk_{i' \to i^*}, CT_\sigma)$, where $CT_\sigma = (C_1, ..., C_5)$ is a "good message" which can be re-encrypted from $pk_{i'}$ to pk_{i^*}.

Let $\sigma' \in \{0,1\}$ denote the output of $\mathcal{A}$, and let T_0 be the event that $\sigma' = \sigma$ in **G0**, so that

$$\mathrm{Adv}^{\text{1st-aCond-CCA}}_{\mathrm{CPRE},\mathcal{A}}(\lambda) = |\Pr[T_0] - 1/2|. \tag{6.7}$$

Game G1: Now we define a modified game **G1**. This game behaves just like game **G0**, except that we use a completely random symmetric key $K^\dagger$ in place of the key $K^* = H(pk^{r'}_{i^*,1})$ in both the encryption and decryption oracles. Let T_1 be the event that $\sigma' = \sigma$ in game **G1**.

Now, we show that there is an oracle query machine $\mathcal{B}$, whose running time is essentially the same as that of $\mathcal{A}$, such that

$$|\Pr[T_1] - \Pr[T_0]| \leq \mathrm{Adv}^{\text{ghdh}}_{\mathrm{Gen},\mathcal{B}}(\lambda) + \mathrm{Adv}^{\text{hash-TCR}}_{\mathsf{TCR}',\mathcal{B}}(\lambda). \tag{6.8}$$

$\mathcal{B}$ runs the adversary $\mathcal{A}$ as a subroutine to solve a random instance of the GHDH problem.

We now give the description of the adversary $\mathcal{B}$. The adversary $\mathcal{B}$ inputs an instance of the GHDH problem, *i.e.* $\mathcal{B}$ inputs the values $(1^k, g, g^a, g^b, Q)$. $\mathcal{B}$' goal is to determine whether $Q = H(g^{ab})$ or $Q \in \{0,1\}^\ell$ is a random bit string. The adversary $\mathcal{B}$ runs adversary $\mathcal{A}$ simulating its view as in the original 1st-aCond-CCA security experiment as follows:

Setup: $\mathcal{B}$ chooses collision-resistant hash functions $\mathsf{TCR} : \mathbb{G} \times \mathbb{G}_T \to \mathbb{Z}_p$ and $\mathsf{TCR}' : \mathbb{G} \to \mathbb{Z}_p$. To generate challenge key, $\mathcal{B}$ chooses randomly $x_{i^*}, \alpha, u, v, d \leftarrow_R \mathbb{Z}_p$, sets $g_1 := pk_{i^*,3}^{-t^*} \cdot g^\alpha$, and defines $pk_{i^*} = (g^{x_{i^*}}, g_1^{x_{i^*}^2}, g^a), A^* = g^b$ (meaning that $sk_{i^*} = (x_{i^*}, a), r^* = b$). $\mathcal{B}$ computes $t^* = \mathsf{TCR}'(A^*)$.

The other parameters are chosen as in the algorithm $\mathbf{Setup}$. The public parameter is $PP = (p, \mathbb{G}, \mathbb{G}_T, g, g_1, u, v, d, \mathsf{e}, H, \mathsf{TCR}, \mathsf{TCR}', F)$. Next, $\mathcal{B}$ generates the uncorrupted-keys and corrupted-keys as follows.

- *Uncorrupted-key generation.* $\mathcal{B}$ chooses randomly $x_i, y_i \leftarrow_R \mathbb{Z}_p$, and defines $pk_i = (g^{x_i}, g_1^{x_i^2}, g^{y_i})$ (meaning that $sk_i = (x_i, y_i)$). $\mathcal{B}$ adds pk_i to $\mathcal{PK}_{un}$.
- *Corrupted-key generation.* $\mathcal{B}$ chooses randomly $x_i, y_i \leftarrow_R \mathbb{Z}_p$, and defines $pk_i = (g^{x_i}, g_1^{x_i^2}, g^{y_i}), sk_i = (x_i, y_i)$. $\mathcal{B}$ adds pk_i to $\mathcal{PK}_{corr}$ and sk_i to $\mathcal{SK}_{corr}$.

$\mathcal{B}$ provides $\mathcal{A}$ with the public parameter PP, two sets of public key $\mathcal{PK} = (\mathcal{PK}_{un}, \mathcal{PK}_{corr})$, $\mathcal{SK}_{corr}$, and the challenge public key pk_{i^*}.

Find stage. The adversary $\mathcal{A}$ issues a series of questions as in 1st-Cond-CCA game. Note that, since $\mathcal{B}$ knows all of the secret keys except $sk_{i^*,2}(= a)$, he can exactly answer every queries of $\mathcal{A}$. In particular, $\mathcal{B}$ answers these queries for $\mathcal{A}$ as follows:

ReKey Oracle $\mathcal{O}_{rk}(pk_i, pk_j, \omega)$: $\mathcal{B}$ runs the algorithm $\mathbf{RKGen}$ using $sk_{i,1}, pk_j$.

Second Level Decryption Oracle $\mathcal{O}_{dec_2}(pk_i, \omega, CT_i)$: $\mathcal{B}$ runs the algorithm $\mathbf{KGen}$ using $sk_{i,1}$.

First Level Decryption Oracle $\mathcal{O}_{dec_1}(pk_i, CT_i)$: If $i \neq i^*$ $\mathcal{B}$ does as $\mathbf{Dec}_1$ to decrypt the ciphertext CT_i using secret key sk_i. Otherwise, $\mathcal{B}$ does as follows.

Let $CT = (A, B, C)$ be an arbitrary ciphertext submitted to the decapsulation oracle $\mathcal{O}_{dec_1}$. $\mathcal{B}$ computes $t = \mathsf{TCR}(A)$, if $\mathbf{e}(A, pk^t_{i^*,3} \cdot g_1) \neq \mathbf{e}(g, B)$ then output $\perp$ indicating an invalid ciphertext. Otherwise, do the following.

Case 1: $t = t^*$ and $A = A^*$: $\mathcal{B}$ rejects the query. In this case $B = (pk^t_{i^*,3} \cdot g_1)^r = (pk^{t-t^*}_{i^*,3} \cdot g^\alpha)^r = A^\alpha = (A^*)^\alpha = B^*$ and hence $C = C^*$ and the query made by $\mathcal{A}$ is illegal. Therefore it may be rejected by $\mathcal{B}$.

Case 2: $t = t^*$ and $A \neq A^*$: $\mathcal{B}$ finds a collision $A \neq A^*$ in TCR' with $\mathsf{TCR}'(A) = \mathsf{TCR}'(A^*)$. In that case $\mathcal{B}$ returns the collision and aborts.

Case 3: $t \neq t^*$: the adversary $\mathcal{B}$ computes the correct session key by $K \leftarrow H((B/(A^\alpha)^{(t-t^*)^{-1}})$. $\mathcal{B}$ using this session key computes $CT_i = \mathbf{SYM.Dec}(K, CT)$, then using $sk_{i^*,1} = x_{i^*}$ computes M as the decryption algorithm.

We have shown that unless $\mathcal{B}$ finds a collision in TCR' (**Case 2**) the simulation of the decryption oracle is always perfect, i.e. the output of the simulated oracle $\mathcal{O}_{dec_1}$ is identically distributed as the output of $\mathbf{Dec}_1$.

Challenge.

1. If $\mathcal{A}$ decides that the challenge ciphertext is the original one: $\mathcal{A}$ outputs two equal-length messages $M_0, M_1 \in \mathbb{G}_T$ and a condition ω^*. $\mathcal{B}$ flips a coin $\sigma \leftarrow_R \{0,1\}$, randomly chooses $r, R, s, \alpha \leftarrow_R \mathbb{Z}_p$ and does as follows:

   $C_2 = F(\omega^*)^r$; $C_3 = \mathbf{e}(g, g_1)^r \cdot M_\sigma$; $t = \mathsf{TCR}(C_2, C_3)$; $C_4 = (u^t v^s d)^r$; $C_5 = s$;

   $C_6 = pk^{rR}_{i^*,2}$; $C_7 = pk^R_{i^*,2}$; $C_8 = (g \cdot F(\omega^*)^\alpha)^{1/R}$; $C_9 = pk^\alpha_{i,1}$ $CT'_{i^*} = C_2||C_3||...||C_8||C_9||\omega^*$; $A^* = g^b (= g^{r^*})$, $t^* = \mathsf{TCR}'(A^*)$, $B^* = (g^b)^\alpha (= (pk^{t^*}_{i^*,3} \cdot g_1)^{r^*})$, $C^* \leftarrow \mathbf{SYM.Enc}(Q, CT'_{i'})$.

   Return $CT^* = (A^*, B^*, C^*)$.

2. If $\mathcal{A}$ decides that the challenge ciphertext is the re-encrypted one:

   $\mathcal{A}$ outputs two "good messages" CT_0, CT_1 which can be re-encrypted from $pk_{i'}$ to pk_{i^*} and a condition ω^*. $\mathcal{B}$ flips a coin $\sigma \leftarrow_R \{0,1\}$, chooses $R \leftarrow_R \mathbb{Z}_p$, and does as follows:

   Parse $CT_\sigma = (C_1, ..., C_5)$, and compute: $rk_{i' \to i^*|\omega^*} \leftarrow \mathbf{RKGen}(sk_{i'}, pk_{i^*}, \omega^*)$,

   $C_6 = C_1^R$, $C_7 = pk^R_{i',1}$; $C_8 = rk_1^{1/R}$, where $rk_{i' \to i^*|\omega^*} = (rk_1, rk_2)$; $CT'_{i'} = C_2||C_3||...||C_8||rk_2||\omega^*$, $A^* = g^b (= g^{r^*})$, $t^* = \mathsf{TCR}'(A^*)$, $B^* = (g^b)^\alpha (= (pk^{t^*}_{i^*,3} \cdot g_1)^{r^*})$, $C^* \leftarrow \mathbf{SYM.Enc}(Q, CT'_{i'})$.

   Return $CT^* = (A^*, B^*, C^*)$.

Guess stage. Eventually, $\mathcal{A}$ outputs a guess $\sigma' \in \{0,1\}$, where $\sigma' = 1$ means that K^* is the correct key. Algorithm $\mathcal{B}$ concludes its own game by outputting $\gamma' = \sigma'$ where $\gamma' = 1$ means that $Q = H(g^{ab})$ and $\gamma' = 0$ means that Q is random.

ANALYSIS. We have shown that as long as there is no hash collision in TCR' found by $\mathcal{B}$, adversary $\mathcal{A}$'s view in the simulation is identically distributed to its view in the real attack game.

Note that A^* is a random element from $\mathbb{G}$ (provided from outside of $\mathcal{B}$'s view), therefore finding a value $A \neq A^*$ with $\mathsf{TCR}'(A) = \mathsf{TCR}'(A^*)$ really contradicts to the security property of the target-collision resistant hash function. The probability that $\mathcal{B}$ finds a collision in the hash function TCR' is bounded by $\mathrm{Adv}^{\text{hash-TCR}}_{\mathsf{TCR}',\mathcal{B}}(\lambda)$.

Assume there was no hash collision found by $\mathcal{B}$. On the one hand, if Q is uniform and independent in $\{0,1\}^\ell$ then the game is exactly the same as the game **G1**. On the other hand, if $Q = H(g^{ab})$, then the game is exactly the same as the game **G0**. Therefore, adversary $\mathcal{B}$'s advantage in the GHDH game is $\mathrm{Adv}^{\text{ghdh}}_{\text{Gen},\mathcal{B}}(\lambda) \geq |\Pr[T_1] - \Pr[T_0]| - \mathrm{Adv}^{\text{hash-TCR}}_{\mathsf{TCR}',\mathcal{B}}(\lambda)$, which proves Eqn. 5.7.

Lastly, we observe that there is an oracle query machine $\mathcal{A}_1$, whose running time is essentially the same as that of $\mathcal{A}$, such that

$$|\Pr[T_1] - 1/2| = \mathrm{Adv}^{\text{CCA}}_{\mathbf{SYM},\mathcal{A}_1}(\lambda). \tag{6.9}$$

To see this, note that in game **G1**, the ciphertext C^* is produced using the random symmetric encryption key $K^\dagger$, and also that some other ciphertexts $C \neq C^*$ are decrypted using $K^\dagger$, but that the key $K^\dagger$ plays no other role in game **G1**. Thus, in game **G1**, the adversary $\mathcal{A}$ essentially just carries out an adaptive chosen ciphertext attack against $\mathbf{SYM}$.

The theorem now follows immediately from the equations 6.10 – 6.12.

6.4.3 Proof of Theorem 13

We prove that our proposed scheme is 2nd-aCond-CCA secure under the 6-AmDBDH problem. We build an algorithm $\mathcal{B}$ which is, given a modified 6-AmDBDH instance $(g, g^a, g^b, g^c, g^{a/c}, g^{c^2}, g^{ac}, g^{ac^2}, g^{ac^3}, g^{ac^4}, Q) \in \mathbb{G}^{10} \times \mathbb{G}_T$, deciding whether $Q = \mathbf{e}(g,g)^{ab/c^2}$, using a 2nd-aCond-CCA adversary $\mathcal{A}$. Adversary $\mathcal{B}$ runs adversary $\mathcal{A}$ simulating its view as in the 2nd-aCond-CCA security experiment as follows:

Setup: $\mathcal{B}$ first sets an integer, $m = 4q$, and chooses an integer, k, uniformly at random between 0 and n. It then chooses a random n-length vector, $\overrightarrow{x} = (\bar{x}_i)$, where the elements of $\overrightarrow{x}$ are chosen uniformly at random from the integers between 0 and $m-1$, and a value $\bar{x}'$ chosen uniformly at random between 0 and $m-1$. Additionally, the simulator chooses a random $\bar{y}' \in \mathbb{Z}_p$ and an n-length vector, $\overrightarrow{y} = (\bar{y}_i)$, where the elements of $\overrightarrow{y}$ are chosen uniformly at random from $\mathbb{Z}_p$.

Again, for a condition ω we will let $\Omega \subseteq \{1, 2, \ldots, n\}$ be the set of all i for which the ith bit of ω is equal 1. For ease of analysis we define three functions. We define $G(\omega) = (p - mk) + \bar{x}' + \sum_{i\in\Omega} \bar{x}_i$ and define $J(\omega) = \bar{y}' + \sum_{i\in\Omega} \bar{y}_i$. Finally, we define a binary function $K(\omega)$ as

$$K(\omega) = \begin{cases} 0 & \text{if } \bar{x}' + \sum_{i\in\Omega} \bar{x}_i \equiv 0 \pmod{m} \\ 1 & \text{otherwise} \end{cases}$$

$\mathcal{B}$ chooses $x_v, x_d, y_u, y_v, y_d \leftarrow_R \mathbb{Z}_p$, sets $g_1 := g^a$, and computes $u = g \cdot (g^{c^2})^{y_u}$, $v = g^{x_v} \cdot (g^{c^2})^{y_v}$, $d = g^{x_d} \cdot (g^{c^2})^{y_d}$, $u' = h_1^{p-km+\bar{x}'} h_2^{\bar{y}'}$, and $u_i = h_1^{\bar{x}_i} h_2^{\bar{y}_i}$, where $h_1 = g^{a/c}$ and $h_2 = g^{c^2}$. The other parameters are chosen as in the algorithm **Setup**. The public parameter is $PP = (p, \mathbb{G}, \mathbb{G}_T, g, g_1, u, v, d, \mathsf{e}, H, \mathsf{TCR}, \mathsf{TCR}', F)$.

Next, $\mathcal{B}$ generates the challenge key, uncorrupted-keys, and corrupted-keys as follows.

- *Challenge-key generation.* $\mathcal{B}$ chooses randomly $x_{i^*}, y_{i^*} \leftarrow_R \mathbb{Z}_p$, and defines $pk_{i^*} = ((g^{c^2})^{x_{i^*}}, (g^{ac^4})^{x_{i^*}^2}, g^{y_{i^*}})$ (meaning that $sk_{i^*} = (c^2 x_{i^*}, y_{i^*})$).
- *Uncorrupted-key generation.* $\mathcal{B}$ chooses randomly $x_i, y_i \leftarrow_R \mathbb{Z}_p$, and defines $pk_i = ((g^c)^{x_i}, (g^{ac^2})^{x_i^2}, g^{y_i})$ (meaning that $sk_i = (cx_i, y_i)$). $\mathcal{B}$ adds pk_i to $\mathcal{PK}_{un}$.
- *Corrupted-key generation.* $\mathcal{B}$ chooses randomly $x_i, y_i \leftarrow_R \mathbb{Z}_p$, and defines $pk_i = (g^{x_i}, (g^a)^{x_i^2}, g^{y_i})$, $sk_i = (x_i, y_i)$. $\mathcal{B}$ adds pk_i to $\mathcal{PK}_{corr}$ and sk_i to $\mathcal{SK}_{corr}$.

$\mathcal{B}$ provides $\mathcal{A}$ with public parameter PP, two sets of public key $\mathcal{PK} = (\mathcal{PK}_{un}, \mathcal{PK}_{corr})$, $\mathcal{SK}_{corr}$, and the challenge public key pk_{i^*}.

Find stage. Adversary $\mathcal{A}$ issues a series of questions as in the 2nd-aCond-CCA game. $\mathcal{B}$ answers these queries for $\mathcal{A}$ as follows:

ReKey Oracle $\mathcal{O}_{rk}(pk_i, pk_j, \omega)$: $\mathcal{B}$ chooses $\alpha \leftarrow_R \mathbb{Z}_p$ and does as follows.

- If $\omega = \omega^*$ (meaning that $F(\omega) = g^{c^2}$):
  1. If $pk_i = pk_{i^*}$ and $pk_j \in \mathcal{PK}_{corr}$, then return $\perp$.
  2. If $pk_i, pk_j \in \mathcal{PK}_{corr}$, then return $rk_{i \to j|\omega} = (g_1^{x_j^2/x_i} \cdot F(\omega)^\alpha, pk_{i,1}^\alpha)$.
  3. If $pk_i, pk_j \in \mathcal{PK}_{un}$, then return $rk_{i \to j|\omega} = ((g^{ac})^{x_j^2/x_i} \cdot F(\omega)^\alpha, pk_{i,1}^\alpha)$.
  4. If $pk_i \in \mathcal{PK}_{un}$ and $pk_j \in \mathcal{PK}_{corr}$, then return $rk_{i \to j|\omega} = ((g^{a/c})^{x_j^2/x_i} \cdot F(\omega)^\alpha, pk_{i,1}^\alpha)$.
  5. If $pk_i \in \mathcal{PK}_{corr}$ and $pk_j \in \mathcal{PK}_{un} \cup \{pk_{i^*}\}$, then return $rk_{i \to j|\omega} = (pk_{j,2}^{1/x_i} \cdot F(\omega)^\alpha, pk_{i,1}^\alpha)$.
  6. If $pk_i = pk_{i^*}$ and $pk_j \in \mathcal{PK}_{un}$, then return $rk_{i^* \to j|\omega} = ((g^{ac})^{x_j^2/x_{i^*}} \cdot F(\omega)^\alpha, pk_{i,1}^\alpha)$.
  7. If $pk_i \in \mathcal{PK}_{un}$ and $pk_j = pk_{i^*}$, then return $rk_{i \to i^*|\omega} = ((g^{ac^3})^{x_{i^*}^2/x_i} \cdot F(\omega)^\alpha, pk_{i,1}^\alpha)$.
- If $\omega \neq \omega^*$:
  - If $pk_i = pk_{i^*}, pk_j \in \mathcal{PK}_{corr}$, then $\mathcal{B}$ does as follows.
    If $K(\omega) = 0$, then $\mathcal{B}$ aborts and randomly chooses its guess $\beta \in \{0, 1\}$. Otherwise, it returns the re-encryption key, $rk_{i^* \to j|\omega}$, as

$$rk_{i^* \to j|\omega} = (R_1, R_2) = \left(\left((g^c)^{\frac{-J(\omega)}{G(\omega)}} F(\omega)^\alpha \right)^{\frac{x_j^2}{x_{i^*}}}, \left((g^c)^{\frac{-1}{G(\omega)}} h_2^\alpha \right)^{x_j^2} \right).$$

Let $\tilde{\alpha} = \left(\alpha - \frac{1}{cG(\omega)}\right) \cdot \frac{x_j^2}{x_{i^*}}$, and note that $h_1 = g^{a/c}, h_2 = g^{c^2}$. Then we have

$$\begin{aligned}
R_1 &= \left((g^c)^{\frac{-J(\omega)}{G(\omega)}} F(\omega)^{\alpha}\right)^{\frac{x_j^2}{x_{i^*}}} \\
&= \left((g^c)^{\frac{-J(\omega)}{G(\omega)}} \left(h_1^{G(\omega)} h_2^{J(\omega)}\right)^{\alpha}\right)^{\frac{x_j^2}{x_{i^*}}} \\
&= \left(h_1^{\frac{1}{c}} \left(h_1^{G(\omega)} h_2^{J(\omega)}\right)^{-\frac{1}{cG(\omega)}} \left(h_1^{G(\omega)} h_2^{J(\omega)}\right)^{\alpha}\right)^{\frac{x_j^2}{x_{i^*}}} \\
&= \left(h_1^{\frac{1}{c}} \left(h_1^{G(\omega)} h_2^{J(\omega)}\right)^{\alpha - \frac{1}{cG(\omega)}}\right)^{\frac{x_j^2}{x_{i^*}}} \\
&= \left(h_1^{\frac{1}{c}} F(\omega)^{\alpha}\right)^{\frac{x_j^2}{x_{i^*}}} \\
&= g^{a\frac{x_j^2}{c^2 x_{i^*}}} F(\omega)^{\tilde{\alpha}} \\
&= pk_{j,2}^{1/sk_{i^*,1}} \cdot F(\omega)^{\tilde{\alpha}}.
\end{aligned}$$

Additionally, we have

$$R_2 = \left((g^c)^{\frac{-1}{G(\omega)}} h_2^{\alpha}\right)^{x_j^2} = \left((g^{c^2})^{\alpha - \frac{1}{cG(\omega)}}\right)^{x_j^2} = \left((g^{c^2 x_{i^*}})^{\alpha - \frac{1}{cG(\omega)}}\right)^{\frac{x_j^2}{x_{i^*}}} = pk_{i^*,1}^{\tilde{\alpha}}.$$

Thus, we get

$$rk_{i^* \to j|\omega} = \left(pk_{j,2}^{1/sk_{i^*,1}} \cdot F(\omega)^{\tilde{\alpha}}, pk_{i^*,1}^{\tilde{\alpha}}\right).$$

Note that, $\mathcal{B}$ will be able to generate this re-encryption key iff $F(\omega) \neq 0 \bmod p$.

– Otherwise, $\mathcal{B}$ does as the above cases in which $\omega = \omega^*$.

First Level Decryption Oracle $\mathcal{O}_{dec_1}(pk_i, CT_i)$: If $pk_i \in \mathcal{PK}_{corr}$, $\mathcal{B}$ does as the $\mathbf{Dec}_1$ to decrypt the ciphertext CT_i using secret key sk_i. Otherwise, $\mathcal{B}$ does as follows. $\mathcal{B}$ computes $t' = \mathsf{TCR}(A)$, and checks whether the following equality hold: $\mathbf{e}(A, pk_{i,3}^{t'} \cdot g_1) = \mathbf{e}(g, B)$. If this verification fails, output $\perp$ indicating an invalid ciphertext. Otherwise, it does the following.

Compute $CT_i' = \mathbf{SYM.Dec}(H(A^{y_i}), C)$;

Parse $CT_i' = C_2||C_3||...||C_8||rk_2||\omega$, if this is not the case, output $\perp$ and halt.

Else, $\mathcal{B}$ computes $t = \mathsf{TCR}(C_2, C_3)$ and does as follows:

1. Check the validity of the ciphertext CT_i as the equations 6.3 – 6.5 in $\mathbf{Dec}_1$.
2. Checks whether $(C_2, C_3) = (C_2^*, C_3^*)$ and $t \neq t^*$. If so, $\mathcal{B}$ aborts and outputs a random bit.

3. Checks whether $t + sx_v + x_d = 0$. If so, $\mathcal{B}$ aborts and outputs a random bit.

If all of these verifications pass, then there exists $r \in \mathbb{Z}_p$ such that $C_1 = pk_{i,1}^r, C_4 = (u^t v^s d)^r, s = C_5$. If $(C_2, C_3, C_4, C_5) = (C_2^*, C_3^*, C_4^*, C_5^*)$, $\mathcal{B}$ outputs $\perp$ which deems CT_i as a derivative of the challenge pair (pk_{i^*}, CT^*). We now assume $C_3 \neq C_3^*$. If $\omega \neq \omega^*$, then $\mathcal{B}$ computes re-encryption key $rk_{j \to i'|\omega}$ as in **ReKey Oracle** $\mathcal{O}_{rk}$, where $i' \in \mathcal{PK}_{corr}$. Using this re-encryption key, $\mathcal{B}$ computes C_6, C_7, C_8 as in the re-encryption algorithm, then computes m as the equation 6.6. Otherwise ($\omega = \omega^*$), we have $F(\omega^*) = h_2^{J(\omega^*)} = (g^{c^2})^{J(\omega^*)}$. Since $C_2 = F(\omega^*)^r = g^{c^2 J(\omega^*) r}$, we have $C_2^{1/J(\omega^*)} = g^{c^2 r}$. Therefore $\mathcal{B}$ can compute $X := C_2^{(ty_u + sy_v + y_d)/J(\omega^*)} = g^{c^2 r(ty_u + sy_v + y_d)}$. Since $C_4 = (u^t v^s d)^r = g^{r(t+sx_v+x_d)} \cdot g^{c^2 r(ty_u+sy_v+y_d)}$ we have $g^r = (C_4/X)^{1/(t+sx_v+x_d)}$. Therefore, $\mathcal{B}$ can compute g^r by computing $(C_4/X)^{1/(t+sx_v+x_d)}$. Finally, $\mathcal{B}$ computes $M = \frac{C_3}{\mathbf{e}(g^r, g_1)}$, and returns it to $\mathcal{A}$.

Second Level Decryption Oracle $\mathcal{O}_{dec_2}(pk_i, \omega, CT_i)$**:** $\mathcal{B}$ does as follows.

- If $\omega = \omega^*$ (meaning that $F(\omega) = g^{c^2}$):
  1. If $pk_i \in \mathcal{PK}_{corr}$, then do as the algorithm $\mathbf{Dec}_2$ to decrypt the ciphertext CT_i using the secret key $sk_i = (x_i, y_i)$, and return the obtained result to $\mathcal{A}$.
  2. If $pk_i \in \mathcal{PK}_{un}$, then $\mathcal{B}$ first computes a re-encryption key $rk_{i \to j|\omega} = ((g^{a/c})^{x_j^2/x_i} \cdot F(\omega)^\alpha, pk_{i,1}^\alpha)$, where $pk_j \in \mathcal{PK}_{corr}$. $\mathcal{B}$ then re-encrypts the ciphertext CT_i by using $rk_{i \to j|\omega}$ to obtain $CT_j \leftarrow \mathbf{ReEnc}(rk_{i \to j|\omega}, CT_i)$. Finally, $\mathcal{B}$ does as the algorithm $\mathbf{Dec}_1$ to decrypt the ciphertext CT_j using the secret key $sk_j = (x_j, y_j)$, and returns the obtained result to $\mathcal{A}$.
  3. If $i = i^*$ and $CT_i \neq CT^*$, then $\mathcal{B}$ first computes a re-encryption key $rk_{i^* \to j|\omega} = ((g^{ac})^{x_j^2/x_{i^*}} \cdot F(\omega)^\alpha, pk_{i,1}^\alpha)$, where $pk_j \in \mathcal{PK}_{un}$. $\mathcal{B}$ then re-encrypts the ciphertext CT_i by using $rk_{i^* \to j}$ to obtain $CT_j \leftarrow \mathbf{ReEnc}(rk_{i^* \to j}, CT_i)$. Finally, $\mathcal{B}$ does as in **First Decryption Oracle** $\mathcal{O}_{dec_1}$ to decrypt the ciphertext CT_j, and returns the obtained result to $\mathcal{A}$.
- If $\omega \neq \omega^*$: $\mathcal{B}$ first computes $rk_{i \to j|\omega}$ as in **Re-Encryption Key Generation Oracle** $\mathcal{O}_{rk}$, where $pk_j \in \mathcal{PK}_{corr}$. $\mathcal{B}$ then re-encrypts the ciphertext CT_i by using $rk_{i \to j|\omega}$ to obtain $CT_j \leftarrow \mathbf{ReEnc}(rk_{i \to j|\omega}, CT_i)$. Finally, $\mathcal{B}$ does as the algorithm $\mathbf{Dec}_1$ to decrypt the ciphertext CT_j using the secret key $sk_j = (x_j, y_j)$, and returns the obtained result to $\mathcal{A}$.

Re Encryption Oracle $\mathcal{O}_{re}(rk_{i \to j|\omega}, CT_i)$**:** $\mathcal{B}$ parses $CT_i = (C_1, C_2, C_3, C_4, C_5)$. If this is not the case, outputs $\perp$ and halts. Otherwise, $\mathcal{B}$ computes $t = \mathsf{TCR}(C_2, C_3)$ and check whether $Check(CT_i) = 1$. If this is not the case, outputs $\perp$ and halts; else $\mathcal{B}$ works according to the following cases:

1. If $pk_i = pk_{i^*}, pk_j \in \mathcal{PK}_{corr}, \omega = \omega^*$ (meaning that $sk_{i^*} = c^2 x_{i^*}$ and $sk_j = x_j$): $\mathcal{B}$ chooses $R', r', \alpha' \leftarrow_R \mathbb{Z}_p$ and computes g^r as in **First Decryption Oracle** $\mathcal{O}_{dec_1}$. Using g^r, $\mathcal{B}$ computes:

   $C_6 = (g^r)^{x_{i^*} R'} (= (g^{c^2 x_{i^*} r})^{R'/c^2} = C_1^R)$, where $R = R'/c^2$;

$C_7 = g^{x_{i^*} R'} (= (g^{c^2 x_{i^*}})^{R'/c^2} = pk_{i^*,1}^R)$;

$C_8 = (g_1^{x_j^2/x_{i^*}})^{1/R'} F(\omega^*)^{\alpha'} (= (g_1^{sk_{j,1}^2/sk_{i^*,1}} F(\omega^*)^{\alpha})^{c^2/R'} = rk_1^{1/R})$, where $\alpha = \alpha' R'/c^2$ and $rk_{i^* \to j|\omega^*} = (rk_1, rk_2)$;

$C_9 = g^{x_{i^*} \alpha' R'} (= pk_{i^*,1}^{\alpha' R'/c^2} = pk_{i^*,1}^{\alpha} = rk_2)$; $CT'_{i^*} = C_2 || C_3 || ... || C_8 || C_9 || \omega^*$;

$A = g^{r'}$, $t' = \mathsf{TCR}'(A)$, $B = (pk_{i^*,3}^{t'} \cdot g_1)^{r'}$, $C \leftarrow \mathbf{SYM.Enc}(H(pk_{i^*,3}^{r'}), CT'_{i^*})$.

Output $CT_j = (A, B, C)$.

2. Otherwise, $\mathcal{B}$ executes the algorithm $\mathbf{ReEnc}$ with re-encryption key computed as in $\mathcal{O}_{rk}(pk_i, pk_j, \omega)$, then returns $CT_j \leftarrow \mathbf{ReEnc}(rk_{i \to j|\omega}, CT_i)$ to $\mathcal{A}$.

Challenge. When $\mathcal{A}$ decides that Phase 1 is over, it outputs two equal-length messages $M_0, M_1 \in \mathbb{G}_T$ and a condition ω^*. At this stage, $\mathcal{B}$ flips a coin $\sigma \leftarrow_R \{0,1\}$ and defines: $C_1^* = (g^b)^{x_{i^*}}, C_2^* = (g^b)^{J(\omega^*)}, C_3^* = Q \cdot M_\sigma, C_5^* = s^* = -(t^* + x_d)/x_v$ where $t^* = \mathsf{TCR}(C_2^*, C_3^*)$, and $C_4^* = (g^b)^{t^* y_u + s^* y_v + y_d}$. Return $CT^* = (C_1^*, C_2^*, C_3^*, C_4^*, C_5^*)$ as the challenge ciphertext to $\mathcal{A}$.

Observe the following to see that, CT^* is indeed a valid challenge ciphertext under public key pk_{i^*} with the random exponent $r = b/c^2$ if $Q = \mathbf{e}(g,g)^{ab/c^2}$.

$C_1^* = (g^b)^{x_{i^*}} = (g^{c^2 x_{i^*}})^{b/c^2} = pk_{i^*,1}^r$; $C_2^* = (g^b)^{J(\omega^*)} = (g^{c^2 J(\omega^*)})^{b/c^2} = F(\omega^*)^r$;

$C_3^* = Q \cdot M_\sigma$; $t^* = \mathsf{TCR}(C_2^*, C_3^*)$; $C_5^* = s^* = -(t^* + x_d)/x_v$;

$C_4^* = (g^b)^{t^* y_u + s^* y_v + y_d} = (g^{c^2 (t^* y_u + s^* y_v + y_d)})^{b/c^2} \cdot (g^{t^* + s^* x_v + x_d})^{b/c^2} = (u^{t^*} v^{s^*} d)^r$;

In contrast, if Q is random in $\mathbb{G}_T$, CT^* perfectly hides m_σ and $\mathcal{A}$ cannot guess σ with better probability than $1/2$.

Guess stage. Adversary $\mathcal{A}$ continues to issue the rest queries. In this stage, $\mathcal{B}$ can respond these queries for $\mathcal{A}$ as in the **Find stage**. Eventually, adversary $\mathcal{A}$ returns a guess $\sigma' \in \{0,1\}$.

Artificial abort. This step is done in exactly the same way as that of Waters [64, Section 5].

At this point, we have to also use the technique of "artificial abort" employed by Waters [64]. The idea is that the probability of aborting up to this is not independent of the adversarial queries. The idea of the artificial abort technique is to allow the simulator to sample the transcript of queries it obtained from the adversary and on certain conditions abort and output a random bit. This increases the total probability of abort and makes it almost equal for all adversarial inputs. This helps in the probability analysis. The effect of sampling the transcript is to increase the runtime of the simulator. See [64] for the details.

Output. If the $\mathcal{B}$ has not aborted at this point it will take check to see if the adversary's guess, $\sigma' = \sigma$. If $\sigma' = \sigma$ then $\mathcal{B}$ outputs 1 to guess $Q = \mathbf{e}(g,g)^{ab/c^2}$; else $\mathcal{B}$ outputs 0 to guess that Q is a random element in $\mathbb{G}_T$.

This completes the description of the simulation.

Next, we analyze the simulation.

In the simulation of **Decryption Oracle**, $\mathcal{B}$ aborts in two cases, those are in the second and third conditions. The probability that $\mathcal{B}$ aborts in the second condition of $\mathcal{O}_{dec_1}$ is at most $\mathrm{Adv}^{\mathrm{TCR}}_{\mathsf{TCR},\mathcal{A}}$. For the challenge ciphertext, the adversary could obtain the information that $t^* + s^* x_v + x_d = 0$. However, there are exactly p possible (x_v, x_d) pairs that satisfy this equation and each of them are equally likely. Thus, information-theoretically, the probability that $t + s x_v + x_d = 0$ is at most $1/p$. Hence, the probability that $\mathcal{B}$ aborts in the third condition of $\mathcal{O}_{dec_1}$ is at most q_{dec}/p. Therefore, the probability that $\mathcal{B}$ aborts during the simulation of oracle $\mathcal{O}_{dec_1}$ is at most $\mathrm{Adv}^{\mathrm{TCR}}_{\mathsf{TCR},\mathcal{A}} + q_{dec}/p$.

Assuming that $\mathcal{B}$ doesn't abort in the simulation of **Decryption Oracle**. Then, in the simulation of **ReKey Oracle**, using exactly the same analysis method of Waters [64], we have the probability of the simulation not aborting by the guess phase is at least $\frac{\varepsilon}{8(n+1)q}$.

Observe that, if $\mathcal{B}$ does not abort during the simulation and $Q = \mathbf{e}(g,g)^{ab/c^2}$, then the simulation of $\mathcal{B}$ is perfect from the view of $\mathcal{A}$. On the other hand, if $\mathcal{B}$ does not abort during the simulation and Q is distributed uniformly and independently over $\mathbb{G}_T$ then the advantage that $\mathcal{A}$ wins the attack game is $1/2$. Since the simulation for $\mathcal{O}_{re}, \mathcal{O}_{rk}$, and using exactly the same analysis method of Waters [64], we have $\mathrm{Adv}^{\text{6-AmDBDH}}_{\mathcal{B}}(\lambda) \geq \frac{\varepsilon}{32(n+1)q} - \mathrm{Adv}^{\mathrm{TCR}}_{\mathsf{TCR},\mathcal{A}} - q_{dec}/p$.

This concludes the proof of the theorem.

6.4.4 Proof of Theorem 14

We will prove the theorem using a sequences of game transitions.

Game G0: This is the original 1st-aCond-CCA game, which is the CCA game as in Definition 22 with the additional constraints from Definition 31. The adversary $\mathcal{A}$ has access to the re-encryption key oracle $\mathcal{O}_{rk}$ and the decryption oracle $\mathcal{O}_{dec_1}$, gets a target ciphertext $CT^* = (A^*, B^*, C^*)$, where CT^* is computed as follows:

1. If the challenge ciphertext is the original one: $CT^* \leftarrow \mathbf{Enc}_1(pk_{i^*}, m_\sigma)$.
2. If the challenge ciphertext is the re-encrypted one: $CT^* \leftarrow \mathbf{ReEnc}(rk_{i' \to i^*}, CT_\sigma)$, where $CT_\sigma = (C_1, ..., C_5)$ is a "good message" which can be re-encrypted from $pk_{i'}$ to pk_{i^*}.

Let $\sigma' \in \{0,1\}$ denote the output of $\mathcal{A}$, and let T_0 be the event that $\sigma' = \sigma$ in **G0**, so that

$$\mathrm{Adv}^{\text{1st-aCond-CCA}}_{\mathrm{CPRE},\mathcal{A}}(\lambda) = |\Pr[T_0] - 1/2|. \tag{6.10}$$

Game G1: Now we define a modified game **G1**. This game behaves just like game **G0**, except that we use a completely random symmetric key $K^\dagger$ in place of the key $K^* = H(pk_{i^*,1}^{r'})$ in both the encryption and decryption oracles. Let T_1 be the event that $\sigma' = \sigma$ in game **G1**.

Now, we show that there is an oracle query machine $\mathcal{B}$, whose running time is essentially the same as that of $\mathcal{A}$, such that

$$|\Pr[T_1] - \Pr[T_0]| \leq \mathrm{Adv}^{\mathrm{ghdh}}_{\mathrm{Gen},\mathcal{B}}(\lambda) + \mathrm{Adv}^{\text{hash-TCR}}_{\mathsf{TCR}',\mathcal{B}}(\lambda). \tag{6.11}$$

$\mathcal{B}$ runs the adversary $\mathcal{A}$ as a subroutine to solve a random instance of the GHDH problem.

We now give the description of the adversary $\mathcal{B}$. The adversary $\mathcal{B}$ inputs an instance of the GHDH problem, *i.e.* $\mathcal{B}$ inputs the values $(1^k, g, g^a, g^b, Q)$. $\mathcal{B}$' goal is to determine whether

$Q = H(g^{ab})$ or $Q \in \{0,1\}^\ell$ is a random bit string. The adversary $\mathcal{B}$ runs adversary $\mathcal{A}$ simulating its view as in the original 1st-aCond-CCA security experiment as follows:

Setup: $\mathcal{B}$ chooses collision-resistant hash functions $\mathsf{TCR} : \mathbb{G} \times \mathbb{G}_T \to \mathbb{Z}_p$ and $\mathsf{TCR}' : \mathbb{G} \to \mathbb{Z}_p$. To generate challenge key, $\mathcal{B}$ chooses randomly $x_{i^*}, \alpha, u, v, d \leftarrow_R \mathbb{Z}_p$, sets $g_1 := pk_{i^*,3}^{-t^*} \cdot g^\alpha$, and defines $pk_{i^*} = (g^{x_{i^*}}, g_1^{x_{i^*}^2}, g^a), A^* = g^b$ (meaning that $sk_{i^*} = (x_{i^*}, a), r^* = b$). $\mathcal{B}$ computes $t^* = \mathsf{TCR}'(A^*)$.

The other parameters are chosen as in the algorithm $\mathbf{Setup}$. The public parameter is $PP = (p, \mathbb{G}, \mathbb{G}_T, g, g_1, u, v, d, \mathbf{e}, H, \mathsf{TCR}, \mathsf{TCR}', F)$. Next, $\mathcal{B}$ generates the uncorrupted-keys and corrupted-keys as follows.

- *Uncorrupted-key generation.* $\mathcal{B}$ chooses randomly $x_i, y_i \leftarrow_R \mathbb{Z}_p$, and defines $pk_i = (g^{x_i}, g_1^{x_i^2}, g^{y_i})$ (meaning that $sk_i = (x_i, y_i)$). $\mathcal{B}$ adds pk_i to $\mathcal{PK}_{un}$.
- *Corrupted-key generation.* $\mathcal{B}$ chooses randomly $x_i, y_i \leftarrow_R \mathbb{Z}_p$, and defines $pk_i = (g^{x_i}, g_1^{x_i^2}, g^{y_i}), sk_i = (x_i, y_i)$. $\mathcal{B}$ adds pk_i to $\mathcal{PK}_{corr}$ and sk_i to $\mathcal{SK}_{corr}$.

$\mathcal{B}$ provides $\mathcal{A}$ with the public parameter PP, two sets of public key $\mathcal{PK} = (\mathcal{PK}_{un}, \mathcal{PK}_{corr})$, $\mathcal{SK}_{corr}$, and the challenge public key pk_{i^*}.

Find stage. The adversary $\mathcal{A}$ issues a series of questions as in 1st-Cond-CCA game. Note that, since $\mathcal{B}$ knows all of the secret keys except $sk_{i^*,2}(= a)$, he can exactly answer every queries of $\mathcal{A}$. In particular, $\mathcal{B}$ answers these queries for $\mathcal{A}$ as follows:

ReKey Oracle $\mathcal{O}_{rk}(pk_i, pk_j, \omega)$: $\mathcal{B}$ runs the algorithm $\mathbf{RKGen}$ using $sk_{i,1}, pk_j$.

Second Level Decryption Oracle $\mathcal{O}_{dec_2}(pk_i, \omega, CT_i)$: $\mathcal{B}$ runs the algorithm $\mathbf{KGen}$ using $sk_{i,1}$.

First Level Decryption Oracle $\mathcal{O}_{dec_1}(pk_i, CT_i)$: If $i \neq i^*$ $\mathcal{B}$ does as $\mathbf{Dec}_1$ to decrypt the ciphertext CT_i using secret key sk_i. Otherwise, $\mathcal{B}$ does as follows.

Let $CT = (A, B, C)$ be an arbitrary ciphertext submitted to the decapsulation oracle $\mathcal{O}_{dec_1}$. $\mathcal{B}$ computes $t = \mathsf{TCR}(A)$, if $\mathbf{e}(A, pk_{i^*,3}^t \cdot g_1) \neq \mathbf{e}(g, B)$ then output $\perp$ indicating an invalid ciphertext. Otherwise, do the following.

Case 1: $t = t^*$ and $A = A^*$: $\mathcal{B}$ rejects the query. In this case $B = (pk_{i^*,3}^t \cdot g_1)^r = (pk_{i^*,3}^{t-t^*} \cdot g^\alpha)^r = A^\alpha = (A^*)^\alpha = B^*$ and hence $C = C^*$ and the query made by $\mathcal{A}$ is illegal. Therefore it may be rejected by $\mathcal{B}$.

Case 2: $t = t^*$ and $A \neq A^*$: $\mathcal{B}$ finds a collision $A \neq A^*$ in TCR' with $\mathsf{TCR}'(A) = \mathsf{TCR}'(A^*)$. In that case $\mathcal{B}$ returns the collision and aborts.

Case 3: $t \neq t^*$: the adversary $\mathcal{B}$ computes the correct session key by $K \leftarrow H((B/(A^\alpha)^{(t-t^*)^{-1}})$. $\mathcal{B}$ using this session key computes $CT_i = \mathbf{SYM.Dec}(K, CT)$, then using $sk_{i^*,1} = x_{i^*}$ computes M as the decryption algorithm.

We have shown that unless $\mathcal{B}$ finds a collision in TCR' (**Case 2**) the simulation of the decryption oracle is always perfect, i.e. the output of the simulated oracle $\mathcal{O}_{dec_1}$ is identically distributed as the output of $\mathbf{Dec}_1$.

Challenge.

1. If $\mathcal{A}$ decides that the challenge ciphertext is the original one: $\mathcal{A}$ outputs two equal-length messages $M_0, M_1 \in \mathbb{G}_T$ and a condition ω^*. $\mathcal{B}$ flips a coin $\sigma \leftarrow_R \{0,1\}$, randomly chooses $r, R, s, \alpha \leftarrow_R \mathbb{Z}_p$ and does as follows:

   $C_2 = F(\omega^*)^r$; $C_3 = \mathbf{e}(g, g_1)^r \cdot M_\sigma$; $t = \mathsf{TCR}(C_2, C_3)$; $C_4 = (u^t v^s d)^r$; $C_5 = s$;

   $C_6 = pk_{i^*,2}^{rR}$; $C_7 = pk_{i^*,2}^{R}$; $C_8 = (g \cdot F(\omega^*)^\alpha)^{1/R}$; $C_9 = pk_{i,1}^{\alpha}$ $CT'_{i^*} = C_2||C_3||...||C_8||C_9||\omega^*$; $A^* = g^b (= g^{r^*})$, $t^* = \mathsf{TCR}'(A^*)$, $B^* = (g^b)^\alpha (= (pk_{i^*,3}^{t^*} \cdot g_1)^{r^*})$, $C^* \leftarrow \mathbf{SYM.Enc}(Q, CT'_{i'})$.

   Return $CT^* = (A^*, B^*, C^*)$.

2. If $\mathcal{A}$ decides that the challenge ciphertext is the re-encrypted one:

   $\mathcal{A}$ outputs two "good messages" CT_0, CT_1 which can be re-encrypted from $pk_{i'}$ to pk_{i^*} and a condition ω^*. $\mathcal{B}$ flips a coin $\sigma \leftarrow_R \{0,1\}$, chooses $R \leftarrow_R \mathbb{Z}_p$, and does as follows:

   Parse $CT_\sigma = (C_1, ..., C_5)$, and compute: $rk_{i' \to i^*|\omega^*} \leftarrow \mathbf{RKGen}(sk_{i'}, pk_{i^*}, \omega^*)$,

   $C_6 = C_1^R$, $C_7 = pk_{i',1}^R$; $C_8 = rk_1^{1/R}$, where $rk_{i' \to i^*|\omega^*} = (rk_1, rk_2)$; $CT'_{i'} = C_2||C_3||...||C_8||rk_2||\omega^*$, $A^* = g^b (= g^{r^*})$, $t^* = \mathsf{TCR}'(A^*)$, $B^* = (g^b)^\alpha (= (pk_{i^*,3}^{t^*} \cdot g_1)^{r^*})$, $C^* \leftarrow \mathbf{SYM.Enc}(Q, CT'_{i'})$.

   Return $CT^* = (A^*, B^*, C^*)$.

Guess stage. Eventually, $\mathcal{A}$ outputs a guess $\sigma' \in \{0,1\}$, where $\sigma' = 1$ means that K^* is the correct key. Algorithm $\mathcal{B}$ concludes its own game by outputting $\gamma' = \sigma'$ where $\gamma' = 1$ means that $Q = H(g^{ab})$ and $\gamma' = 0$ means that Q is random.

ANALYSIS. We have shown that as long as there is no hash collision in TCR' found by $\mathcal{B}$, adversary $\mathcal{A}$'s view in the simulation is identically distributed to its view in the real attack game.

Note that A^* is a random element from $\mathbb{G}$ (provided from outside of $\mathcal{B}$'s view), therefore finding a value $A \neq A^*$ with $\mathsf{TCR}'(A) = \mathsf{TCR}'(A^*)$ really contradicts to the security property of the target-collision resistant hash function. The probability that $\mathcal{B}$ finds a collision in the hash function TCR' is bounded by $\mathrm{Adv}^{\text{hash-TCR}}_{\mathsf{TCR}',\mathcal{B}}(\lambda)$.

Assume there was no hash collision found by $\mathcal{B}$. On the one hand, if Q is uniform and independent in $\{0,1\}^\ell$ then the game is exactly the same as the game **G1**. On the other hand, if $Q = H(g^{ab})$, then the game is exactly the same as the game **G0**. Therefore, adversary $\mathcal{B}$'s advantage in the GHDH game is $\mathrm{Adv}^{\text{ghdh}}_{\text{Gen},\mathcal{B}}(\lambda) \geq |\Pr[T_1] - \Pr[T_0]| - \mathrm{Adv}^{\text{hash-TCR}}_{\mathsf{TCR}',\mathcal{B}}(\lambda)$, which proves Eqn. 5.7.

Lastly, we observe that there is an oracle query machine $\mathcal{A}_1$, whose running time is essentially the same as that of $\mathcal{A}$, such that

$$|\Pr[T_1] - 1/2| = \mathrm{Adv}^{\text{CCA}}_{\mathbf{SYM},\mathcal{A}_1}(\lambda). \tag{6.12}$$

To see this, note that in game **G1**, the ciphertext C^* is produced using the random symmetric encryption key $K^\dagger$, and also that some other ciphertexts $C \neq C^*$ are decrypted using $K^\dagger$, but that the key $K^\dagger$ plays no other role in game **G1**. Thus, in game **G1**, the adversary $\mathcal{A}$ essentially just carries out an adaptive chosen ciphertext attack against $\mathbf{SYM}$.

The theorem now follows immediately from the equations 6.10–6.12.

6.5 Summary

In this chapter, we have presented the CPRE schemes that are secure against the selective and adaptive condition chosen-ciphertext attack, respectively. These schemes are extended from the CCA-secure PRE scheme proposed in Section 5.3. Our schemes rely on mild complexity assumptions in bilinear groups without random oracles. We have also shown that the CPRE scheme proposed by Liang et al. [40] is vulnerable to chosen-ciphertext attack, and presented a concrete attack to break the CCA security of it. Therefore, our schemes are the first CPRE schemes secure against condition chosen-ciphertext attack, and these are answers to the problem left by Weng et al. [68].

Chapter 7

Conclusion

In this thesis, we have focused on constructing CCA-secure PRE schemes, and made the following contributions:

- In Chapter 3, we have proposed new CCA security definitions for both bidirectional single-hop, and unidirectional single-hop PRE by extending those of the previous works. In order to point out why our security models are stronger than those of the previous works, we have presented concrete PRE schemes which are secure in their model, but are not secure in our models. We have also proposed new condition CCA security definitions for unidirectional single-hop CPRE.

- In Chapter 4, we have proposed the first factoring-based PRE schemes. In particular, we have presented three PRE schemes. The first is a CPA-secure bidirectional multi-hop PRE scheme. The second is a CCA-secure bidirectional single-hop PRE scheme. The last is a CCA-secure unidirectional single-hop PRE scheme. The first is in the standard model, and the others are in the random oracle model. In order to construct the second and the third schemes, we have proposed a new factoring-based strong signature scheme which is a variant of the Schnorr signature scheme.

- In Chapter 5, we first have shown that the scheme proposed by Shao, Liu, and Zhou [55] is vulnerable to chosen-ciphertext attack. We then have presented the first unidirectional single-hop PRE scheme which is secure in the full CCA security model. The scheme is obtained by extending the PKE scheme proposed by Lai, Deng, Liu, and Kou [37]. Our scheme relies on mild complexity assumptions in bilinear groups without random oracles.

- In Chapter 6, we have presented the CPRE schemes that are secure against the selective and adaptive condition chosen-ciphertext attack, respectively. These schemes are extended from the CCA-secure PRE scheme proposed in Section 5.3. Our schemes rely on mild complexity assumptions in bilinear groups without random oracles. We have also shown that the CPRE scheme proposed by Liang et al. [40] is vulnerable to chosen-ciphertext attack, and presented a concrete attack to break the CCA security of it. Therefore, our schemes are the first CPRE schemes secure against condition chosen-ciphertext attack, and these are answers to the problem left by Weng et al. [68].

Future Works. We propose several future works as follows.

Firstly, there are two future works on constructing factoring-based PRE schemes. In this thesis, we have given only a bidirectional multi-hop scheme which is CPA secure. It is an interesting problem of constructing a CCA-secure bidirectional multi-hop scheme, even in the random oracle model. The other is to construct a CCA-secure factoring-based PRE scheme without random oracles.

Secondly, it would be interesting to construct a PRE scheme without bilinear maps in the standard model. All of the existing CCA-secure schemes in the standard model have used bilinear maps.

Finally, it is still an open problem of constructing conditional schemes without bilinear maps, even in the random oracle model.

Bibliography

[1] Aono, Y., Boyen, X., Phong, L.T., Wang, L.: Key-private proxy re-encryption under lwe. In: Paul, G., Vaudenay, S. (eds.) INDOCRYPT 2013. LNCS, vol. 8250, pp. 1–18. Springer, Heidelberg (2013)

[2] Ateniese, G., Benson, K., Hohenberger, S.: Key-private proxy re-encryption. In: Fischlin, M. (ed.) CT-RSA 2009. LNCS, vol. 5473, pp. 279–294. Springer, Heidelberg (2009)

[3] Ateniese, G., Fu, K., Green, M., Hohenberger, S.: Improved proxy re-encryption schemes with applications to secure distributed storage. In: NDSS (2005)

[4] Ateniese, G., Fu, K., Green, M., Hohenberger, S.: Improved proxy re-encryption schemes with applications to secure distributed storage. In: ACM Trans. Inf. Syst. Secur., 9(1). pp. 1–30 (2006)

[5] Ateniese, G., Hohenberger, S.: Proxy re-signatures: New definitions, algorithms and applications. In: ACM Conference on Computer and Communications Security. pp. 310–319 (2005)

[6] Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: ACMCCS 1993. pp. 62–73. ACM Press (1993)

[7] Bellare, M., Desai, A., Jokipii, E., Rogaway, P.: A concrete security treatment of symmetric encryption. In: 38th Annual Symposium on Foundations of Computer Science. pp. 394–403. IEEE Computer Society Press (1997)

[8] Blaze, M., Bleumer, G., Strauss, M.: Divertible protocols and atomic proxy cryptography. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 127–144. Springer, Heidelberg (1998)

[9] Blum, L., Blum, M., Shub, M.: A simple unpredictable pseudo- random number generator. SIAM Journal on Computing 15(2), 364–383 (1986)

[10] Boneh, D., Boyen, X., Goh, E.J.: Hierarchical identity-based encryption with constant size ciphertext. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 440–456. Springer, Heidelberg (2005)

[11] Boneh, D., Franklin, M.: Identity-based encryption from the Weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001)

[12] Boneh, D., Franklin, M.: Identity-based encryption from the weil pairing. SIAM Journal on Computing 32(3), 586–615 (2003)

[13] Boneh, D., Boyen, X.: Efficient selective-id secure identity based encryption without random oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 223–238. Springer, Heidelberg (2004)

[14] Canard, S., Devigne, J., Laguillaumie, F.: Improving the security of an efficient unidirectional proxy re-encryption scheme. Journal of Internet Services and Information Security 1(2), 140–160 (2011)

[15] Canetti, R., Halevi, S., Katz, J.: A forward secure public key encryption scheme. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 254–271. Springer, Heidelberg (2003)

[16] Canetti, R., Hohenberger, S.: Chosen-ciphertext secure proxy re-encryption. In: ACM Conference on Computer and Communications Security. pp. 185–194. ACM Press (2007)

[17] Chow, S., Weng, J., Yang, Y., Deng, H.: Efficient unidirectional proxy re-encryption. In: Bernstein, D.J., Lange, T. (eds.) AFRICACRYPT 2010. LNCS, vol. 6055, pp. 316–332. Springer, Heidelberg (2010)

[18] Chu, C.K., Tzeng, W.G.: Identity-based proxy re-encryption without random oracles. In: Garay, J., Lenstra, A., Mambo, M., Peralta, R. (eds.) ISC 2007. LNCS, vol. 4779, pp. 189–202. Springer, Heidelberg (2007)

[19] Chu, C.K., Weng, J., Chow, S.S., Zhou, J., Deng, R.H.: Conditional proxy broadcast re-encryption. In: Boyd, C., Nieto, J.G. (eds.) ACISP 2009. LNCS, vol. 5594, pp. 327–342. Springer-Verlag, Heidelberg (2009)

[20] Deng, R., Weng, J., Liu, S., Chen, K.: Chosen-ciphertext secure proxy re-encryption without pairings. In: Franklin, M.K., Hui, L.C.K., Wong, D.S. (eds.) CANS 2008. LNCS, vol. 5339, pp. 1–17. Springer, Heidelberg (2008)

[21] Elgamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. In: Blakley, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 10–18. Springer, Heidelberg (1985)

[22] Fang, L., Susilo, W., Wang, J.: Anonymous conditional proxy re-encryption without random oracle. In: Pieprzyk, J., Zhang, F. (eds.) ProvSec 2009. LNCS, vol. 5848, pp. 47–60. Springer, Heidelberg (2009)

[23] Fujisaki, E., Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. In: Wiener, M.J. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 537–554. Springer, Heidelberg (1999)

[24] Green, M., Ateniese, G.: Identity-based proxy re-encryption. In: Katz, J., Yung, M. (eds.) ACNS 2007. LNCS, vol. 4521, pp. 288–306. Springer, Heidelberg (2007)

[25] Hanaoka, G., Kawai, Y., Kunihiro, N., Matsuda, T., Weng, J., Zhang, R., Zhao, Y.: Generic construction of chosen cipertext secure proxy re-encryption. In: Dunkelman, O. (ed.) CT-RSA 2012. LNCS, vol. 7178, pp. 349–364. Springer, Heidelberg (2012)

[26] Hayashi, R., Matsushita, T., Yoshida, T., Fujii, Y., Okada, K.: Unforgeability of re-encryption keys against collusion attack in proxy re-encryption. In: Iwata, T., Nishigaki, M. (eds.) IWSEC 2011. LNCS, vol. 7038, pp. 210–229. Springer, Heidelberg (2011)

[27] He, Y., Chim, T., Hui, L., Yiu, S.: Non-transferable proxy re-encryption. In: Cryptology ePrint archive. `http://eprint.iacr.org/2010/192` (2010)

[28] Hofheinz, D., Kiltz, E.: The group of signed quadratic residues and applications. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 637–653. Springer, Heidelberg (2009)

[29] Hofheinz, D., Kiltz, E.: Practical chosen ciphertext secure encryption from factoring. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5749, pp. 313–332. Springer, Heidelberg (2009)

[30] Hohenberger, S., Rothblum, G.N., Shelat, A., Vaikuntanathan, V.: Securely obfuscating re-encryption. In: Vadhan, S. (ed.) TCC 2007. LNCS, vol. 4392, pp. 233–252. Springer, Heidelberg (2007)

[31] Isshiki, T., Nguyen, M.H., Tanaka, K.: Attacks to the proxy re-encryption schemes from iwsec2011. In: Sakiyama, K., Terada, M. (eds.) IWSEC 2013. LNCS, vol. 8231, pp. 290–302. Springer, Heidelberg (2013)

[32] Isshiki, T., Nguyen, M.H., Tanaka, K.: Factoring-based proxy re-encryption schemes. In: Susilo, W., Reyhanitabar, R. (eds.) ProvSec 2013. LNCS, vol. 8209, pp. 309–329. Springer, Heidelberg (2013)

[33] Isshiki, T., Nguyen, M.H., Tanaka, K.: Proxy re-encryption in a stronger security model extended from CT-RSA2012. In: Dawson, E. (ed.) CT-RSA 2013. LNCS, vol. 7779, pp. 277–292. Springer, Heidelberg (2013)

[34] Ivan, A., Dodis, Y.: Proxy cryptography revisited. In: NDSS. The Internet Society (2003)

[35] Jakobsson, M.: On quorum controlled asymmetric proxy re-encryption. In: Imai, H., Zheng, Y. (eds.) PKC 1999. LNCS, vol. 1560, pp. 112–121. Springer, Heidelberg (1999)

[36] Kiltz, E.: Chosen-ciphertext secure key-encapsulation based on gap hashed Diffie-Hellman. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 282–297. Springer, Heidelberg (2007)

[37] Lai, J., Deng, H., Liu, S., Kou, W.: Efficient CCA-secure PKE from identity-based techniques. In: Pieprzyk, J. (ed.) CT-RSA 2010. LNCS, vol. 5985, pp. 132–147. Springer, Heidelberg (2010)

[38] Lai, J., Zhu, W., Deng, R., Liu, S., Kou, W.: New constructions for identity-based unidirectional proxy re-encryption. Journal of Computer Science and Technology pp. 793–806 (2010)

[39] Liang, K., Huang, Q., Schlegel, R., Wong, D.S., Tang, C.: A conditional proxy broadcast re-encryption scheme supporting timed-release. In: Deng, R., Feng, T. (eds.) ISPEC 2013. LNCS, vol. 7863, pp. 132–146. Springer, Heidelberg (2013)

[40] Liang, K., Liu, Z., Tan, X., Wong, D.S., Tang, C.: A CCA-secure identity-based conditional proxy re-encryption without random oracles. In: Kwon, T., Lee, M.K., Kwon, D. (eds.) ICISC 2012. LNCS, vol. 7839, pp. 231–246. Springer, Heidelberg (2012)

[41] Libert, B., Vergnaud, D.: Tracing malicious proxies in proxy re-encryption. In: Galbraith, S., Paterson, K. (eds.) Pairing 2008. LNCS, vol. 5209, pp. 332–353. Springer, Heidelberg (2008)

[42] Libert, B., Vergnaud, D.: Unidirectional chosen-ciphertext secure proxy re-encryption. In: Cramer, R. (ed.) PKC 2008. LNCS, vol. 4939, pp. 360–379. Springer, Heidelberg (2008)

[43] Luo, S., Hu, J., Chen, Z.: New construction of identity-based proxy re-encryption. In: Proceedings of the Tenth Annual ACM Workshop on Digital Rights Management, DRM 2010. pp. 47–50. ACM, New York (2010)

[44] Mambo, M., Okamoto, E.: Proxy cryptosystems: Delegation of the power to decrypt ciphertexts. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences E80-A(1) (1997)

[45] Matsuda, T., Nishimaki, R., Tanaka, K.: CCA proxy re-encryption without bilinear maps in the standard model. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 261–278. Springer, Heidelberg (2010)

[46] Matsuo, T.: Proxy re-encryption systems for identity-based encryption. In: Takagi, T., Okamoto, T., Okamoto, E., Okamoto, T. (eds.) Pairing 2007. LNCS, vol. 4575, pp. 247–267. Springer, Heidelberg (2007)

[47] Nishimaki, R.: A CCA-secure proxy re-encryption scheme with short ciphertexts. In: SCIS 2011, 3F3-2 in Japanese (2011)

[48] Pointcheval, D., Stern, J.: Security proofs for signature schemes. In: Maurer, U. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 387–398. Springer, Heidelberg (1996)

[49] Sahai, A., Waters, B.: Fuzzy identity-based encryption. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (2005)

[50] Schnorr, C.P.: Efficient signature generation by smart cards. J. Cryptology 4(3), 161–174 (1991)

[51] Seo, J.W., Yum, D.H., Lee, P.J.: Proxy-invisible CCA-secure type-based proxy re-encryption without random oracles. Theoretical computer science 491, 83–93 (2013)

[52] Shamir, A.: On the generation of cryptographically strong pseudorandom sequences. ACM Trans. Comput. Syst. 1(1) (1983)

[53] Shamir, A.: Identity based cryptosystems and signature schemes. In: Blakley, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 47–53. Springer, Heidelberg (1984)

[54] Shao, J., Cao, Z.: CCA-secure proxy re-encryption without pairings. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 357–376. Springer, Heidelberg (2009)

[55] Shao, J., Liu, P., Zhou, Y.: Achieving key privacy without losing CCA security in proxy re-encryption. Journal of Systems and Software 85(3), 655–665 (2012)

[56] Shao, J., Wei, G., Ling, Y., Xie, M.: Identity-based conditional proxy re-encryption. In: IEEE ICC 2011 (2011)

[57] Shao, J.: Anonymous id-based proxy re-encryption. In: Susilo, W., Mu, Y., Seberry, J. (eds.) ACISP 2012. LNCS, vol. 7372, pp. 364–375. Springer, Heidelberg (2012)

[58] Shao, J., Cao, Z.: Multi-use unidirectional identity-based proxy re-encryption from hierarchical identity-based encryption. Information Sciences 206, 83–95 (2012)

[59] Shao, J., Liu, P., Wei, G., Ling, Y.: Anonymous proxy re-encryption. Security and Communication Networks 5(5), 439–449 (2012)

[60] Taban, G., Cárdenas, A., Gligor, V.D.: Towards a secure and inter-operable DRM architecture. In: DRM'06, ACM Workshop Digital Rights Manage. pp. 69–76 (2006)

[61] Vivek, S.S., Selvi, S.S.D., Radhakishan, V., Rangan, C.P.: Conditional proxy re-encryption - A more efficient construction. In: Wyld, D.C., Wozniak, M., Chaki, N., Meghanathan, N., Nagamalai, D. (eds.) CNSA 2011. CCIS, vol. 196, pp. 502–512. Springer-Verlag (2011)

[62] Wang, H., Cao, Z., Wang, L.: Multi-use and unidirectional identity-based proxy re-encryption schemes. Information Sciences 180, 4042–4059 (2010)

[63] Wang, L., Wang, L., Mambo, M., Okamoto, E.: New identity-based proxy re-encryption schemes to prevent collusion attacks. In: Joye, M., Miyaji, A., Otsuka, A. (eds.) Pairing 2010. LNCS, vol. 6486, pp. 327–346. Springer, Heidelberg (2010)

[64] Waters, B.: Efficient identity-based encryption without random oracles. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 114–127. Springer, Heidelberg (2005)

[65] Wee, H.: Efficient chosen-ciphertext security via extractable hash proofs. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 314–332. Springer, Heidelberg (2010)

[66] Weng, J., Deng, R., Ding, X., Chu, C.K., Lai, J.: Conditional proxy re-encryption secure against chosen-ciphertext attack. In: ASIACCS 2009. vol. 322-332 (2009)

[67] Weng, J., Zhao, Y., Hanaoka, G.: On the security of a bidirectional proxy re-encryption scheme from PKC 2010. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 284–295. Springer, Heidelberg (2011)

[68] Weng, J., Yang, Y., Tang, Q., Deng, R.H., Bao, F.: Efficient conditional proxy re-encryption with chosen-ciphertext security. In: Samarati, P., Yung, M., Martinelli, F., Ardagna, C.A. (eds.) ISC 2009. LNCS, vol. 5735, pp. 151–166. Springer, Heidelberg (2009)

[69] Zheng, Q., Zhu, W., Zhu, J., Zhang, X.: Improved anonymous proxy re-encryption with CCA security. In: ASIACCS 2014 (2014)