Ph.D. Thesis

博士論文

Provable Security of Applied Public Key Cryptosystems and Heuristically Secure Protocols

（高機能公開鍵暗号技術と経験的なセキュリティプロトコルの証明可能安全性）

Satsuya Ohata

大畑 幸矢

Acknowledgement

Firstly, I would like to express my gratitude to my supervisor, Professor Kanta Matsuura, for his guidance during my graduated school days. I have learned a lot from him, including how to consider not only the research but also the education. I would also like to thank chief and deputy examiners, Professor Tohru Asami, Kaoru Sezaki, Kenjiro Taura, Masashi Toyoda, and Toshihiko Yamasaki for their helpful comments and feedbacks on my research and this thesis.

I would like to thank the many present and past Matsuura-lab members including the secretaries and visitors of the meeting for many useful discussion. I was able to discuss and learn a lot of things about information security.

I would like to thank the researchers at Advanced Cryptography Research Group at AIST and many other participants of the study group (Shin-Akarui-Angou-Benkyou-Kai). Especially, I would like to thank Takahiro Matsuda and Goichiro Hanaoka for their attentive guidance, suggestions, and discussion.

Finally, I would like to thank my family (including the dog and cats) for a great deal of support.

Abstract

The information on the Internet is always exposed to the threat of eavesdropping and falsification. Even in these conditions, we can ensure the confidentiality and integrity of information by using cryptosystems. In the research of cryptosystems, especially in public key cryptosystems, it is strongly required that we should rigorously prove its security. When we prove the security of cryptosystems, we usually reduce its security to the difficulty of mathematical problems. In this framework, we can objectively judge the security. Although not all the cryptosystems without security proof are insecure, the concept of provable security is useful in many aspects. In this thesis, we show two types of results related to the provable security.

First, we denote the results of provably secure applied cryptosystems in Chapters 3-5. In general, it often appears many entities, keys, and ciphertexts in applied cryptosystems. Therefore, we have to consider a complex model to deal with various attacks. If we fail this modeling, it is meaningless to prove the security. In these chapters, we show the results with respect to threshold public key encryption and proxy re-encryption under the extended models and security definitions. In Chapter 3, we show three new constructions of threshold public key encryption schemes with key re-splittability. In Chapter 4, we show a generic construction of a proxy re-encryption scheme with new functionality called re-encryption verifiability. In Chapter 5, we show a construction of a multi-hop and unidirectional proxy re-encryption scheme based on a cryptographic obfuscator. In this thesis, we discuss practical / theoretical meaning of new models and security definitions.

Next, in Chapter 6, we extend the provable security to the security protocols other than cryptosystems. More concretely, we show the result with respect to a protocol that we call "password reset protocol". We define models and security definitions, propose generic constructions, prove its security, and implement a prototype to evaluate its efficiency. This result can improve the security of real world protocols. Moreover, we can expect progress of theoretical analysis for password reset protocol based on this result.

Contents

Chapter 1

Introduction

1.1 Background and Motivation

These days, we exchange a lot of information through the Internet which is an open network. The information on the Internet is always exposed to the threat of eavesdropping and falsification. Even in these conditions, we can assure the confidentiality and integrity of information by using cryptosystems.

It is an appropriate security evaluation that is important in information security research. Research without security evaluation is not security research. In cryptosystems, the most desirable security notion is the security against adversaries with unbounded computational power. However, this is impossible to achieve in the context of public key cryptosystems. Moreover, it is also despairingly difficult to derive the lower bound of computational cost to break the cryptosystems because this means the settlement of P versus NP problem. Therefore, in cryptosystems (especially in public key cryptosystems), a framework of security evaluation called "provable security" is widely spread. In this framework, we can objectively judge the security within some model by reducing its security of cryptosystems to the hardness of mathematical problems. Thanks to this property, governmental organization (e.g. CRYPTREC [39]) can announce recommended ciphers lists and estimate cryptographic parameters to provide enough security. We cannot easily conclude that the cryptosystems without security proofs are insecure ones. However, we have found that some cryptosystems without security proof (e.g. RSA PKCS#1 version 1.5, ISO/IEC 9796-2 (Scheme 1), etc.) that have been already standardized and widely used in practice have been already broken. Moreover, as we can objectively judge the security, we can also (to some extent) terminate the cat-and-mouse game between attacks and heuristic countermeasures. Therefore, proving the security of cryptosystems is now indispensable when we propose new ones.

To prove the security of cryptosystems, we have to define the models of security goal, adversaries, correctness, and format of mathematical problems which are used as security assumptions. We usually use the mathematical problems which are widely believed to be difficult (e.g. factoring problem, discrete logarithm problem, etc.). When we prove the security of cryptosystems, we have to show the fact that "there exists no algorithm that can break the security of the cryptosystems with non-negligible probability if the mathematical assumptions hold". In practice, we show the contraposition of this. That is, we show the fact that "If there exist an probabilistic polynomial time algorithm $\mathcal{A}$ that can break the security of the cryptosystems with non-negligible probability, we can solve the mathematical problems by using $\mathcal{A}$ as subroutine". Here, probabilistic polynomial time algorithm shows the ability of algorithms that is existing in the real world. When we define a security model, we consider a "security goal" and "attack model" separately. We show an example as follows: As a security goal, we require that a ciphertext does not leak any information of plaintext. This security goal is defined as "INDistinguishability". As an attack model, we consider the adversary that can choose arbitrary plaintexts, encrypt them using public key (in the setting of public key encryption, anyone can obtain the public key), and obtain the polynomially many pairs of plaintext and ciphertext. This attack model is defined as "Chosen Plaintext Attacks". Finally, we can define the indistinguishability against chosen plaintext attacks (IND-CPA, for short) for public key encryption by combining these security goal and attack model. Since we have already found that there exist stronger attacks than chosen plaintext attacks, we usually consider the indistinguishability against adaptive chosen ciphertext attacks (IND-CCA, for short). We define these security models by using a security game which is executed by adversary and challenger. More concretely, please see the Section 2.9.

These days, the theory of provable security has been progressed. More concretely, we formalize various attacks and situations (e.g. side channel attacks, human errors, and misuse) and introduce them into the security model (e.g. key-dependent message security [18], leakage resilient security [3], selective opening security [45], and related-key security [12]). Cryptosystems with above security have (to some extent) resistance against attacks even if the security of hardware is partially broken. From this observation, we find that these cryptosystems are important to improve the security level of whole systems.

As we can see above, the concept of provable security is a useful. However, there exist at least two points that we should be noted. First, we have to rigorously discuss the hardness of mathematical problems that are used as the destination of security reduction. We should avoid introducing new (and hence, suspicious) assumptions without consideration. In addition, we have to correctly estimate and decide the parameters of those problems because their hardness depend on the size of parameters. This note has been continued since the beginning of research on cryptosystems.

The other note is that we have to closely examine whether the security model captures realistic threats or not. In the concept of provable security, modeling an adversary is extremely important. If we fail this modeling, security proof becomes meaningless. However, it is not easy for applied cryptosystems to set an appropriate security model. We can point out some reasons of above difficulty.

Diversification of usage environment: In the past, cryptosystems had been used only to protect the confidentiality on communication channels. Recently, however, we use cryptosystems in many ways. For example, the functionality of hard disk encryption have been equipped in the standard computer to prevent information disclosures. Since a key for encryption exists in the hard disk drive itself, plaintext contains the key for encryption in this case. Standard security definitions for public key encryption (e.g. IND-CCA security) do not capture this situation.

Enrichment of attack environment: Side channel attacks (that is, cryptanalysis using physical information) have become more strong. For example, recently, we have to consider the leakage of not only a secret key itself, but also a randomness which is used in the encryption algorithm.

Existence of too many entities: In general, it often appears many entities, keys, and ciphertexts in applied cryptosystems and we have to consider many situations. That is, we have to consider who may become adversaries, what types of attacks should be considered, and how security should be assured (or not be assured). In the research on group signature (privacy preserving signature) before 2012, for example, we had only considered the security property that the users are not falsely accused by other malicious ones. However, we had not considered the signature hijacking. That is, even in the previous group signature schemes with security proofs, malicious users may be able to claim that he/she generated the signature even if it is generated by other user.

If there exist many security models that we cannot compare, we also cannot compare the security and efficiency between concrete schemes even in the same primitives. Therefore, it is important to construct an appropriate security models. Here, "appropriate" contains many meanings. For example, it captures realistic threats, we can prove enough security, and it is well-defined, and so on. This problem arises recently because many applied cryptosystems have been proposed and more and more functionalities/security on them have been defined these days.

We pointed out some negative points and problems of provable security so far. On the other hand, we can also point out positive aspects of provable security. That is, we can consider extending its possibility. It seems that there is no reason to apply this useful concept only for cryptosystems. In heuristically secure protocols, in fact, cat-and-mouse game between attacks and heuristic countermeasures has been occurred. Therefore, it will be meaningful if we can introduce the concept of provably security to the heuristically secure protocols.

1.2 Outline and Summary of This Thesis

In this thesis, we show the results of three cryptosystems and a heuristically secure protocol to address the above problems.

- In Chapter 2, we explain the basic notations and cryptographic primitives which are used in this thesis.

- In Chapter 3, we denote the constructions of threshold public key encryption with the functionality called "re-splittability". In this setting, we can split a secret key polynomially many times. This property is useful in practice. Moreover, this cryptosystems plays an important role in the generic construction of a proxy re-encryption scheme which is denoted in Chapter 4. The schemes that we propose in this chapter can be obtained by extending the previous standard threshold public key encryption schemes. However, we note that we cannot extend all the previous standard threshold public key encryption schemes to the re-splittable ones.

- In Chapter 4, we denote a proxy re-encryption scheme with new functionality called "re-encryption verifiability". In verifiable proxy re-encryption, we can detect the malicious activities of a proxy. We define a model and security definitions for verifiable proxy re-encryption. Not only we add a functionality of re-encryption verifiability, but also we show the backward compatibility between new security definitions and previous ones. That is, the scheme secure under our new security definitions automatically satisfy the previous ones of standard proxy re-encryption. In addition, we propose a generic construction of proxy re-encryption that satisfies the new security definitions.

- In Chapter 5, we propose a construction of a multi-hop uni-directional proxy re-encryption scheme. In the proxy re-encryption scheme which is denoted in Chapter 4, the number of re-encryptions is limited to only once. As there exists no model and security definitions for multi-hop uni-directional proxy re-encryption, we define them and construct a concrete scheme via a cryptographic obfuscator. We cannot expect the practicality of this scheme now because the obfuscator does not work within realistic time. On the other hand, the technique which is used in the security proof seems somewhat interesting.

- In Chapter 6, we extend the concept of provable security to security protocols other than cryptosystems. As we denoted in Section 1.1, it seems there is no reason to apply the concept of provable security only for cryptosystems. More concretely, this is a result about a backup authentication protocol that we call "password reset protocol". Although most online services that adopt a password-based user authentication system support a mechanism with which a user can reset a password, the fact that the most popular password reset mechanism called secret question does not provide enough security in practice has been already showed. Although the security of password reset protocol has been evaluated in a heuristic manner, we introduce a concept of provable security to this field. We define the model and security definitions, propose a generic construction, prove its security, and implement a prototype to evaluate the efficiency of our protocol. We introduce a special key for password reset to overcome the barrier for provably secure protocols. We consider not only the standard impersonation attacks, but also the illegal registration attacks for password reset protocols. This result can improve the security of real world protocols. Moreover, we can expect the progress of theoretical analysis for password reset protocol based on this results.

- In Chapter 7, we conclude the contributions of this thesis and describe the future prospects that can be considered from this thesis.

Chapter 2

Preliminaries

In this chapter, we review the basic notation, definitions, and cryptographic primitives which are used in this thesis.

2.1 Basic Notations

$\mathbb{N}$ denotes the set of all natural numbers, and for $n \in \mathbb{N}$, we let $[n] := \{1, \ldots, n\}$. "$x \leftarrow y$" denotes that x is chosen uniformly at random from y if y is a finite set, x is output from y if y is a function or an algorithm, or y is assigned to x otherwise. "$x \| y$" denotes a concatenation of x and y. "$|x|$" denotes the size of the set if x is a finite set or bit length of x if x is a string. "PPT" stands for probabilistic polynomial-time. If $\mathcal{A}$ is a probabilistic algorithm then "$y \leftarrow \mathcal{A}(x)$" denotes that $\mathcal{A}$ computes y as output by taking x as input. "$x := y$" denotes that x is defined as y. Without loss of generality, we consider that a secret key contains the information of the corresponding public key. "k" always denotes the security parameter. ϕ denotes the empty string. A function $f(k) : \mathbf{N} \rightarrow [0, 1]$ is said to be *negligible* if for all positive polynomials p and all sufficiently large $k \in \mathbf{N}$, we have $f(k) < 1/p(k)$. If $\mathcal{A}$ is a protocol between interactive algorithms P and Q, we write $(p_{out}, q_{out}) \leftarrow \mathcal{A}(P(p_{in}) \leftrightarrow Q(q_{in}))$ to mean that P and Q take p_{in} and q_{in} as input, respectively, interact with each other, and finally P and Q locally output p_{out} and q_{out}, respectively.

2.2 Bilinear Maps

Groups $(\mathbb{G}, \mathbb{G}_T)$ of prime order p are called *bilinear groups* if there is a mapping $\boldsymbol{e} : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ with the following properties:

1. bilinearity: $\boldsymbol{e}(g^a, g^b) = \boldsymbol{e}(g, g)^{ab}$ for any $g \in \mathbb{G}$ and $a, b \in \mathbb{Z}$;

2. efficient computability: given $g, h \in \mathbb{G}$, $\boldsymbol{e}(g, h) \in \mathbb{G}_T$ is efficiently computable.

3. non-degeneracy: $\boldsymbol{e}(g, g) \neq 1_{\mathbb{G}_T}$ whenever $g \neq 1_{\mathbb{G}}$;

For convenience, we introduce a bilinear group generator BG that takes 1^k as input and outputs a description $(p, \mathbb{G}, \mathbb{G}_T, \boldsymbol{e})$ of bilinear groups where p is a k-bit prime. This process is written as $(p, \mathbb{G}, \mathbb{G}_T, \boldsymbol{e}) \leftarrow \mathsf{BG}(1^k)$.

2.3 Mathematical Assumptions

In this section, we explain the mathematical assumptions which are used in this thesis.

2.3.1 Decisional Bilinear Diffie-Hellman (DBDH) Assumption

The decisional bilinear Diffie-Hellman (DBDH) assumption on $(\mathbb{G}, \mathbb{G}_T)$ posits the hardness of distinguishing $\boldsymbol{e}(g,g)^{abc}$ from a random group element in $\mathbb{G}_T$ given (g, g^a, g^b, g^c). More formally, we define the advantage of an adversary $\mathcal{A}$ in solving the DBDH problem as follows.

$$\mathsf{Adv}_{\mathcal{A}}^{\mathrm{DBDH}}(k) = |\Pr[\mathcal{A}(g, g^a, g^b, g^c, \boldsymbol{e}(g,g)^{abc}) = 1] - \Pr[\mathcal{A}(g, g^a, g^b, g^c, \boldsymbol{e}(g,g)^{z}) = 1]|$$

where, $(p, \mathbb{G}, \mathbb{G}_T, \boldsymbol{e}) \leftarrow \mathsf{BG}(1^k)$, $g \leftarrow \mathbb{G}$, and $a, b, c, z \leftarrow \mathbb{Z}_p$. We say that the DBDH assumption holds when the advantage $\mathsf{Adv}_{\mathcal{A}}^{\mathrm{DBDH}}(k)$ is negligible for any PPT adversary $\mathcal{A}$.

2.3.2 Decisional Linear (DLIN) Assumption

The decisional linear (DLIN) assumption on $(\mathbb{G}, \mathbb{G}_T)$ posits the hardness of distinguishing g_3^{a+b} from random group element in $\mathbb{G}$ given $(g_1, g_2, g_3, g_1^a, g_2^b)$. More formally, we define the advantage of an adversary $\mathcal{A}$ in solving the DLIN problem as follows.

$$\mathsf{Adv}_{\mathcal{A}}^{\mathrm{DLIN}}(k) = |\Pr[\mathcal{A}(g_1, g_2, g_3, g_1^a, g_2^b, g_3^{a+b}) = 1] - \Pr[\mathcal{A}(g_1, g_2, g_3, g_1^a, g_2^b, g_3^{z}) = 1]|$$

where, $(p, \mathbb{G}, \mathbb{G}_T, \boldsymbol{e}) \leftarrow \mathsf{BG}(1^k)$, $g_1, g_2, g_3 \leftarrow \mathbb{G}$, and $a, b, z \leftarrow \mathbb{Z}_p$. We say that the DLIN assumption holds when the advantage $\mathsf{Adv}_{\mathcal{A}}^{\mathrm{DLIN}}(k)$ is negligible for any PPT adversary $\mathcal{A}$.

2.3.3 Hashed Diffie-Hellman (HDH) Assumption

Here, H is a hash function that takes an element in $\mathbb{G}$ as input, and outputs a string in $\{0,1\}^l$. The hashed Diffie-Hellman (HDH) assumption on $(\mathbb{G}, \mathbb{G}_T, H)$ posits the hardness of distinguishing $H(g^{ab})$ from a random string given (g, g^a, g^b). More formally, we define the advantage of an adversary $\mathcal{A}$ in solving the HDH problem as follows.

$$\mathsf{Adv}_{\mathcal{A}}^{\mathrm{HDH}}(k) = |\Pr[\mathcal{A}(g, g^a, g^b, H(g^{ab})) = 1] - \Pr[\mathcal{A}(g, g^a, g^b, R) = 1]|$$

where, $(p, \mathbb{G}, \mathbb{G}_T, \boldsymbol{e}) \leftarrow \mathsf{BG}(1^k)$, $g \leftarrow \mathbb{G}$, $a, b \leftarrow \mathbb{Z}_p$, and $R \leftarrow \{0,1\}^l$. We say that the HDH assumption holds when the advantage $\mathsf{Adv}_{\mathcal{A}}^{\mathrm{HDH}}(k)$ is negligible for any PPT adversary $\mathcal{A}$.

Kiltz [73] introduced the gap HDH (GHDH) assumption, and constructed an efficient key encapsulation mechanism. We remark that when we consider the GHDH assumption *on (symmetric) bilinear groups*, the GHDH and the ordinary HDH assumptions become equivalent. One of our proposed constructions uses the HDH assumption on (symmetric) bilinear groups.

2.4 Pseudorandom Number Generator

A pseudorandom number generator [111] $PRG : \{0,1\}^k \to \{0,1\}^{2k}$ is a deterministic function that takes k-bit length string $s \in \{0,1\}^k$ as input, outputs a $2k$-bit length string. We define the advantage of an adversary $\mathcal{A}$ for PRG as follows:

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{PRG}}(k) := |\Pr[\mathcal{A}(PRG(s)) \to 1 | s \leftarrow \{0,1\}^k] - \Pr[\mathcal{A}(r) \to 1 | r \leftarrow \{0,1\}^{2k}]|$$

We say that a PRG is computationally secure, if for any PPT adversary $\mathcal{A}$, $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{PRG}}(k)$ is negligible.

2.5 Collision Resistant Hash Function

We define the advantage of an adversary $\mathcal{A}$ in finding the collision of a hash function H as follows.

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{CRHF}}(k) = \Pr[(m_0, m_1) \leftarrow \mathcal{A}(H) : \ H(m_0) = H(m_1) \wedge m_0 \neq m_1]$$

where, H is picked randomly. We say that H is a collision resistant hash function if for any PPT adversary $\mathcal{A}$, $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{CRHF}}(k)$ is negligible.

2.6 Pseudorandom Function

We say that F is a pseudorandom function (PRF) if

1. there exists an efficient algorithm that takes a key $K \in \{0,1\}^k$ and a string x as input, and outputs $F(K, x)$.

2. for all PPT adversaries $\mathcal{A}$, the advantage in the following PRF game played with the challenger $\mathcal{B}$ is negligible: First, $\mathcal{B}$ picks the challenge bit $b \in \{0,1\}$ and a key $K \in \{0,1\}^k$ uniformly at random. $\mathcal{A}$ can adaptively make queries x (without loss of generality, we assume that $\mathcal{A}$ does not make the same query twice). If $b = 0$, then $\mathcal{B}$ responds with $F(K, m)$. Otherwise (that is, $b = 1$), $\mathcal{B}$ returns a random string in the domain of F. Finally, $\mathcal{A}$ outputs a guess bit b' for b. We define the advantage of $\mathcal{A}$ in this game by $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{PRF}}(k) = |\Pr[b' = b] - 1/2|$.

2.7 Puncturable Pseudorandom Function

A puncturable pseudorandom function (PPRF) F consists of the following three Turing machines $(\mathsf{Key}_F, \mathsf{Punc}_F, \mathsf{Eval}_F)$ and a pair of computable functions $(n(\cdot), m(\cdot))$ satisfying the following conditions:

Key_F: This is a key generation algorithm that takes 1^k as input, and outputs a key K.[1] This process is written as $K \leftarrow \mathsf{Key}_F(1^k)$.

Punc_F: This is a puncture algorithm that takes a key K and puncture points $S \subseteq \{0,1\}^{n(k)}$ as input, and outputs a punctured key $K(S)$. This process is written as $K(S) \leftarrow \mathsf{Punc}_F(K, S)$.

Eval_F: This is an evaluation algorithm that takes a key K and a string x as input, and outputs a evaluation value of x $\mathsf{Eval}_F(K, x)$. Here, $n(\cdot)$ means input length and $m(\cdot)$ means output length. For simplicity, we use $F(K, x)$ to represent $\mathsf{Eval}_F(K, x)$.

[1]As shown in [106], in fact, it is sufficient to choose a $K \in \{0,1\}^k$ from a uniform distribution instead of running the Key_F algorithm. For simplicity, in this paper, we also write like that in the description of our scheme and the security proof.

We say that a PPRF is secure if the following two conditions are satisfied.

Preserving evaluation value at unpunctured point: We consider the all PPT adversaries $\mathcal{A}$ that takes 1^k as input, and outputs puncture points $S \subseteq \{0,1\}^{n(k)}$. For all $x \in \{0,1\}^{n(k)} \backslash S$, we have:

$$\Pr[\mathsf{Eval}_F(K,x) = \mathsf{Eval}_F(K(S),x) | K \leftarrow \mathsf{Key}_F(1^k), K(S) \leftarrow \mathsf{Punc}_F(K,S)] = 1.$$

Pseudorandomness at punctured points: We consider the all PPT adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$. Here, τ is state information. $\mathcal{A}_1$ takes 1^k as input, and outputs puncture points $S \subseteq \{0,1\}^{n(k)}$. For all $K \leftarrow \mathsf{Key}_F(1^k)$ and $K(S) \leftarrow \mathsf{Punc}_F(K,S)$, we have:

$$\begin{aligned}\mathsf{Adv}^{\mathsf{PPRF}}_{(\mathcal{A}_1,\mathcal{A}_2)}(k) = |\Pr[\mathcal{A}_2(\tau, K(S), S, &\mathsf{Eval}_F(K,S)) = 1] \\ &- \Pr[\mathcal{A}_2(\tau, K(S), S, U_{m(k)\cdot|S|}) = 1]| \leq \varepsilon(k).\end{aligned}$$

Here, the distribution of $\mathsf{Eval}_F(K,S)$ is $\{\mathsf{Eval}_F(K,x) | x \in S\}$, and U_ℓ denotes the uniform distribution over ℓ bits.

2.8 Indistinguishability Obfuscation

Let $\mathcal{C}_k$ be the class of circuits of size at most k. A uniform PPT algorithm $i\mathcal{O}$ is called an indistinguishability obfuscator ($i\mathcal{O}$) [50, 98] for a circuit class $\mathcal{C}_k$ if the following conditions are satisfied:

1. For all security parameters $k \in \mathbf{N}$, for all $\mathcal{C} \in \mathcal{C}_k$, for all input x, we have:

$$\Pr[\mathcal{C}'(x) = \mathcal{C}(x) : \mathcal{C}' \leftarrow i\mathcal{O}(k, \mathcal{C})] = 1.$$

2. For any (not necessarily uniform) PPT algorithms $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, there exists a negligible function ε such that the following holds: if $\mathcal{A}_1$ always outputs $(\mathcal{C}_0, \mathcal{C}_1, \sigma)$ such that $\mathcal{C}_0, \mathcal{C}_1 \in \mathcal{C}_k$ and $\mathcal{C}_0(x) = \mathcal{C}_1(x)$ for all x, we have:

$$\begin{aligned}\mathsf{Adv}^{i\mathcal{O}}_{(\mathcal{A}_1,\mathcal{A}_2)}(k) := |&\Pr[\mathcal{A}_2(\sigma, i\mathcal{O}(k, \mathcal{C}_0)) = 1 : (\mathcal{C}_0, \mathcal{C}_1, \sigma) \leftarrow \mathcal{A}_1(1^k)] \\ &- \Pr[\mathcal{A}_2(\sigma, i\mathcal{O}(k, \mathcal{C}_1)) = 1 : (\mathcal{C}_0, \mathcal{C}_1, \sigma) \leftarrow \mathcal{A}_1(1^k)]| \leq \varepsilon(k).\end{aligned}$$

Such (candidate) constructions of $i\mathcal{O}$ for all polynomial-size circuits that satisfies the above conditions were given in [50].

2.9 Public Key Encryption

A public key encryption scheme (PKE) consists of the following three algorithms (PKG, PEnc, PDec).

PKG This is the key generation algorithm that takes 1^k as input, and outputs a pair of decryption key dk and public key pk.

PEnc This is the encryption algorithm that takes a public key pk and a plaintext m as input, and outputs a ciphertext c.

PDec This is the decryption algorithm that takes a decryption key dk and a ciphertext c as input, and outputs a decryption result m (which could be the special symbol $\perp$ meaning that c is invalid).

We require the standard correctness for a PKE scheme, namely, for any $(dk, pk) \leftarrow \mathsf{PKG}(1^k)$ and any plaintext m, we have $m = \mathsf{PDec}(dk, \mathsf{PEnc}(pk, m))$.

Chosen Plaintext Security [54] We recall the definition of chosen plaintext security (CPA security, for short) of PKE, which is defined by the following game between the challenger and an adversary $\mathcal{A}$: First, the challenger picks the challenge bit $b \in \{0, 1\}$, computes $(dk, pk) \leftarrow \mathsf{PKG}(1^k)$, and gives 1^k and pk to $\mathcal{A}$. $\mathcal{A}$ can make a challenge query (only once). For a challenge query (m_0, m_1), where (m_0, m_1) is a message pair of equal length, the challenger computes $c^* \leftarrow \mathsf{PEnc}(pk, m_b)$, and then returns the challenge ciphertext c^* to $\mathcal{A}$. Finally, $\mathcal{A}$ outputs a guess bit b' for b. $\mathcal{A}$ wins the game if $b = b'$. We define the advantage of $\mathcal{A}$ by $\mathsf{Adv}_{\mathcal{A}}^{\mathtt{CPA\text{-}PKE}}(k) = |\Pr[b' = b] - 1/2|$. We say a PKE scheme is *CPA secure*, if for any PPT adversary $\mathcal{A}$, $\mathsf{Adv}_{\mathcal{A}}^{\mathtt{CPA\text{-}PKE}}(k)$ is negligible.

Chosen Ciphertext Security [83, 88, 44] We recall the definition of chosen ciphertext security (CCA security, for short) of PKE, which is defined by the following game between the challenger and an adversary $\mathcal{A}$: First, the challenger picks the challenge bit $b \in \{0, 1\}$, computes $(dk, pk) \leftarrow \mathsf{PKG}(1^k)$, and gives 1^k and pk to $\mathcal{A}$. $\mathcal{A}$ can adaptively make a challenge query (only once) and decryption queries. For a challenge query (m_0, m_1), where (m_0, m_1) is a message pair of equal length, the challenger computes $c^* \leftarrow \mathsf{PEnc}(pk, m_b)$, and then returns the challenge ciphertext c^* to $\mathcal{A}$. For a decryption query c, the challenger responds with $m \leftarrow \mathsf{PDec}(dk, c)$, except that if c is the challenge ciphertext c^*, then the challenger returns $\perp$ to $\mathcal{A}$. Finally, $\mathcal{A}$ outputs a guess bit b' for b. $\mathcal{A}$ wins the game if $b = b'$. We define the advantage of $\mathcal{A}$ by $\mathsf{Adv}_{\mathcal{A}}^{\mathtt{CCA\text{-}PKE}}(k) = |\Pr[b' = b] - 1/2|$. We say a PKE scheme is *CCA secure*, if for any PPT adversary $\mathcal{A}$, $\mathsf{Adv}_{\mathcal{A}}^{\mathtt{CCA\text{-}PKE}}(k)$ is negligible.

Chosen Ciphertext Security in the Multi-user Setting In this paper, we will also use the multi-user version of the CCA security for a PKE scheme, which was introduced by Bellare, Boldyreva, and Micali [9]. We briefly recall it here. CCA security in the multi-user setting is also defined by the game that is parameterized by an integer n, and is played between the challenger and an adversary: Firstly, the challenger picks the challenge bit $b \in \{0, 1\}$, computes $(dk_i, pk_i) \leftarrow \mathsf{PKG}(1^k)$ for $i \in [n]$, and gives 1^k and $(pk_1, \dots, pk_n)$ to $\mathcal{A}$. $\mathcal{A}$ can adaptively make "left-or-right" (LR) queries and decryption queries. An LR query is of the form $(j \in [n], m_0, m_1)$ such that $|m_0| = |m_1|$, and the challenger responds to it with $\mathsf{Enc}(pk_j, m_b)$. A decryption query is of the form (j, c), and the challenger responds to it with $\mathsf{PDec}(sk_j, c)$, except that if c is a ciphertext that is some of the answers to previous LR queries, the challenger answers with $\perp$. Finally, $\mathcal{A}$ outputs a guess bit b' for b. $\mathcal{A}$ wins the game if $b = b'$. We define the advantage of $\mathcal{A}$ by $\mathsf{Adv}_{(\mathcal{A},n)}^{\mathtt{CCA\text{-}PKE}}(k) = |\Pr[b' = b] - 1/2|$.

We say a PKE scheme is *CCA secure in the multi-user setting*, if for all positive polynomials $n = n(k)$ and for any PPT adversary $\mathcal{A}$, $\mathsf{Adv}_{(\mathcal{A},n)}^{\mathtt{CCA\text{-}PKE}}(k)$ is negligible.

Bellare, Boldyreva, and Micali [9] showed that ordinary CCA security and CCA security in the multi-user setting are polynomially equivalent.

Multi-challenge Chosen Ciphertext Security We recall the definition of indistinguishability against multi-challenge chosen ciphertext attacks (mIND-CCA, for short) [9] of PKE, which is defined by the following game between the challenger $\mathcal{B}$ and an adversary $\mathcal{A}$: First, $\mathcal{B}$ picks the challenge bit $b \in \{0, 1\}$, computes $(pk, dk) \leftarrow \mathsf{PKG}(1^k)$, and gives pk to $\mathcal{A}$. $\mathcal{A}$ can adaptively make encryption queries. For an encryption query (m_0, m_1), where (m_0, m_1) is a message pair of equal length, $\mathcal{B}$ computes $c^* \leftarrow \mathsf{PEnc}(pk, m_b)$, and then returns the ciphertext c^* to $\mathcal{A}$. For a decryption query c, $\mathcal{B}$ responds with $m \leftarrow \mathsf{PDec}(dk, c)$, except that if c is one of the challenge ciphertexts c^*, as a response to the encryption queries, then the challenger returns $\perp$ to $\mathcal{A}$. Finally, $\mathcal{A}$ outputs a guess bit b' for b. $\mathcal{A}$ wins the game if $b = b'$. We define the advantage of $\mathcal{A}$ by $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{mINDCCA\text{-}PKE}}(k) = |\Pr[b' = b] - 1/2|$. We say a PKE scheme is multi-challenge IND-CCA secure, if for all PPT adversaries $\mathcal{A}$, $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{mINDCCA\text{-}PKE}}(k)$ is negligible. It was showed that ordinary IND-CCA security and multi-challenge IND-CCA security are polynomially equivalent [9].

2.10 Signature

A signature scheme consists of the following three algorithms $(\mathsf{SKG}, \mathsf{Sign}, \mathsf{SVer})$.

SKG This is the key generation algorithm that takes 1^k as input, and outputs a signing key sk and a verification key vk.

Sign This is the signing algorithm that takes a signing key sk and a message m as input, and outputs a signature σ.

SVer This is the verification algorithm that takes a verification key vk, a message m, and a signature σ as input, and outputs either $\top$ or $\perp$ (indicating whether the signature is valid or not).

We require the standard correctness for a signature scheme, namely, for any $(sk, vk) \leftarrow \mathsf{SKG}(1^k)$ and any message m, we have $\mathsf{SVer}(vk, m, \mathsf{Sign}(sk, m)) = \top$.

Strong Unforgeability [4] We recall the definition of strong unforgeability [4] of a signature scheme, which is defined by the following game between the challenger and an adversary $\mathcal{A}$. First, the challenger computes $(sk, vk) \leftarrow \mathsf{SKG}(1^k)$, and gives 1^k and vk to $\mathcal{A}$. $\mathcal{A}$ can adaptively make signing queries. For the i-th signing query on a message m_i, the challenger computes $\sigma_i \leftarrow \mathsf{Sign}(sk, m_i)$, returns σ_i to $\mathcal{A}$, and stores (m_i, σ_i). Finally, $\mathcal{A}$ outputs a message/signature pair (m^*, σ^*). $\mathcal{A}$ wins the game if $\mathsf{SVer}(vk, m^*, \sigma^*) = \top$ and $(m^*, \sigma^*) \neq (m_i, \sigma_i)$ for all i. We define the advantage of $\mathcal{A}$ by $\mathsf{Adv}_{\mathcal{A}}^{\mathtt{SUF\text{-}SIG}}(k) = \Pr[\mathcal{A} \text{ wins}]$.

We say a signature scheme is *strongly unforgeable*, if for any PPT adversary $\mathcal{A}$, $\mathsf{Adv}_{\mathcal{A}}^{\mathtt{SUF\text{-}SIG}}(k)$ is negligible.

One-time Strong Unforgeability. [4] One-time strong unforgeability [4] is defined by the following game between the challenger and an adversary $\mathcal{A}$. First, the challenger computes $(svk, ssk) \leftarrow \mathsf{SKG}(1^k)$, and gives 1^k and svk to $\mathcal{A}$. $\mathcal{A}$ can make a signing query only once. For the signing query on a message m', the challenger computes $\sigma' \leftarrow \mathsf{Sign}(ssk, m')$, returns σ' to $\mathcal{A}$, and stores (m', σ'). Finally, $\mathcal{A}$ outputs a message/signature pair (m^*, σ^*). $\mathcal{A}$ wins the game if $\mathsf{SVer}(svk, m^*, \sigma^*) = \top$ and $(m^*, \sigma^*) \neq (m', \sigma')$. We define the advantage of $\mathcal{A}$ by $\mathsf{Adv}_{\mathcal{A}}^{\mathtt{OT\text{-}sEUF\text{-}CMA}}(k) = \Pr[A \ wins]$.

We say that a signature scheme is *one-time strongly unforgeable*, if for any PPT adversary $\mathcal{A}$, $\mathsf{Adv}_{\mathcal{A}}^{\mathtt{OT\text{-}sEUF\text{-}CMA}}(k)$ is negligible.

Chapter 3

Re-splittable Threshold Public Key Encryption

3.1 Introduction

Threshold public key encryption (TPKE) is a kind of public key encryption where a secret key is distributed among n decryption servers so that partial decryption shares from at least t servers are needed for decryption. In TPKE, an entity called *combiner* has a ciphertext c that it wishes to decrypt. The combiner sends c to the decryption servers and receives partial decryption shares from at least t out of n decryption servers. Then, the combiner combines these t partial decryption shares into a complete decryption of c. No information about the plaintext is leaked, even if the number of the corrupted users is up to $t - 1$. Ideally, there is no other interaction in the system, namely the servers need not talk to each other during the decryption. Such threshold encryption systems are called non-interactive. We usually require that a TPKE scheme has *robustness*. That is, if decryption of a valid ciphertext fails, the combiner can identify the decryption servers that supplied invalid partial decryption shares.

Not only the functionality of TPKE is useful in the real world, but TPKE is used as a building block in a generic construction of another cryptosystem. Hanaoka et al. [58] showed a generic construction of proxy re-encryption (PRE) using TPKE as one of the building blocks. (Later, Ohata et al. [84] extended it to verifiable PRE.) To achieve the functionality and prove the security of the PRE scheme in [58, 84], the underlying TPKE scheme should have a special (but natural) property called *key re-splittability*. In TPKE with key re-splittability (re-splittable TPKE, for short), a secret key can be split into a set of secret key shares not only once, but also multiple times, and the security of the TPKE scheme is guaranteed as long as the number of corrupted secret key shares under the same splitting is smaller than the threshold. Since TPKE can control the distribution of information in the same way as a secret sharing, one would expect even more convenience of TPKE by considering the use of it to construct other cryptosystems such as multi-party protocols. The example of PRE suggests that the impact of TPKE would be significantly enhanced by the key re-splittability if it is also an essential property to prove the security of such cryptosystems based on TPKE. We can understand that re-splittable TPKE is a cryptosystem in which the mechanism of a proactive secret sharing [60] is applied to its secret key. That is, we can re-distribute the secret key shares without changing the original public/secret keys. As mentioned in [60], it is difficult in practice to protect secret information over a long period. To make matters worse, these days, threats of information leakage (by malware, for example) are greatly increasing. We consider that refreshing secret information is one of the promising countermeasures. Therefore, studying the re-splittability of TPKE is important.

Hanaoka et al. gave a formal security definition of re-splittable TPKE, and proposed a concrete construction by extending the ordinary TPKE scheme by Arita and Tsurudome [5]. In [58], however, only one construction was given. The main motivation of this paper is to show more constructions of re-splittable TPKE schemes.

3.1.1 Our Contribution

In this paper, we propose several new constructions of re-splittable TPKE by extending the existing (ordinary) TPKE schemes. All of these schemes can be proven secure under the well-established number-theoretic assumptions in the standard model. By directly applying the result of [58], we can also obtain new CCA secure proxy re-encryption schemes based on these assumptions.

Concretely, we present the following constructions:

1. **Re-splittable TPKE scheme based on the *decisional linear (DLIN) assumption*:** We construct the first re-splittable TPKE scheme based on the DLIN assumption on bilinear groups by extending Arita and Tsurudome's TPKE scheme [5]. In terms of computational costs and parameter sizes, our DLIN-based scheme is not as efficient as the re-splittable TPKE scheme of [58] based on the decisional bilinear Diffie-Hellman (DBDH) assumption. However, the message space of our scheme is not the target group but the source group of bilinear groups. Hence, our scheme is considered to be convenient when used with bilinear-group-based zero-knowledge proof systems such as the Groth-Sahai proofs [56].

2. **Re-splittable TPKE scheme based on the DBDH assumption *without signature*:** We construct a re-splittable TPKE scheme based on the DBDH assumption by extending Lai et al's TPKE scheme [76]. The scheme based on [58] is also based on the DBDH assumption, and needs a one-time strongly unforgeable signature in the construction. Our construction, however, does not need a one-time signature scheme. Ciphertext size of this scheme is smaller than that of [58] because we do not have to contain a signature and its verification key into a ciphertext of this re-splittable TPKE scheme.

3. **Re-splittable TPKE scheme based on the *hashed Diffie-Hellman (HDH) assumption*:** We construct a re-splittable TPKE scheme based on the HDH assumption on bilinear groups by extending Gan et al's TPKE scheme [47]. This scheme also does not need a signature scheme in the construction. Though the assumption on which the security is based is different from our second construction, all of the algorithms of this scheme are more efficient than it.

The efficiency of our schemes is not degraded at all from the corresponding underlying TPKE schemes. In general, when we extend an encryption scheme to achieve an additional functionality, the resultant scheme often suffers from unfavorable changes (e.g. stronger or less popular assumptions to prove its security, longer communication overheads, and/or higher computational cost). However, our extensions do not bring such changes. Our results thus suggest that key re-splittability of discrete logarithm (DL)-based TPKE is a very natural property which can be provided ordinarily. On the other hand, we cannot extend the previous TPKE schemes based on multiple encryption (e.g. [42, 99]) to the re-splittable ones through the methodology used in this paper. Therefore, we cannot assert that re-splittability is a trivial property in TPKE.

3.1.2 Related Work

We briefly review related works. Desmedt and Frankel [40] introduced the concept of TPKE. Their scheme needs an interaction among decryption servers to decrypt a ciphertext. The first non-interactive and CCA secure TPKE scheme was proposed by Shoup and Gennaro [104]. They proposed two constructions, one is based on the computational Diffie-Hellman (CDH) assumption and the other is based on the decisional Diffie-Hellman (DDH) assumption. To prove the security of these schemes, we need a random oracle. Later, Boneh, Boyen, and Halevi [22] proposed a TPKE scheme based on the DBDH assumption in the standard model. We can understand that this scheme is realized by the combination of the identity-based encryption scheme by Boneh and Boyen [21] and the CPA-to-CCA conversion technique by Canetti, Halevi, and Katz [32]. Subsequently, Arita and Tsurudome [5] proposed two more efficient TPKE schemes than [22] by using tag-based encryption [72]. One of their schemes is based on the DBDH assumption and the other is based on the DLIN assumption. Later, Lai et al. [76] proposed a new technique to construct an efficient public key encryption scheme and applied it to construct a TPKE scheme. Though [22] and [5] use a signature scheme in the constructions, the scheme in [76] does not use it. Subsequently, Gan et al. [47] proposed a TPKE scheme based on the HDH assumption. We can understand that this scheme can be seen as the scheme by [76] in which the bilinear map used as a key derivation (to generate a mask for hiding a plaintext) is replaced with a hash function. (The relation between Gan et al.'s scheme [47] and Lai et al.'s scheme [76] is similar to the relation between the Kiltz' key encapsulation mechanism (KEM) based on the gap HDH (GHDH) assumption [73] and Boyen et al.'s KEM based on the DBDH assumption [28]. Note that on the bilinear groups, the GHDH assumption and the HDH assumption become equivalent.) Libert and Yung [78] proposed an adaptively secure TPKE scheme by using composite order groups. The same authors [79] also proposed a framework for the construction of adaptively secure TPKE based on the so-called all-but-one perfectly sound threshold hash proof systems, from which we can obtain concrete instantiations based on the DLIN, the external Diffie-Hellman (XDH) assumptions in prime order bilinear groups, and one based on some assumption in composite order bilinear groups.

There are some schemes based on non-discrete logarithm type assumptions. Wee [108] showed a CCA secure TPKE scheme based on the factoring assumption in the random oracle model. Dodis and Katz [42] and Sakai et al. [99] proposed a generic construction of TPKE via multiple encryption technique. Note that the model of TPKE [42] is different from [22], (arguably) currently a more popular model of TPKE on which the security model of re-splittable TPKE is based.

In summary, to the best of our knowledge, none of the previous works other than [58] considered the key re-splittability of TPKE.

3.1.3 Chapter Organization

The remainder of this chapter is organized as follows. In Section 3.2, we review the model and security definitions of re-splittable TPKE. In Section 3.3, we present our new constructions of re-splittable TPKE schemes. In Section 3.4, we give a comparison on the efficiency among re-splittable TPKE schemes. Section 3.5 is the conclusion of this chapter.

3.2 Re-splittable Threshold Public Key Encryption

In this section, we review the model and the security definitions of re-splittable TPKE. The concept of re-splittable threshold public key encryption (TPKE) was introduced in [58]. It is a special class of TPKE in which a secret key can be split multiple times, and security of the scheme is maintained as long as the number of corrupted secret key shares *under the same splitting* is less than the threshold.

3.2.1 Model

A re-splittable TPKE scheme consists of the following six PPT algorithms:

TKG This is the key generation algorithm that takes 1^k, n, and t such that $0 < t \leq n$ as input, and outputs a public key tpk and a secret key tsk. This process is written as $(tpk, tsk) \leftarrow \mathsf{TKG}(1^k, n, t)$.

TEnc This is the encryption algorithm that takes tpk and a plaintext m as input, and outputs a ciphertext c. This process is written as $c \leftarrow \mathsf{TEnc}(tpk, m)$.

TSplit This is the key-splitting algorithm that takes tsk as input, and outputs n secret key shares $tsk_1, \cdots, tsk_n$ and a verification key tvk. This process is written as $(tsk_1, \cdots, tsk_n, tvk) \leftarrow \mathsf{TSplit}(tsk)$.

TShDec This is the share-decryption algorithm that takes tpk, a secret key share tsk_i (where $i \in [n]$) output by TSplit, and c as input, and outputs a decryption share μ_i (which could be the special symbol $\perp$ meaning that c is invalid). This process is written as $\mu_i \leftarrow \mathsf{TShDec}(tpk, tsk_i, c)$.

TShVer This is the share-verification algorithm that takes tpk, tvk, c, an index $i \in [n]$, and a decryption share μ as input, and outputs $\top$ or $\perp$. When the output is $\top$, we say that μ is a valid decryption share of the ciphertext c. This process is written as $\top/\perp \leftarrow \mathsf{TShVer}(tpk, tvk, c, i, \mu)$.

TCom This is the combining algorithm that takes tpk, tvk, c, and t decryption shares (generated under distinct secret key shares) as input, and outputs a decryption result m (which could be the special symbol $\perp$). This process is written as $m/\perp \leftarrow \mathsf{TCom}(tpk, tvk, c, \{\mu_1, \cdots, \mu_t\})$.

Correctness: For any $k \in \mathbb{N}$, any polynomials t, n such that $0 < t \leq n$, any $(tpk, tsk) \leftarrow \mathsf{TKG}(1^k, n, t)$ and any $(tsk_1, \cdots, tsk_n, tvk) \leftarrow \mathsf{TSplit}(tsk)$, we require the following two correctness properties:

1. For any ciphertext c, if $\mu = \mathsf{TShDec}(tpk, tsk_i, c)$, then we have $\mathsf{TShVer}(tpk, tvk, c, i, \mu) = \top$.
2. For any m, if c is output from $\mathsf{TEnc}(tpk, m)$ and $S = \{\mu_{s_1}, \cdots, \mu_{s_t}\}$ is a set of decryption shares (i.e. $\mu_{s_i} = \mathsf{TShDec}(tpk, tsk_{s_i}, c)$ for all $i \in [t]$), then we have $\mathsf{TCom}(tpk, tvk, c, S) = m$.

3.2.2 Security Definitions

In this subsection, we review the two security definitions for re-splittable TPKE which are defined in [58].

Chosen Ciphertext Security: CCA security of a re-splittable TPKE scheme is defined using the following game between a challenger and an adversary $\mathcal{A}$: The challenger first runs $(tpk, tsk) \leftarrow \mathsf{TKG}(1^k, n, t)$ and gives tpk to $\mathcal{A}$. Then $\mathcal{A}$ can adaptively make the following types of queries.

Split&corruption query: On input a set of indices $I = \{s_1, \cdots, s_{t-1}\}$, the challenger runs $(tsk_1, \cdots, tsk_n, tvk) \leftarrow \mathsf{TSplit}(tsk)$ and returns $(tsk_{s_1}, \cdots, tsk_{s_{t-1}}, tvk)$ to $\mathcal{A}$. The challenger also stores $\{tsk_i\}_{i \in [n]}$ and tvk for later share decryption queries from $\mathcal{A}$.

Share decryption query: On input (tvk, i, c), where tvk is required to be one of the answers to previously asked split&corruption queries, $i \in [n]$, and $c \neq c^*$, the challenger finds tsk_i that is previously generated together with tvk, and returns a decryption share $\mu_i \leftarrow \mathsf{TShDec}(tpk, tsk_i, c)$ to $\mathcal{A}$.

Challenge query: This query is asked only once. On input (m_0, m_1), the challenger randomly picks $b \in \{0, 1\}$ and returns $c^* \leftarrow \mathsf{TEnc}(tpk, m_b)$ to $\mathcal{A}$.

Finally, $\mathcal{A}$ outputs its guess b' for b, and wins the game if $b = b'$. We define the advantage of $\mathcal{A}$ by $\mathsf{Adv}^{\mathtt{CCA\text{-}RS\text{-}TPKE}}_{(\mathcal{A},n,t)}(k) = |\Pr[b = b'] - 1/2|$.

Definition 1. *We say that a re-splittable TPKE scheme is* CCA secure*, if for any PPT adversary $\mathcal{A}$ and for any polynomials t and n with $0 < t \leq n$, $\mathsf{Adv}^{\mathtt{CCA\text{-}RS\text{-}TPKE}}_{(\mathcal{A},n,t)}(k)$ is negligible.*

Decryption Consistency: Decryption consistency is defined using the game which is defined in the same way as the CCA game, except that the challenge query is not considered.

The adversary $\mathcal{A}$ finally outputs a ciphertext c, a verification key tvk, and two sets of decryption shares $S = \{\mu_{s_1}, \ldots, \mu_{s_t}\}$ and $S' = \{\mu'_{s'_1}, \ldots, \mu'_{s'_t}\}$. $\mathcal{A}$ wins the game if the following four conditions are simultaneously satisfied.

(a) tvk is one of verification keys returned as a response to one of $\mathcal{A}$'s split&corruption queries;

(b) All shares in S and S' are valid for a ciphertext c under tvk. That is, $\mathsf{TShVer}(tpk, tvk, c, s_i, \mu_{s_i}) = \mathsf{TShVer}(tpk, tvk, c, s'_i, \mu'_{s'_i}) = \top$ for all $i \in [t]$;

(c) S and S' are sets that are distinct regardless of re-ordering the elements;

(d) $\mathsf{TCom}(tpk, tvk, c, S) \neq \mathsf{TCom}(tpk, tvk, c, S')$.

We let $\mathsf{Adv}^{\mathtt{DC\text{-}RS\text{-}TPKE}}_{(\mathcal{A},n,t)}(k)$ denote the probability of $\mathcal{A}$ winning in this game.

Definition 2. *We say that a re-splittable TPKE scheme has* decryption consistency*, if for any PPT adversary $\mathcal{A}$ and for any polynomials t and n such that $0 < t \leq n$, $\mathsf{Adv}^{\mathtt{DC\text{-}RS\text{-}TPKE}}_{(\mathcal{A},n,t)}(k)$ is negligible.*

3.3 Proposed Re-splittable TPKE Schemes

In this section, we propose several concrete re-splittable TPKE schemes, and prove their security. Each of our schemes is an extension of the existing ordinary TPKE schemes, and we show how to implement the TSplit algorithm for each of them. For each scheme, the assumption on which the security is based is exactly the same as the one on which the original scheme is based. Furthermore, the efficiency (ciphertext size and the computational costs) of each scheme is also essentially the same as the corresponding underlying scheme.

3.3.1 Construction Based on Arita and Tsurudome Scheme

We show the construction of a re-splittable TPKE scheme based on the Arita and Tsurudome scheme [5]. Actually, Arita and Tsurudome showed two TPKE schemes, one is based on the DBDH assumption and the other is based on the DLIN assumption. The DBDH-based TPKE scheme was already extended to re-splittable TPKE by Hanaoka et al. [58]. Here, therefore, we show the construction of re-splittable TPKE scheme based on the DLIN assumption in [5].

Our proposed DLIN-based scheme is as in Fig.3.1. Here, we call it eAT.

Theorem 1. *If the DLIN assumption on $(\mathbb{G}, \mathbb{G}_T)$ holds, the signature scheme is one-time strongly unforgeable, and H is a collision resistant hash function, then the re-splittable TPKE scheme eAT is CCA secure.*

Proof of Theorem 1. Let $\mathcal{A}$ be an arbitrary PPT adversary that attacks the re-splittable TPKE scheme eAT in the game of the CCA security. Using the adversary $\mathcal{A}$, we build an algorithm $\mathcal{B}$ that solves the DLIN problem on $(\mathbb{G}, \mathbb{G}_T)$. Given $(1^k, p, \mathbb{G}, \mathbb{G}_T, \boldsymbol{e}, g_1, g_2, g_3, g_1^a, g_2^b, W)$ as input, $\mathcal{B}$ proceeds as follows.

1. **Setup:** Algorithm $\mathcal{B}$ does the following.

   (a) $\mathcal{B}$ selects a collision resistant hash function H.

   (b) $\mathcal{B}$ executes $(svk^*, ssk^*) \leftarrow \mathsf{SKG}(1^k)$ and sets $\tau^* := H(svk^*)$.

   (c) $\mathcal{B}$ chooses the random $c_1, c_2 \leftarrow \mathbb{Z}_p$ and sets $u_1 := g_3^{-\tau^*} g_1^{c_1}, u_2 := g_3^{-\tau^*} g_2^{c_2}$. $\mathcal{B}$ sets $tpk := (p, \mathbb{G}, \mathbb{G}_T, \boldsymbol{e}, H, g_1, g_2, g_3, u_1, u_2)$.

```
$\mathsf{TKG}(1^k, n, t)$ :
    $(p, \mathbb{G}, \mathbb{G}_T, \boldsymbol{e}) \leftarrow \mathsf{BG}(1^k)$
    Select a collision resistant hash function $H : \{0,1\}^* \to \mathbb{Z}_p$.
    $g_1 \leftarrow \mathbb{G}$;   $x_1, x_2, y_1, y_2 \leftarrow \mathbb{Z}_p$
    $g_3 := g_1^{x_1}$;   $g_2 := g_3^{\frac{1}{x_2}}$;   $u_1 := g_1^{y_1}$;   $u_2 := g_2^{y_2}$
    $tpk := (p, \mathbb{G}, \mathbb{G}_T, \boldsymbol{e}, H, g_1, g_2, g_3, u_1, u_2)$,   $tsk := (x_1, x_2)$
    Return $(tpk, tsk)$.
$\mathsf{TEnc}(tpk, m)$ :
    $(p, \mathbb{G}, \mathbb{G}_T, \boldsymbol{e}, H, g_1, g_2, g_3, u_1, u_2) \leftarrow tpk$
    $(svk, ssk) \leftarrow \mathsf{SKG}(1^k)$
    $r_1, r_2 \leftarrow \mathbb{Z}_p$
    $C_1 := g_1^{r_1}$;   $C_2 := g_2^{r_2}$;   $D_1 := (g_3^{H(svk)} u_1)^{r_1}$
    $D_2 := (g_3^{H(svk)} u_2)^{r_2}$;   $E := m \cdot g_3^{r_1 + r_2}$
    $\sigma \leftarrow \mathsf{Sign}(ssk, C_1 \| C_2 \| D_1 \| D_2 \| E)$
    Return $c := (C_1, C_2, D_1, D_2, E, \sigma, svk)$.
$\mathsf{TSplit}(tsk)$ :
    $(x_1, x_2) \leftarrow tsk$
    $f_1, f_2 \leftarrow \mathbb{Z}_p[X]$ satisfying $\deg(f_1) = \deg(f_2) = t - 1$,
                                                  $f_1(0) = x_1$, and $f_2(0) = x_2$.
    $\forall i \in [n] : tsk_i := (tsk_{1.i}, tsk_{2.i}) := (f_1(i), f_2(i))$
    $\forall i \in [n] : (tvk_{1.i}, tvk_{2.i}) := (g_1^{f_1(i)}, g_2^{f_2(i)})$
    $tvk := (tvk_{1.i}, tvk_{2.i})_{i \in [n]}$
    Return $((tsk_i)_{i \in [n]}, tvk)$.
$\mathsf{TShDec}(tpk, tsk_i, c)$ :
    $(p, \mathbb{G}, \mathbb{G}_T, \boldsymbol{e}, H, g_1, g_2, g_3, u_1, u_2) \leftarrow tpk$;   $(tsk_{1.i}, tsk_{2.i}) \leftarrow tsk_i$
    $(C_1, C_2, D_1, D_2, \sigma, svk) \leftarrow c$
    If $\mathsf{SVer}(svk, C_1 \| C_2 \| D_1 \| D_2 \| E, \sigma) = \bot$ then return $\mu_i := \bot$.
    If $\boldsymbol{e}(C_1, g_3^{H(svk)} u_1) \neq \boldsymbol{e}(D_1, g_1)$ or $\boldsymbol{e}(C_2, g_3^{H(svk)} u_2) \neq \boldsymbol{e}(D_2, g_2)$
                                                  then return $\mu_i := \bot$.
    Return $\mu_i := (\mu_{1.i}, \mu_{2.i}) := (C_1^{tsk_{1.i}}, C_2^{tsk_{2.i}})$.
$\mathsf{TShVer}(tpk, tvk, c, i, \mu_i)$ :
    $(p, \mathbb{G}, \mathbb{G}_T, \boldsymbol{e}, H, g_1, g_2, g_3, u_1, u_2) \leftarrow tpk$;   $(tvk_{1.i}, tvk_{2.i})_{i \in [n]} \leftarrow tvk$
    $(C_1, C_2, D_1, D_2, E, \sigma, svk) \leftarrow c$
    If $\mathsf{SVer}(svk, \sigma, C_1 \| C_2 \| D_1 \| D_2 \| E) = \bot$ or
    $\boldsymbol{e}(C_1, g_3^{H(svk)} u_1) \neq \boldsymbol{e}(D_1, g_1)$ or $\boldsymbol{e}(C_2, g_3^{H(svk)} u_2) \neq \boldsymbol{e}(D_2, g_2)$ then
        If $\mu_i = \bot$ then return $\top$ else return $\bot$.
    End if
    $(\mu_{1.i}, \mu_{2.i}) \leftarrow \mu_i$
    If $\boldsymbol{e}(\mu_{1.i}, g_1) = \boldsymbol{e}(C_1, tvk_{1.i})$ and $\boldsymbol{e}(\mu_{2.i}, g_2) = \boldsymbol{e}(C_2, tvk_{2.i})$
                                                  then return $\top$ else return $\bot$.
$\mathsf{TCom}(tpk, tvk, c, \{\mu_{s_i}\}_{i \in [t]})$ where $\{s_i\}_{i \in [t]} \subset [n]$ :
    If $\exists i \in [t] : \mathsf{TShVer}(tpk, tvk, c, s_i, \mu_{s_i}) = \bot$ or $\mu_{s_i} = \bot$ then return $\bot$.
    $\forall i \in [t] : (\mu_{1.s_i}, \mu_{2,s_i}) \leftarrow \mu_{s_i}$
    Return $m = E / \Big( (\prod_{i=1}^{t} \mu_{1.s_i}^{\lambda_{1.i}}) \cdot (\prod_{i=1}^{t} \mu_{2.s_i}^{\lambda_{2.i}}) \Big)$ using
        Lagrange coefficients $\{\lambda_{1.i}\}_{i \in [t]}$ satisfying $f_1(0) = \sum_{i=1}^{t} \lambda_{1.i} f_1(s_i)$
                        and $\{\lambda_{2.i}\}_{i \in [t]}$ satisfying $f_2(0) = \sum_{i=1}^{t} \lambda_{2.i} f_2(s_i)$.
```

Figure 3.1: The re-splittable TPKE scheme eAT.

(d) $\mathcal{B}$ gives tpk to $\mathcal{A}$.

2. **Split&corruption query:** When $\mathcal{A}$ issues a split&corruption query $I = \{s_i\}_{i\in[t-1]}$, $\mathcal{B}$ does the following.

   (a) $\mathcal{B}$ picks $2t-2$ random integers $\alpha_{s_1}, \beta_{s_1}, \cdots, \alpha_{s_{t-1}}, \beta_{s_{t-1}} \leftarrow \mathbb{Z}_p$. We let $f_1, f_2 \in \mathbb{Z}_p[X]$ be two polynomials of degree $t-1$ defined by $f_1(0) := a$, $f_1(s_i) := \alpha_{s_i}$ for all $i \in [t-1]$, $f_2(0) := b$, $f_2(s_i) := \beta_{s_i}$ for all $i \in [t-1]$. (Note that $\mathcal{B}$ does not know f_1 and f_2 entirely.) For each $i \in [t-1]$, $\mathcal{B}$ sets $tsk_{s_i} := (\alpha_{s_i}, \beta_{s_i})$.

   (b) i. For $j \in I$, $\mathcal{B}$ sets $v_{1.j} := g_1^{\alpha_j}$ and $v_{2.j} := g_2^{\beta_j}$.

      ii. For $j \notin I$, $\mathcal{B}$ computes $v_{1.j} = g_3^{\lambda_0}(g_1^{\alpha_{s_1}})^{\lambda_1}\cdots(g_1^{\alpha_{s_{t-1}}})^{\lambda_{t-1}}$ and $v_{2.j} = g_3^{\lambda'_0}(g_2^{\beta_{s_1}})^{\lambda'_1}\cdots(g_2^{\beta_{s_{t-1}}})^{\lambda'_{t-1}}$, where $\{\lambda_0, \cdots, \lambda_{t-1}\}$ and $\{\lambda'_0, \cdots, \lambda'_{t-1}\}$ are the Lagrange coefficients satisfying $f_1(j) = \lambda_0 f_1(0) + \Sigma_{\ell=1}^{t-1}\lambda_\ell f_1(s_\ell)$ and $f_2(j) = \lambda'_0 f_2(0) + \Sigma_{\ell=1}^{t-1}\lambda'_\ell f_2(s_\ell)$ for the polynomials f_1 and f_2. Note that $(v_{1.j}, v_{2.j})$ satisfies $v_{1.j} = g_1^{f_1(j)}$ and $v_{2.j} = g_2^{f_2(j)}$.

      iii. $\mathcal{B}$ sets $tvk := \{(v_{1.i}, v_{2.i})\}_{i\in[n]}$.

   (c) $\mathcal{B}$ returns $((tsk_{s_i})_{i\in[t-1]}, tvk)$ to $\mathcal{A}$.

3. **Challenge query:** Adversary $\mathcal{A}$ outputs two equal-length messages m_0 and m_1. $\mathcal{B}$ flips a fair coin $\theta \in \{0,1\}$, and sets $(C_1^*, C_2^*, D_1^*, D_2^*, E^*) := (g_1^a, g_2^b, g_1^{ac_1}, g_2^{bc_2}, m_\theta W)$. Then, $\mathcal{B}$ calculates $\sigma^* \leftarrow \mathsf{Sign}(ssk^*, C_1^*\|C_2^*\|D_1^*\|D_2^*\|E^*)$. $\mathcal{B}$ returns the challenge ciphertext $c^* := (C_1^*, C_2^*, D_1^*, D_2^*, E^*, \sigma^*, svk^*)$ to $\mathcal{A}$.

4. **Share decryption query:** When $\mathcal{A}$ issues a share decryption query (tvk, i, c), $\mathcal{B}$ does the following, where $c = (C_1, C_2, D_1, D_2, E, \sigma, svk)$.

   (a) If $\mathsf{SVer}(svk, C_1\|C_2\|D_1\|D_2\|E, \sigma) = \bot$, then $\mathcal{B}$ returns $\bot$ to $\mathcal{A}$.

   (b) Else if $H(svk) = H(svk^*)$, then $\mathcal{B}$ outputs a random bit and aborts.

   (c) Else $\mathcal{B}$ first tests $\boldsymbol{e}(C_1, z^{H(svk)}u_1) \overset{?}{=} \boldsymbol{e}(D_1, g_1)$ and $\boldsymbol{e}(C_2, z^{H(svk)}u_2) \overset{?}{=} \boldsymbol{e}(D_2, g_2)$ to check the validity of the ciphertext. If this validity test fails, $\mathcal{B}$ returns $\bot$ to $\mathcal{A}$.

   (d) If $\boldsymbol{e}(C_1, z^{H(svk)}u_1) \neq \boldsymbol{e}(D_1, g_1)$ or $\boldsymbol{e}(C_2, z^{H(svk)}u_2) \neq \boldsymbol{e}(D_2, g_2)$, $\mathcal{B}$ returns $\bot$ to $\mathcal{A}$.

   (e) Let $I = \{s_\ell\}_{\ell\in[t-1]}$ be the set of indices corresponding to tvk. $\mathcal{B}$ proceeds as follows:

      i. If $j \in I$, $\mathcal{B}$ computes $\mu_{1.j} = C_1^{f_1(j)}$ and $\mu_{2.j} = C_2^{f_2(j)}$. Then, $\mathcal{B}$ returns $\mu_j := (\mu_{1.j}, \mu_{2.j})$ to $\mathcal{A}$.

      ii. Otherwise, $\mathcal{B}$ computes the Lagrange coefficients $\{\lambda_1, \cdots, \lambda_t\}$ and λ^* satisfying $f_1(0) = \lambda^* f_1(j) + \Sigma_{\ell=1}^{t-1}\lambda_\ell f_1(s_\ell)$ for the polynomial f_1. Then, $\mathcal{B}$ sets

$$\mu_{1.j} := \left(\frac{\left(\frac{D_1}{C_1^{c_1}}\right)^{\frac{1}{H(svk)-H(svk^*)}}}{C_1^{\Sigma_{\ell=1}^{t-1}\lambda_\ell\alpha_{s_\ell}}}\right)^{\frac{1}{\lambda^*}}.$$

Similarly, $\mathcal{B}$ computes the Lagrange coefficients $\{\lambda'_1, \cdots, \lambda'_t\}$ and λ'^* satisfying $f_2(0) = \lambda'^* f_2(j) + \Sigma_{\ell=1}^{t-1} \lambda'_\ell f_2(s_\ell)$ for the polynomial f_2. Then, $\mathcal{B}$ sets

$$\mu_{2.j} := \left(\frac{\left(\frac{D_2}{C_2^{c_2}}\right)^{\frac{1}{H(svk)-H(svk^*)}}}{C_2^{\Sigma_{\ell=1}^{t-1}\lambda'_\ell \beta_{s_\ell}}} \right)^{\frac{1}{\lambda'^*}}.$$

Finally, $\mathcal{B}$ returns $\mu_j := (\mu_{1.j}, \mu_{2.j})$ to $\mathcal{A}$.

Finally, when $\mathcal{A}$ terminates with its guess bit $\theta' \in \{0,1\}$, $\mathcal{B}$ outputs 1 if $\theta = \theta'$, otherwise outputs 0 and terminates.

The above completes the description of $\mathcal{B}$. Because of the one-time strong unforgeability of the signature scheme and collision resistance of the hash function H, the probability that $\mathcal{B}$ aborts during the simulation is negligible. More specifically, the case $H(svk) = H(svk^*)$ can be further divided into two cases $svk = svk^*$ and $svk \neq svk^*$, where the probability that the former case occurs can be shown to be negligible due to the one-time strong unforgeability of the signature scheme, and the probability that the latter case occurs can be shown to be negligible due to the collision resistance of H. Except for that, $\mathcal{B}$ perfectly simulates the CCA game for $\mathcal{A}$ if $W = g_3^{a+b}$. When W is a random element, the view of $\mathcal{B}$ is independent of $\mathcal{A}$'s challenge bit θ. Therefore, $\mathcal{B}$'s advantage of solving the DLIN problem can be estimated as $\mathsf{Adv}_{\mathcal{B}}^{\mathrm{DLIN}} \geq \mathsf{Adv}_{(\mathcal{A},n,t)}^{\texttt{CCA-RS-TPKE}}(k) - (negl.)$. This completes the proof of Theorem 1. $\square$

Theorem 2. *The re-splittable TPKE scheme* eAT *has decryption consistency unconditionally.*

Proof of Theorem 2. Let $\mathcal{A}$ be an arbitrary adversary attacking the TPKE scheme eAT in the decryption consistency game. Suppose $\mathcal{A}$ outputs $c = (C_1, C_2, D_1, D_2, E, \sigma, svk), tvk$, $S = (\mu_{s_1} = (\mu_{1.s_1}, \mu_{2.s_1}), \cdots, \mu_{s_t} = (\mu_{1.s_t}, \mu_{2.s_t}))$, $S' = (\mu'_{s'_1} = (\mu'_{1.s'_1}, \mu'_{2.s'_1}), \cdots, \mu'_{s'_t} = (\mu'_{1.s'_t}, \mu'_{2.s'_t}))$.

If the shares μ_{s_i} in S are valid, they must satisfy $\boldsymbol{e}(\mu_{1.s_i}, g_1) = \boldsymbol{e}(C_1, g_1^{f_1(s_i)})$ and $\boldsymbol{e}(\mu_{2.s_i}, g_2) = \boldsymbol{e}(C_2, g_2^{f_2(s_i)})$. Therefore, $\mu_{1.s_i} = C_1^{f_1(s_i)}$ and $\mu_{2.s_i} = C_2^{f_2(s_i)}$. Then, it holds $\prod_{i=1}^t \mu_{1.s_i}^{\lambda_{1.i}} = C_1^{\Sigma_{i=1}^t \lambda_{1.i} f_1(s_i)} = C_1^{x_1}$ and $\prod_{i=1}^t \mu_{2.s_i}^{\lambda_{2.i}} = C_2^{\Sigma_{i=1}^t \lambda_{2.i} f_2(s_i)} = C_2^{x_2}$, where $\{\lambda_{1.i}\}_{i\in[t]}$ and $\{\lambda_{2.i}\}_{i\in[t]}$ are Lagrange coefficients computed in TCom. Therefore, $\mathsf{TCom}(tpk, tvk, c, S)$ $= E/\Big((\prod_{i=1}^t \mu_{1.s_i}^{\lambda_{1.i}}) \cdot (\prod_{i=1}^t \mu_{2.s_i}^{\lambda_{2.i}}) \Big) = E/(C_1^{x_1} \cdot C_2^{x_2})$.

Similarly, if the shares $\mu'_{s'_i}$ in S' are valid, we have $\prod_{i=1}^t {\mu'}_{1.s'_i}^{\lambda'_{1.i}} = C_1^{\Sigma_{i=1}^t \lambda'_{1.i} f_1(s'_i)} = C_1^{x_1}$ and $\prod_{i=1}^t {\mu'}_{2.s'_i}^{\lambda'_{2.i}} = C_2^{\Sigma_{i=1}^t \lambda'_{2.i} f_2(s'_i)} = C_2^{x_2}$, where $\{\lambda'_{1.i}\}_{i\in[t]}$ and $\{\lambda'_{2.i}\}_{i\in[t]}$ are Lagrange coefficients computed in TCom. Therefore, $\mathsf{TCom}(tpk, tvk, c, S') = E/\Big((\prod_{i=1}^t \mu_{1.s_i}^{\lambda'_{1.i}}) \cdot (\prod_{i=1}^t \mu_{2.s_i}^{\lambda'_{2.i}}) \Big) = E/(C_1^{x_1} \cdot C_2^{x_2})$.

We have seen that the combined values from the sets S and S' cannot be distinct. Thus, we have $\mathsf{Adv}_{(\mathcal{A},n,t)}^{\texttt{DC-RS-TPKE}}(k) = 0$ for any (even computationally unbounded) adversary $\mathcal{A}$. $\square$

3.3.2 Construction Based on Lai et al's scheme

Here, we show the construction of a re-splittable TPKE scheme based on the Lai et al. scheme [76] as in Fig.3.2. Here, we call it eLDLK. The scheme based on [58] is also based on the DBDH assumption, and needs a one-time strongly unforgeable signature in the construction. Our construction, however, does not need a signature scheme.

Theorem 3. *If the DBDH assumption holds in $(\mathbb{G}, \mathbb{G}_T)$ and H is a collision resistant hash function, then the re-splittable TPKE scheme* eLDLK *is CCA secure.*

Proof of Theorem 3. Let $\mathcal{A}$ be an arbitrary adversary that attacks the re-splittable TPKE scheme eLDLK in the game of CCA security. Using the adversary $\mathcal{A}$, we build an algorithm $\mathcal{B}$ that solves the DBDH problem on $(\mathbb{G}, \mathbb{G}_T)$. Given $(1^k, p, \mathbb{G}, \mathbb{G}_T, \boldsymbol{e}, g, g^a, g^b, g^c, W)$ as input, algorithm $\mathcal{B}$ proceeds as follows. (The aim of $\mathcal{B}$ is to distinguish whether $W = \boldsymbol{e}(g,g)^{abc}$ or not.)

1. **Setup:** Algorithm $\mathcal{B}$ does the following.

   (a) $\mathcal{B}$ selects a collision resistant hash function H and chooses $x_1, x_2, x_3, x_4, x_5, x_6 \leftarrow \mathbb{Z}_p$.

   (b) $\mathcal{B}$ sets $g_1 := g^a, h := g^b, u := g_1^{x_1} g^{x_2}, v := g_1^{x_3} g^{x_4}, d := g_1^{x_5} g^{x_6}$, and sets $tpk := (p, \mathbb{G}, \mathbb{G}_T, \boldsymbol{e}, H, g, g_2, Z = \boldsymbol{e}(g_1, h), u, v, d)$.

   (c) $\mathcal{B}$ gives tpk to $\mathcal{A}$.

2. **Split&Corruption query:** When $\mathcal{A}$ issues a split&corruption query $I = \{s_i\}_{i \in [t-1]}$, $\mathcal{B}$ does the following.

   (a) $\mathcal{B}$ picks $t-1$ random integers $\alpha_{s_1}, \cdots, \alpha_{s_t} \leftarrow \mathbb{Z}_p$. We let $f \in \mathbb{Z}_p[X]$ be a polynomial of degree $t-1$ defined by $f(0) := a$, $f(s_i) := \alpha_{s_i}$ for all $i \in [t-1]$. (Note that $\mathcal{B}$ does not know f entirely.) $\mathcal{B}$ sets $\{tsk_{s_i}\} := \{\alpha_{s_i}\}$ for all $i \in [t-1]$.

   (b) i. For $j \in I$, $\mathcal{B}$ sets $tvk_j := g^{\alpha_j}$.

   ii. For $j \notin I$, $\mathcal{B}$ computes $v_j = g_1^{\lambda_0}(g^{\alpha_{s_1}})^{\lambda_1} \cdots (g^{\alpha_{s_{t-1}}})^{\lambda_{t-1}}$, where $\lambda_0, \cdots, \lambda_{t-1} \in \mathbb{Z}_p$ are the Lagrange coefficients satisfying $f(j) = \lambda_0 f(0) + \Sigma_{\ell=1}^{t-1} \lambda_\ell f(s_\ell)$ for the polynomial f. Note that v_j satisfies $v_j = g^{f(j)}$.

   iii. $\mathcal{B}$ sets $tvk := (v_j)_{j \in [n]}$.

   (c) $\mathcal{B}$ returns $(\{tsk_{s_i}\}_{i \in [t-1]}, tvk)$ to $\mathcal{A}$.

3. **Challenge query:** Adversary $\mathcal{A}$ outputs two equal-length messages m_0 and m_1. First, $\mathcal{B}$ flips a fair coin $\theta \in \{0,1\}$, and sets $(C_0^*, C_1^*) := (m_\theta W, g^c)$. Then, $\mathcal{B}$ computes $\omega^* = H(C_0^*, C_1^*)$ and $\tau^* = \frac{x_1 \omega^* + x_5}{x_3} \mod p$. $\mathcal{B}$ sets $C_2^* := (g^c)^{\omega x_2 + \tau^* x_4 + x_6}$. $\mathcal{B}$ returns the challenge ciphertext $c^* := (C_0^*, C_1^*, C_2^*, \tau^*)$ to $\mathcal{A}$.

4. **Share decryption query:** When $\mathcal{A}$ issues a share decryption query (tvk, i, c), $\mathcal{B}$ does the following, where $c = (C_0, C_1, C_2, \tau)$.

   (a) If $c = c^*$, then $\mathcal{B}$ returns $\mu_i := \perp$ to $\mathcal{A}$.

```
$\mathsf{TKG}(1^k, n, t)$ :
    $(p, \mathbb{G}, \mathbb{G}_T, \boldsymbol{e}) \leftarrow \mathsf{BG}(1^k)$
    Select a collision resistant hash function $H : \{0,1\}^* \to \mathbb{Z}_p$.
    $g, g_2, u, v, d \leftarrow \mathbb{G}$;   $x \leftarrow \mathbb{Z}_p$;   $g_1 := g^x$;   $Z := \boldsymbol{e}(g_1, g_2)$
    $tpk := (p, \mathbb{G}, \mathbb{G}_T, \boldsymbol{e}, H, g, g_2, Z, u, v, d)$;   $tsk := x$
    Return $(tpk, tsk)$.
$\mathsf{TEnc}(tpk, m)$ :
    $(p, \mathbb{G}, \mathbb{G}_T, \boldsymbol{e}, H, g, g_2, Z, u, v, d) \leftarrow tpk$
    $\tau, r \leftarrow \mathbb{Z}_p$
    $C_0 := m \cdot Z^r$;   $C_1 := g^r$;   $\omega := H(C_0, C_1)$;   $C_2 := (u^\omega v^\tau d)^r$
    Return $c := (C_0, C_1, C_2, \tau)$.
$\mathsf{TSplit}(tsk)$ :
    $x \leftarrow tsk$
    $f \leftarrow \mathbb{Z}_p[X]$ satisfying $\deg(f) = t-1$ and $f(0) = x$
    $\forall i \in [n] : tsk_i := g_2^{f(i)}$
    $\forall i \in [n] : tvk_i := g^{f(i)}$
    $tvk := (tvk_i)_{i \in [n]}$
    Return $((tsk_i)_{i \in [n]}, tvk)$.
$\mathsf{TShDec}(tpk, tsk_i, c)$ :
    $(p, \mathbb{G}, \mathbb{G}_T, \boldsymbol{e}, H, g, g_2, Z, u, v, d) \leftarrow tpk$;   $(C_0, C_1, C_2, \tau) \leftarrow c$
    $\omega := H(C_0, C_1)$
    If $\boldsymbol{e}(C_1, u^\omega v^\tau d) \neq \boldsymbol{e}(g, C_2)$ then return $\mu_i := \bot$.
    $\gamma \leftarrow \mathbb{Z}_p$
    Return $\mu_i := (\mu_{1.i}, \mu_{2.i}) = (tsk_i \cdot (u^\omega v^\tau d)^\gamma, g^\gamma)$.
$\mathsf{TShVer}(tpk, tvk, c, i, \mu_i)$ :
    $(p, \mathbb{G}, \mathbb{G}_T, \boldsymbol{e}, H, g, g_2, Z, u, v, d) \leftarrow tpk$;   $(tvk_i)_{i \in [n]} \leftarrow tvk$;   $(C_0, C_1, C_2, \tau) \leftarrow c$
    $\omega := H(C_0, C_1)$
    If $\boldsymbol{e}(C_1, u^\omega v^\tau d) \neq \boldsymbol{e}(g, C_2)$ then
        If $\mu_i = \bot$ then return $\top$ else return $\bot$.
    End if
    If $\boldsymbol{e}(g, \mu_{1.i}) = \boldsymbol{e}(tvk_i, g_2) \cdot \boldsymbol{e}(u^\omega v^\tau d, \mu_{2.i})$ then return $\top$ else return $\bot$.
$\mathsf{TCom}(tpk, tvk, c, \{\mu_{s_i}\}_{i \in [t]})$ where $\{s_i\}_{i \in [t]} \subset [n]$ :
    If $\exists i \in [t] : \mathsf{TShVer}(tpk, tvk, c, s_i, \mu_{s_i}) = \bot$ or $\mu_{s_i} = \bot$ then return $\bot$.
    $(C_0, C_1, C_2, \tau) \leftarrow c$
    $\forall i \in [t] : (\mu_{1.s_i}, \mu_{2.s_i}) \leftarrow \mu_{s_i}$
    Return $m = C_0 \cdot \boldsymbol{e}(C_2, \prod_{i=1}^t \mu_{2.s_i}^{\lambda_i}) / \boldsymbol{e}(C_1, \prod_{i=1}^t \mu_{1.s_i}^{\lambda_i})$ using
                    Lagrange coefficients $\{\lambda_i\}_{i \in [t]}$ satisfying $f(0) = \sum_{i=1}^t \lambda_i f(s_i)$.
```

Figure 3.2: The re-splittable TPKE scheme eLDLK.

(b) $\mathcal{B}$ computes $\omega = H(C_0, C_1)$ and tests $\boldsymbol{e}(C_1, u^\omega v^r d) \overset{?}{=} \boldsymbol{e}(g, C_2)$ to check the validity of the ciphertext. If this validity check fails, $\mathcal{B}$ returns $\bot$ to $\mathcal{A}$.

(c) Let $I = \{s_\ell\}_{\ell \in [t-1]}$ be the set of indices corresponding to tvk. $\mathcal{B}$ proceeds as follows:

i. If $j \in I$, $\mathcal{B}$ chooses $\gamma \leftarrow \mathbb{Z}_p$. Then, $\mathcal{B}$ sets $\mu_j = (\mu_{1.j}, \mu_{2.j}) := (C_1^{\gamma\alpha_j}, g^{\gamma})$ and returns μ_j to $\mathcal{A}$.

ii. If $\omega x_1 + \tau x_3 + x_5 \mod p = 0$, then $\mathcal{B}$ outputs a random bit and aborts.

iii. If $\omega = H(C_0^*, C_1^*)$ and $(C_0, C_1) \neq (C_0^*, C_1^*)$, then $\mathcal{B}$ outputs a random bit and aborts.

iv. (Note that at this point, it is guaranteed that $j \notin I$.) $\mathcal{B}$ choose $\gamma \leftarrow \mathbb{Z}_p$. Then, $\mathcal{B}$ calculates $\mu_j = (\mu_{1.j}, \mu_{2.j}) := \big((\frac{C_2}{C_1^{\omega x_2 + \tau x_4 + x_6}})^{\frac{\lambda_0}{\omega x_1 + \tau x_3 + x_5}} \cdot \chi, g^{\gamma} \big)$, where $\chi = C_1^{\Sigma_{\ell=1}^{t-1} \lambda_\ell \alpha_{s_\ell}}$ and $\lambda_0, \cdots, \lambda_{t-1} \in \mathbb{Z}_p$ are the Lagrange coefficients satisfying $f(j) = \lambda_0 f(0) + \Sigma_{\ell=1}^{t-1} \lambda_\ell f(s_\ell)$ for the polynomial f. Then, $\mathcal{B}$ returns μ_j to $\mathcal{A}$.

Finally, when $\mathcal{A}$ terminates with its guess bit $\theta' \in \{0,1\}$, $\mathcal{B}$ outputs 1 if $\theta = \theta'$, otherwise outputs 0 and terminates.

The above completes the description of $\mathcal{B}$. For the challenge ciphertext, the adversary could obtain the information that $\omega x_1 + \tau x_3 + x_5 \mod p = 0$. However, there are exactly p possible (x_1, x_3, x_5) pairs that satisfy $\omega x_1 + \tau x_3 + x_5 \mod p = 0$. In addition, each of them is equally likely. Thus, information-theoretically, the probability that $\mathcal{B}$ aborts is at most $1/p$ for each query. Therefore, taking the union bound over all $\mathcal{A}$'s share decryption queries, the probability that $\mathcal{B}$ aborts is at most q/p, where q is the number of share decryption queries, which is negligible. Because of the collision resistance of the hash function H and the above discussion, the probability that $\mathcal{B}$ aborts during the simulation is negligible. Except for that, $\mathcal{B}$ perfectly simulates the CCA game for $\mathcal{A}$ if $W = \boldsymbol{e}(g,g)^{abc}$. When W is a random element, the view of $\mathcal{B}$ is independent of $\mathcal{A}$'s challenge bit θ. Therefore, $\mathcal{B}$'s advantage in solving the DBDH problem can be estimated as $\mathsf{Adv}_{\mathcal{B}}^{\mathrm{DBDH}} \geq \mathsf{Adv}_{(\mathcal{A},n,t)}^{\texttt{CCA-RS-TPKE}}(k) - (negl.)$. This completes the proof of Theorem 3. □

Theorem 4. *The re-splittable TPKE scheme* eLDLK *has decryption consistency unconditionally.*

Proof of Theorem 4. Let $\mathcal{A}$ be an arbitrary adversary attacking the TPKE scheme eLDLK in the decryption consistency game. Suppose $\mathcal{A}$ outputs $c = (C_0, C_1, C_2, \tau)$, $tvk = (g^{f(i)})_{i \in [n]}$, $S = \{\mu_{s_i}\}_{i \in [t]} = \{(\mu_{1.s_i}, \mu_{2.s_i})\}_{i \in [t]}$, and $S' = \{\mu'_{s'_i}\}_{i \in [t]} = \{(\mu'_{1.s'_i}, \mu'_{2.s'_i})\}_{i \in [t]}$, satisfying the winning condition for the decryption consistency game, where f is the polynomial that was generated for computing tvk. (Note that tvk must be one of the verification keys returned to $\mathcal{A}$ as an answer to one of $\mathcal{A}$'s split&corruption queries.) This means that c must be valid, i.e. it satisfies

$$\boldsymbol{e}(C_1, u^{\omega} v^{\tau} d) = \boldsymbol{e}(g, C_2), \tag{3.1}$$

where $\omega = H(C_1, C_2)$. (Otherwise, both $\mathsf{TCom}(tpk, tvk, c, S) = \mathsf{TCom}(tpk, tvk, c, S') = \bot$ and hence the results cannot be different.) Furthermore, the shares $\mu_{s_i} = (\mu_{1.s_i}, \mu_{2.s_i})$ in S are valid, and hence we have $\boldsymbol{e}(g, \mu_{1.s_i}) = \boldsymbol{e}(g^{f(s_i)}, g_2) \cdot \boldsymbol{e}(u^{\omega} v^{\tau} d, \mu_{2.s_i})$, which implies

$$\mu_{1.s_i} = g^{f(i)} \cdot (u^{\omega} v^{\tau} d)^{\log \mu_{2.s_i}}.$$

This in turn implies

$$\begin{aligned}\boldsymbol{e}(C_1, \prod_{i=1}^{t} \mu_{1.s_i}^{\lambda_i}) &= \boldsymbol{e}(C_1, g_2^{\Sigma_{i=1}^{t} f(i)\lambda_i} \cdot (u^\omega v^\tau d)^{\Sigma_{i=1}^{t}\lambda_i \cdot \log \mu_{2.s_i}}) \\ &= \boldsymbol{e}(C_1, g_2^x) \cdot \boldsymbol{e}(C_1, u^\omega v^\tau d)^{\Sigma_{i=1}^{t}\lambda_i \cdot \log \mu_{2.s_i}} \\ &\overset{(*)}{=} \boldsymbol{e}(C_1, g_2^x) \cdot \boldsymbol{e}(g, C_2)^{\Sigma_{i=1}^{t}\lambda_i \cdot \log \mu_{2.s_i}} \\ &= \boldsymbol{e}(C_1, g_2^x) \cdot \boldsymbol{e}(\prod_{i=1}^{t} \mu_{2.s_i}^{\lambda_i}, C_2),\end{aligned}$$

where $\{\lambda_i\}_{i\in[t]}$ are the Lagrange coefficients satisfying $f(0) = x = \Sigma_{i=1}^{t}\lambda_i f(s_i)$, and in the equation (*) we used Eq. (3.1). Therefore, we have

$$\begin{aligned}\mathsf{TCom}(tpk, tvk, c, S) &= C_0 \cdot \boldsymbol{e}(C_2, \prod_{i=1}^{t} \mu_{2.s_i}^{\lambda_i}) / \boldsymbol{e}(C_1, \prod_{t=1}^{t} \mu_{1.s_i}^{\lambda_i}) \\ &= C_0 / \boldsymbol{e}(C_1, g_2^x)\end{aligned}$$

Similarly, the shares $\mu'_{s'_i} = (\mu'_{1.s'_i}, \mu'_{2.s'_i})$ in S' are also valid, from which we obtain $\boldsymbol{e}(g, {\mu'}_{1.s'_i}) = \boldsymbol{e}(g^{f(s'_i)}, g_2) \cdot \boldsymbol{e}(u^\omega v^\tau d, {\mu'}_{2.s'_i})$. From this result, with a similar calculation to the above, we have

$$e(C_1, \prod_{i=1}^{t} {\mu'}_{1.s'_i}^{\lambda'_i}) = \boldsymbol{e}(C_1, g_2^x) \cdot \boldsymbol{e}(\prod_{i=1}^{t} {\mu'}_{2.s'_i}^{\lambda'_i}, C_2),$$

where $\{\lambda'_i\}_{i\in[t]}$ are the Lagrange coefficients satisfying $f(0) = x = \Sigma_{i=1}^{t}\lambda'_i f(s'_i)$. Hence, we have

$$\begin{aligned}\mathsf{TCom}(tpk, tvk, c, S') &= C_0 \cdot \boldsymbol{e}(C_2, \prod_{i=1}^{t} {\mu'}_{2.s'_i}^{\lambda'_i}) / \boldsymbol{e}(C_1, \prod_{t=1}^{t} {\mu'}_{1.s'_i}^{\lambda'_i}) \\ &= C_0 / \boldsymbol{e}(C_1, g_2^x).\end{aligned}$$

That is, the combined values from the sets S and S' are identical, and we rearch a contradiction. Therefore, there cannot be two distinct sets of valid shares whose combined values disagree. Thus, we have $\mathsf{Adv}_{(\mathcal{A},n,t)}^{\mathtt{DC\text{-}RS\text{-}TPKE}}(k) = 0$ for any (even computationally unbounded) adversary $\mathcal{A}$. □

3.3.3 Construction Based on Gan et al's scheme

Here, we show the construction of a re-splittable TPKE scheme based on the Gan et al. scheme [47] as in Fig.3.3. Here, we call it $\mathsf{eGWW+}$. As with the eLDLK scheme, this scheme also does not need a signature scheme in the construction. Though the assumption on which the security is based is different from eLDLK scheme, all of the algorithms of this scheme is more efficient than it.

Theorem 5. *If the HDH assumption on $(\mathbb{G}, \mathbb{G}_T, H_2)$ holds and H_1 is a collision resistant hash function, then the re-splittable TPKE scheme* $\mathsf{eGWW+}$ *is CCA secure.*

```
$\mathsf{TKG}(1^k, n, t)$ :
    $(p, \mathbb{G}, \mathbb{G}_T, H_2, \boldsymbol{e}) \leftarrow \mathsf{BG}_{\mathrm{HDH}}(1^k)$, where $H_2 : \mathbb{G} \to \{0,1\}^{|m|}$
    Select a collision resistant hash function $H_1 : \{0,1\}^* \to \mathbb{Z}_p$.
    $g, u, v, d \leftarrow \mathbb{G}$;   $x \leftarrow \mathbb{Z}_p$;   $g_1 := g^x$
    $tpk := (p, \mathbb{G}, \mathbb{G}_T, \boldsymbol{e}, H_1, H_2, g, g_1, u, v, d)$;   $tsk := x$;
    Return $(tpk, tsk)$.
$\mathsf{TEnc}(tpk, m)$ :
    $(p, \mathbb{G}, \mathbb{G}_T, \boldsymbol{e}, H_1, H_2, g, g_1, u, v, d) \leftarrow tpk$;   $r, \tau \leftarrow \mathbb{Z}_p$
    $C_0 := m \oplus H_2(g_1^r)$;   $C_1 := g^r$;   $\omega := H_1(C_0, C_1)$;   $C_2 := (u^\omega v^\tau d)^r$
    Return $c := (C_0, C_1, C_2, \tau)$.
$\mathsf{TSplit}(tsk)$ :
    $x \leftarrow tsk$
    $f \leftarrow \mathbb{Z}_p[X]$ satisfying $\deg(f) = t-1$ and $f(0) = x$.
    $\forall i \in [n] : tsk_i := f(i)$
    $\forall i \in [n] : tvk_i := g^{f(i)}$
    $tvk := (tvk_i)_{i \in [n]}$
    Return $(\{tsk_i\}_{i \in [n]}, tvk)$.
$\mathsf{TShDec}(tpk, tsk_i, c)$ :
    $(p, \mathbb{G}, \mathbb{G}_T, \boldsymbol{e}, H_1, H_2, g, g_1, u, v, d) \leftarrow tpk$;   $(C_0, C_1, C_2, \tau) \leftarrow c$
    $\omega := H_1(C_0, C_1)$
    If $\boldsymbol{e}(C_1, u^\omega v^\tau d) \neq \boldsymbol{e}(g, C_2)$ then return $\perp$.
    Return $\mu_i := C_1^{tsk_i}$.
$\mathsf{TShVer}(tpk, tvk, c, i, \mu_i)$ :
    $(p, \mathbb{G}, \mathbb{G}_T, \boldsymbol{e}, H_1, H_2, g, g_1, u, v, d) \leftarrow tpk$;   $(tvk_i)_{i \in [n]} \leftarrow tvk$
    $(C_0, C_1, C_2, \tau) \leftarrow c$;   $\omega := H_1(C_0, C_1)$
    If $\boldsymbol{e}(C_1, u^\omega v^\tau d) \neq \boldsymbol{e}(g, C_2)$ then
        If $\mu_i = \perp$ then return $\top$ else return $\perp$.
    End if
    If $\boldsymbol{e}(\mu_i, g) = \boldsymbol{e}(C_1, tvk_i)$ then return $\top$ else return $\perp$.
$\mathsf{TCom}(tpk, tvk, c, \{\mu_{s_i}\}_{i \in [t]})$ where $\{s_i\}_{i \in [t]} \subset [n]$ :
    If $\exists i \in [t] : \mathsf{TShVer}(tpk, tvk, c, s_i, \mu_{s_i}) = \perp$ or $\mu_{s_i} = \perp$ then return $\perp$.
    $(C_0, C_1, C_2, \tau) \leftarrow c$
    Return $m = C_0 \oplus H_2(\prod_{i=1}^t \mu_{s_i}^{\lambda_i})$ using
            Lagrange coefficients $\{\lambda_i\}_{i \in [t]}$ satisfying $f(0) = \sum_{i=1}^t \lambda_i f(s_i)$.
```

Figure 3.3: The re-splittable TPKE scheme eGWW+.

Proof of Theorem 5. Let $\mathcal{A}$ be an arbitrary adversary that attacks the re-splittable TPKE scheme eGWW+ in the game of CCA security. Using the adversary $\mathcal{A}$, we build an algorithm $\mathcal{B}$ that solves the HDH problem on $(\mathbb{G}, \mathbb{G}_T, H_2)$. Given $(1^k, p, \mathbb{G}, \mathbb{G}_T, \boldsymbol{e}, g, g^a, g^b, W)$ as input, algorithm $\mathcal{B}$ proceeds as follows. (The aim of $\mathcal{B}$ is to distinguish whether $W = H_2(g^{ab})$ or not.)

1. **Setup:** Algorithm $\mathcal{B}$ does the following.

(a) $\mathcal{B}$ selects a collision resistant hash function H_1 and chooses $x_1, x_2, x_3, x_4, x_5, x_6 \leftarrow \mathbb{Z}_p$.

(b) $\mathcal{B}$ sets $g_1 := g^a, u := g_1^{x_1} g^{x_2}, v := g_1^{x_3} g^{x_4}, d := g_1^{x_5} g^{x_6}$, and sets $tpk := (p, \mathbb{G}, \mathbb{G}_T, \boldsymbol{e}, H_1, H_2, g, g_2, Z = \boldsymbol{e}(g_1, g_2), u, v, d)$.

(c) $\mathcal{B}$ gives tpk to $\mathcal{A}$.

2. **Split&Corruption query:** When $\mathcal{A}$ issues a split&corruption query $I = \{s_i\}_{i \in [t-1]}$, $\mathcal{B}$ does the following.

   (a) $\mathcal{B}$ picks $t-1$ random integers $\alpha_{s_1} \cdots, \alpha_{s_{t-1}} \leftarrow \mathbb{Z}_p$. We let $f \in \mathbb{Z}_p[X]$ be a polynomial of degree $t-1$ satisfying by $f(0) := a$, $f(s_i) := \alpha_{s_i}$ for all $i \in [t-1]$. (Note that $\mathcal{B}$ does not know f entirely.) $\mathcal{B}$ sets $\{tsk_{s_i}\} := \{\alpha_{s_i}\}$ for all $i \in [t-1]$.

   (b) i. For $j \in I$, $\mathcal{B}$ sets $v_j := g^{\alpha_j}$.

   ii. For $j \notin I$, $\mathcal{B}$ computes $v_j = g_1^{\lambda_0}(g^{\alpha_{s_1}})^{\lambda_1} \cdots (g^{\alpha_{s_{t-1}}})^{\lambda_{t-1}}$, where $\{\lambda_0, \cdots, \lambda_{t-1}\}$ are the Lagrange coefficients satisfying $f(j) = \lambda_0 f(0) + \Sigma_{\ell=1}^{t-1} \lambda_\ell f(s_\ell)$ for the polynomial f. Note that v_j satisfies $v_j = g_1^{f_1(j)}$.

   iii. $\mathcal{B}$ sets $tvk := (v_i)_{i \in [n]}$.

   (c) $\mathcal{B}$ returns $(\{tsk_{s_i}\}_{i \in [t-1]}, tvk)$ to $\mathcal{A}$.

3. **Challenge query:** Adversary $\mathcal{A}$ outputs two equal-length messages m_0 and m_1. First, $\mathcal{B}$ flips a fair coin $\theta \in \{0,1\}$, and sets $(C_0^*, C_1^*) := (m_\theta \oplus W, g^b)$. Then, $\mathcal{B}$ computes $\omega^* = H_1(C_0^*, C_1^*)$ and $\tau^* = \frac{x_1 \omega^* + x_5}{x_3} \mod p$. $\mathcal{B}$ sets $C_2^* := (g^b)^{\omega x_2 + \tau^* x_4 + x_6}$. $\mathcal{B}$ returns the challenge ciphertext $c^* := (C_0^*, C_1^*, C_2^*, \tau^*)$ to $\mathcal{A}$.

4. **Share decryption query:** When $\mathcal{A}$ issues a share decryption query (tvk, i, c), $\mathcal{B}$ does the following, where $c = (C_0, C_1, C_2, \tau)$.

   (a) If $c = c^*$, then $\mathcal{B}$ returns $\mu_i := \perp$ to $\mathcal{A}$.

   (b) If $\mathcal{B}$ computes $\omega = H_1(C_0, C_1)$ and tests $\boldsymbol{e}(C_1, u^\omega v^r d) \overset{?}{=} \boldsymbol{e}(g, C_2)$. to check the validity of the ciphertext. If this validity check fails, $\mathcal{B}$ returns $\perp$ to $\mathcal{A}$.

   (c) Let $I = \{s_\ell\}_{\ell \in [t-1]}$ be the set of indices corresponding to tvk. $\mathcal{B}$ proceeds as follows:

   i. If $j \in I$, $\mathcal{B}$ chooses $\gamma \leftarrow \mathbb{Z}_p$. Then, $\mathcal{B}$ sets $\mu_j := C_1^{\gamma \alpha_j}$ and returns μ_j to $\mathcal{A}$.

   ii. If $\omega x_1 + \tau x_3 + x_5 \mod p = 0$, then $\mathcal{B}$ outputs a random bit and aborts.

   iii. If $\omega = H_1(C_0^*, C_1^*)$ and $(C_0, C_1) \neq (C_0^*, C_1^*)$, then $\mathcal{B}$ outputs random bit and aborts.

   iv. (Note that at this point, it is guaranteed that $j \notin I$.) $\mathcal{B}$ choose $\gamma \leftarrow \mathbb{Z}_p$. Then, $\mathcal{B}$ calculates $\mu_j := \big((\frac{C_2}{C_1^{\omega x_2 + \tau x_4 + x_6}})^{\frac{\lambda_0}{\omega x_1 + \tau x_3 + x_5}} \cdot \chi \big)$, where $\chi = (C_1)^{\Sigma_{\ell=1}^{t-1} \lambda_\ell \alpha_{s_\ell}}$ and $\lambda_0, \cdots, \lambda_{t-1} \in \mathbb{Z}_p$ are the Lagrange coefficients satisfying $f(j) = \lambda_0 f(0) + \Sigma_{\ell=1}^{t-1} \lambda_\ell f(s_\ell)$ for the polynomial f. Then, $\mathcal{B}$ returns μ_j to $\mathcal{A}$.

Finally, when $\mathcal{A}$ terminates with its guess bit $\theta' \in \{0,1\}$, $\mathcal{B}$ outputs 1 if $\theta = \theta'$, otherwise outputs 0 and terminates.

The above completes the description of $\mathcal{B}$. Because of the collision resistance of the hash function H_1 and a similar argument in [76], the probability that $\mathcal{B}$ aborts during the simulation is negligible. Except for that, $\mathcal{B}$ perfectly simulates the CCA game for $\mathcal{A}$ if $W = H_2(g^{ab})$. When W is a random string, the view of $\mathcal{B}$ is independent of the choice b. Therefore, $\mathcal{B}$'s advantage in solving HDH problem can be estimated as $\mathsf{Adv}^{\mathrm{HDH}}_{\mathcal{B}} \geq \mathsf{Adv}^{\texttt{CCA-RS-TPKE}}_{(\mathcal{A},n,t)}(k) - (negl.)$. This completes the proof of Theorem 5. □

Theorem 6. *The re-splittable TPKE scheme* eGWW+ *has decryption consistency unconditionally.*

Proof of Theorem 6. Let $\mathcal{A}$ be an arbitrary adversary attacking the re-splittable TPKE scheme eGWW+ in the decryption consistency game. Suppose $\mathcal{A}$ outputs $c = (C_0, C_1, C_2, \tau)$, $tvk = (f(i))_{i\in[n]}$, $S = \{\mu_{s_i}\}_{i\in[t]}$, and $S' = \{\mu'_{s'_i}\}_{i\in[t]}$, where f is the polynomial that was generated when computing tvk.

If the shares μ_{s_i} in S are valid, they must satisfy $\boldsymbol{e}(\mu_{s_i}, g) = \boldsymbol{e}(C_1, g^{f(s_i)})$, which implies $\mu_{s_i} = C_1^{f(s_i)}$. This in turn implies $\prod_{i=1}^{t} \mu_{s_i}^{\lambda_i} = C_1^x$, where $\{\lambda_i\}_{i\in[t]}$ are the Lagrange coefficients satisfying $f(0) = x = \Sigma_{i=1}^{t}\lambda_i f(s_i)$. Therefore, it holds that

$$\mathsf{TCom}(tpk, tvk, c, S) = C_0 \oplus H(\prod_{i=1}^{t} \mu_{s_i}^{\lambda_i}) = C_0 \oplus H(C_1^x).$$

Similarly, if the shares $\mu'_{s'_i}$ in S' are valid, we have $\boldsymbol{e}(\mu'_{s'_i}, g) = \boldsymbol{e}(C_1, g^{f(s'_i)})$, which implies $\mu'_{s'_i} = C_1^{f(s'_i)}$. This in turn implies $\prod_{i=1}^{t} {\mu'}_{s'_i}^{\lambda'_i} = C_1^x$, where $\{\lambda'_i\}_{i\in[t]}$ are the Lagrange coefficients satisfying $f(0) = x = \Sigma_{i=1}^{t}\lambda'_i f(s'_i)$. Therefore, we also have

$$\mathsf{TCom}(tpk, tvk, c, S') = C_0 \oplus H(\prod_{i=1}^{t} {\mu'}_{s'_i}^{\lambda'_i}) = C_0 \oplus H(C_1^x).$$

We have seen that the combined values from the sets S and S' cannot be distinct. Thus, we have $\mathsf{Adv}^{\texttt{DC-RS-TPKE}}_{(\mathcal{A},n,t)}(k) = 0$ for any (even computationally unbounded) adversary $\mathcal{A}$. □

Extension of eGWW+ Scheme. In this section, we propose an extension of the eGWW+ scheme. Here, we call this extended scheme eGWW++. This eGWW++ scheme is secure under the computational Diffie-Hellman (CDH) assumption. We can obtain this scheme by replacing the hash function H_2 which is used in the eGWW+ scheme to a hardcore function for the CDH problem (in bilinear groups). The existence of such a hardcore function was shown by Boneh and Shparlinski [25]. This is the first scheme that is secure under the CDH assumption. Although the message space of this scheme is limited to $\{0,1\}$, we can extend this message space to $O(\log_2 k)$ bits by applying the general hardcore function by Goldreich and Levin [53].

3.4 Comparisons

The Tables 3.1 and 3.2 are comparisons of the previous re-splittable TPKE scheme [58] and our proposed schemes. Here, e means the number of pairing computation, and E

Table 3.1: Efficiency comparisons of computational cost (pairing computation and exponentiation)

	TKG	TEnc	TSplit	TShDec	TShVer	TCom
Hanaoka et al. [58]	$3E$	$1e + 7E$	nE	$2e + 5E$	$4e + 4E$	$(4t + 1)e + (5t)E$
eAT	$4E$	$10E$	$2nE$	$4e + 6E$	$8e + 4E$	$(8t)e + (6t)E$
eLDLK	$1e + 1E$	$5E$	$2nE$	$2e + 4E$	$5e + 2E$	$(5t + 2)e + (4t)E$
eGWW+	$1E$	$5E$	nE	$2e + 3E$	$4e + 2E$	$(4t)e + (3t)E$
eGWW++	$1E$	$5E$	nE	$2e + 3E$	$4e + 2E$	$(4t)e + (3t)E$

means the number of exponentiation, ciphertext overhead means $|ciphertext| - |plaintext|$, and we ignore the computation costs for all other operations such as modular additions/multiplications and hash functions. For the re-splittable TPKE schemes that use a one-time signature scheme as a building block, we concretely implement the signature scheme by using Wee's scheme [109, Section 5.1], which, to the best of our knowledge, is the most efficient one-time signature scheme based on the discrete logarithm assumption.

As is clear from the tables, the eGWW+ scheme is the most efficient in almost all aspects. Therefore, we can obtain an efficient PRE scheme by using this re-splittable TPKE scheme as a building block in the construction of PRE schemes in [58] and [84].

Table 3.2: Efficiency comparisons of communication cost (Note that the message space of eGWW+$'$ is limited to $\{0, 1\}$.)

	tpk	tsk	$tskshare$	tvk	$ciphertext$	$dec.share$
Hanaoka et al. [58]	$4\|\mathbb{G}\|$	$1\|\mathbb{Z}_p\|$	$1\|\mathbb{Z}_p\|$	$n\|\mathbb{G}\|$	$5\|\mathbb{G}\| + 1\|\mathbb{G}_T\| + 2\|\mathbb{Z}_p\|$	$1\|\mathbb{G}\|$
eAT	$5\|\mathbb{G}\|$	$2\|\mathbb{Z}_p\|$	$2\|\mathbb{Z}_p\|$	$2n\|\mathbb{G}\|$	$7\|\mathbb{G}\| + 1\|\mathbb{G}_T\| + 2\|\mathbb{Z}_p\|$	$2\|\mathbb{G}\|$
eLDLK	$5\|\mathbb{G}\| + 1\|\mathbb{G}_T\|$	$1\|\mathbb{Z}_p\|$	$1\|\mathbb{G}\|$	$n\|\mathbb{G}\|$	$2\|\mathbb{G}\| + 1\|\mathbb{G}_T\| + 1\|\mathbb{Z}_p\|$	$2\|\mathbb{G}\|$
eGWW+	$5\|\mathbb{G}\|$	$1\|\mathbb{Z}_p\|$	$1\|\mathbb{Z}_p\|$	$n\|\mathbb{G}\|$	$2\|\mathbb{G}\| + 1\|m\| + 1\|\mathbb{Z}_p\|$	$1\|\mathbb{G}\|$
eGWW++	$5\|\mathbb{G}\|$	$1\|\mathbb{Z}_p\|$	$1\|\mathbb{Z}_p\|$	$n\|\mathbb{G}\|$	$2\|\mathbb{G}\| + 1\|\mathbb{Z}_p\|$	$1\|\mathbb{G}\|$

3.5 Conclusion

In this chapter, we proposed three new constructions of re-splittable TPKE schemes. It seems that the re-splittable TPKE is close to the standard TPKE and we can easily extend the standard TPKE schemes to the re-splittable ones. As we denoted in Section 3.1.1, however, we cannot extend the previous TPKE schemes based on multiple encryption to the re-splittable ones through the methodology used in this chapter. Moreover, the property that we can re-distribute decryption key shares is useful not only in the generic construction of proxy re-encryption but also in practical cases that we use TPKE in the real world. From the view point of security, we can regard as the conditions of attacks become harder because the adversary have to gather the t and more decryption (key) shares which are generated in the same timing. From the view point of practicality, we can handle the flexible change of the number of the entities or the threshold by changing the values of t and n. In our model, it seems that we have to decide these values in the TKG algorithm. However, we can avoid it and decide those values in the TSplit algorithm. We cannot see this property in the standard TPKE.

A proposal of new applications for re-splittable TPKE is a future work. Same as the secret sharing, we can realize the distributed key management by (re-splittable) TPKE. We are expecting that the proposal of theoretical and/or practical applications of re-splittable TPKE.

Chapter 4

Proxy Re-encryption with Re-encryption Verifiability

4.1 Introduction

4.1.1 Background and Motivation

Proxy re-encryption (PRE) which was formalized by Blaze et al. [19] is an interesting extension of traditional public key encryption (PKE) and has received much attentions in recent years. In PRE, in addition to the normal operations of PKE, with a dedicated re-encryption key (which is generated by receiver A), a semi-trusted third party called *proxy* can change the destination of ciphertexts destined for user A into those for user B. A noticeable property of PRE is that the proxy can carry out the transform of a ciphertext without decrypting it and is totally ignorant of the plaintext. There are many models as well as implementations [19, 6, 33, 77, 91, 37, 58, 65] of PRE. The type of PRE we focus on in this chapter is "single-hop" and "uni-directional", where a ciphertext[1] can be transformed only once, and a re-encryption key used to transform a ciphertext for user A to that for user B cannot be used for the transform of the opposite direction.

In ordinary PRE schemes, a proxy is considered as a semi-trusted party, and is typically assumed to perform the re-encryption process honestly. This means that we have to put relatively high level of trust on proxies, and it may be undesirable for some applications of PRE, e.g. cloud-based file sharing systems. In this chapter, we study a mechanism that enables us to reduce the level of trust we have to put on proxies in PRE systems.

To motivate it further, consider a cloud storage service, one of the major applications of PRE, in which users store a (possibly large) encrypted data c. PRE allows an easy way to share the encrypted data in the cloud with another user: if an owner (say user A) of the encrypted data c wants to share it with user B, it can simply give a re-encryption key $rk_{A \rightarrow B}$ to the cloud manager, and can go off-line.; When later B requests the data for the cloud manager, the manager can transform c into a re-encrypted ciphertext $\widehat{c}$ that can be decrypted by user B. *However, in this situation, can user B be sure if $\widehat{c}$ is actually a re-encryption of c? Can B detect whether the cloud manager (proxy) has misbehaved?*

[1]In the context of single-hop uni-directional PRE, an original ciphertext (which can be re-encrypted) and a re-encrypted ciphertext (which cannot be re-encrypted further) are typically called a *second-level* ciphertext and a *first-level* ciphertext, respectively [77, 58], and we will also use the names.

Note that an ordinary PRE scheme is not required to support the functionality to check the relation between an original ciphertext c and a re-encrypted ciphertext $\widehat{c}$ (if user A reveals its secret key to user B, then B can check the relation, but it is clearly undesirable). It is therefore desirable if there is a PRE scheme in which the relation between original and re-encrypted ciphertexts can be checked efficiently by a recipient of a re-encrypted ciphertext (user B in this example), without the help of the other entities.

4.1.2 Our Contribution

In this chapter, we introduce a new functionality for PRE that we call *re-encryption verifiability.* In a PRE scheme with re-encryption verifiability (which we simply call verifiable PRE, or VPRE), a receiver of a re-encrypted ciphertext can verify whether the received ciphertext is correctly transformed from an original ciphertext by a proxy, and thus can detect an illegal activity of the proxy. We may even expect that the existence of re-encryption verifiability suppresses proxy's illegal activities, and this functionality enables us to relax the level of trust that we have to put on proxies. We achieve re-encryption verifiability by introducing a new algorithm that we call the *re-encryption verification algorithm*, into the syntax of (single-hop, uni-directional) PRE. This algorithm takes two ciphertexts c and $\widehat{c}$, a secret key sk_B (of the receiver B) and a public key pk_A (of another user A) as input, and can tell whether $\widehat{c}$ is transformed from c using a re-encryption key that transforms a ciphertext from user A to user B. We stress that this algorithm needs not only a re-encrypted ciphertext $\widehat{c}$ but also a (candidate) original ciphertext c (while to normally decrypt a re-encrypted ciphertext, original ciphertext c is not required). Note that such a situation is natural in the applications of PRE which we explained earlier.

We formalize the security model for a VPRE scheme. In particular, in order for the re-encryption verification algorithm to be meaningful, in addition to ordinary chosen ciphertext (CCA) security (for both original/transformed ciphertexts), we introduce a new security notion that we call *soundness.* Our security model for CCA security is based on the one used by Hanaoka et al. [58], and is extended to take the existence of the re-encryption verification algorithm into account. For "backward compatibility" with the model of ordinary PRE (without re-encryption verification algorithm), we show that a VPRE scheme secure in our model is in general secure as a PRE scheme in the model of [58]. Then, we show that the PRE scheme by Hanaoka et al. [58] (which we call "HKK^+") can be extended to a VPRE scheme (which we call "eHKK^+"), by augmenting the HKK^+ scheme with the dedicated re-encryption verification algorithm. To prove the security of the VPRE scheme eHKK^+, we need the property that we call *strong smoothness* (which is essentially the same notion as that introduced in [46] with the name γ*-uniformity*) for the underlying TPKE scheme. This property is unconditionally satisfied by natural TPKE schemes, and thus is not a strong assumption at all. For more details, see Section 4.3.

Naive Approaches and Their Problems. Although one might think that the problem of checking dishonest behaviors of a proxy can be resolved by using a *signature* scheme in a PRE scheme (that is, by considering a proxy re-"signcryption" scheme), we argue that this approach does *not* work. Specifically, consider a situation where a sender holds a key pair of a signature scheme, and consider the typical "Sign-then-Encrypt"-style construction of a proxy re-signcryption scheme, i.e. the construction where a ciphertext is generated by first signing the plaintext, and then the plaintext together with the signature are encrypted by the PRE scheme. Note that what is verified in such a proxy re-signcryption scheme (by a recipient of a re-encrypted ciphertext) is that the original plaintext has not been modified and that it is indeed generated by the sender, but *not* that the transformed ciphertext resulted from re-encryption performed by the proxy. For example, such a construction is vulnerable to the following attack: a sender generates several ciphertexts to the proxy, then the proxy re-encrypts one of them, and sends it to the recipient. The recipient may find that the plaintext recovered from the received ciphertext indeed comes from the sender, but he will not be sure which of the ciphertexts the proxy owns was re-encrypted (and even that whether the received ciphertext is a re-encryption of one of the ciphertexts). In the "Encrypt-then-Sign"-style construction, i.e. the construction where the sender first encrypts a plaintext and then generates a signature on the ciphertext, the situation is worse, because the signature attached to the original ciphertext will not be a valid signature for a re-encrypted ciphertext. Furthermore, these proxy re-signcryption-style approaches also have a potential drawback that the receiver needs to be aware of the sender who generates the original ciphertext, which is not necessary in our VPRE model (and in an ordinary PRE scheme), and may be a barrier for some applications of (V)PRE. In summary, we emphasize that what is achieved by proxy re-signcryption-style approaches and what we achieve in this research (i.e. verifiability of a dishonest behavior of a proxy) are two very different properties, and one approach cannot be a solution to the other.

On the Choice of Security Models on which Our Security Definitions Are Based. We note that, as mentioned above, our definitions for VPRE are based on those of PRE adopted by Hanaoka et al. [58]. Their security definitions (of chosen ciphertext security) are known to be one of the strongest in the literature of PRE. Notably, besides capturing the chosen ciphertext security (not re-playable variant [34]), the security models in [58] do not assume the so-called *knowledge-of-secret-key* (KOSK) assumption [20], in which an adversary can use any public key for corrupted users, without revealing the corresponding secret key. The KOSK assumption typically appears in security models of cryptographic primitives in which multiple users are inherently involved (e.g. multi-receiver PKE [8, 87], multi-signature [20, 15, 89]). The KOSK assumption does not reflect the reality quite well, and there are several critiques on this assumption (e.g. in the discussions in [15, 89, 87]). To the best of our knowledge, the Hanaoka et al. model is the only security definitions for PRE that do not assume the KOSK assumption, and thus we base our security definitions on theirs.

As far as we are aware, most popular PRE schemes without random oracles are secure only under the KOSK assumption (e.g. [77, 65]). [2] Therefore, we do not think these schemes can be shown to achieve re-encryption verifiability in our model. However, we do not rule out the possibility that these existing PRE schemes can be extended to VPRE

[2]To be more precise, in the security models adopted in these papers, public keys (of even a corrupted user) that can be used in the security games (say, in a re-encryption key generation and/or re-encryption queries) are generated by the challenger, who always generates these keys honestly. Therefore, the KOSK assumption is automatically assumed in these security models.

schemes that can be shown to be secure in the security models that are appropriately extended from the security models in which the original PRE schemes are proved secure. Especially, the pairing-based schemes (e.g. [77, 65]) seem to allow strong validity checking properties between a re-encrypted ciphertext and an original ciphertext, and we think they are good candidates of VPRE schemes. We would like to leave it as our future work whether these existing PRE schemes can be extended to VPRE schemes and can be proven secure in security models appropriately extended from the original security models.

Theoretical Aspect of Re-encryption Verifiability. There are two types of ciphertexts in single-hop unidirectional PRE (i.e. first-level ciphertexts and second-level ciphertexts), and consequently we have to consider "confidentiality" for each type of ciphertexts. Since PRE is an extension of ordinary PKE, it is natural to consider (a suitably extended version of) chosen ciphertext security for PRE. However, it is known that CCA security implies non-malleability of ciphertexts [10, 44] (in the case of ordinary PKE), while the "re-encryption" functionality directly contradicts "non-malleability" because it allows a "meaningful" modification of a ciphertext, and therefore, even the definition of CCA security for second-level ciphertexts of PRE has been non-trivial.

The difficulty is reflected in the security definition of second-level ciphertexts. Specifically, in the security game for defining second-level CCA security, an adversary is allowed to make decryption queries for both types of ciphertexts, and re-encryption and re-encryption key generation queries. To capture as powerful adversaries as possible and yet avoid unachievable security definitions, we allow arbitrary queries for an adversary, as long as the queries do not allow the adversary to trivially win, such as that it should not submit the challenge ciphertext itself as a decryption query for second-level ciphertexts. Here, note that an adversary can also trivially win if there is no restriction in decryption queries, because an adversary may transform the challenge second-level ciphertext into a first-level ciphertext (via a re-encryption query or using a re-encryption key obtained by making the re-encryption key generation query). However, in ordinary PRE, it is not clear what ciphertexts should be considered as a "re-encrypted ciphertext" of the challenge ciphertext.

The previous definitions [77, 58] resolve the issues by (partly) adopting the "replayable"-CCA (RCCA) security security [34], which is a relaxed notion of CCA security. In the RCCA game, if an adversary submits a ciphertext whose decryption result is one of the plaintexts that the adversary has used as its challenge, then as the answer to the decryption query the adversary receives not the decryption result, but some special symbol that tells the adversary that the decryption result was one of the challenge plaintexts (but does not tell which). The security definitions by Libert and Vergnaud [77] consider RCCA security for both types of ciphertexts. The security definitions by Hanaoka et al. [58] strengthened the definitions of [77], but still have a RCCA-security-like treatment in the security game. Roughly speaking, what the definition of CCA security for second-level ciphertexts in [58] achieves is CCA security against all entities but a proxy (who has a re-encryption key that can convert ciphertexts for the challenge users to others) and RCCA security against the proxy.

However, we naturally succeed in removing the RCCA-security-like treatment from the definition of CCA security for second-level ciphertexts. Specifically, in our security definition, thanks to the functionality of re-encryption verification, we can unambiguously define what should be considered as a re-encryption of the challenge ciphertext (and hence the challenger does not answer to it by simply returning some symbol (e.g. $\perp$) that tells an adversary that it has queried an unallowable ciphertext). For more details, see the security definitoins in Section 4.2.2.

4.1.3 Related Work

We briefly review the related work. Mambo and Okamoto introduced the concept of proxy decryption [81]. Later, Ivan and Dodis [66] proposed a generic construction of proxy cryptography based on sequential multiple encryption. Blaze, Bleumer, and Strauss formulated the concept of PRE cryptosystems [19] and proposed the first bidirectional PRE scheme based on the ElGamal PKE scheme. Subsequently, Ateniese et al. [6], Canetti and Hohenberger [33], Libert and Vergnaud [77], and Chow et al. [37], proposed different PRE schemes with various properties. Shao and Cao [91] proposed a PRE scheme without pairings. Later, however, Zhang et al. [93] pointed out that it is not secure in the Libert-Vergnaud security model [77]; that is, it does not provide master key security. Subsequently, Matsuda et al. proposed a PRE scheme without pairings [82], but later, Weng, Zhao, and Hanaoka [92] pointed out that their scheme is not chosen-ciphertext secure. Hanaoka et al. [58] proposed a new definition of CCA security in PRE and showed a generic construction of uni-directional PRE. Isshiki et al. [65] proposed a CCA secure PRE scheme.[3] Kirshanova [74] proposed a lattice-based PRE scheme. To the best of our knowledge, none of the previous works considered the re-encryption verifiability.

4.1.4 Chapter Organization

The remainder of this chapter is organized as follows. In Section 4.2, we introduce the model and security definitions for VPRE and discuss the relations with the security definitions (for ordinary PRE) by Hanaoka et al. [58]. In Section 4.3, we present our VPRE scheme. In Section 4.4, we prove the security of our VPRE scheme. Section 4.5 is the conclusion of this chapter.

4.2 Verifiable Proxy Re-Encryption

In this section, we present the model and the security definitions of VPRE. Note that we only focus on a single-hop uni-directional scheme.

This section is organized as follows: In Section 4.2.1, we define the syntax of a VPRE scheme. In Section 4.2.2, based on the definitions given in [58] for (ordinary) PRE, we define three kinds of security definitions of VPRE. In particular, we introduce *soundness*, which plays an important role for VPRE. We also explain the difference between our definitions and those of [58]. Finally, in Section 4.2.4, we show that a VPRE secure in our

[3]Although it is claimed that their security model is stronger than that of [58], they are actually incomparable. The security model for a transformed ciphertext (first-level ciphertext) in [65] allows an adversary a slightly more flexible challenge query than that of [58]. However, all public keys in the security models of [65] that can be used for re-encryption key generation and re-encryption queries must be generated by the challenger, and such restriction is not posed in the model of [58].

definitions is also secure (as an ordinary PRE scheme) in the definitions of [58], and thus our definitions have "backward compatibility" with [58].

4.2.1 Model

Here, we define the syntax of VPRE. As mentioned earlier, the main feature of VPRE is the re-encryption verification algorithm $\mathsf{REncVer}$.

Formally, a VPRE scheme consists of the following seven algorithms:

KG This is the key generation algorithm that takes 1^k as input, and outputs a secret key sk and a public key pk. This process is written as $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KG}(1^k)$.

RKG This is the re-encryption key generation algorithm that takes a secret key sk_i of user i and a public key pk_j of user j as input, and outputs a re-encryption key $rk_{i \to j}$. This process is written as $rk_{i \to j} \leftarrow \mathsf{RKG}(\mathsf{sk}_i, \mathsf{pk}_j)$.

Enc This is the encryption algorithm that takes a public key pk and a plaintext m as input, and outputs a *second-level* ciphertext c that can be re-encrypted for another party. This process is written as $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m)$.

REnc This is the re-encryption algorithm that takes a second-level ciphertext c (for user i) and a re-encryption key $rk_{i \to j}$ as input, and outputs a *first-level* ciphertext $\widehat{c}$ (for user j) or the special symbol $\perp$ meaning that ($rk_{i \to j}$ or) c is invalid. This process is written as $\widehat{c}$ (or $\perp$) $\leftarrow \mathsf{REnc}(rk_{i \to j}, c)$.

$\mathsf{REncVer}$ This is the re-encryption verification algorithm that takes a public key pk_i of user i, a secret key sk_j of user j, a second-level ciphertext c, and a first-level ciphertext $\widehat{c}$ as input, and outputs $\top$ (meaning that $\widehat{c}$ is a valid re-encrypted ciphertext of c_i) or $\perp$. This process is written as $\top$ (or $\perp$) $\leftarrow \mathsf{REncVer}(\mathsf{pk}_i, \mathsf{sk}_j, c, \widehat{c})$.

Dec_1 This is the first-level decryption algorithm that takes a secret key sk and a first-level ciphertext $\widehat{c}$ as input, and outputs a decryption result m (which could be the special symbol $\perp$ meaning that $\widehat{c}$ is invalid). This process is written as $m \leftarrow \mathsf{Dec}_1(\mathsf{sk}, \widehat{c})$.

Dec_2 This is the second-level decryption algorithm that takes a secret key sk and a second-level ciphertext c as input, and outputs a decryption result m (which could be $\perp$ as above). This process is written as $m \leftarrow \mathsf{Dec}_2(\mathsf{sk}, c)$.

The $\mathsf{REncVer}$ algorithm needs not only a re-encrypted ciphertext $\widehat{c}$ but also a (candidate) original ciphertext c. We again stress that such a situation is natural in the applications of PRE which we explained in Section 6.1. We remark that as in [58], we do not consider the direct first-level encryption algorithm (that generates a first-level ciphertext that cannot be re-encrypted further), because such a functionality can be realized by just using a CCA secure PKE scheme in addition to a (V)PRE scheme.

We say that a VPRE scheme is *correct* if for all $(\mathsf{sk}_i, \mathsf{pk}_i)$ and $(\mathsf{sk}_j, \mathsf{pk}_j)$ output from $\mathsf{KG}(1^k)$, all plaintexts m, all $rk_{i \to j} \leftarrow \mathsf{RKG}(\mathsf{sk}_i, \mathsf{pk}_j)$, all $c_i \leftarrow \mathsf{Enc}(pk_i, m)$, and all $\widehat{c}_j \leftarrow \mathsf{REnc}(rk_{i \to j}, c_i)$, we have: (1) $\mathsf{Dec}_2(\mathsf{sk}_i, c_i) = m$, (2) $\mathsf{Dec}_1(\mathsf{sk}_j, \widehat{c}_j) = m$, and (3) $\mathsf{REncVer}(\mathsf{pk}_i, \mathsf{sk}_j, c_i, \widehat{c}_j) = \top$.

4.2.2 Security Definitions

In this subsection, we give the formal security definitions of VPRE.

Soundness. According to the correctness requirement, an algorithm that outputs 1 for any input is "correct" as the re-encryption verification algorithm $\mathsf{REncVer}$. However, this is clearly not what we expect for $\mathsf{REncVer}$. To avoid such triviality and a meaningless definition, here we introduce *soundness* of the $\mathsf{REncVer}$ algorithm. Roughly, our soundness definition guarantees that if an adversary who owns a re-encryption key $rk_{i \to j}$ and is given an original (second-level) ciphertext c, it can produce only a re-encrypted ciphertext $\widehat{c}$ that can decrypt to the same value as the decryption result of c. Furthermore, if an adversary does not have the re-encryption key $rk_{i \to j}$, then it cannot produce a valid re-encrypted ciphertext $\widehat{c}$ at all.

Formally, we define the soundness of re-encryption with the following game which is parameterized by an integer $n \in \mathbf{N}$ and is played between the challenger and an adversary $\mathcal{A}$: Firstly, the challenger generates honest users' key pairs $(\mathsf{sk}_i, \mathsf{pk}_i) \leftarrow \mathsf{KG}(1^k)$ for $i \in [n]$, and sets $\mathcal{PK} = \{\mathsf{pk}_i\}_{i \in [n]}$. Next, the challenger generates a challenge user's key pair $(\mathsf{sk}_{i^*}, \mathsf{pk}_{i^*}) \leftarrow \mathsf{KG}(1^k)$. Then, the challenger gives 1^k and $\mathcal{PK}^* = \mathcal{PK} \cup \{\mathsf{pk}_{i^*}\}$ to $\mathcal{A}$. Then, $\mathcal{A}$ can adaptively make the following types of queries:

Re-encryption key generation (RKG) query: On input $(\mathsf{pk}_i \in \mathcal{PK}^*, \mathsf{pk}_j)$, where pk_j is an arbitrary public key of $\mathcal{A}$'s choice (for which $\mathcal{A}$ is not required to reveal the corresponding secret key), the challenger responds as follows: If $\mathsf{pk}_i = \mathsf{pk}_{i^*}$ and $\mathsf{pk}_j \notin \mathcal{PK}^*$, then the challenger responds with $\perp$. Otherwise, the challenger responds with $\mathsf{RKG}(\mathsf{sk}_i, \mathsf{pk}_j)$.

Re-encryption (REnc) query: On input $(\mathsf{pk}_i \in \mathcal{PK}^*, \mathsf{pk}_j, c)$, where pk_j is an arbitrary public key of $\mathcal{A}$'s choice (for which $\mathcal{A}$ is not required to reveal the corresponding secret key), the challenger responds with $\mathsf{REnc}(\mathsf{RKG}(\mathsf{sk}_i, \mathsf{pk}_j), c)$.

Re-encryption verification ($\mathsf{REncVer}$) query: On input $(\mathsf{pk}_i, \mathsf{pk}_j \in \mathcal{PK}^*, c, \widehat{c})$, where pk_i is an arbitrary public key of $\mathcal{A}$'s choice (for which $\mathcal{A}$ is not required to reveal the corresponding secret key), the challenger responds with $\mathsf{REncVer}(\mathsf{pk}_i, \mathsf{sk}_j, c, \widehat{c})$.

Challenge query: This query is asked only once. On input m^*, the challenger runs $c^* \leftarrow \mathsf{Enc}(\mathsf{pk}_{i^*}, m^*)$, and returns c^* to $\mathcal{A}$.

First-level decryption (Dec_1) query: On input $(\mathsf{pk}_j \in \mathcal{PK}^*, \widehat{c})$, the challenger responds with $\mathsf{Dec}_1(\mathsf{sk}_j, \widehat{c})$.

Second-level decryption (Dec_2) query: On input $(\mathsf{pk}_i \in \mathcal{PK}^*, c)$, the challenger responds with $\mathsf{Dec}_2(\mathsf{sk}_i, c)$.

Finally, $\mathcal{A}$ outputs $(\mathsf{pk}_j \in \mathcal{PK}^*, \widehat{c}^*)$ and wins the game if they satisfy the following three conditions:

1. $\mathsf{REncVer}(\mathsf{pk}_{i^*}, \mathsf{sk}_j, c^*, \widehat{c}^*) = \top$
2. $\widehat{c}^*$ is not an answer to some of $\mathcal{A}$'s REnc queries of the form $(\mathsf{pk}_{i^*}, \mathsf{pk}_j, c^*)$

3. Either of the following conditions is satisfied:
    - In the case that $\mathcal{A}$ has submitted a RKG query of the form $(\mathsf{pk}_{i^*}, \mathsf{pk}_j)$ and obtained a re-encryption key $rk_{i^* \to j}$: $\mathsf{Dec}_1(\mathsf{sk}_j, \widehat{c}^*) \neq m^*$.
    - Otherwise: $\mathsf{Dec}_1(\mathsf{sk}_j, \widehat{c}^*) \neq \bot$.

We define the advantage of $\mathcal{A}$ by $\mathsf{Adv}^{\mathtt{SND\text{-}VPRE}}_{(\mathcal{A},n)}(k) = \Pr[\mathcal{A} \text{ wins}]$.

Definition 3 (Soundness of Re-encryption Verification). *We say that a VPRE scheme satisfies* soundness*, if for any PPT adversary $\mathcal{A}$ and for all positive polynomials n, $\mathsf{Adv}^{\mathtt{SND\text{-}VPRE}}_{(\mathcal{A},n)}(k)$ is negligible.*

Second-Level CCA Security. Here, we define the security for second-level ciphertexts (*second-level CCA security*) with the following game which is parameterized by an integer $n \in \mathbf{N}$ and is played between the challenger and an adversary $\mathcal{A}$: Firstly, the challenger generates honest users' key pairs $(\mathsf{sk}_i, \mathsf{pk}_i) \leftarrow \mathsf{KG}(1^k)$ for $i \in [n]$, and sets $\mathcal{PK} = \{\mathsf{pk}_i\}_{i \in [n]}$. Next, the challenger generates the challenge user's key pair $(\mathsf{sk}_{i^*}, \mathsf{pk}_{i^*}) \leftarrow \mathsf{KG}(1^k)$. Then, the challenger gives 1^k and $\mathcal{PK}^* = \mathcal{PK} \cup \{\mathsf{pk}_{i^*}\}$ to $\mathcal{A}$. Then, $\mathcal{A}$ can adaptively make the following types of queries:

Re-encryption key generation (RKG) and Re-encryption verification ($\mathsf{REncVer}$) queries:
These are the same as those in the soundness game.

Re-encryption (REnc) query: On input $(\mathsf{pk}_i \in \mathcal{PK}^*, \mathsf{pk}_j, c)$, where pk_j is an arbitrary public key of $\mathcal{A}$'s choice (for which $\mathcal{A}$ is not required to reveal the corresponding secret key), the challenger responds as follows. If $(\mathsf{pk}_i, c) = (\mathsf{pk}_{i^*}, c^*)$ and $\mathsf{pk}_j \notin \mathcal{PK}^*$, then the challenger returns $\bot$ to $\mathcal{A}$. Otherwise, the challenger responds with $\mathsf{REnc}(\mathsf{RKG}(\mathsf{sk}_i, \mathsf{pk}_j), c)$.

Challenge query: This query is asked only once. On input (m_0, m_1), the challenger picks a bit $b \in \{0,1\}$ uniformly at random, and computes $c^* \leftarrow \mathsf{Enc}(\mathsf{pk}_{i^*}, m_b)$. Then it gives c^* to $\mathcal{A}$.

First-level decryption (Dec_1) query: On input $(\mathsf{pk}_j \in \mathcal{PK}^*, \widehat{c})$, the challenger responds as follow: If $\mathsf{REncVer}(\mathsf{pk}_{i^*}, \mathsf{sk}_j, c^*, \widehat{c}) = \top$, then the challenger returns $\bot$ to $\mathcal{A}$. Otherwise, the challenger responds with $\mathsf{Dec}_1(\mathsf{sk}_j, \widehat{c})$.

Second-level decryption (Dec_2) query: On input $(\mathsf{pk}_i \in \mathcal{PK}^*, c)$, the challenger responds with $\mathsf{Dec}_2(\mathsf{sk}_i, c)$, except that if $(\mathsf{pk}_i, c) = (\mathsf{pk}_{i^*}, c^*)$, then the challenger returns the special symbol $\bot$ to $\mathcal{A}$.

Finally, $\mathcal{A}$ outputs its guess b' for b and wins the game if $b = b'$. We define the advantage of $\mathcal{A}$ by $\mathsf{Adv}^{\mathtt{second\text{-}VPRE}}_{(\mathcal{A},n)}(k) = |\Pr[b = b'] - 1/2|$.

Definition 4 (Second-Level CCA Security). *We say that a VPRE scheme is* second-level CCA secure*, if for any PPT adversary $\mathcal{A}$ and all positive polynomials n, $\mathsf{Adv}^{\mathtt{second\text{-}VPRE}}_{(\mathcal{A},n)}(k)$ is negligible.*

First-Level CCA Security. Next, we define the security for first-level ciphertexts (*first-level CCA security*) with the following game between the challenger and an adversary $\mathcal{A}$: Firstly, the challenger generates the challenge key pair $(\mathsf{sk}^*, \mathsf{pk}^*) \leftarrow \mathsf{KG}(1^k)$, and gives 1^k and pk^* to $\mathcal{A}$. Then, $\mathcal{A}$ can adaptively make the following types of queries:

Re-encryption key generation (RKG) query: On input pk, where pk is an arbitrary public key of $\mathcal{A}$'s choice (for which $\mathcal{A}$ is not required to reveal the corresponding secret key), the challenger responds with $\mathsf{RKG}(\mathsf{sk}^*, \mathsf{pk})$.

Re-encryption verification ($\mathsf{REncVer}$) query: On input $(\mathsf{pk}, c, \widehat{c})$, where pk is an arbitrary public of $\mathcal{A}$'s choice (for which $\mathcal{A}$ is not required to reveal the corresponding secret key), the challenger responds with $\mathsf{REncVer}(\mathsf{pk}, \mathsf{sk}^*, c, \widehat{c})$.

Challenge query: This query is asked only once. On input $(\mathsf{sk}_{\mathcal{A}}, \mathsf{pk}_{\mathcal{A}}, m_0, m_1)$ where $(\mathsf{sk}_{\mathcal{A}}, \mathsf{pk}_{\mathcal{A}})$ is required to be a valid key pair[4], the challenger picks the challenge bit $b \in \{0, 1\}$ randomly and runs $c \leftarrow \mathsf{Enc}(\mathsf{pk}_{\mathcal{A}}, m_b)$ and $\widehat{c}^* \leftarrow \mathsf{REnc}(\mathsf{RKG}(\mathsf{sk}_{\mathcal{A}}, \mathsf{pk}^*), c)$. It then returns $\widehat{c}^*$ to $\mathcal{A}$.

First-level decryption (Dec_1) query: On input $\widehat{c}$, the challenger responds with $\mathsf{Dec}_1(\mathsf{sk}^*, \widehat{c})$, except that if $\widehat{c} = \widehat{c}^*$, then the challenger returns the special symbol $\perp$ to $\mathcal{A}$.

Second-level decryption (Dec_2) query: On input c, the challenger responds with $\mathsf{Dec}_2(\mathsf{sk}^*, c)$.

Finally, $\mathcal{A}$ outputs its guess b' for b and wins the game if $b = b'$. We define the advantage of $\mathcal{A}$ by $\mathsf{Adv}_{\mathcal{A}}^{\mathtt{first\text{-}VPRE}}(k) = |\Pr[b = b'] - 1/2|$.

Definition 5 (First-Level CCA Security)**.** *We say that a VPRE scheme is first-level CCA secure, if for any PPT adversary $\mathcal{A}$, $\mathsf{Adv}_{\mathcal{A}}^{\mathtt{first\text{-}VPRE}}(k)$ is negligible.*

4.2.3 Security Definitions for PRE by [58]

Here, we recall the security definitions of [58]. In order not to mix up with our security definitions for VPRE, we put the prefix "HKK$^+$" for the security definitions of [58].

Second-Level CCA Security [58, Definition 1]. The security for second-level ciphertexts in [58] (*HKK$^+$-second-level CCA security*) is defined using the following game which is parameterized by an integer $n \in \mathbf{N}$ and is played between the challenger and an adversary $\mathcal{A}$: Firstly, the challenger generates honest users' key pairs $(\mathsf{sk}_i, \mathsf{pk}_i) \leftarrow \mathsf{KG}(1^k)$ for $i \in [n]$, and sets $\mathcal{PK} = \{\mathsf{pk}_i\}_{i \in [n]}$. Next, the challenger generates the challenge user's key pair $(\mathsf{sk}_{i^*}, \mathsf{pk}_{i^*}) \leftarrow \mathsf{KG}(1^k)$. Then, the challenger gives 1^k and $\mathcal{PK}^* = \mathcal{PK} \cup \{\mathsf{pk}_{i^*}\}$ to $\mathcal{A}$. Then, $\mathcal{A}$ can adaptively make the following types of queries:

Re-encryption key generation (RKG) query: On input $(\mathsf{pk}_i \in \mathcal{PK}^*, \mathsf{pk}_j)$, where pk_j is an arbitrary public key of $\mathcal{A}$'s choice (for which $\mathcal{A}$ is not required to reveal the corresponding secret key), the challenger responds as follows: If $\mathsf{pk}_i = \mathsf{pk}_{i^*}$ and $\mathsf{pk}_j \notin \mathcal{PK}^*$, then the challenger responds with $\perp$. Otherwise, the challenger responds with $\mathsf{RKG}(\mathsf{sk}_i, \mathsf{pk}_j)$.

[4]That is, $(\mathsf{sk}_{\mathcal{A}}, \mathsf{pk}_{\mathcal{A}})$ is required to be in the range of $\mathsf{KG}(1^k)$.

Re-encryption (REnc) query: On input $(\mathsf{pk}_i \in \mathcal{PK}^*, \mathsf{pk}_j, c)$, where pk_j is an arbitrary public key of $\mathcal{A}$'s choice (for which $\mathcal{A}$ is not required to reveal the corresponding secret key), the challenger responds as follows. If $(\mathsf{pk}_i, c) = (\mathsf{pk}_{i^*}, c^*)$ and $\mathsf{pk}_j \notin \mathcal{PK}^*$, then the challenger returns $\perp$ to $\mathcal{A}$. Otherwise, the challenger responds with $\mathsf{REnc}(\mathsf{RKG}(\mathsf{sk}_i, \mathsf{pk}_j), c)$.

Challenge query: This query is asked only once. On input (m_0, m_1), the challenger picks a bit $b \in \{0,1\}$ uniformly at random, and computes $c^* \leftarrow \mathsf{Enc}(\mathsf{pk}_{i^*}, m_b)$. Then it gives c^* to $\mathcal{A}$.

First-level decryption (Dec_1) query: On input $(\mathsf{pk}_j \in \mathcal{PK}^*, \widehat{c})$, the challenger responds as follow: If $\mathcal{A}$ has asked a re-encryption query $\mathsf{pk}_{i^*}, \mathsf{pk}_j \in \mathcal{PK}^*, c_{i^*}$ and obtained $\widehat{c}$ previously, then the challenger returns $\perp$ to $\mathcal{A}$. Else if $\mathcal{A}$ has asked a re-encryption key generation query $\mathsf{pk}_{i^*}, \mathsf{pk}_j \in \mathcal{PK}^*$ previously and $\mathsf{Dec}_1(\mathsf{sk}_j, \widehat{c}) \in \{m_0, m_1\}$, then the challenger returns the special symbol *test* to $\mathcal{A}$. Otherwise, the challenger responds with $\mathsf{Dec}_1(\mathsf{sk}_j, \widehat{c})$.

Second-level decryption (Dec_2) query: On input $(\mathsf{pk}_i \in \mathcal{PK}^*, c)$, the challenger responds with $\mathsf{Dec}_2(\mathsf{sk}_i, c)$, except that if $(\mathsf{pk}_i, c) = (\mathsf{pk}_{i^*}, c^*)$, then the challenger returns the special symbol $\perp$ to $\mathcal{A}$.

Finally, $\mathcal{A}$ outputs its guess b' for b and wins the game if $b = b'$. We define the advantage of $\mathcal{A}$ by $\mathsf{Adv}^{\texttt{HKK+-second-PRE}}_{(\mathcal{A},n)}(k) = |\Pr[b = b'] - 1/2|$.

Definition 6 (Second-Level CCA Security)**.** *We say that a PRE scheme is* second-level CCA secure*, if for any PPT adversary $\mathcal{A}$ and all positive polynomials n, $\mathsf{Adv}^{\texttt{HKK+-second-PRE}}_{(\mathcal{A},n)}(k)$ is negligible.*

First-Level CCA Security [58, Definition 2]. The security for first-level ciphertexts (*HKK$^+$-first-level CCA security*) with the following game between the challenger and an adversary $\mathcal{A}$: Firstly, the challenger generates the challenge key pair $(\mathsf{sk}^*, \mathsf{pk}^*) \leftarrow \mathsf{KG}(1^k)$, and gives 1^k and pk^* to $\mathcal{A}$. Then, $\mathcal{A}$ can adaptively make the following types of queries:

Re-encryption key generation (RKG) query: On input pk, where pk is an arbitrary public key of $\mathcal{A}$'s choice (for which $\mathcal{A}$ is not required to reveal the corresponding secret key), the challenger responds with $\mathsf{RKG}(\mathsf{sk}^*, \mathsf{pk})$.

Challenge query: This query is asked only once. On input $(\mathsf{sk}_{\mathcal{A}}, \mathsf{pk}_{\mathcal{A}}, m_0, m_1)$ where $(\mathsf{sk}_{\mathcal{A}}, \mathsf{pk}_{\mathcal{A}})$ is required to be a valid key pair[5], the challenger picks the challenge bit $b \in \{0,1\}$ randomly and runs $c \leftarrow \mathsf{Enc}(\mathsf{pk}_{\mathcal{A}}, m_b)$ and $\widehat{c}^* \leftarrow \mathsf{REnc}(\mathsf{RKG}(\mathsf{sk}_{\mathcal{A}}, \mathsf{pk}^*), c)$. It then returns $\widehat{c}^*$ to $\mathcal{A}$.

First-level decryption (Dec_1) query: On input $\widehat{c}$, the challenger responds with Dec_1 $(\mathsf{sk}^*, \widehat{c})$, except that if $\widehat{c} = \widehat{c}^*$, then the challenger returns the special symbol $\perp$ to $\mathcal{A}$.

Second-level decryption (Dec_2) query: On input c, the challenger responds with Dec_2 (sk^*, c).

[5]That is, $(\mathsf{sk}_{\mathcal{A}}, \mathsf{pk}_{\mathcal{A}})$ is required to be in the range of $\mathsf{KG}(1^k)$.

Finally, $\mathcal{A}$ outputs its guess b' for b and wins the game if $b = b'$. We define the advantage of $\mathcal{A}$ by $\mathsf{Adv}_{\mathcal{A}}^{\texttt{HKK+-first-PRE}}(k) = |\Pr[b = b'] - 1/2|$.

Definition 7 (First-Level CCA Security)**.** *We say that a VPRE scheme is first-level CCA secure, if for any PPT adversary $\mathcal{A}$, $\mathsf{Adv}_{\mathcal{A}}^{\texttt{HKK+-first-PRE}}(k)$ is negligible.*

Difference with the Definitions (for PRE) in [58]. Here, we explain the differences of our security definitions for VPRE with the definitions (for PRE) by Hanaoka et al. [58]. (We use the prefix "HKK$^+$" for the names of the security definitions in [58].) Soundness is a new security definition for VPRE. Furthermore, regarding the definition of first-level CCA security, we naturally allow $\mathsf{REncVer}$ queries for an adversary in addition to queries allowed in the first-level CCA definition of Definition 7.

For the security definition of second-level ciphertexts, we also allow an adversary to make $\mathsf{REncVer}$ queries. Furthermore, there is a remarkable difference in the response to Dec_1 queries. The response to Dec_1 queries in the second-level CCA security game defined in Definition 6 is as follows (where we *emphasize* the difference).

First-level decryption query (of [58]) : On input $(\mathsf{pk}_j \in \mathcal{PK}^*, \widehat{c})$, the challenger responds as follows: *If $\mathcal{A}$ has asked a* REnc *query of the form $(\mathsf{pk}_{i^*}, \mathsf{pk}_j \in \mathcal{PK}, c^*)$ and obtained $\widehat{c}_i$ previously, then the challenger returns $\perp$ to $\mathcal{A}$. Else if $\mathcal{A}$ has asked a* RKG *query of the form $(\mathsf{pk}_{i^*}, \mathsf{pk}_j \in \mathcal{PK})$ previously and $\mathsf{Dec}_1(\mathsf{sk}_i, \widehat{c}) \in \{m_0, m_1\}$ holds, then the challenger returns the special symbol* test *to $\mathcal{A}$.* Otherwise, the challenger responds with $\mathsf{Dec}_1(\mathsf{sk}_i, \widehat{c})$. (We note that here, test is a symbol that is distinguished from $\perp$.)

Note that in a CCA security definition, what we expect is that an adversary can ask any ciphertext that does not trivially allow it to decrypt the challenge ciphertext. The emphasized sentences above are the definitional approach taken in [58] to avoid such a "self-broken" definition considered in [58]. On the other hand, our definition of second-level CCA security given in this subsection uses $\mathsf{REncVer}$ for deciding "prohibited" decryption queries, and thus is simplified (of course, we additionally need *soundness* in order for $\mathsf{REncVer}$ to be meaningful). Our use of $\mathsf{REncVer}$ for deciding "prohibited" queries in the CCA security game for an encryption scheme has some similarity with *secretly detectable re-playable CCA security* of [34] and *detectable PKE* of [62], and we believe these connections to be interesting.

4.2.4 Implications to the Definitions of [58]

Here, we show a "backward compatibility" of our security definitions. Namely, we show that if a VPRE scheme satisfies security definitions given in Section 4.2.2, then it is also secure as a (V)PRE under the definitions of [58].

Theorem 7. *If a VPRE scheme is first-level CCA secure in the sense of Definition 5, then the VPRE scheme is first-level CCA secure in the sense of Definition 7.*

This is obvious from the definition. In particular, an adversary in our first-level CCA security definition is only more powerful than that of Definition 7 (our adversary can make $\mathsf{REncVer}$ queries which are not considered in [58]).

Theorem 8. *If a VPRE scheme is second-level CCA secure in the sense of Definition 4 and satisfies soundness (Definition 3), then the VPRE scheme is second-level CCA secure in the sense of Definition 6.*

Proof of Theorem 8 Let $n > 0$ be a polynomial, and $\mathcal{A}$ be any PPT adversary that attacks a VPRE scheme in the sense of [58, Definition 1] and makes $Q > 0$ Dec_1 queries. (Since $\mathcal{A}$ is PPT, Q is polynomial.) Consider the following games.

Game 0. The second-level CCA game of Definition 6.

Game 1. Same as Game 0, except that if $\mathcal{A}$ submits a Dec_1 query $(\mathsf{pk}_j, \widehat{c})$ such that (1) $\mathsf{REncVer}(\mathsf{pk}_{i^*}, \mathsf{sk}_j, c_{i^*}, \widehat{c}) = \top$, and (2) $\widehat{c}$ is not an answer to some of $\mathcal{A}$'s REnc queries of the form $(\mathsf{pk}_{i^*}, \mathsf{pk}_j, c^*)$, then the challenger responds as follows:

If $\mathcal{A}$ has submitted a RKG query $(\mathsf{pk}_{i^*}, \mathsf{pk}_j)$ before, then the challenger returns test to $\mathcal{A}$. Otherwise, the challenger returns $\perp$ to $\mathcal{A}$.

For $i \in \{0, 1\}$, let Succ_i be the event that in Game i $\mathcal{A}$ succeeds in guessing the challenge bit (i.e. $b' = b$ occurs), and let Bad_i be the event that in Game i, $\mathcal{A}$ submits at least one Dec_1 query $(\mathsf{pk}_j, \widehat{c})$ such that it satisfies the following conditions simultaneously:

1. $\mathsf{REncVer}(\mathsf{pk}_{i^*}, \mathsf{sk}_j, c^*, \widehat{c}) = \top$.
2. $\widehat{c}$ has not appeared as an answer to some of $\mathcal{A}$'s previous REnc queries of the form $(\mathsf{pk}_{i^*}, \mathsf{pk}_j, c^*)$.
3. Either of the following conditions is satisfied:
   - In the case that $\mathcal{A}$ has submitted a RKG query $(\mathsf{pk}_{i^*}, \mathsf{pk}_j)$ and obtained a re-encryption key $rk_{i^* \to j}$: $\mathsf{Dec}_1(\mathsf{sk}_j, \widehat{c}) \notin \{m_0, m_1\}$.
   - Otherwise: $\mathsf{Dec}_1(\mathsf{sk}_j, \widehat{c}) \neq \perp$.

$\mathcal{A}$'s advantage (in the second-level CCA definition of 6) is calculated as follows:

$$|\Pr[\mathsf{Succ}_0] - \frac{1}{2}| \le |\Pr[\mathsf{Succ}_0] - \Pr[\mathsf{Succ}_1]| + |\Pr[\mathsf{Succ}_1] - \frac{1}{2}|.$$

Thus, it suffices to show that each term in the right hand side of the above inequality is negligible.

Firstly, note that Game 0 and Game 1 proceed identically unless Bad_0 or Bad_1 occurs in the corresponding games. Hence, we have $|\Pr[\mathsf{Succ}_0] - \Pr[\mathsf{Succ}_1]| \le \Pr[\mathsf{Bad}_0] = \Pr[\mathsf{Bad}_1]$. Then, we show that we can construct a soundness adversary $\mathcal{B}$ such that $\mathsf{Adv}^{\mathtt{SND\text{-}VPRE}}_{(\mathcal{B},n)}(k) \ge (1/Q) \cdot \Pr[\mathsf{Bad}_1]$, which implies that $\Pr[\mathsf{Bad}_1]$ is negligible.

The construction of $\mathcal{B}$ is as follows: First, $\mathcal{B}$ is given public keys $(\mathsf{pk}_1, \cdots, \mathsf{pk}_n, \mathsf{pk}_{i^*})$ from the soundness challenger. Then $\mathcal{B}$ forwards them to $\mathcal{A}$.

$\mathcal{B}$ answers to $\mathcal{A}$'s queries (except for the challenge query) exactly as specified in Game 1. This is possible because $\mathcal{B}$ can also query to $\mathcal{B}$'s challenger except for the challenge query. When $\mathcal{A}$ submits the challenge query (m_0, m_1), $\mathcal{B}$ randomly picks $d \leftarrow \{0, 1\}$, submits m_d as $\mathcal{B}$'s challenge to $\mathcal{B}$'s challenger, receives c^*, and returns c^* to $\mathcal{A}$.

When $\mathcal{A}$ terminates, from $\mathcal{A}$'s Dec_1 queries, $\mathcal{B}$ randomly picks one query $(pk_j, \widehat{c})$, and terminates with output $(pk_j, \widehat{c})$.

The above completes the description of $\mathcal{B}$. It is not hard to see that $\mathcal{B}$ simulates Game 1 perfectly for $\mathcal{A}$ until $\mathcal{A}$ submits a Dec_1 query satisfying the conditions of the event Bad_1. Therefore, the probability that $\mathcal{A}$ submits a Dec_1 query satisfying the conditions of Bad_1 in the game simulated by $\mathcal{B}$ is exactly the same as the probability of this event occurring in Game 1. Furthermore, once $\mathcal{A}$ makes such a query, $\mathcal{B}$ can pick it with probability at least $1/Q$. Therefore, we have $\mathsf{Adv}^{\mathtt{SND\text{-}VPRE}}_{(\mathcal{B},n)}(k) \geq (1/Q) \cdot \Pr[\mathsf{Bad}_1]$. Hence, $\Pr[\mathsf{Bad}_1]$ is negligible. This in turn implies that $|\Pr[\mathsf{Succ}_1] - \Pr[\mathsf{Succ}_2]|$ is negligible.

To prove Theorem 8, it remains to show that $|\Pr[\mathsf{Succ}_1] - 1/2|$ is negligible. However, it is straightforward from the definition of the second-level CCA security of the VPRE scheme (in the sense of Definition 4). In particular, a second-level CCA adversary (in the sense of Definition 4) can simulate Game 1 perfectly for $\mathcal{A}$, and such an adversary has advantage exactly $|\Pr[\mathsf{Succ}_1] - 1/2|$.

This completes the proof of Theorem 8. $\square$

4.3 A Concrete VPRE Scheme

In this section, we show a concrete VPRE scheme and intuitions for its security. Specifically, our VPRE scheme is a simple extension of the PRE scheme by Hanaoka et al. (which we denote by HKK^+) [58], and we show how to implement the re-encryption verification algorithm for it.

Intuition to Achieve the Functionality of Re-encryption Verification. Consider a situation in which a second-level ciphertext c_A for user A is re-encrypted into a first-level ciphertext $\widehat{c}_B$ for user B. In order to achieve PRE with re-encryption verifiability, one promising approach is to design a PRE scheme with the following properties: (1) When re-encrypting c_A into $\widehat{c}_B$, c_A is somehow embedded into $\widehat{c}_B$ in such a way that when user B decrypts $\widehat{c}_B$, the embedded second-level ciphertext c_A can be extracted. (2) In re-encryption verification (between $\widehat{c}_B$ and a candidate second-level ciphertext c'_A) user B checks whether an extracted ciphertext c_A is equal to the given candidate c'_A. We observe that the HKK^+ scheme has the desirable properties, and this is the reason why we focus on the PRE scheme. We next explain how we extend it into a VPRE scheme.

Extending the Hanaoka et al. PRE [58] to VPRE. Recall that the PRE scheme HKK^+ is a generic construction from a re-splittable TPKE scheme, an (ordinary) PKE scheme, and a signature scheme. We observe that a re-encrypted ciphertext (i.e. first-level ciphertext) $\widehat{c}$ of the HKK^+ scheme contains the information on an original ciphertext (i.e. second-level ciphertext) c which is just a ciphertext of the underlying TPKE scheme. Our re-encryption verification algorithm is thus fairly simple: On input $(\mathsf{pk}_i, \mathsf{sk}_j, c, \widehat{c})$, it executes the first-level decryption algorithm of the HKK^+ scheme partway to recover the "embedded" second-level ciphertext c', and checks whether $c = c'$ holds.

Now, we formally describe the VPRE scheme, which we denote by eHKK^+ (which stands for "extended HKK^+"). Let (TKG, TEnc, TSplit, TShDec, TShVer, TCom) be a re-splittable TPKE scheme, (PKG, PEnc, PDec) be a PKE scheme, and (SKG, Sign, SVer)

be a signature scheme. Using these as building blocks, the VPRE scheme eHKK^+ is constructed as in Fig. 4.1.

```
$\mathsf{KG}(1^k)$ :
    $(tsk, tpk) \leftarrow \mathsf{TKG}(1^k, 2, 2)$
    $(\widehat{dk}, \widehat{pk}) \leftarrow \mathsf{PKG}(1^k)$
    $(dk, pk) \leftarrow \mathsf{PKG}(1^k)$
    $(sk, vk) \leftarrow \mathsf{SKG}(1^k)$
    $\mathsf{sk} \leftarrow (tsk, \widehat{dk}, dk, sk)$
    $\mathsf{pk} \leftarrow (tpk, \widehat{pk}, pk, vk)$
    Return $(\mathsf{sk}, \mathsf{pk})$.

$\mathsf{Enc}(\mathsf{pk}_i, m)$ :
    $(tpk_i, \widehat{pk}_i, pk_i, vk_i) \leftarrow \mathsf{pk}_i$
    Return $c \leftarrow \mathsf{TEnc}(tpk_i, m)$.

$\mathsf{RKG}(\mathsf{sk}_i, \mathsf{pk}_j)$ :
    $(tsk_i, \widehat{dk}_i, dk_i, sk_i) \leftarrow \mathsf{sk}_i$
    $(tpk_j, \widehat{pk}_j, pk_j, vk_j) \leftarrow \mathsf{pk}_j$
    $(tsk_{i.1}, tsk_{i.2}, tvk_i) \leftarrow \mathsf{TSplit}(tsk_i)$
    $\psi \leftarrow \mathsf{PEnc}(pk_j, tsk_{i.1})$
    $\sigma \leftarrow \mathsf{Sign}(sk_i, \langle \psi \| tvk_i \| \mathsf{pk}_i \| \mathsf{pk}_j \rangle)$
    $rk_{i \to j} \leftarrow (\mathsf{pk}_i, \mathsf{pk}_j, tsk_{i.2}, \psi, tvk_i, \sigma)$
    Return $rk_{i \to j}$.

$\mathsf{Dec}_1(\mathsf{sk}_j, \widehat{c}_j)$ :
    $(tsk_j, \widehat{dk}_j, dk_j, sk_j) \leftarrow \mathsf{sk}_j$
    $\widehat{M} \leftarrow \mathsf{PDec}(\widehat{dk}_j, \widehat{c}_j)$
    If $\widehat{M} = \perp$ then return $\perp$.
    $\langle \mathsf{pk}'_i \| \mathsf{pk}'_j \| c_i \| \mu_2 \| \psi \| tvk_i \| \sigma \rangle \leftarrow \widehat{M}$
    If $\mathsf{pk}'_j \neq \mathsf{pk}_j$ then return $\perp$.
    $(tpk_i, \widehat{pk}_i, pk_i, vk_i) \leftarrow \mathsf{pk}'_i$
    If $\mathsf{SVer}(vk_i, \langle \psi \| tvk_i \| \mathsf{pk}'_i \| \mathsf{pk}'_j \rangle, \sigma) = \perp$
        then return $\perp$.
    $tsk_{i.1} \leftarrow \mathsf{PDec}(dk_j, \psi)$
    If $tsk_{i.1} = \perp$ then return $\perp$.
    $\mu_1 \leftarrow \mathsf{TShDec}(tpk_i, tsk_{i.1}, c_i)$
    If $\mu_1 = \perp$ then return $\perp$.
    If $\mathsf{TShVer}(tpk_i, tvk_i, c_i, 2, \mu_2) = \perp$
        then return $\perp$.
    $m \leftarrow \mathsf{TCom}(tpk_i, tvk_i, c_i, \{\mu_1, \mu_2\})$
    Return $m$.

$\mathsf{REnc}(rk_{i \to j}, c_i)$ :
    $(\mathsf{pk}_i, \mathsf{pk}_j, tsk_{i.2}, \psi, tvk_i, \sigma) \leftarrow rk_{i \to j}$
    $(tpk_i, \widehat{pk}_i, pk_i, vk_i) \leftarrow \mathsf{pk}_i$
    If $\mathsf{SVer}(vk_i, \langle \psi \| tvk_i \| \mathsf{pk}_i \| \mathsf{pk}_j \rangle, \sigma) = \perp$
        then return $\perp$.
    $(tpk_j, \widehat{pk}_j, pk_j, vk_j) \leftarrow \mathsf{pk}_j$
    $\mu_2 \leftarrow \mathsf{TShDec}(tpk_i, tsk_{i.2}, c_i)$
    If $\mu_2 = \perp$ then return $\perp$.
    $\widehat{M} \leftarrow \langle \mathsf{pk}_i \| \mathsf{pk}_j \| c_i \| \mu_2 \| \psi \| tvk_i \| \sigma \rangle$
    Return $\widehat{c}_j \leftarrow \mathsf{PEnc}(\widehat{pk}_j, \widehat{M})$.

$\mathsf{Dec}_2(\mathsf{sk}_i, c)$ :
    $(tpk_i, \widehat{pk}_i, pk_i, vk_i) \leftarrow \mathsf{pk}_i$
    $(tsk_i, \widehat{dk}_i, dk_i, sk_i) \leftarrow \mathsf{sk}_i$
    $(tsk_{i.1}, tsk_{i.2}, tvk_i) \leftarrow \mathsf{TSplit}(tsk_i)$
    $\mu_1 \leftarrow \mathsf{TShDec}(tpk_i, tsk_{i.1}, c)$
    If $\mu_1 = \perp$ then return $\perp$.
    $\mu_2 \leftarrow \mathsf{TShDec}(tpk_i, tsk_{i.2}, c)$
    If $\mu_2 = \perp$ then return $\perp$.
    $m \leftarrow \mathsf{TCom}(tpk_i, tvk_i, c, \{\mu_1, \mu_2\})$
    Return $m$.

$\mathsf{REncVer}(\mathsf{pk}_i, \mathsf{sk}_j, c'_i, \widehat{c}_j)$ :
    $(tpk_i, \widehat{pk}_i, pk_i, vk_i) \leftarrow \mathsf{pk}_i$
    $(tsk_j, \widehat{dk}_j, dk_j, sk_j) \leftarrow \mathsf{sk}_j$
    $\widehat{M} \leftarrow \mathsf{PDec}(\widehat{dk}_j, \widehat{c}_j)$
    If $\widehat{M} = \perp$ then return $\perp$.
    $\langle \mathsf{pk}'_i \| \mathsf{pk}'_j \| c_i \| \mu_2 \| \psi \| tvk_i \| \sigma \rangle \leftarrow \widehat{M}$
    If $(\mathsf{pk}'_i, \mathsf{pk}'_j) \neq (\mathsf{pk}_i, \mathsf{pk}_j)$ then return $\perp$.
    If $\mathsf{SVer}(vk_i, \langle \psi \| tvk_i \| \mathsf{pk}'_i \| \mathsf{pk}'_j \rangle, \sigma) = \perp$
        then return $\perp$.
    $tsk_{i.1} \leftarrow \mathsf{PDec}(dk_j, \psi)$
    If $tsk_{i.1} = \perp$ then return $\perp$.
    $\mu_1 \leftarrow \mathsf{TShDec}(tpk_i, tsk_{i.1}, c_i)$
    If $\mu_1 = \perp$ then return $\perp$.
    If $\mathsf{TShVer}(tpk_i, tvk_i, c_i, 2, \mu_2) = \perp$
        then return $\perp$.
    If $c'_i = c_i$ then return $\top$ else return $\perp$.
```

Figure 4.1: The VPRE scheme eHKK^+ based on the PRE scheme by Hanaoka et al. [58]. Since Dec_2 described above needs to run TSplit, it is probabilistic. However, it can be made deterministic by running $(tsk_1, tsk_2) \leftarrow \mathsf{TSplit}(tsk)$ in KG (instead of running it in Dec_2) and including (tsk_1, tsk_2) into sk. We do not take this approach in the above so that the description is kept close to the original one shown in [58].

Security. We show that eHKK^+ satisfies the three kinds of security of VPRE.

Theorem 9. *If the PKE scheme is CCA secure, the signature scheme is strongly unforgeable, and the re-splittable TPKE scheme has decryption consistency, then the VPRE scheme* eHKK^+ *satisfies soundness.*

Intuition. The formal proof is given in Section 4.4.1. Here, we explain an intuition of the proof of soundness. Recall that the third winning condition of an adversary $\mathcal{A}$ who outputs a pair $(\mathsf{pk}_j, \widehat{c}^*)$ in the soundness game is different depending on whether $\mathcal{A}$ has obtained a re-encryption key $rk_{i^* \to j}$ by making a RKG query of the form $(\mathsf{pk}_{i^*}, \mathsf{pk}_j)$. If $\mathcal{A}$ has issued such a RKG query, then the condition is "$\mathsf{Dec_1}(\mathsf{sk}_j, \widehat{c}^*) \neq m^*$", where m^* is the challenge message, while if $\mathcal{A}$ has not done so, then the condition is "$\mathsf{Dec_1}(\mathsf{sk}_j, \widehat{c}^*) \neq \perp$".

We will show that the probability of the adversary $\mathcal{A}$ coming up with the pair $(\mathsf{pk}_j, \widehat{c}^*)$ in the latter case is negligible, mainly due to the strong unforgeability of the signature scheme. Intuitively this can be shown because if $\mathcal{A}$ can output $(\mathsf{pk}_j, \widehat{c}^*)$ such that $\mathsf{Dec_1}(\mathsf{sk}_j, \widehat{c}^*) \neq \perp$ without using a re-encryption key $rk_{i^* \to j}$, (among other things) $\mathcal{A}$ must have generated a forged signature in the plaintext of $\widehat{c}^*$, without relying on RKG queries. However, note that $\mathcal{A}$ may indirectly obtain $rk_{i^* \to j}$ through a REnc query of the form $(\mathsf{pk}_{i^*}, \mathsf{pk}_j, c)$ where c is some second-level ciphertext. Therefore, we also need to use the CCA security of the PKE scheme to guarantee that REnc queries of the above form do not help $\mathcal{A}$ to indirectly obtain $rk_{i^* \to j}$.

To show that the probability of the adversary $\mathcal{A}$ coming up with a ciphertext $\widehat{c}^*$ such that $\mathsf{Dec_1}(\mathsf{sk}_j, \widehat{c}^*) \neq m^*$ in case $\mathcal{A}$ has obtained $rk_{i^* \to j}$ (via a RKG query), we will use the decryption consistency of the re-splittable TPKE scheme. In doing so, as above we have to use the CCA security of the PKE scheme to guarantee that REnc queries do not help, and also to guarantee that the information of $tsk_{i^*.1}$ does not leak from a re-encryption key $rk_{i^* \to j}$ that is obtained by $\mathcal{A}$ through the RKG query that $\mathcal{A}$ issued. Finally, note that the decryption consistency is guaranteed only under an honestly generated verification key tvk_{i^*}, but $\mathcal{A}$ may have generated the ciphertext $\widehat{c}^*$ in such a way that tvk_{i^*} is generated maliciously by $\mathcal{A}$. To prevent it, we will again rely on the strong unforgeability of the signature scheme, which ensures that the only way to generate a valid re-encrypted ciphertext is to use a re-encryption key which is generated honestly (and thus tvk_{i^*} is also honestly generated).

Theorem 10. *If the PKE scheme is CCA secure, the signature scheme is strongly unforgeable, and the re-splittable TPKE scheme is CCA secure, then the VPRE scheme* eHKK^+ *is second-level CCA secure.*

Intuition. The formal proof is given in Section 4.4.2. Here, we explain an intuition of the proof of second-level CCA security. The proof follows closely to the proof of soundness, and the original security proof of the HKK^+ scheme [58]. More specifically, the difference is that we calculate the (differences of the) probabilities of $\mathcal{A}$ succeeding in guessing the challenge bit (instead of the event that an adversary succeeds in breaking the conditions of soundness). In the final game, we can show that there exists a PPT CCA adversary $\mathcal{B}$ against the re-splittable TPKE scheme such that its advantage $\mathsf{Adv}^{\mathtt{CCA\text{-}TPKE}}_{(\mathcal{B},n)}(k)$ is exactly the difference between the success probability of $\mathcal{A}$ in the final game and 1/2.

Theorem 11. *If the PKE scheme is CCA secure and the re-splittable TPKE scheme has strong smoothness, then the VPRE scheme* eHKK^+ *is first-level CCA secure.*

Intuition. The formal security proof is given in Section 4.4.3. Here, we explain an intuition of the proof of first-level CCA security. As shown in [58], a first-level ciphertext in the eHKK^+ scheme is wrapped entirely by the underlying PKE scheme (regarding $\widehat{pk}$), and thus its CCA security naturally leads to first-level CCA security, if it were not for re-encryption verification queries.

```
$\mathsf{MiniDec}(tpk, tvk, tsk_1, \mu_2, c)$:
  $\mu_1 \leftarrow \mathsf{TShDec}(tpk, tsk_1, c)$
  If $\mu_1 = \bot$ then return $\bot$.
  If $\mathsf{TShVer}(tpk, tvk, c, 2, \mu_2) = \bot$
    then return $\bot$.
  $m \leftarrow \mathsf{TCom}(tpk, tvk, c, \{\mu_1, \mu_2\})$
  Return $m$.
```

```
$\mathsf{MiniREncVer}(tpk, tvk, tsk_1, \mu_2, c, c')$:
  $\mu_1 \leftarrow \mathsf{TShDec}(tpk, tsk_1, c)$
  If $\mu_1 = \bot$ then return $\bot$.
  If $\mathsf{TShVer}(tpk, tvk, c, 2, \mu_2) = \bot$
    then return $\bot$.
  If $c' = c$ then return $\top$
    else return $\bot$.
```

Figure 4.2: The algorithms $\mathsf{MiniDec}$ (left) and $\mathsf{MiniREncVer}$ (right).

The main difference from the proof in [58] is that we need the strong smoothness of the underlying re-splittable TPKE scheme, which was not necessary in the original proof of [58], in order to deal with $\mathsf{REncVer}$ queries. More specifically, recall that in the first-level CCA security game, an adversary $\mathcal{A}$ can choose a key pair $(\mathsf{sk}_{\mathcal{A}}, \mathsf{pk}_{\mathcal{A}})$ for the second-level encryption of the challenge query. In particular, $\mathcal{A}$ can know $tsk_{\mathcal{A}}$. Now, suppose that this TPKE scheme has a "weak plaintext" m_w in the sense that it is easy to find given $tsk_{\mathcal{A}}$, and its encryption $c_w \leftarrow \mathsf{TEnc}(tpk_{\mathcal{A}}, m_w)$ is easy to guess. (Such a property does not contradict the CCA security of the TPKE scheme, because m_w could be hard to find without $tsk_{\mathcal{A}}$.) Then $\mathcal{A}$ can choose such m_w as one of the challenge plaintexts, submit it with $(\mathsf{sk}_{\mathcal{A}}, \mathsf{pk}_{\mathcal{A}})$ as a challenge query, and obtain the challenge ciphertext $\widehat{c}^*$. Then $\mathcal{A}$ by itself calculates the "easy-to-guess" ciphertext c_w corresponding to m_w, and submits a $\mathsf{REncVer}$ query $(\mathsf{pk}_{\mathcal{A}}, c_w, \widehat{c}^*)$, which by definition reveals the challenge bit (because its answer essentially tells whether "$\widehat{c}^*$ is a re-encryption of c_w"). However, if the underlying re-splittable TPKE scheme is guaranteed to have strong smoothness, such weak plaintexts cannot exist, and hence we can conclude that $\mathsf{REncVer}$ queries do not help $\mathcal{A}$.

4.4 Proof of Theorems

In this section, we prove the theorems in Section 4.3.

4.4.1 Proof of Theorem 9

Here, we state the proof of soundness in Theorem 9. For the security proofs of soundness and second-level CCA security, it is convenient to introduce the following algorithms $\mathsf{MiniDec}$ and $\mathsf{MiniREncVer}$:

$\mathsf{MiniDec}$ is the sub-procedure of Dec_1 that starts from the step "$\mu_1 \leftarrow \mathsf{TShDec}(tpk_i, tsk_{i.1}, c)$" of Dec_1. More specifically, it takes a TPKE public key tpk, a TPKE verification key tvk, a secret key share tsk_1, a decryption share μ_2, and a second-level ciphertext c, and runs as in Fig. 4.2(left).

$\mathsf{MiniREncVer}$ is the $\mathsf{REncVer}$-analogue of $\mathsf{MiniDec}$. Namely, this algorithm takes a tuple $(tpk, tvk, tsk_1, \mu_2, c)$, and another second-level ciphertext c' as input, and runs as in Fig. 4.2(right).

Let $n = n(k) > 0$ be any polynomial, and $\mathcal{A}$ be any PPT soundness adversary that attacks the soundness of the VPRE scheme eHKK^+. Consider the following sequence of games:

Game 0. This is the soundness game regarding eHKK^+. Since the subsequent games consider $\mathcal{A}$'s queries of some special types, without loss of generality we let the challenger generate two empty lists L^*_{RKG} and L^*_{REnc} at the beginning, and store the values that appear in the response to a re-encryption key generation query and a re-encryption query of special types. More concretely,

- If $\mathcal{A}$ issues a RKG query of the form $(\mathsf{pk}_{i^*}, \mathsf{pk}_j)$ with $\mathsf{pk}_j \in \mathcal{PK}$, then the challenger stores the values $(\mathsf{pk}_j, \psi, tvk_{i^*}, \sigma, tsk_{i^*.1})$ into L^*_{RKG}, where $(\psi, tvk_{i^*}, \sigma, tsk_{i^*.1})$ are the values generated when calculating $rk_{i^* \to j} \leftarrow \mathsf{RKG}(\mathsf{sk}_{i^*}, \mathsf{pk}_j)$.
- If $\mathcal{A}$ issues a REnc query of the form $(\mathsf{pk}_{i^*}, \mathsf{pk}_j, c)$ with $\mathsf{pk}_j \in \mathcal{PK}$ and the answer $\widehat{c}$ to this query is not $\perp$, then the challenger stores the values $(\mathsf{pk}_j, \widehat{c}, c, \mu_2, tvk_{i^*}, tsk_{i^*.1})$ into L^*_{REnc}, where $(\mu_2, tvk_{i^*}, tsk_{i^*.1})$ are the values generated when calculating $\widehat{c} \leftarrow \mathsf{REnc}(\mathsf{RKG}(\mathsf{sk}_{i^*}, \mathsf{pk}_j), c)$.

Game 1. Same as Game 0, except for the following changes to the response to $\mathsf{REncVer}$ queries and Dec_1 queries: For $\mathsf{REncVer}$ queries $(\mathsf{pk}_i, \mathsf{pk}_j, c', \widehat{c})$, the challenger responds as follows:

- (1) If $(\mathsf{pk}_j, \widehat{c}, c, \mu_2, tvk_{i^*}, tsk_{i^*.1}) \in L^*_{\mathsf{REnc}}$ for some $(c, \mu_2, tvk_{i^*}, tsk_{i^*.1})$, then:
  - (1a) If $\mathsf{pk}_i \neq \mathsf{pk}_{i^*}$, then return $\perp$.
  - (1b) Otherwise (i.e. $\mathsf{pk}_i = \mathsf{pk}_{i^*}$), run $\mathsf{MiniREncVer}(tpk_{i^*}, tvk_{i^*}, tsk_{i^*.1}, \mu_2, c, c')$ and return the result.
- (2) Otherwise (i.e. $(\mathsf{pk}_j, \widehat{c}, *, *, *, *) \notin L^*_{\mathsf{REnc}}$), run $\mathsf{REncVer}(\mathsf{pk}_i, \mathsf{sk}_j, c, \widehat{c})$ and return the result.

For Dec_1 queries $(\mathsf{pk}_j, \widehat{c})$, the challenger responds as follows:

- (1) If $(\mathsf{pk}_j, \widehat{c}, c, \mu_2, tvk_{i^*}, tsk_{i^*.1}) \in L^*_{\mathsf{REnc}}$ for some $(c, \mu_2, tvk_{i^*}, tsk_{i^*.1})$, then execute $m \leftarrow \mathsf{MiniDec}(tpk_{i^*}, tvk_{i^*}, tsk_{i^*.1}, \mu_2, c)$, and return m to $\mathcal{A}$.
- (2) Otherwise (i.e. $(\mathsf{pk}_j, \widehat{c}, *, *, *, *) \notin L^*_{\mathsf{REnc}}$, then execute $m \leftarrow \mathsf{Dec}_1(\mathsf{sk}_j, \widehat{c})$, and return $\perp$ to $\mathcal{A}$.

We would like to emphasize that from this game on, the challenger need not perform $\mathsf{PDec}(\widehat{dk}_j, \widehat{c})$ for a $\mathsf{REncVer}$ query $(*, \mathsf{pk}_j, *, \widehat{c})$ and a Dec_1 query $(\mathsf{pk}_j, \widehat{c})$ such that $(\mathsf{pk}_j, \widehat{c}, *, *, *, *) \in L^*_{\mathsf{REnc}}$.

Game 2. Same as Game 1, except that in this game, a re-encrypted ciphertext $\widehat{c}$ which is from the challenge key pk_{i^*} to an honest user key $\mathsf{pk}_j \in \mathcal{PK}$, is generated in such a way that $\widehat{c}$ contains no information.

More precisely, if $\mathcal{A}$ submits a REnc query of the form $(\mathsf{pk}_{i^*}, \mathsf{pk}_j, c)$ with $\mathsf{pk}_j \in \mathcal{PK}$, then the challenger responds as follows:

- (1) Compute $(tsk_{i^*.1}, tsk_{i^*.2}, tvk_{i^*}) \leftarrow \mathsf{TSplit}(tsk_{i^*})$.
- (2) Compute $\mu_2 \leftarrow \mathsf{TShDec}(tpk_{i^*}, tsk_{i^*.2}, c)$, and return $\perp$ to $\mathcal{A}$ if $\mu_2 = \perp$.
- (3) Compute $\widehat{c} \leftarrow \mathsf{PEnc}(\widehat{pk}_j, \langle \mathsf{pk}_{i^*} \| \mathsf{pk}_j \| \mathbf{0} \rangle)$ where $\mathbf{0}$ is the zero-string of appropriate length.

- (4) Return $\widehat{c}$ to $\mathcal{A}$ and store the values $(\mathsf{pk}_j, \widehat{c}, c, \mu_2, tvk_{i^*}, tsk_{i^*.1})$ into L^*_{REnc}, where $tsk_{i^*.1}$ is the secret key share corresponding to $(tsk_{i^*.2}, tvk_{i^*})$ that appears in the above step (1).

Game 3. Same as Game 2, except that in this game, if a re-encrypted ciphertext $\widehat{c}$ which is from the challenge key pk_{i^*} to an honest user key $\mathsf{pk}_j \in \mathcal{PK}$ is submitted as a $\mathsf{REncVer}$ query (with pk_j and some c) or as a Dec_1 query (with pk_j), then $\widehat{c}$ is immediately answered with $\perp$ unless (a) it is an answer to a previously asked REnc query of the form $(\mathsf{pk}_{i^*}, \mathsf{pk}_j, c)$, or (b) it is a re-encryption using a re-encryption key $rk_{i^* \to j}$ that is returned as an answer to a previously asked RKG query.

More precisely, in this game, the challenger responds to $\mathsf{REncVer}$ queries $(\mathsf{pk}_i, \mathsf{pk}_j, c', \widehat{c})$ as follows:

- (1) If $(\mathsf{pk}_j, \widehat{c}, *, *, *, *) \in L^*_{\mathsf{REnc}}$, then respond as in Game 1.
- (2) Run $\widehat{M} = (\mathsf{pk}'_i \| \mathsf{pk}'_j \| c \| \mu_2 \| \psi \| tvk_i \| \sigma) \leftarrow \mathsf{PDec}(\widehat{dk_j}, \widehat{c})$, and return $\perp$ to $\mathcal{A}$ if $\widehat{M} = \perp$, $(\mathsf{pk}'_i, \mathsf{pk}'_j) \neq (\mathsf{pk}_i, \mathsf{pk}_j)$, or $\mathsf{SVer}(vk_i, \langle \psi \| tvk_i \| \mathsf{pk}'_i \| \mathsf{pk}'_j \rangle, \sigma) = \perp$.
- (3) If $\mathsf{pk}_i \neq \mathsf{pk}_{i^*}$ then execute $\mathsf{REncVer}(\mathsf{pk}_i, \mathsf{sk}_j, c', \widehat{c})$, and return the result to $\mathcal{A}$.
- (4) If $\mathsf{pk}'_i = \mathsf{pk}_{i^*}$ and $(\mathsf{pk}_j, \psi, tvk_{i^*}, \sigma, *) \notin L^*_{\mathsf{RKG}}$, then return $\perp$ to $\mathcal{A}$.
- (5) Otherwise (i.e. $\mathsf{pk}'_i = \mathsf{pk}_{i^*}$ and $(\mathsf{pk}_j, \psi, tvk_{i^*}, \sigma, tsk_{i^*.1}) \in L^*_{\mathsf{RKG}}$ for some $tsk_{i^*.1}$), as in the above step (3), execute the remaining procedure of $\mathsf{REncVer}$, and output the result to $\mathcal{A}$.

Furthermore, the challenger responds to Dec_1 queries $(\mathsf{pk}_j, \widehat{c})$ as follows:

- (1) If $(\mathsf{pk}_j, \widehat{c}, *, *, *, *) \in L^*_{\mathsf{REnc}}$, then respond as in Game 1.
- (2) Run $\widehat{M} = (\mathsf{pk}'_i \| \mathsf{pk}'_j \| c \| \mu_2 \| \psi \| tvk_i \| \sigma) \leftarrow \mathsf{PDec}(\widehat{dk_j}, \widehat{c})$, and return $\perp$ to $\mathcal{A}$ if $\widehat{M} = \perp$ or $\mathsf{pk}'_j \neq \mathsf{pk}_j$.
- (3) Parse pk'_i as $(tpk_i, \widehat{pk}_i, pk_i, vk_i)$, and return $\perp$ if $\mathsf{SVer}(vk_i, \langle \psi \| tvk_i \| \mathsf{pk}'_i \| \mathsf{pk}'_j \rangle, \sigma)$ $= \perp$.
- (4) If $\mathsf{pk}'_i \neq \mathsf{pk}_{i^*}$ then calculate m by following the remaining procedure of Dec_1 (i.e. from the step "$tsk_{i.1} \leftarrow \mathsf{PDec}(dk_j, \psi)$"), and return m to $\mathcal{A}$.
- (5) If $\mathsf{pk}'_i = \mathsf{pk}_{i^*}$ and $(\mathsf{pk}_j, \psi, tvk_i, \sigma, *) \notin L^*_{\mathsf{RKG}}$, then return $\perp$ to $\mathcal{A}$.
- (6) Otherwise (i.e. $\mathsf{pk}'_i = \mathsf{pk}_{i^*}$ and $(\mathsf{pk}_j, \psi, tvk_i, \sigma, tsk_{i^*.1}) \in L^*_{\mathsf{RKG}}$ for some $tsk_{i^*.1}$), as in the above step (4), calculate m by following the remaining procedure of Dec_1.

Game 4. Same as Game 3, except that in this game, if $\mathcal{A}$ issues a $\mathsf{REncVer}$ query $(\mathsf{pk}_i, \mathsf{pk}_j, c', \widehat{c})$ or Dec_1 query $(\mathsf{pk}_j, \widehat{c})$, such that $\widehat{c}$ is a re-encrypted ciphertext from the challenge key pk_{i^*} to pk_j using a re-encryption key $rk_{i^* \to j}$ that is an answer to a previously asked RKG query of the form $(\mathsf{pk}_{i^*}, \mathsf{pk}_j)$ (which can be checked using L^*_{RKG} as in Game 3), then the query is answered using the information of $tsk_{i^*.1}$ found in L^*_{RKG}.

More precisely, in this game, the challenger responds to $\mathsf{REncVer}$ queries $(\mathsf{pk}_i, \mathsf{pk}_j, c', \widehat{c})$ as follows:

- (1), (2), (3), and (4): Same as in Game 3.
- (5): Here, it is guaranteed that $\mathsf{pk}'_i = \mathsf{pk}_{i^*}$ and $(\mathsf{pk}_j, \psi, tvk_{i^*}, \sigma, tsk_{i^*.1}) \in L^*_{\mathsf{RKG}}$ for some $tsk_{i^*.1}$. Execute $\mathsf{MiniREncVer}(tpk_{i^*}, tvk_{i^*}, tsk_{i^*.1}, \mu_2, c, c')$, and return the result to $\mathcal{A}$.

Furthermore, the challenger responds to the Dec_1 queries $(\mathsf{pk}_j, \widehat{c})$ in the following way:

- (1), (2), (3), (4), and (5): Same as in Game 3.
- (6): Here, it is guaranteed that $\mathsf{pk}'_i = \mathsf{pk}_{i^*}$ and $(\mathsf{pk}_j, \psi, tvk_{i^*}, \sigma, tsk_{i^*.1}) \in L^*_{\mathsf{RKG}}$ for some $tsk_{i^*.1}$. Run $m \leftarrow \mathsf{MiniDec}(tpk_{i^*}, tvk_{i^*}, tsk_{i^*.1}, \mu_2, c)$, and return the result to $\mathcal{A}$.

We would like to emphasize that from this game on, the challenger need not perform $\mathsf{PDec}(dk_j, \psi)$ for a $\mathsf{REncVer}$ query $(\mathsf{pk}_i, \mathsf{pk}_j, c', \widehat{c})$ and a Dec_1 query $(\mathsf{pk}_j, \widehat{c})$ that are processed at their steps (6), namely, those queries that satisfy $(\mathsf{pk}_j, \widehat{c}, *, *, *, *) \notin L^*_{\mathsf{REnc}}$, $\mathsf{PDec}(\widehat{dk}_j, \widehat{c}) = \langle \mathsf{pk}_{i^*} \| \mathsf{pk}_j \| c \| \mu_2 \| \psi \| tvk_{i^*} \| \sigma \rangle \neq \bot$, $\mathsf{SVer}(vk_{i^*}, \langle \psi \| tvk_{i^*} \| \mathsf{pk}_{i^*} \| \mathsf{pk}_j \rangle, \sigma) = \top$, and $(\mathsf{pk}_j, \psi, tvk_{i^*}, \sigma, *) \in L^*_{\mathsf{RKG}}$.

Game 5. Same as Game 4, except that in this game, if $\mathcal{A}$ issues a RKG query of the form $(\mathsf{pk}_{i^*}, \mathsf{pk}_j)$ with $\mathsf{pk}_j \in \mathcal{PK}$, then the component ψ in a re-encryption key $rk_{i^* \to j}$ is generated in such a way that it contains no information.

More precisely, for this query, the challenger generates $rk_{i^* \to j} = (\mathsf{pk}_{i^*}, \mathsf{pk}_j, tsk_{i^*.2}, \psi, tvk_{i^*}, \sigma)$ by following the procedure of $\mathsf{RKG}(\mathsf{sk}_{i^*}, \mathsf{pk}_j)$ except that ψ is generated by $\psi \leftarrow \mathsf{PEnc}(pk_j, 0^{|tsk_{i^*.1}|})$. Then the challenger returns $rk_{i^* \to j}$ to $\mathcal{A}$ and stores the values $(\mathsf{pk}_j, \psi, tvk_{i^*}, \sigma, tsk_{i^*.1})$ into L^*_{RKG}.

In Game 0 (i.e. the original soundness game), we define the event Win_0 as the event that $\mathcal{A}$ wins, i.e. the following conditions are satisfied: (where $(\mathsf{pk}_j, \widehat{c}^*)$ represents $\mathcal{A}$'s output)

(1) $\mathsf{REncVer}(\mathsf{pk}_{i^*}, \mathsf{sk}_j, c^*, \widehat{c}^*) = \top$

(2) $\widehat{c}^*$ is not an answer to some of $\mathcal{A}$'s REnc queries of the form $(\mathsf{pk}_{i^*}, \mathsf{pk}_j, c^*)$

(3) Either of the following conditions is satisfied:

- In the case that $\mathcal{A}$ has submitted a RKG query of the form $(\mathsf{pk}_{i^*}, \mathsf{pk}_j)$ and obtained a re-encryption key $rk_{i^* \to j}$: $\mathsf{Dec}_1(\mathsf{sk}_j, \widehat{c}^*) \neq m^*$
- Otherwise: $\mathsf{Dec}_1(\mathsf{sk}_j, \widehat{c}^*) \neq \bot$

Furthermore, for $i \in [5]$, we also define the event Win_i in Game i, in the same way as Win_0 except that the condition of "$\mathsf{REncVer}(\mathsf{pk}_{i^*}, \mathsf{sk}_j, c^*, \widehat{c}^*) = \top$" is replaced with "The response to the $\mathsf{REncVer}$ query $(\mathsf{pk}_{i^*}, \mathsf{pk}_j, c^*, \widehat{c}^*)$ in Game i is $\top$", and the condition "Dec_1 $(\mathsf{sk}_j, \widehat{c}^*) \neq m^*$" (resp. "$\mathsf{Dec}_1(\mathsf{sk}_j, \widehat{c}^*) \neq \bot$") is replaced with the condition "The response to the Dec_1 query $(\mathsf{pk}_j, \widehat{c}^*)$ is not m^* (resp. $\bot$)".

Finally, for $i \in \{0, \dots, 5\}$ let Ask_i be the event that $\mathcal{A}$ issues a RKG query of the form $(\mathsf{pk}_{i^*}, \mathsf{pk}_j)$ where pk_j is used as the output of $\mathcal{A}$ in the soundness game. (Whether this event has occurred is determined when $\mathcal{A}$ outputs $(\mathsf{pk}_j, \widehat{c}^*)$ and terminates.)

Note that by definition, for any $i \in \{0, \dots, 5\}$, we have

$$\Pr[\mathsf{Win}_i] = \Pr[\mathsf{Win}_i \wedge \overline{\mathsf{Ask}_i}] + \Pr[\mathsf{Win}_i \wedge \mathsf{Ask}_i].$$

The soundness advantage of $\mathcal{A}$ is, by definition, $\mathsf{Adv}^{\mathtt{SND\text{-}VPRE}}_{(\mathcal{A},n)}(k) = \Pr[\mathsf{Win}_0]$. By the above equation and the triangle inequality, we have:

$$\mathsf{Adv}^{\mathtt{SND\text{-}VPRE}}_{(\mathcal{A},n)}(k) \leq \sum_{i \in \{0,1,2\}} |\Pr[\mathsf{Win}_i] - \Pr[\mathsf{Win}_{i+1}]| + \Pr[\mathsf{Win}_3 \wedge \overline{\mathsf{Ask}_3}]$$
$$+ \sum_{i \in \{3,4\}} |\Pr[\mathsf{Win}_i \wedge \mathsf{Ask}_i] - \Pr[\mathsf{Win}_{i+1} \wedge \mathsf{Ask}_{i+1}]| + \Pr[\mathsf{Win}_5 \wedge \mathsf{Ask}_5]. \quad (4.1)$$

We complete the proof by upperbounding each term in the right-hand side of the above inequality to be negligible.

Lemma 1. $\Pr[\mathsf{Win}_0] = \Pr[\mathsf{Win}_1]$.

Proof of Lemma 1. Note that the difference between Game 0 and Game 1 is only in how the challenger responds to a $\mathsf{REncVer}$ query $(\mathsf{pk}_i, \mathsf{pk}_j, c', \widehat{c})$ and a Dec_1 query $(\mathsf{pk}_j, \widehat{c})$ such that there is an entry $(\mathsf{pk}_j, \widehat{c}, c, \mu_2, tvk_{i^*}, tsk_{i^*.1})$ for some $(c, \mu_2, tvk_{i^*}, tsk_{i^*.1})$ in the list L^*_{REnc}. Recall that according the definition of Game 0 (and Game 1), the values $(\mathsf{pk}_j, \widehat{c}, c, \mu_2, tvk_{i^*}, tsk_{i^*.1})$ are stored into the list L^*_{REnc} if and only if $\mathcal{A}$ makes a re-encryption query of the form $(\mathsf{pk}_{i^*}, \mathsf{pk}_j, c)$ satisfying (1) $\mathsf{pk}_j \in \mathcal{PK}$ and (2) $\mathsf{TShDec}(tpk_{i^*}, tsk_{i^*.2}, c) = \mu_2 \neq \perp$ where $tsk_{i^*.2}$ is the secret key share that is generated for answering the re-encryption query.

Therefore, it is sufficient to show that the answer to the $\mathsf{REncVer}$ query $(\mathsf{pk}_i, \mathsf{pk}_j, c', \widehat{c})$ and a Dec_1 query of the above type in Game 0 and that in Game 1 are always the same, where $\widehat{c}$ is an output of $\mathsf{REnc}(\mathsf{RKG}(\mathsf{sk}_{i^*}, \mathsf{pk}_j), c)$, $\mathsf{pk}_j \in \mathcal{PK}$, and c satisfies the above condition (2).

Firstly, we consider how a $\mathsf{REncVer}$ query $(\mathsf{pk}_i, \mathsf{pk}_j, c', \widehat{c})$ is answered in both Game 0 and Game 1. We know that $\widehat{c}$ is a correctly generated re-encrypted ciphertext, and thus it holds that $\mathsf{PDec}(\widehat{dk_j}, \widehat{c}) = \langle \mathsf{pk}_{i^*} \| \mathsf{pk}_j \| c \| \mu_2 \| \psi \| tvk_{i^*} \| \sigma \rangle$, $\mathsf{SVer}(vk_{i^*}, \langle \psi \| tvk_{i^*} \| \mathsf{pk}_{i^*} \| \mathsf{pk}_j \rangle, \sigma) = \top$, and $\mathsf{PDec}(dk_j, \psi) = tsk_{i^*.1} \neq \perp$, due to the correctness of the building block PKE scheme and signature scheme. If $\mathsf{pk}_i \neq \mathsf{pk}_{i^*}$, then by definition the query is answered with $\perp$ in Game 1. This is also the case in Game 0, because we know that $\widehat{c}$ contains pk_{i^*} in its plaintext, and thus the check performed in the sixth line in the $\mathsf{REncVer}$ algorithm cannot be passed. Otherwise (i.e. $\mathsf{pk}_i = \mathsf{pk}_{i^*}$), note that $tsk_{i^*.1}$ recovered from ψ and $tsk_{i^*.1}$ in the entry corresponding to $(\mathsf{pk}_j, \widehat{c})$ in L^*_{REnc} are identical (due to the correctness of the PKE scheme), and thus the procedure of $\mathsf{REncVer}(\mathsf{pk}_{i^*}, \mathsf{pk}_j, c', \widehat{c})$ after the step "$tsk_{i^*.1} \leftarrow \mathsf{PDec}(dk_j, \psi)$" and the procedure of $\mathsf{MiniREncVer}(tpk_{i^*}, tvk_{i^*}, tsk_{i^*.1}, \mu_2, c, c')$ are exactly the same. The explanation here implies that the result of a $\mathsf{REncVer}$ query in Game 0 and that in Game 1 agree.

With a very similar observation to the above, we can also show that the result of a Dec_1 query $(\mathsf{pk}_j, \widehat{c})$ in Game 0 and that in Game 1 agree. Specifically, the procedure of $\mathsf{Dec}_1(\mathsf{sk}_j, \widehat{c})$ after the step "$tsk_{i^*.1} \leftarrow \mathsf{PDec}(dk_j, \psi)$" and the procedure of $\mathsf{MiniDec}(tpk_{i^*}, tvk_{i^*}, tsk_{i^*.1}, \mu, c)$ are exactly the same, and thus the result of a Dec_1 query in Game 0 and that in Game 1 agree.

We have seen that the answer to a $\mathsf{REncVer}$ query and that of Dec_1 query agree in both Game 0 and Game 1. This completes the proof of Lemma 1. $\square$

Lemma 2. *If the PKE scheme is CCA secure in the multi-user setting,* $|\Pr[\mathsf{Win}_1] - \Pr[\mathsf{Win}_2]|$ *is negligible.*

Proof of Lemma 2. We show that we can construct a multi-user CCA adversary $\mathcal{B}$ against the underlying PKE scheme such that $\mathsf{Adv}^{\mathtt{CCA\text{-}PKE}}_{(\mathcal{B},n)}(k) = |\Pr[\mathsf{Win}_1]-\Pr[\mathsf{Win}_2]|$. By the multi-user CCA security of the underlying PKE scheme (which is equivalent to the ordinary CCA security), the above implies that $|\Pr[\mathsf{Win}_1] - \Pr[\mathsf{Win}_2]|$ is negligible, which proves the lemma. The description of $\mathcal{B}$ is as follows:

First, $\mathcal{B}$ is given 1^k and public keys $(\widehat{pk}_1, \dots, \widehat{pk}_n)$ from the challenger. Then $\mathcal{B}$ generates other key materials of the honest users (except $\{\widehat{dk}_i\}_{i\in[n]}$) as well as the challenge key pair $(\mathsf{sk}_{i^*}, \mathsf{pk}_{i^*}) \leftarrow \mathsf{KG}(1^k)$. $\mathcal{B}$ then sets $\mathcal{PK} = \{\mathsf{pk}_i\}_{i\in[n]}$ and $\mathcal{PK}^* = \{\mathsf{pk}_{i^*}\} \cup \mathcal{PK}$, and gives 1^k and $\mathcal{PK}^*$ to $\mathcal{A}$. $\mathcal{B}$ also generates an empty list L^*_{REnc}. (Since the list L^*_{RKG} does not play any role in Game 1 and Game 2, $\mathcal{B}$ need not generate it.)

When $\mathcal{A}$ makes a REnc query $(\mathsf{pk}_i \in \mathcal{PK}^*, \mathsf{pk}_j, c)$, $\mathcal{B}$ responds as follows: (Recall that $\mathcal{B}$ does not need the knowledge of $\{\widehat{dk}_i\}_{i\in[n]}$ for answering to re-encryption queries.) (1) If $\mathsf{pk}_i \neq \mathsf{pk}_{i^*}$ or $\mathsf{pk}_j \notin \mathcal{PK}$, then $\mathcal{B}$ calculates $\widehat{c} \leftarrow \mathsf{REnc}(\mathsf{RKG}(\mathsf{sk}_i, \mathsf{pk}_j), c)$, and returns $\widehat{c}$ to $\mathcal{A}$. (2) Otherwise (i.e. $\mathsf{pk}_i = \mathsf{pk}_{i^*}$ and $\mathsf{pk}_j \in \mathcal{PK}$), $\mathcal{B}$ proceeds as follows:

(2a) Execute $(tsk_{i^*.1}, tsk_{i^*.2}, tvk_{i^*}) \leftarrow \mathsf{TSplit}(tsk_{i^*})$ and $\mu_2 \leftarrow \mathsf{TShDec}(tpk_{i^*}, tsk_{i^*.2}, c)$, and return $\perp$ to $\mathcal{A}$ if $\mu_2 = \perp$.

(2b) Execute $\psi \leftarrow \mathsf{PEnc}(pk_j, tsk_{i^*.1})$ and $\sigma \leftarrow \mathsf{Sign}(sk_{i^*}, \langle\psi\|tvk_{i^*}\|\mathsf{pk}_{i^*}\|\mathsf{pk}_j\rangle)$.

(2c) Set $M_0 = \langle \mathsf{pk}_{i^*}\|\mathsf{pk}_j\|c\|\mu_2\|\psi\|tvk_{i^*}\|\sigma\rangle$ and $M_1 = \langle \mathsf{pk}_{i^*}\|\mathsf{pk}_j\|\mathbf{0}\rangle$, where $\mathbf{0}$ is the zero-string such that it holds that $|M_0| = |M_1|$. Then, submit (j, M_0, M_1) as an LR query to the challenger, and receive $\widehat{c} \leftarrow \mathsf{PEnc}(\widehat{pk}_j, M_b)$ as the response (where b is the challenge bit for $\mathcal{B}$).

(2d) Return $\widehat{c}$ to $\mathcal{A}$, and store the values $(\mathsf{pk}_j, \widehat{c}, c, \mu_2, tvk_{i^*}, tsk_{i^*.1})$ into the list L^*_{REnc}.

$\mathcal{B}$ answers to all other queries from $\mathcal{A}$ in the same way as the challenger in Game 1 (and thus as in Game 2) does. This is possible because $\mathcal{B}$ holds all secret key materials except $\{\widehat{dk}_i\}_{i\in[n]}$, and when $\mathcal{B}$ needs to run $\mathsf{PDec}(\widehat{dk}_j, \widehat{c})$ with $j \in [n]$, $\mathcal{B}$ submits a decryption query $(j, \widehat{c})$ to the challenger, and uses the received result. Here, as described in the description of Game 1, $\mathcal{B}$ need not perform $\mathsf{PDec}(\widehat{dk}_j, \widehat{c})$ such that $\widehat{c}$ is obtained as a response to some of $\mathcal{B}$'s LR queries. (Such queries will be answered using the list L^*_{REnc}, as described in Game 1.)

Finally, when $\mathcal{A}$ terminates with output $(\mathsf{pk}_j \in \mathcal{PK}, \widehat{c}^*)$, $\mathcal{B}$ proceeds as follows. If $\widehat{c}^*$ is an answer to some of $\mathcal{A}$'s REnc queries of the form $(\mathsf{pk}_{i^*}, \mathsf{pk}_j, c^*)$, then $\mathcal{B}$ outputs 0 and terminates. Otherwise, it is guaranteed that $\widehat{c}^*$ is different from any answer to LR queries that $\mathcal{B}$ receives as an answer to its LR queries. $\mathcal{B}$ checks whether the pair $(\mathsf{pk}_j, \widehat{c}^*)$ satisfies the winning condition of Win_1 (which is the same as Win_2) by simulating the response to the $\mathsf{REncVer}$ query $(\mathsf{pk}_{i^*}, \mathsf{pk}_j, c^*, \widehat{c}^*)$ and the response to the Dec_1 query $(\mathsf{pk}_{i^*}, \widehat{c}^*)$ by itself. If this is the case, then $\mathcal{B}$ outputs 1, otherwise outputs 0, and terminates.

The above completes the description of $\mathcal{B}$. Note that $\mathcal{B}$ submits a LR query of the form (j, M_0, M_1) only if $\mathcal{A}$ submits a REnc query of the form $(\mathsf{pk}_{i^*}, \mathsf{pk}_j, c)$ satisfying $\mathsf{pk}_j \in \mathcal{PK}$ and $\mathsf{TShDec}(tpk_{i^*}, tsk_{i^*.2}, c) \neq \perp$. Note also that $\mathcal{B}$ never submits a decryption query $(j, \widehat{c})$ such that $\widehat{c}$ is an answer to some of $\mathcal{B}$'s LR queries of the form (j, M_0, M_1) (with the same j).

Let b be $\mathcal{B}$'s challenge bit. Let $\mathsf{Win}_{\mathcal{B}}$ be the event that $\mathcal{A}$'s output $(\mathsf{pk}_j, \widehat{c}^*)$ satisfies the three conditions of Win_1 (which is the same as Win_2) in the experiment simulated by $\mathcal{B}$. The multi-user CCA advantage of $\mathcal{B}$ can be calculated as follows:

$$\begin{aligned}\mathsf{Adv}^{\mathtt{CCA\text{-}PKE}}_{(\mathcal{B},n)}(k) &= |\Pr[b = b'] - \frac{1}{2}| \\ &= \frac{1}{2}|\Pr[b' = 1|b = 0] - \Pr[b' = 1|b = 1]| \\ &= \frac{1}{2}|\Pr[\mathsf{Win}_{\mathcal{B}}|b = 0] - \Pr[\mathsf{Win}_{\mathcal{B}}|b = 1]|\end{aligned}$$

Now, consider the case when $b = 0$. In this case, a re-encrypted ciphertext $\widehat{c}$ from the challenge public key pk_{i^*} to a honest user key $\mathsf{pk}_j \in \mathcal{PK}$ is generated as in Game 1. Moreover, it is easy to see that all the other values are calculated as in Game 1. Therefore, $\mathcal{B}$ simulates Game 1 perfectly for $\mathcal{A}$. Under this situation, the probability that the event $\mathsf{Win}_{\mathcal{B}}$ occurs is exactly the same as the probability that Win_1 occurs in Game 1. That is, $\Pr[\mathsf{Win}_{\mathcal{B}}|b = 0] = \Pr[\mathsf{Win}_1]$.

On the other hand, when $b = 1$, a re-encrypted ciphertext $\widehat{c}$ from pk_{i^*} to $\mathsf{pk}_j \in \mathcal{PK}$ is an encryption of $\langle \mathsf{pk}_{i^*} \| \mathsf{pk}_j \| \mathbf{0} \rangle$, where $\mathbf{0}$ is the zero-string of appropriate length, which is exactly how it is generated in Game 2. Since this is the only difference from the case $b = 0$, with a similar argument to the above we have $\Pr[\mathsf{Win}_{\mathcal{B}}|b = 1] = \Pr[\mathsf{Win}_2]$.

In summary we have $\mathsf{Adv}^{\mathtt{CCA\text{-}PKE}}_{(\mathcal{B},n)}(k) = \frac{1}{2}|\Pr[\mathsf{Win}_1] - \Pr[\mathsf{Win}_2]|$, as required. This completes the proof of Lemma 2. $\square$

Lemma 3. *If the signature scheme is strongly unforgeable,* $|\Pr[\mathsf{Win}_2] - \Pr[\mathsf{Win}_3]|$ *is negligible.*

Proof of Lemma 3. For $i \in \{2, 3\}$, let Forge_i be the event that in Game i, $\mathcal{A}$ submits at least one $\mathsf{REncVer}$ query of the form $(\mathsf{pk}_{i^*}, \mathsf{pk}_j, c', \widehat{c})$ or at least one Dec_1 query $(\mathsf{pk}_j, \widehat{c})$ satisfying the following conditions:

(a) $(\mathsf{pk}_j, \widehat{c}, *, *, *, *) \notin L^*_{\mathsf{REnc}}$

(b) $\mathsf{PDec}(\widehat{dk}_j, \widehat{c}) = \langle \mathsf{pk}_{i^*} \| \mathsf{pk}_j \| c \| \mu_2 \| \psi \| tvk_{i^*} \| \sigma \rangle \neq \perp$

(c) $(\mathsf{pk}_j, \psi, tvk_{i^*}, \sigma, *) \notin L^*_{\mathsf{RKG}}$

(d) $\mathsf{SVer}(vk_{i^*}, \langle \psi \| tvk_{i^*} \| \mathsf{pk}_{i^*} \| \mathsf{pk}_j \rangle, \sigma) = \top$.

Game 2 and Game 3 proceed identically until the event Forge_2 or Forge_3 occurs in the corresponding games. Therefore we have $|\Pr[\mathsf{Win}_2] - \Pr[\mathsf{Win}_3]| \leq \Pr[\mathsf{Forge}_2] = \Pr[\mathsf{Forge}_3]$.

Now, we show that we can construct another adversary $\mathcal{B}$ against the strong unforgeability of the underlying signature scheme such that $\mathsf{Adv}^{\mathtt{SUF\text{-}SIG}}_{\mathcal{B}}(k) \geq \Pr[\mathsf{Forge}_3]$. By the strong unforgeability of the signature scheme, the above inequation implies that $\Pr[\mathsf{Forge}_3]$

is negligible, which in turn implies that $|\Pr[\mathsf{Win}_2]-\Pr[\mathsf{Win}_3]|$ is negligible, and thus proves the lemma. The description of $\mathcal{B}$ is as follows.

First, $\mathcal{B}$ is given 1^k and a verification key vk_{i^*} from the challenger. $\mathcal{B}$ generates other key materials of the the honest users' keys $\mathcal{PK}$ and the challenge key pair $(\mathsf{sk}_{i^*}, \mathsf{pk}_{i^*})$ except the signing key sk_{i^*} corresponding to vk_{i^*}. Then, $\mathcal{B}$ sets $\mathcal{PK} = \{\mathsf{pk}_i\}_{i \in [n]}$ and $\mathcal{PK}^* = \{\mathsf{pk}_{i^*}\} \cup \mathcal{PK}$, and gives 1^k and $\mathcal{PK}^*$ to $\mathcal{A}$. $\mathcal{B}$ also generates two empty lists L^*_{RKG} and L^*_{REnc}.

$\mathcal{B}$ answers to $\mathcal{A}$'s queries exactly as in Game 3. This is possible because $\mathcal{B}$ holds all key materials except the signing key sk_{i^*} corresponding to vk_{i^*}, and whenever $\mathcal{B}$ has to compute $\sigma \leftarrow \mathsf{Sign}(sk_{i^*}, \langle \psi \| tvk_{i^*} \| \mathsf{pk}_{i^*} \| \mathsf{pk}_j \rangle)$ (for answering RKG or REnc queries) $\mathcal{B}$ submits $\langle \psi \| tvk_{i^*} \| \mathsf{pk}_{i^*} \| \mathsf{pk}_j \rangle$ as a signing query to the challenger and uses the obtained result σ.

When $\mathcal{A}$ terminates with output $(pk_j, \widehat{c}^*)$, from the $\mathsf{REncVer}$ queries of the form $(\mathsf{pk}_{i^*}, \mathsf{pk}_j, c', \widehat{c})$ and the Dec_1 queries $(\mathsf{pk}_j, \widehat{c})$ made by $\mathcal{A}$, $\mathcal{B}$ tries to find a query that satisfies the conditions (a) to (d) satisfied by a query that causes the event Forge_3. Namely: (a) $(\mathsf{pk}_j, \widehat{c}_j, *, *, *, *) \notin L^*_{\mathsf{REnc}}$, (b) $\mathsf{PDec}(\widehat{dk}_j, \widehat{c}) = \langle \mathsf{pk}_{i^*} \| \mathsf{pk}_j \| c \| \mu_2 \| \psi \| tvk_{i^*} \| \sigma \rangle \neq \bot$, (c) $(\mathsf{pk}_j, \psi, tvk_{i^*}, \sigma, *) \notin L^*_{\mathsf{RKG}}$, and (d) $\mathsf{SVer}(vk_{i^*}, \langle \psi \| tvk_{i^*} \| \mathsf{pk}_{i^*} \| \mathsf{pk}_j \rangle, \sigma) = \top$. If such a $\mathsf{REncVer}$ query $(\mathsf{pk}_{i^*}, \mathsf{pk}_j, c, \widehat{c})$ or a Dec_1 query $(\mathsf{pk}_j, \widehat{c})$ is found, then $\mathcal{B}$ terminates with output the message $\langle \psi \| tvk_{i^*} \| \mathsf{pk}_{i^*} \| \mathsf{pk}_j \rangle$ and the signature σ as a forgery pair, where these values are the ones that appear in the plaintext of $\widehat{c}$ (decrypted using PDec). If there is no such query, then $\mathcal{B}$ simply gives up and aborts.

The above completes the description of $\mathcal{B}$. It is not hard to see that $\mathcal{B}$ simulates Game 3 perfectly for $\mathcal{A}$. We note that $\mathcal{B}$ submits a signing query when $\mathcal{A}$ submits a RKG query of the form $(\mathsf{pk}_{i^*}, \mathsf{pk}_j)$ or when $\mathcal{A}$ submits a REnc query of the form $(\mathsf{pk}_{i^*}, \mathsf{pk}_j, c)$ with $\mathsf{pk}_j \notin \mathcal{PK}$.

Now, we argue that whenever $\mathcal{A}$ submits a $\mathsf{REncVer}$ query $(\mathsf{pk}_{i^*}, \mathsf{pk}_j, c', \widehat{c})$ or a Dec_1 query $(\mathsf{pk}_j, \widehat{c})$ satisfying the above conditions (a) to (d), $\mathcal{B}$ breaks the strong unforgeability of the building block signature scheme: In order for $\mathcal{B}$ to break the strong unforgeability, $\mathcal{B}$ has to come up with a valid message-signature pair that has not appeared in $\mathcal{B}$'s signing query/answer pairs. Here, the condition (d) guarantees that the message-signature pair $(\langle \psi \| tvk_{i^*} \| \mathsf{pk}_{i^*} \| \mathsf{pk}_j \rangle, \sigma)$ is valid, and the condition (c) guarantees that the message-signature pair output by $\mathcal{B}$ is different from all signing query/answer pairs made/received by $\mathcal{B}$. (Here, we note that although the list L^*_{RKG} does not contain the signing queries (and the answers to them) that are made for generating a re-encryption key $rk_{i^* \to j'}$ for a corrupted user j' (which is generated during the response to REnc queries $(\mathsf{pk}_{i^*}, \mathsf{pk}_{j'}, c)$), these signing queries $M = \langle \psi \| tvk_{i^*} \| \mathsf{pk}_{i^*} \| \mathsf{pk}_{j'} \rangle$ always satisfy $\mathsf{pk}_{j'} \notin \mathcal{PK}$ and thus as a message-signature pair, the pair finally output by $\mathcal{B}$ will always be different from signing queries/answers of this type.)

This means $\mathcal{B}$'s strong unforgeability advantage is at least the probability $\Pr[\mathsf{Forge}_3]$, as required. This completes the proof of Lemma 3. $\square$

Lemma 4. $\Pr[\mathsf{Win}_3 \wedge \overline{\mathsf{Ask}_3}] = 0$.

Proof of Lemma 4. Note that the conditions of the event $\mathsf{Win}_3 \wedge \overline{\mathsf{Ask}_3}$ implies that for $\mathcal{A}$'s output $(\mathsf{pk}_j, \widehat{c}^*)$ in Game 3, (among other conditions) the answer to Dec_1 query of the form $(\mathsf{pk}_j, \widehat{c}^*)$ is not $\bot$. This in turn implies that (1) $\mathsf{PDec}(\widehat{dk}_j, \widehat{c}^*) = \langle \mathsf{pk}_{i^*} \| \mathsf{pk}_j \| c \| \mu_2 \| \psi \| tvk_i \| \sigma \rangle$

for some $(c, \mu_2, \psi, tvk_i, \sigma)$, and (2) $(\mathsf{pk}_j, \psi, tvk_i, \sigma, *) \in L^*_{\mathsf{RKG}}$.

However, if $\overline{\mathsf{Ask}_3}$ occurs, the above (1) and (2) cannot be satisfied simultaneously, because $\mathcal{A}$ has not issued a RKG query of the form $(\mathsf{pk}_{i^*}, \mathsf{pk}_j)$, and thus there is no entry of the form $(\mathsf{pk}_j, *, *, *, *)$ in L^*_{RKG}. This completes the proof of Lemma 4. □

Lemma 5. $\Pr[\mathsf{Win}_3 \wedge \mathsf{Ask}_3] = \Pr[\mathsf{Win}_4 \wedge \mathsf{Ask}_4]$.

Proof of Lemma 5. Notice that the difference between Game 3 and Game 4 is only in how a $\mathsf{REncVer}$ query $(\mathsf{pk}_{i^*}, \mathsf{pk}_j, c', c)$ and a Dec_1 query $(\mathsf{pk}_j, \widehat{c})$ satisfying the following conditions:

(a) $(\mathsf{pk}_j, \widehat{c}, *, *, *, *) \notin L^*_{\mathsf{REnc}}$

(b) $\mathsf{PDec}(\widehat{dk}_j, \widehat{c}) = \langle \mathsf{pk}_{i^*} \| \mathsf{pk}_j \| c \| \mu_2 \| \psi \| tvk_{i^*} \| \sigma \rangle$

(c) $(\mathsf{pk}_j, \psi, tvk_{i^*}, \sigma, *) \in L^*_{\mathsf{RKG}}$

are answered. More concretely, the difference in these games is only in whether the challenger checks the signature σ and decrypt ψ that appear in the plaintext of $\widehat{c}$ as is done in $\mathsf{REncVer}(\mathsf{pk}_{i^*}, \mathsf{sk}_j, c', \widehat{c})$ and $\mathsf{Dec}_1(\mathsf{sk}_j, \widehat{c})$, or ignores ψ and σ and just decrypt the second level ciphertext c using the values $(tvk_{i^*}, tsk_{i^*.1})$ found in L^*_{RKG}. However, with the similar argument in the proof of Lemma 1, the results in Game 3 and Game 4 always agree, due to the correctness of the underlying PKE scheme and the underlying signature scheme.

Therefore, Game 3 and Game 4 are identical, which in particular implies the Lemma 5. □

Lemma 6. *If the PKE scheme is CCA secure in the multi-user setting,* $|\Pr[\mathsf{Win}_4 \wedge \mathsf{Ask}_4] - \Pr[\mathsf{Win}_5 \wedge \mathsf{Ask}_5]|$ *is negligible.*

Proof of Lemma 6. We show that we can construct a multi-user CCA adversary $\mathcal{B}$ (against the underlying PKE scheme) such that $\mathsf{Adv}^{\mathtt{CCA\text{-}PKE}}_{(\mathcal{B},n)}(k) = |\Pr[\mathsf{Win}_4 \wedge \mathsf{Ask}_4] - \Pr[\mathsf{Win}_5 \wedge \mathsf{Ask}_5]|$. By the multi-user CCA security of the underlying PKE scheme (which is equivalent to the ordinary CCA security), the above implies that $|\Pr[\mathsf{Win}_4 \wedge \mathsf{Ask}_4] - \Pr[\mathsf{Win}_5 \wedge \mathsf{Ask}_5]|$ is negligible, which proves the lemma. The description of $\mathcal{B}$ is as follows:

First, $\mathcal{B}$ is given 1^k and public keys $(pk_1, \ldots, pk_n)$ from the challenger. $\mathcal{B}$ generates other key materials of the challenge key pk_{i^*} and the honest users' keys $\mathcal{PK}$ except $\{dk_i\}_{i \in [n]}$. $\mathcal{B}$ then gives 1^k and $\mathcal{PK}^* = \{\mathsf{pk}_{i^*}\} \cup \mathcal{PK}$ to $\mathcal{A}$. $\mathcal{B}$ also generates two empty lists L^*_{RKG} and L^*_{REnc}.

When $\mathcal{A}$ makes a RKG query $(\mathsf{pk}_i \in \mathcal{PK}^*, \mathsf{pk}_j)$, $\mathcal{B}$ responds as follows: (Recall that $\mathcal{B}$ does not need the knowledge of $\{dk_i\}_{i \in [n]}$ for answering to RKG queries.) (1) If $\mathsf{pk}_i \neq \mathsf{pk}_{i^*}$ then $\mathcal{B}$ calculates $rk_{i^* \to j}$ by faithfully following the procedure of $\mathsf{RKG}(\mathsf{sk}_i, \mathsf{pk}_j)$, and returns $rk_{i^* \to j}$ to $\mathcal{A}$. (2) If $\mathsf{pk}_i = \mathsf{pk}_{i^*}$ and $\mathsf{pk}_j \notin \mathcal{PK}$, then $\mathcal{B}$ returns $\perp$ to $\mathcal{A}$. (3) Otherwise (i.e. $\mathsf{pk}_i = \mathsf{pk}_{i^*}$ and $\mathsf{pk}_j \in \mathcal{PK}$), $\mathcal{B}$ proceeds as follows:

(3a) Execute $(tsk_{i^*.1}, tsk_{i^*.2}, tvk_{i^*}) \leftarrow \mathsf{TSplit}(tsk_{i^*})$.

(3b) Set $M_0 = tsk_{i^*.1}$ and $M_1 = 0^{|tsk_{i^*.1}|}$, submit (j, M_0, M_1) as a LR query to the challenger, and receive $\psi \leftarrow \mathsf{PEnc}(pk_j, M_b)$ as the response (where b is the challenge bit for $\mathcal{B}$).

(3c) Compute $\sigma \leftarrow \mathsf{Sign}(sk_{i^*}, \langle \psi \| tvk_{i^*} \| \mathsf{pk}_{i^*} \| \mathsf{pk}_j \rangle)$ and set $rk_{i^* \to j} = (\mathsf{pk}_{i^*}, \mathsf{pk}_{j^*}, tsk_{i^*.2}, \psi, tvk_{i^*}, \sigma)$.

(3d) Return $rk_{i^* \to j}$ to $\mathcal{A}$, and store the values $(\mathsf{pk}_j, \psi, tvk_i, \sigma, tsk_{i^*.1})$ into the list L_{RKG}.

$\mathcal{B}$ answers to all other queries from $\mathcal{A}$ in the same way as the challenger in Game 4 (and thus as in Game 5) does. This is possible because $\mathcal{B}$ holds all secret key materials except $\{dk\}_{i \in [n]}$, and when $\mathcal{B}$ needs to run $\mathsf{PDec}(dk_j, \psi)$ with $j \in [n]$, $\mathcal{B}$ submits a decryption query (j, ψ) to the challenger, and uses the received result. As emphasized in the description of Game 4, $\mathcal{B}$ need not perform $\mathsf{PDec}(dk_j, \psi)$ for ψ that it receives as an answer to some of $\mathcal{B}$'s LR queries. (Such a case is dealt with by the use of the list L^*_{RKG}.)

Finally, when $\mathcal{A}$ terminates with output $(\mathsf{pk}_j \in \mathcal{PK}, \widehat{c}^*)$, $\mathcal{B}$ proceeds as follows. If $\widehat{c}^*$ is an answer to some of $\mathcal{A}$'s REnc queries of the form $(\mathsf{pk}_{i^*}, \mathsf{pk}_j, c^*)$, or $\mathcal{A}$ has not asked a RKG query of the form $(\mathsf{pk}_{i^*}, \mathsf{pk}_j)$, then $\mathcal{B}$ outputs 0 and terminates. Otherwise, $\mathcal{B}$ simulates the response to the $\mathsf{REncVer}$ query $(\mathsf{pk}_{i^*}, \mathsf{pk}_j, c^*, \widehat{c}^*)$ and the response to the $\mathsf{Dec_1}$ query $(\mathsf{pk}_{i^*}, \widehat{c}^*)$ by itself, and checks whether the pair $(\mathsf{pk}_j, \widehat{c}^*)$ satisfies the winning condition of Win_4 (which is the same as Win_5) for the case that $\mathcal{A}$ has asked the RKG query $(\mathsf{pk}_{i^*} \mathsf{pk}_j)$ (i.e. whether the result of the $\mathsf{REncVer}$ query $(\mathsf{pk}_{i^*}, \mathsf{pk}_j, c^*, \widehat{c}^*)$ is $\top$ and the $\mathsf{Dec_1}$ query $(\mathsf{pk}_j, \widehat{c}^*)$ is different from m^*). It this is the case, then $\mathcal{B}$ outputs 1, otherwise output 0, and terminates.

The above completes the description of $\mathcal{B}$. Note that $\mathcal{B}$ submits an LR query of the form $(j, M_0 = tsk_{i^*.1}, M_1 = 0^{|tsk_{i^*.1}|})$ only when $\mathcal{A}$ submits a RKG query of the form $(\mathsf{pk}_{i^*}, \mathsf{pk}_j)$ with $\mathsf{pk}_j \in \mathcal{PK}$. Moreover, note also that all the ciphertexts ψ that $\mathcal{B}$ receives as an answer to a LR query of the form (j, M_0, M_1) are stored into L^*_{RKG}, and all the $\mathsf{REncVer}$ queries $(\mathsf{pk}_{i^*}, \mathsf{pk}_j, c, \widehat{c})$ and all the $\mathsf{Dec_1}$ queries $(\mathsf{pk}_j, \widehat{c})$ such that the plaintext of $\widehat{c}$ contains ψ that appears in L^*_{RKG} are answered with either $\perp$ or using $\mathsf{MiniREncVer}$ and $\mathsf{MiniDec}$, respectively. Therefore, $\mathcal{B}$ never submits a decryption query (j, ψ) such that ψ is an answer to some of $\mathcal{B}$'s LR query of the form (j, M_0, M_1) (with the same j).

Let b be $\mathcal{B}$'s challenge bit. Let $\mathsf{Win}_{\mathcal{B}}$ be the event that $\mathcal{A}$'s output $(\mathsf{pk}_j, \widehat{c}^*$ satisfies the condition of Win_4 in the experiment simulated by $\mathcal{B}$, and let $\mathsf{Ask}_{\mathcal{B}}$ be the event $\mathcal{A}$ asks a RKG query of the form $(\mathsf{pk}_{i^*}, \mathsf{pk}_j)$ in the experiment simulated by $\mathcal{B}$. Note that by our construction of $\mathcal{B}$, it outputs 1 only when both $\mathsf{Win}_{\mathcal{B}}$ and $\mathsf{Ask}_{\mathcal{B}}$ occur.

The multi-user CCA advantage of $\mathcal{B}$ can be calculated as follows:

$$\begin{aligned}
\mathsf{Adv}^{\mathtt{CCA\text{-}PKE}}_{(\mathcal{B},n)}(k) &= |\Pr[b = b'] - \frac{1}{2}| \\
&= \frac{1}{2}|\Pr[b' = 1|b = 0] - \Pr[b' = 1|b = 1]| \\
&= \frac{1}{2}|\Pr[\mathsf{Win}_{\mathcal{B}} \wedge \mathsf{Ask}_{\mathcal{B}}|b = 0] - \Pr[\mathsf{Win}_{\mathcal{B}} \wedge \mathsf{Ask}_{\mathcal{B}}|b = 1]|
\end{aligned}$$

Then, a similar analysis to the proof of Lemma 2 shows that $\Pr[\mathsf{Win}_{\mathcal{B}} \wedge \mathsf{Ask}_{\mathcal{B}}|b = 0] = \Pr[\mathsf{Win}_4 \wedge \mathsf{Ask}_4]$ and $\Pr[\mathsf{Win}_{\mathcal{B}} \wedge \mathsf{Ask}_{\mathcal{B}}|b = 1] = \Pr[\mathsf{Win}_5 \wedge \mathsf{Ask}_5]$. Using these in the above inequality, and recalling the assumption that the underlying PKE scheme is CCA secure in the multi-user setting, we conclude that $|\Pr[\mathsf{Win}_4 \wedge \mathsf{Ask}_4] - \Pr[\mathsf{Win}_5 \wedge \mathsf{Ask}_5]|$ is negligible.

□

Lemma 7. *If the re-splittable TPKE scheme has decryption consistency,* $\Pr[\mathsf{Win}_5 \wedge \mathsf{Ask}_5]$ *is negligible.*

Proof of Lemma 7. We show that we can construct a PPT adversary $\mathcal{B}$ against the decryption consistency of the underly re-splittable TPKE scheme such that $\mathsf{Adv}^{\mathtt{DC\text{-}TPKE}}_{(\mathcal{B},2,2)}(k) = \Pr[\mathsf{Win}_5 \wedge \mathsf{Ask}_5]$. By the decryption consistency of the TPKE scheme, the above equation implies that $\Pr[\mathsf{Win}_5 \wedge \mathsf{Ask}_5]$ is negligible, which proves the lemma. The description of $\mathcal{B}$ is as follows.

First, $\mathcal{B}$ is given 1^k and a public key tpk_{i^*} from the challenger. $\mathcal{B}$ generates other key materials of the the honest users' keys $\mathcal{PK}$ and the challenge key pair $(\mathsf{sk}^*, \mathsf{pk}^*)$ except for the secret key tsk_{i^*} corresponding to tpk_{i^*}. Then, $\mathcal{B}$ gives 1^k and $\mathcal{PK}^* = \{\mathsf{pk}_{i^*}\} \cup \mathcal{PK}$ to $\mathcal{A}$. $\mathcal{B}$ also generates two empty lists L^*_{RKG} and L^*_{REnc}.

When $\mathcal{A}$ submits a challenge query m^*, $\mathcal{B}$ just encrypts $c^* \leftarrow \mathsf{TEnc}(tpk_{i^*}, m^*)$, and returns c^* to $\mathcal{A}$.

When $\mathcal{A}$ submits a RKG query $(\mathsf{pk}_i \in \mathcal{PK}^*, \mathsf{pk}_j)$, $\mathcal{B}$ responds as follows: (1) If $\mathsf{pk}_i \neq \mathsf{pk}_{i^*}$, then $\mathcal{B}$ runs $rk_{i\to j} \leftarrow \mathsf{RKG}(\mathsf{sk}_i, \mathsf{pk}_j)$, and returns $rk_{i\to j}$ to $\mathcal{A}$. (2) If $\mathsf{pk}_i = \mathsf{pk}_{i^*}$ and $\mathsf{pk}_j \notin \mathcal{PK}$, then $\mathcal{B}$ returns $\perp$ to $\mathcal{A}$. (3) Otherwise (i.e. $\mathsf{pk}_i = \mathsf{pk}_{i^*}$ and $\mathsf{pk}_j \in \mathcal{PK}$), $\mathcal{B}$ proceeds as follows:

(3a) Submit a split&corruption query that asks for the second secret key share to the challenger, and receive $(tsk_{i^*.2}, tvk_{i^*})$.

(3b) Compute $\psi \leftarrow \mathsf{PEnc}(pk_j, 0^{|tsk_{i^*.1}|})$ and $\sigma \leftarrow \mathsf{Sign}(sk_{i^*}, \langle \psi \| tvk_{i^*} \| \mathsf{pk}_{i^*} \| \mathsf{pk}_j \rangle)$.

(3c) Set $rk_{i^*\to j} = (\mathsf{pk}_{i^*}, \mathsf{pk}_j, tsk_{i^*.2}, \psi, tvk_{i^*}, \sigma)$, return $rk_{i^*\to j}$ to $\mathcal{A}$, and store the values $(\mathsf{pk}_j, \psi, tvk_{i^*}, \sigma, -)$ into the list L^*_{RKG}, where "$-$" is a "blank".

When $\mathcal{A}$ submits a REnc query $(\mathsf{pk}_i \in \mathcal{PK}^*, \mathsf{pk}_j, c)$, $\mathcal{B}$ responds as follows:

(1) If $\mathsf{pk}_i \neq \mathsf{pk}_{i^*}$, then execute $\widehat{c} \leftarrow \mathsf{REnc}(\mathsf{RKG}(\mathsf{sk}_i, \mathsf{pk}_j), c)$, and return $\widehat{c}$ to $\mathcal{A}$.

(2) (From here on it holds that $\mathsf{pk}_i = \mathsf{pk}_{i^*}$.) Submit a split&corruption query that asks for the first key share to the challenger, and receive $(tsk_{i^*,1}, tvk_{i^*})$ as the response.

(3) Execute $\psi \leftarrow \mathsf{PEnc}(pk_j, tsk_{i^*.1})$ and $\sigma \leftarrow \mathsf{Sign}(sk_{i^*}, \langle \psi \| tvk_{i^*} \| \mathsf{pk}_{i^*} \| \mathsf{pk}_j \rangle)$.

(4) If $\mathsf{pk}_j \notin \mathcal{PK}$ and $c = c^*$, then return $\perp$ (recall that a REnc query $(\mathsf{pk}_i, \mathsf{pk}_j, c)$ with $\mathsf{pk}_i = \mathsf{pk}_{i^*}$, $\mathsf{pk}_j \notin \mathcal{PK}$ and $c = c^*$ is answered with $\perp$).

(5) Else if $\mathsf{pk}_j \notin \mathcal{PK}$ and $c \neq c^*$, then submit a share decryption query $(tvk_{i^*}, 2, c)$, and receive μ_2 as the response. Here, (5-1) if $\mu_2 = \perp$ then return $\perp$ to $\mathcal{A}$. (5-2) Otherwise, execute $\widehat{c} \leftarrow \mathsf{PEnc}(\widehat{pk}_j, \langle \mathsf{pk}_{i^*} \| \mathsf{pk}_j \| c \| \mu \| tvk_{i^*} \| \sigma \rangle)$ and return $\widehat{c}$ to $\mathcal{A}$.

(6) Otherwise (i.e. $\mathsf{pk}_j \in \mathcal{PK}$), execute $\widehat{c} \leftarrow \mathsf{PEnc}(\widehat{pk}_j, \langle \mathsf{pk}_{i^*} \| \mathsf{pk}_j \| \mathbf{0} \rangle$, return $\widehat{c}$ to $\mathcal{A}$, and store the values $(\mathsf{pk}_j, \widehat{c}, c, \mu_2, tvk_{i^*}, tsk_{i^*.1})$ into the list L^*_{REnc}.

$\mathcal{B}$ answers to $\mathsf{REncVer}$, Dec_1, and Dec_2 queries from $\mathcal{A}$ exactly as in Game 5. This is possible because $\mathcal{B}$ holds all key materials except the tsk_{i^*} corresponding to tpk^*, and whenever $\mathcal{B}$ has to compute $\mathsf{TSplit}(tsk_i)$ or $\mathsf{TShDec}(tpk_i, tsk_{i.\gamma}, c)$ ($\gamma \in \{1,2\}$), $\mathcal{B}$ can submit a split&corruption/share decryption query to the challenger and use the obtained result. Here, we stress that $\mathcal{B}$ never falls into the situation where both of the secret shares $tsk_{i^*.1}$ and $tsk_{i^*.2}$ (under the same splitting) are required. In particular, to answer to Dec_2 queries, $\mathcal{B}$ does not need the exact values of $tsk_{i^*.1}$ and $tsk_{i^*.2}$, but the decryption shares computed using them, and thus Dec_2 queries can be simulated by utilizing the share decryption queries.

Finally, when $\mathcal{A}$ terminates with output $(\mathsf{pk}_j, \widehat{c}^*)$, $\mathcal{B}$ checks whether the conditions of Win_5 and Ask_5 hold for this $(\mathsf{pk}_j, \widehat{c}^*)$ by simulating the response to the $\mathsf{REncVer}$ query $(\mathsf{pk}_{i^*}, \mathsf{pk}_j, c^*, \widehat{c}^*)$ and the response to the Dec_1 query $(\mathsf{pk}_j, \widehat{c}^*)$ by itself. If the conditions of Win_5 and Ask_5 are not satisfied, then $\mathcal{B}$ gives up and aborts.

Otherwise (i.e. the conditions of Win_5 and Ask_5 are satisfied), let $\{\mu_1, \mu_2\}$, $tsk_{i^*.1}$, and tvk_{i^*} be the decryption shares, the secret key share, and the verification key that appear when simulating the response to the Dec_1 query $(\mathsf{pk}_j, \widehat{c}^*)$. Note that due to how $\mathcal{B}$ answers to the RKG queries of the form $(\mathsf{pk}_{i^*}, \mathsf{pk}_j)$, if Win_5 and Ask_5 occur, then the following conditions are satisfied:

(a) $\mathsf{TShVer}(tpk_{i^*}, tvk_{i^*}, c^*, 2, \mu_2) = \top$.

(b) $\mu_1 = \mathsf{TShDec}(tpk_{i^*}, tsk_{i^*.1}, c^*)$.

(c) tvk_{i^*} is one that is obtained as an answer to one of the split&corruption queries made by $\mathcal{B}$ (during the response to the RKG query $(\mathsf{pk}_{i^*}, \mathsf{pk}_j)$).

Note also that these three conditions also imply $\mathsf{TShVer}(tpk_{i^*}, tvk_{i^*}, c^*, 1, \mu_1) = \top$, because of the correctness property of the TPKE scheme.

Then, $\mathcal{B}$ submits a share decryption query $(tvk_{i^*}, 2, c^*)$ and receives the result μ_2'. (Note that that Win_5 and Ask_5 occur in particular implies that $\mathsf{TCom}(tpk_{i^*}, tvk_{i^*}, c^*, \{\mu_1, \mu_2'\}) \neq \mathsf{TCom}(tpk_{i^*}, tvk_{i^*}, c^*, \{\mu_1, \mu_2\}) = m^*$ and $\mu_2 \neq \mu_2'$.)

Finally, $\mathcal{B}$ terminates with output c^*, tvk_{i^*}, and two sets of decryption shares $\{\mu_1, \mu_2\}$ and $\{\mu_1, \mu_2'\}$.

The above completes the description of $\mathcal{B}$. It is not hard to see that $\mathcal{B}$ perfectly simulates Game 5 for $\mathcal{A}$. In particular, as explained above, $\mathcal{B}$ never falls into the situation where both of the secret shares $tsk_{i^*.1}$ and $tsk_{i^*.2}$ (under the same splitting) are required. Hence, the probability that $\mathcal{A}$ and $\mathcal{A}$'s output $(\mathsf{pk}_j, \widehat{c}^*)$ satisfy the conditions of Win_5 and Ask_5 in the experiment simulated by $\mathcal{B}$ is exactly the same as the probability that $\mathcal{A}$ and $\mathcal{A}$'s output satisfy those in the actual Game 5.

Furthermore, as we explained in the description of $\mathcal{B}$, whenever $\mathcal{A}$ and $\mathcal{A}$'s output satisfy the conditions of Win_5 and Ask_5, $\mathcal{B}$ can always output c^*, tvk_{i^*}, and $\{\mu_1, \mu_2\}$ and $\{\mu_1, \mu_2'\}$ satisfying the conditions of violating the decryption consistency, namely,

(a) tvk_{i^*} is returned one of the split&corruption queries made by $\mathcal{B}$.

(b) $\mathsf{TShVer}(tpk_{i^*}, tvk_{i^*}, c^*, 1, \mu_1) = \top$, $\mathsf{TShVer}(tpk_{i^*}, tvk_{i^*}, c^*, 2, \mu_2) = \top$, and TShVer $(tpk_{i^*}, tvk_{i^*}, c^*, 2, \mu_2') = \top$ (where the last equation is due to the correctness of the TPKE scheme and the fact that μ_2' is the result of the share decryption query of the form $(tvk_{i^*}, 2, c^*)$

(c) $\mu_2 \neq \mu_2'$ (and thus $\{\mu_1, \mu_2\} \neq \{\mu_1, \mu_2'\}$)

(d) $\mathsf{TCom}(tpk_{i^*}, tvk_{i^*}, c^*, \{\mu_1, \mu_2\}) \neq \mathsf{TCom}(tpk_{i^*}, tvk_{i^*}, c^*, \{\mu_1, \mu_2'\})$

Putting everything together, $\mathcal{B}$'s advantage in breaking the decryption consistency is $\mathsf{Adv}^{\mathtt{DC\text{-}TPKE}}_{(\mathcal{B},2,2)}(k) = \Pr[\mathsf{Win}_5 \wedge \mathsf{Ask}_5]$, as required. This completes the proof of Lemma 7. $\square$

Lemmas 1 to 7 guarantee that the right hand side of the inequality (4.1) is negligible, and thus $\mathcal{A}$ has negligible advantage in the soundness game. Since the negligible upper-bound of the advantage can be shown for any soundness adversary $\mathcal{A}$ and any polynomial n, we conclude that the VPRE scheme eHKK^+ has soundness. This completes the proof of Theorem 9. $\square$

4.4.2 Proof of Theorem 10

Let $n = n(k) > 0$ be any polynomial, and $\mathcal{A}$ be any PPT second-level CCA adversary against the VPRE scheme eHKK^+. We will consider the following sequence of games, which are very similar to the games we used in the proof of Theorem 4.4.1. The difference is that we will treat the share decryption of c^* slightly differently in several places, which is to guarantee that a reduction algorithm for attacking the CCA security of the re-splittable TPKE scheme will not fall into the situation in which it needs to perform the share decryption of c^*.

Game 0. This is the second-level CCA security game regarding eHKK^+. Since the subsequent games consider $\mathcal{A}$'s queries of some special types, without loss of generality we let the challenger generate two empty lists L^*_{RKG} and L^*_{REnc} at the beginning, and store the values that appear in the response to a re-encryption key generation query and a re-encryption query of special types. More concretely,

- If $\mathcal{A}$ issues a RKG query of the form $(\mathsf{pk}_{i^*}, \mathsf{pk}_j)$ with $\mathsf{pk}_j \in \mathcal{PK}$, then the challenger stores the values $(\mathsf{pk}_j, \psi, tvk_{i^*}, \sigma, tsk_{i^*.1})$ into L^*_{RKG}, where $(\psi, tvk_{i^*}, \sigma, tsk_{i^*.1})$ are the values generated when calculating $rk_{i^* \to j} \leftarrow \mathsf{RKG}(\mathsf{sk}_{i^*}, \mathsf{pk}_j)$.
- If $\mathcal{A}$ issues a REnc query of the form $(\mathsf{pk}_{i^*}, \mathsf{pk}_j, c)$ with $\mathsf{pk}_j \in \mathcal{PK}$ and the answer $\widehat{c}$ to this query is not $\perp$, then the challenger stores the values $(\mathsf{pk}_j, \widehat{c}, c, \mu_2, tvk_{i^*}, tsk_{i^*.1})$ into L^*_{REnc}, where $(\mu_2, tvk_{i^*}, tsk_{i^*.1})$ are the values generated when calculating $\widehat{c} \leftarrow \mathsf{REnc}(\mathsf{RKG}(\mathsf{sk}_{i^*}, \mathsf{pk}_j), c)$.

Game 1. Same as Game 0, except the following changes to the response to $\mathsf{REncVer}$ queries and Dec_1 queries: For $\mathsf{REncVer}$ queries $(\mathsf{pk}_i, \mathsf{pk}_j, c', \widehat{c})$, $\mathcal{A}$ responds as follows:

- (1) If $(\mathsf{pk}_j, \widehat{c}, c, \mu_2, tvk_{i^*}, tsk_{i^*.1}) \in L^*_{\mathsf{REnc}}$ for some $(c, \mu_2, tvk_{i^*}, tsk_{i^*.1})$, then:
  - (1a) If $\mathsf{pk}_i \neq \mathsf{pk}_{i^*}$, then return $\perp$.
  - (1b) Else if $\mathsf{pk}_i = \mathsf{pk}_{i^*}$ and $c' = c^*$, then: If $c = c^*$ and $\mathsf{TShVer}(tpk_{i^*}, tvk_{i^*}, c^*, 2, \mu_2) = \top$ then return $\top$, else return $\perp$.
  - (1c) Otherwise (i.e. $\mathsf{pk}_i = \mathsf{pk}_{i^*}$ and $c' \neq c^*$) run $\mathsf{MiniREncVer}(tpk_{i^*}, tvk_{i^*}, tsk_{i^*.1}, \mu_2, c, c')$ and return the result.
- (2) Otherwise (i.e. $(\mathsf{pk}_j, \widehat{c}, *, *, *, *) \notin L^*_{\mathsf{REnc}}$), run $\mathsf{REncVer}(\mathsf{pk}_i, \mathsf{sk}_j, c, \widehat{c})$ and return the result.

For Dec_1 queries $(\mathsf{pk}_j, \widehat{c})$, the challenger responds as follows: (Here, note that we consider the zero-th step corresponding to checking whether $\mathsf{REncVer}(\mathsf{pk}_{i^*}, \mathsf{pk}_j, c^*, \widehat{c}) = \top$.)

- (0) Simulate the response to the $\mathsf{REncVer}$ query of the form $(\mathsf{pk}_{i^*}, \mathsf{pk}_j, c^*, \widehat{c})$ for this game, and return $\perp$ if the result of the response is $\top$. (If $\mathcal{A}$ has not made the challenge query, this step is skipped.)
- (1) If $(\mathsf{pk}_j, \widehat{c}, c, \mu_2, tvk_{i^*}, tsk_{i^*.1}) \in L^*_{\mathsf{REnc}}$ for some $(c, \mu_2, tvk_{i^*}, tsk_{i^*.1})$, then:
  - (1a) If $c = c^*$ then return $\perp$.[6]
  - (1b) Otherwise (i.e. $c \neq c^*$), execute $m \leftarrow \mathsf{MiniDec}(tpk_{i^*}, tvk_{i^*}, tsk_{i^*.1}, \mu_2, c)$, and return m to $\mathcal{A}$.
- (2) Otherwise (i.e. $(\mathsf{pk}_j, \widehat{c}, *, *, *, *) \notin L^*_{\mathsf{REnc}}$), then execute $m \leftarrow \mathsf{Dec_1}(\mathsf{sk}_j, \widehat{c})$, and return m to $\mathcal{A}$.

We would like to emphasize that from this game on, the challenger need not perform $\mathsf{PDec}(\widehat{dk}_j, \widehat{c})$ for a $\mathsf{REncVer}$ query $(*, \mathsf{pk}_j, *, \widehat{c})$ and a $\mathsf{Dec_1}$ query $(\mathsf{pk}_j, \widehat{c})$ such that $(\mathsf{pk}_j, \widehat{c}, *, *, *, *) \in L^*_{\mathsf{REnc}}$.

Game 2. Same as Game 1, except that in this game, a re-encrypted ciphertext $\widehat{c}$ which is from the challenge key pk_{i^*} to an honest user key $\mathsf{pk}_j \in \mathcal{PK}$, is generated in such a way that $\widehat{c}$ contains no information.

More precisely, if $\mathcal{A}$ submits a REnc query of the form $(\mathsf{pk}_{i^*}, \mathsf{pk}_j, c)$ with $\mathsf{pk}_j \in \mathcal{PK}$, then the challenger responds as follows:

- (1) Compute $(tsk_{i^*.1}, tsk_{i^*.2}, tvk_{i^*}) \leftarrow \mathsf{TSplit}(tsk_{i^*})$.
- (2) Compute $\mu_2 \leftarrow \mathsf{TShDec}(tpk_{i^*}, tsk_{i^*.2}, c)$, and return $\perp$ to $\mathcal{A}$ if $\mu_2 = \perp$.
- (3) Compute $\widehat{c} \leftarrow \mathsf{PEnc}(\widehat{pk}_j, \langle \mathsf{pk}_{i^*} \| \mathsf{pk}_j \| \mathbf{0} \rangle)$ where $\mathbf{0}$ is the zero-string of appropriate length.
- (4) Return $\widehat{c}$ to $\mathcal{A}$ and store the values $(\mathsf{pk}_j, \widehat{c}, c, \mu_2, tvk_{i^*}, tsk_{i^*.1})$ into L^*_{REnc}, where $tsk_{i^*.1}$ is the secret key share corresponding to $(tsk_{i^*.2}, tvk_{i^*})$ that appears in the above step (1).

Game 3. Same as Game 2, except that in this game, if a re-encrypted ciphertext $\widehat{c}$ which is from the challenge key pk_{i^*} to an honest user key $\mathsf{pk}_j \in \mathcal{PK}$ is submitted as a $\mathsf{REncVer}$ query (with pk_j and some c) or as a $\mathsf{Dec_1}$ query (with pk_j), then $\widehat{c}$ is immediately answered with $\perp$ unless (a) it is an answer to a previously asked REnc query of the form $(\mathsf{pk}_{i^*}, \mathsf{pk}_j, c)$, or (b) it is a re-encryption using a re-encryption key $rk_{i^* \rightarrow j}$ that is returned as an answer to a previously asked RKG query.

More precisely, in this game, the challenger responds to $\mathsf{REncVer}$ queries $(\mathsf{pk}_i, \mathsf{pk}_j, c', \widehat{c})$ as follows:

- (1) If $(\mathsf{pk}_j, \widehat{c}, *, *, *, *) \in L^*_{\mathsf{REnc}}$, then respond as in Game 1.
- (2) Run $\widehat{M} = (\mathsf{pk}'_i \| \mathsf{pk}'_j \| c \| \mu_2 \| \psi \| tvk_i \| \sigma) \leftarrow \mathsf{PDec}(\widehat{dk}_j, \widehat{c})$, and return $\perp$ to $\mathcal{A}$ if $\widehat{M} = \perp$, or $(\mathsf{pk}'_i, \mathsf{pk}'_j) \neq (\mathsf{pk}_i, \mathsf{pk}_j)$, or $\mathsf{SVer}(vk_i, \langle \psi \| tvk_i \| \mathsf{pk}'_i \| \mathsf{pk}'_j \rangle, \sigma) = \perp$.

[6]This case will correspond to the situation where $\mathcal{A}$ asks a $\mathsf{Dec_1}$ query $(\mathsf{pk}_j, \widehat{c})$ such that $\widehat{c}$ is an answer to some of $\mathcal{A}$'s previous REnc query of the form $(\mathsf{pk}_{i^*}, \mathsf{pk}_j, c^*)$, but such a ciphertext $\widehat{c}$ satisfies $\mathsf{REncVer}(\mathsf{pk}_{i^*}, \mathsf{sk}_j, c^*, \widehat{c}) = \top$, and thus has already been answered with $\perp$ at the zero-th step. And thus, the condition $c = c^*$ checked here is never satisfied in Game 1. This is introduced just to clarify the later proof of Lemma 13 where we need to ensure that $\mathsf{TShDec}(tpk_i, tsk_{i^*.1}, c^*)$ is never performed.

- (3) If $\mathsf{pk}_i \neq \mathsf{pk}_{i^*}$ then execute $\mathsf{REncVer}(\mathsf{pk}_i, \mathsf{sk}_j, c', \widehat{c})$, and return the result to $\mathcal{A}$.
- (4) If $\mathsf{pk}'_i = \mathsf{pk}_{i^*}$ and $(\mathsf{pk}_j, \psi, tvk_{i^*}, \sigma, *) \notin L^*_{\mathsf{RKG}}$, then return $\perp$ to $\mathcal{A}$.
- (5) Otherwise (i.e. $\mathsf{pk}'_i = \mathsf{pk}_{i^*}$ and $(\mathsf{pk}_j, \psi, tvk_{i^*}, \sigma, tsk_{i^*.1}) \in L^*_{\mathsf{RKG}}$ for some $tsk_{i^*.1}$), as in the above step 4, execute the remaining procedure of $\mathsf{REncVer}$, and output the result to $\mathcal{A}$.

Furthermore, the challenger responds to Dec_1 queries $(\mathsf{pk}_j, \widehat{c})$ as follows:

- (0) Simulate the response to the $\mathsf{REncVer}$ query of the form $(\mathsf{pk}_{i^*}, \mathsf{pk}_j, c^*, \widehat{c})$ for this game, and return $\perp$ if the result of the response is $\top$. (If $\mathcal{A}$ has not made the challenge query, this step is skipped.)
- (1) If $(\mathsf{pk}_j, \widehat{c}, *, *, *, *) \in L^*_{\mathsf{REnc}}$, then respond as in Game 1.
- (2) Run $\widehat{M} = (\mathsf{pk}'_i \| \mathsf{pk}'_j \| c \| \mu_2 \| \psi \| tvk_i \| \sigma) \leftarrow \mathsf{PDec}(\widehat{dk_j}, \widehat{c})$, and return $\perp$ to $\mathcal{A}$ if $\widehat{M} = \perp$ or $\mathsf{pk}'_j \neq \mathsf{pk}_j$.
- (3) Parse pk'_i as $(tpk_i, \widehat{pk_i}, pk_i, vk_i)$, and return $\perp$ if $\mathsf{SVer}(vk_i, \langle \psi \| tvk_i \| \mathsf{pk}'_i \| \mathsf{pk}'_j \rangle, \sigma) = \perp$.
- (4) If $\mathsf{pk}'_i \neq \mathsf{pk}_{i^*}$ then calculate m by following the remaining procedure of Dec_1 (i.e. from the step "$tsk_{i.1} \leftarrow \mathsf{PDec}(dk_j, \psi)$"), and return m to $\mathcal{A}$.
- (5) If $\mathsf{pk}'_i = \mathsf{pk}_{i^*}$ and $(\mathsf{pk}_j, \psi, tvk_i, \sigma, *) \notin L^*_{\mathsf{RKG}}$, then return $\perp$ to $\mathcal{A}$.
- (6) Otherwise (i.e. $\mathsf{pk}'_i = \mathsf{pk}_{i^*}$ and $(\mathsf{pk}_j, \psi, tvk_i, \sigma, tsk_{i^*.1}) \in L^*_{\mathsf{RKG}}$ for some $tsk_{i^*.1}$), as in the above step 4, calculate m by following the remaining procedure of Dec_1.

Game 4. Same as Game 3, except that in this game, if $\mathcal{A}$ issues a $\mathsf{REncVer}$ query $(\mathsf{pk}_i, \mathsf{pk}_j, c', \widehat{c})$ or Dec_1 query $(\mathsf{pk}_j, \widehat{c})$, such that $\widehat{c}$ is a re-encrypted ciphertext from the challenge key pk_{i^*} to pk_j using a re-encryption key $rk_{i^* \to j}$ that is an answer to a previously asked RKG query of the form $(\mathsf{pk}_{i^*}, \mathsf{pk}_j)$ (which can be checked using L^*_{RKG} as in Game 3), then the query is answered using the information of $tsk_{i^*.1}$ found in L^*_{RKG}.

More precisely, in this game, the challenger responds to $\mathsf{REncVer}$ queries $(\mathsf{pk}_i, \mathsf{pk}_j, c', \widehat{c})$ as follows:

- (1), (2), (3), and (4): Same as in Game 3.
- (5): Here, it is guaranteed that $\mathsf{pk}'_i = \mathsf{pk}_{i^*}$ and $(\mathsf{pk}_j, \psi, tvk_{i^*}, \sigma, tsk_{i^*.1}) \in L^*_{\mathsf{RKG}}$ for some $tsk_{i^*.1}$. The challenger proceeds as follows:
  - (5a) If $c' = c^*$ then: If $\mathsf{TShVer}(tpk_{i^*}, tvk_{i^*}, c^*, 2, \mu_2) = \top$ then return $\top$ else return $\perp$ to $\mathcal{A}$.
  - (5b) Otherwise (i.e. $c' \neq c^*$), execute $\mathsf{MiniREncVer}(tpk_{i^*}, tvk_{i^*}, tsk_{i^*.1}, \mu_2, c, c')$, and return the result to $\mathcal{A}$.

Furthermore, the challenger responds to the Dec_1 queries $(\mathsf{pk}_j, \widehat{c})$ in the following way:

- (0), (1), (2), (3), (4), and (5): Same as in Game 3.

- (6): Here, it is guaranteed that $\mathsf{pk}'_i = \mathsf{pk}_{i^*}$ and $(\mathsf{pk}_j, \psi, tvk_{i^*}, \sigma, tsk_{i^*.1}) \in L^*_{\mathsf{RKG}}$ for some $tsk_{i^*.1}$. The challenger proceeds as follows:
  - (6a) If $c = c^*$, then return $\perp$.[7]
  - (6b) Otherwise (i.e. $c \neq c^*$), run $m \leftarrow \mathsf{MiniDec}(tpk_{i^*}, tvk_{i^*}, tsk_{i^*.1}, \mu_2, c)$, and return the result to $\mathcal{A}$.

We would like to emphasize that from this game on, the challenger need not perform $\mathsf{PDec}(dk_j, \psi)$ for a $\mathsf{REncVer}$ query $(\mathsf{pk}_i, \mathsf{pk}_j, c', \widehat{c})$ and a Dec_1 query $(\mathsf{pk}_j, \widehat{c})$ that are processed at their steps (5), namely, those queries that satisfy $(\mathsf{pk}_j, \widehat{c}, *, *, *, *) \notin L^*_{\mathsf{REnc}}$, $\mathsf{PDec}(\widehat{dk}_j, \widehat{c}) = \langle \mathsf{pk}_{i^*} \| \mathsf{pk}_j \| c \| \mu_2 \| \psi \| tvk_{i^*} \| \sigma \rangle \neq \perp$, $\mathsf{SVer}(vk_{i^*}, \langle \psi \| tvk_{i^*} \| \mathsf{pk}_{i^*} \| \mathsf{pk}_j \rangle, \sigma) = \top$, and $(\mathsf{pk}_j, \psi, tvk_{i^*}, \sigma, *) \in L^*_{\mathsf{RKG}}$.

Game 5. Same as Game 4, except that in this game, if $\mathcal{A}$ issues a RKG query of the form $(\mathsf{pk}_{i^*}, \mathsf{pk}_j)$ with $\mathsf{pk}_j \in \mathcal{PK}$, then the component ψ in a re-encryption key $rk_{i^* \to j}$ is generated in such a way that it contains no information.

More precisely, for this query, the challenger generates $rk_{i^* \to j} = (\mathsf{pk}_{i^*}, \mathsf{pk}_j, tsk_{i^*.2}, \psi, tvk_{i^*}, \sigma)$ by following the procedure of $\mathsf{RKG}(\mathsf{sk}_{i^*}, \mathsf{pk}_j)$ except that ψ is generated by $\psi \leftarrow \mathsf{PEnc}(pk_j, 0^{|tsk_{i^*.1}|})$. Then the challenger returns $rk_{i^* \to j}$ to $\mathcal{A}$ and stores the values $(\mathsf{pk}_j, \psi, tvk_{i^*}, \sigma, tsk_{i^*.1})$ into L^*_{RKG}.

For $i \in \{0, \ldots, 5\}$, let Succ_i be the event that $\mathcal{A}$ succeeds in guessing the challenge bit in Game i, namely, $b = b'$ occurs. Then, the second-level CCA advantage of $\mathcal{A}$ is estimated as:

$$\begin{aligned} \mathsf{Adv}^{\texttt{second-VPRE}}_{(\mathcal{A},n)}(k) &= |\Pr[\mathsf{Succ}_0] - \frac{1}{2}| \\ \leq \sum_{i \in \{0,1,2,3,4\}} &|\Pr[\mathsf{Succ}_i] - \Pr[\mathsf{Succ}_{i+1}]| + |\Pr[\mathsf{Succ}_5] - \frac{1}{2}|. \end{aligned} \tag{4.2}$$

We complete the proof by upperbounding each term in the right-hand side of the above inequality.

Lemma 8. $\Pr[\mathsf{Succ}_0] = \Pr[\mathsf{Succ}_1]$.

This lemma can be shown in almost the same way as the proof of Lemma 1, and thus we omit a formal proof. In the response to $\mathsf{REncVer}$ query of Game 1, we treat the challenge ciphertext c^* (namely, the steps (1b) and (1c)). However, note that we know that the challenge ciphertext c^* is always correctly generated, and thus that the challenger does not run $\mathsf{TShDec}(tsk_{i^*.1}, tsk_{i^*.1}, c)$ in case $c' = c^*$ does not change the output of $\mathsf{MiniREncVer}(tpk_{i^*}, tvk_{i^*}, tsk_{i^*.1}, \mu_2, c, c')$. The rest of the argument is exactly the same as the proof of Lemma 1.

[7]This case implies that c^* is contained in the plaintext of $\widehat{c}$, and thus $\widehat{c}$ is potentially a re-encryption of c^*. However, before this step, the challenger has performed the zero-th step test and thus have checked whether $\mathsf{REncVer}(\mathsf{pk}_{i^*}, \mathsf{sk}_j, c^*, \widehat{c}) = \top$. If the result of $\mathsf{REncVer}$ was $\top$, then the query must have been answered with $\perp$ at its zero-th step. Therefore, that this step (6a) is performed means that $\widehat{c}$ was not a valid re-encryption of c^*, and thus $\widehat{c}$ containing c^* must be an invalid ciphertext whose decryption result is $\perp$. With a similar reason explained in the previous footnote, this step is introduced to ensure that $\mathsf{TShDec}(tpk_{i^*}, tsk_{i^*.1}, c^*)$ is never performed, which will be important in the proof of Lemma 13.

Lemma 9. *If the PKE scheme is CCA secure in the multi-user setting,* $|\Pr[\mathsf{Succ}_1] - \Pr[\mathsf{Succ}_2]|$ *is negligible.*

Proof of Lemma 9. We show that we can construct a multi-user CCA adversary $\mathcal{B}$ against the underlying PKE scheme such that $\mathsf{Adv}^{\texttt{CCA-PKE}}_{(\mathcal{B},n)}(k) = |\Pr[\mathsf{Succ}_1] - \Pr[\mathsf{Succ}_2]|$. By the multi-user CCA security of the underlying PKE scheme (which is equivalent to the ordinary CCA security), the above implies that $|\Pr[\mathsf{Succ}_1] - \Pr[\mathsf{Succ}_2]|$ is negligible, which proves the lemma. The description of $\mathcal{B}$ is as follows:

First, $\mathcal{B}$ is given 1^k and public keys $(\widehat{pk}_1, \dots, \widehat{pk}_n)$ from the challenger. Then $\mathcal{B}$ generates other key materials of the honest users (except $\{\widehat{dk}_i\}_{i\in[n]}$) as well as the challenge key pair $(\mathsf{sk}_{i^*}, \mathsf{pk}_{i^*}) \leftarrow \mathsf{KG}(1^k)$. $\mathcal{B}$ then sets $\mathcal{PK} = \{\mathsf{pk}_i\}_{i\in[n]}$ and $\mathcal{PK}^* = \{\mathsf{pk}_{i^*}\} \cup \mathcal{PK}$, and gives 1^k and $\mathcal{PK}^*$ to $\mathcal{A}$. $\mathcal{B}$ also generates an empty list L^*_{REnc}. (Since the list L^*_{RKG} does not play any role in Game 1 and Game 2, $\mathcal{B}$ need not generate it.)

When $\mathcal{A}$ makes a challenge query (m_0, m_1), $\mathcal{B}$ picks a random bit $d \in \{0,1\}$, encrypts $c^* \leftarrow \mathsf{TEnc}(tpk_{i^*}, m_d)$, and returns c^* to $\mathcal{A}$.

$\mathcal{B}$ answers to $\mathcal{A}$'s queries except the challenge query in exactly the same way as $\mathcal{B}$ in the proof of Lemma 2 does.

Finally, when $\mathcal{A}$ outputs the guess bit d', $\mathcal{B}$ outputs $b' = 0$ if $d = d'$, otherwise outputs $b' = 1$.

The above completes the description of $\mathcal{B}$. Note that $\mathcal{B}$ submits a LR query of the form (j, M_0, M_1) only if $\mathcal{A}$ submits a RKG query of the form $(\mathsf{pk}_{i^*}, \mathsf{pk}_j, c)$ satisfying $\mathsf{pk}_j \in \mathcal{PK}$ and $\mathsf{TShDec}(tpk_{i^*}, tsk_{i^*.2}, c) \neq \bot$. Note also that $\mathcal{B}$ never submits a decryption query $(j, \widehat{c})$ such that $\widehat{c}$ is an answer to some of $\mathcal{B}$'s LR queries of the form (j, M_0, M_1) (with the same j).

The multi-user CCA advantage of $\mathcal{B}$ can be calculated as follows:

$$\begin{aligned}
\mathsf{Adv}^{\texttt{CCA-PKE}}_{(\mathcal{B},n)}(k) &= |\Pr[b = b'] - \frac{1}{2}| \\
&= \frac{1}{2}|\Pr[b' = 0|b = 0] - \Pr[b' = 0|b = 1]| \\
&= \frac{1}{2}|\Pr[d = d'|b = 0] - \Pr[d = d'|b = 1]|
\end{aligned}$$

Now, consider the case when $b = 0$. In this case, a re-encrypted ciphertext $\widehat{c}$ from the challenge public key pk_{i^*} to a honest user key $\mathsf{pk}_j \in \mathcal{PK}$ is generated as in Game 1. Moreover, it is easy to see that all the other values are calculated as in Game 1. Under this situation, the event $d = d'$ corresponds to the event that $\mathcal{A}$ succeeds in guessing the challenge bit in Game 1, and thus we have $\Pr[d = d'|b = 0] = \Pr[\mathsf{Succ}_1]$.

On the other hand, when $b = 1$, a re-encrypted ciphertext $\widehat{c}$ from pk_{i^*} to $\mathsf{pk}_j \in \mathcal{PK}$ is an encryption of $\langle \mathsf{pk}_{i^*} || \mathsf{pk}_j || \mathbf{0} \rangle$, where $\mathbf{0}$ is the zero-string of appropriate length, which is exactly how it is generated in Game 2. Since this is the only difference from the case $b = 0$, with a similar argument to the above we have $\Pr[d = d'|b = 1] = \Pr[\mathsf{Succ}_2]$.

In summary we have $\mathsf{Adv}^{\texttt{CCA-PKE}}_{(\mathcal{B},n)}(k) = \frac{1}{2}|\Pr[\mathsf{Succ}_1] - \Pr[\mathsf{Succ}_2]|$, as required. □

Lemma 10. *If the signature scheme is strongly unforgeable,* $|\Pr[\mathsf{Succ}_2] - \Pr[\mathsf{Succ}_3]|$ *is negligible.*

Proof Sketch of Lemma 10. For $i \in \{2, 3\}$, let Forge_i be the event that in Game i, $\mathcal{A}$ submits at least one Dec_1 query $(\mathsf{pk}_j, \widehat{c})$ or a $\mathsf{REncVer}$ query of the form $(\mathsf{pk}_{i^*}, \mathsf{pk}_j, c', \widehat{c})$ satisfying the following conditions:

(a) $(\mathsf{pk}_j, \widehat{c}, *, *, *, *) \notin L^*_{\mathsf{REnc}}$

(b) $\mathsf{PDec}(\widehat{dk}_j, \widehat{c}) = \langle \mathsf{pk}_{i^*} \| \mathsf{pk}_j \| c \| \mu_2 \| \psi \| tvk_{i^*} \| \sigma \rangle \neq \bot$

(c) $(\mathsf{pk}_j, \psi, tvk_{i^*}, \sigma, *) \notin L^*_{\mathsf{RKG}}$

(d) $\mathsf{SVer}(vk_{i^*}, \langle \psi \| tvk_{i^*} \| \mathsf{pk}_{i^*} \| \mathsf{pk}_j \rangle, \sigma) = \top$.

Game 2 and Game 3 proceed identically until the event Forge_2 or Forge_3 occurs in the corresponding games. Therefore we have

$$|\Pr[\mathsf{Succ}_2] - \Pr[\mathsf{Succ}_3]| \leq \Pr[\mathsf{Forge}_2] = \Pr[\mathsf{Forge}_3].$$

Then, we can show that there is another PPT adversary $\mathcal{B}$ against the strong unforgeability of the underlying signature scheme such that $\mathsf{Adv}^{\mathtt{SUF\text{-}SIG}}_{\mathcal{B}}(k) \geq \Pr[\mathsf{Forge}_3]$. By the strong unforgeability of the underlying signature scheme, the above implies that $\Pr[\mathsf{Forge}_3]$ is negligible, and thus $|\Pr[\mathsf{Succ}_2] - \Pr[\mathsf{Succ}_3]|$ is negligible, proving the lemma. Since the description of $\mathcal{B}$ and the analysis of $\mathcal{B}$'s advantage are essentially the same as those of the reduction algorithm $\mathcal{B}$ we used in the proof of Lemma 3 we omit a formal proof. $\square$

Lemma 11. $\Pr[\mathsf{Succ}_3] = \Pr[\mathsf{Succ}_4]$

The proof for this lemma is essentially the same as the proof of Lemma 5, and thus we omit a formal proof.

Lemma 12. *If the PKE scheme is CCA secure in the multi-user setting,* $|\Pr[\mathsf{Succ}_4] - \Pr[\mathsf{Succ}_5]|$ *is negligible*

Proof of Lemma 12. We show that we can construct a multi-user CCA adversary $\mathcal{B}$ (against the underlying PKE scheme) such that $\mathsf{Adv}^{\mathtt{CCA\text{-}PKE}}_{(\mathcal{B},n)}(k) = |\Pr[\mathsf{Succ}_4] - \Pr[\mathsf{Succ}_5]|$. By the multi-user CCA security of the underlying PKE scheme (which is equivalent to the ordinary CCA security), the above implies that $|\Pr[\mathsf{Succ}_4] - \Pr[\mathsf{Succ}_5]|$ is negligible, which proves the lemma. The description of $\mathcal{B}$ is as follows:

First, $\mathcal{B}$ is given 1^k and public keys $(pk_1, \ldots, pk_n)$ from the challenger. $\mathcal{B}$ generates other key materials of the challenge key pk_{i^*} and the honest users' keys $\mathcal{PK}$ except $\{dk_i\}_{i \in [n]}$. $\mathcal{B}$ then gives 1^k and $\mathcal{PK}^* = \{\mathsf{pk}_{i^*}\} \cup \mathcal{PK}$ to $\mathcal{A}$. $\mathcal{B}$ also generates two empty lists L^*_{RKG} and L^*_{REnc}.

When $\mathcal{A}$ makes a challenge query (m_0, m_1), $\mathcal{B}$ picks a random bit $d \in \{0, 1\}$, encrypts $c^* \leftarrow \mathsf{TEnc}(tpk_{i^*}, m_d)$, and returns c^* to $\mathcal{A}$.

$\mathcal{B}$ answers to $\mathcal{A}$'s queries except the challenge query in exactly the same way as $\mathcal{B}$ in the proof of Lemma 6 does.

Finally, when $\mathcal{A}$ outputs the guess bit d', $\mathcal{B}$ outputs $b' = 0$ if $d = d'$, otherwise outputs $b' = 1$.

The above completes the description of $\mathcal{B}$. Note that $\mathcal{B}$ submits an LR query of the form $(j, M_0 = tsk_{i^*.1}, M_1 = 0^{|tsk_{i^*.1}|})$ only when $\mathcal{A}$ submits a RKG query of the form

$(\mathsf{pk}_{i^*}, \mathsf{pk}_j)$ with $\mathsf{pk}_j \in \mathcal{PK}$. Moreover, note also that all the ciphertexts ψ that $\mathcal{B}$ receives as an answer to a LR query of the form (j, M_0, M_1) are stored into L^*_{RKG}, and all the $\mathsf{REncVer}$ queries $(\mathsf{pk}_{i^*}, \mathsf{pk}_j, c, \widehat{c})$ and all the Dec_1 queries $(\mathsf{pk}_j, \widehat{c})$ such that the plaintext of $\widehat{c}$ contains ψ that appears in L^*_{RKG} are answered with either $\perp$ or using $\mathsf{MiniREncVer}$ and $\mathsf{MiniDec}$, respectively. Therefore, $\mathcal{B}$ never submits a decryption query (j, ψ) such that ψ is an answer to some of $\mathcal{B}$'s LR query of the form (j, M_0, M_1) (with the same j).

The multi-user CCA advantage of $\mathcal{B}$ can be calculated as follows:

$$\begin{aligned}
\mathsf{Adv}^{\mathtt{CCA\text{-}PKE}}_{(\mathcal{B},n)}(k) &= |\Pr[b = b'] - \frac{1}{2}| \\
&= \frac{1}{2}|\Pr[b' = 0|b = 1] - \Pr[b' = 0|b = 0]| \\
&= \frac{1}{2}|\Pr[d = d'|b = 1] - \Pr[d = d'|b = 0]|
\end{aligned}$$

Then, a similar analysis to the proof of Lemma 6 shows that $\Pr[\mathsf{Succ}_4] = \Pr[d' = d|b = 0]$ and $\Pr[\mathsf{Succ}_5] = \Pr[d' = d|b = 1]$. Using these in the above inequality, and recalling the assumption that the underlying PKE scheme is CCA secure in the multi-user setting, we conclude that $|\Pr[\mathsf{Succ}_4] - \Pr[\mathsf{Succ}_5]|$ is negligible. □

Lemma 13. *If the re-splittable TPKE scheme is CCA secure,* $|\Pr[\mathsf{Succ}_5] - 1/2|$ *is negligible.*

Proof of Lemma 13. We show that we can construct a CCA adversary $\mathcal{B}$ against the underly TPKE scheme such that $\mathsf{Adv}^{\mathtt{CCA\text{-}TPKE}}_{(\mathcal{B},2,2)}(k) = |\Pr[\mathsf{Succ}_5] - 1/2|$. By the CCA security of the TPKE scheme, the above equation implies that $|\Pr[\mathsf{Succ}_5] - 1/2|$ is negligible, which proves the lemma. The description of $\mathcal{B}$ is as follows.

First, $\mathcal{B}$ is given 1^k and a public key tpk_{i^*} from the challenger. $\mathcal{B}$ generates other key materials of the the honest users' keys $\mathcal{PK}$ and the challenge key pair $(\mathsf{sk}^*, \mathsf{pk}^*)$ except for the secret key tsk_{i^*} corresponding to tpk_{i^*}. Then, $\mathcal{B}$ gives 1^k and $\mathcal{PK}^* = \{\mathsf{pk}_{i^*}\} \cup \mathcal{PK}$ to $\mathcal{A}$. $\mathcal{B}$ also generates two empty lists L^*_{RKG} and L^*_{REnc}.

When $\mathcal{A}$ submits a challenge query (m_0, m_1), $\mathcal{B}$ submits the same pair (m_0, m_1) as a challenge query to the challenger and obtains the challenge ciphertext c^*. Then, $\mathcal{B}$ returns this c^* to $\mathcal{A}$.

$\mathcal{B}$ answers to $\mathcal{A}$'s RKG and REnc queries in exactly the same way as $\mathcal{B}$ in the proof of Lemma 7 does. This is possible because an adversary attacking the decryption consistency and an adversary attacking CCA security for a re-splittable TPKE scheme have the same interface (except an output).

$\mathcal{B}$ answers to $\mathsf{REncVer}$, Dec_1, and Dec_2 queries from $\mathcal{A}$ exactly as in Game 5. This is possible because $\mathcal{B}$ holds all key materials except the tsk_{i^*} corresponding to tpk^*, and whenever $\mathcal{B}$ has to compute $\mathsf{TSplit}(tsk_i)$ or $\mathsf{TShDec}(tpk_i, tsk_{i.\gamma}, c)$ $(\gamma \in \{1, 2\})$, $\mathcal{B}$ can submit a split&corruption/share decryption query to the challenger and use the obtained result. Here, we stress that $\mathcal{B}$ never falls into the situation where both of the secret shares $tsk_{i^*.1}$ and $tsk_{i^*.2}$ (under the same splitting) are required, or the situation where $\mathcal{B}$ has to compute $\mathsf{TShDec}(tpk_{i^*}, tsk_{i^*.\gamma}, c^*)$ for some $\gamma \in \{1, 2\}$. These are guaranteed by the definition of Game 5 (see also the footnotes 4 and 5).

Finally, when $\mathcal{A}$ terminates with its guess bit $b' \in \{0, 1\}$, $\mathcal{B}$ uses this b' as its guess for the challenge bit and terminates.

The above completes the description of $\mathcal{B}$. It is not hard to see that $\mathcal{B}$ perfectly simulates Game 5 so that $\mathcal{A}$'s challenge bit is that of $\mathcal{B}$. (In particular, as explained above, $\mathcal{B}$ never submits a prohibited query c^* as a share decryption query.) Therefore, the probability that $\mathcal{B}$ succeeds in guessing its challenge bit is exactly the probability that $\mathcal{A}$ succeeds in guessing the challenge bit in Game 5. Therefore, $\mathcal{B}$'s CCA advantage can be calculated as

$$\mathsf{Adv}^{\texttt{CCA-TPKE}}_{(\mathcal{B},2,2)}(k) = |\Pr[\mathsf{Succ}_5] - 1/2|,$$

as required. This completes the proof of Lemma 13. □

Lemmas 8 to 13 guarantee that the right hand side of the inequality (4.2) is negligible, and thus $\mathcal{A}$ has negligible advantage in the second-level CCA game. Since the negligible upperbound of the advantage can be shown for any second-level CCA adversary $\mathcal{A}$ and any polynomial n, we conclude that the VPRE scheme eHKK^+ is second-level CCA secure. This completes the proof of Theorem 10. □

4.4.3 Proof of Theorem 11

Let $\mathcal{A}$ be any PPT adversary that attacks the first-level CCA security of the VPRE scheme eHKK^+ and makes in total Q $\mathsf{REncVer}$ queries. (Since $\mathcal{A}$ is PPT, Q is polynomial.) Consider the following games, where the values with asterisk (*) are those related to the challenge ciphertext $\widehat{c}^*$ of $\mathcal{A}$:

Game 0. The first-level CCA game of the VPRE scheme eHKK^+.

Game 1. Same as Game 0, except that if $\mathcal{A}$ submits a $\mathsf{REncVer}$ query $(\mathsf{pk}, c, \widehat{c})$ satisfying $\widehat{c} = \widehat{c}^*$, then without actually executing $\mathsf{REncVer}$, the query is answered with $\top$ if $(\mathsf{pk}, c) = (\mathsf{pk}_{\mathcal{A}}, c^*)$ or with $\perp$ otherwise.

Game 2. Same as Game 1, except that if $\mathcal{A}$ submits a $\mathsf{REncVer}$ query $(\mathsf{pk}, c, \widehat{c})$ satisfying $c = c^*$, then it is answered with $\perp$. This change in particular implies that now all $\mathsf{REncVer}$ queries $(\mathsf{pk}, c, \widehat{c})$ with $\widehat{c} = \widehat{c}^*$ are always answered with $\perp$.

Game 3. Same as Game 2, except that $\widehat{c}^*$ is generated in such a way that it does not contain any information on c^*. More precisely, $\widehat{c}^*$ is generated so that $\widehat{c}^* \leftarrow \mathsf{PEnc}(\widehat{dk}^*, \langle \mathsf{pk}_{\mathcal{A}} \| \mathsf{pk}^* \| 0^{\ell} \| \psi^* \| tvk^*_{\mathcal{A}} \| \sigma^* \rangle)$, where $\ell = |c^*| + |\mu^*_2|$.

For $i \in \{0, 1, 2\}$, let Succ_i be the event that in Game i $\mathcal{A}$ succeeds in guessing the challenge bit (i.e. $b' = b$ occurs), and let Query_i be the event that in Game i, $\mathcal{A}$ submits at least one $\mathsf{REncVer}$ query $(\mathsf{pk}, c, \widehat{c})$ satisfying $c = c^*$.

$\mathcal{A}$'s first-level CCA advantage is calculated as follows:

$$\begin{aligned}\mathsf{Adv}^{\texttt{first-VPRE}}_{\mathcal{A}}(k) &= |\Pr[\mathsf{Succ}_0] - \frac{1}{2}| \\ &\le \sum_{i \in \{0,1,2\}} |\Pr[\mathsf{Succ}_i] - \Pr[\mathsf{Succ}_{i+1}]| + |\Pr[\mathsf{Succ}_3] - \frac{1}{2}|. \end{aligned} \tag{4.3}$$

Thus, it suffices to show that each term in the right hand side of the above inequality is negligible.

Firstly, note that in Game 3, the information on the challenge bit is not at all contained in $\widehat{c}^*$ or answers to $\mathcal{A}$'s queries, and hence $\mathcal{A}$'s view is independent of the challenge bit. Therefore, we have $\Pr[\mathsf{Succ}_3] = 1/2$. Note also that it holds that $\Pr[\mathsf{Succ}_0] = \Pr[\mathsf{Succ}_1]$, due to the correctness of the building block PKE scheme. Specifically, $\widehat{c}^*$ contains (in its plaintext) $\mathsf{pk}_{\mathcal{A}}$ and c^*, and thus a $\mathsf{REncVer}$ query $(\mathsf{pk}, c, \widehat{c}^*)$ with $(\mathsf{pk}, c) \neq (\mathsf{pk}_{\mathcal{A}}, c^*)$ cannot make $\mathsf{REncVer}$ output $\top$.

Next, we show that $|\Pr[\mathsf{Succ}_2] - \Pr[\mathsf{Succ}_3]|$ is negligible, due to the CCA security of the building block PKE scheme. To see this, consider the following CCA adversary $\mathcal{B}$ that simulates Game 2 or Game 3 perfectly for $\mathcal{A}$ depending on $\mathcal{B}$'s challenge bit:

At the beginning of the CCA game, $\mathcal{B}$ is given $\widehat{pk}^*$. $\mathcal{B}$ generates the challenge public/secret key pair $(\mathsf{sk}^*, \mathsf{pk}^*)$ except $\widehat{dk}^*$, by executing $(tsk^*, tpk^*) \leftarrow \mathsf{TKG}(1^k, 2, 2)$, $(dk^*, pk^*) \leftarrow \mathsf{PKG}(1^k)$, and $(sk^*, vk^*) \leftarrow \mathsf{SKG}(1^k)$, and setting $\mathsf{sk}^* \leftarrow (tsk^*, \perp, dk^*, sk^*)$ and $\mathsf{pk}^* = (tpk^*, \widehat{pk}^*, pk^*, vk^*)$. Now, since $\mathcal{B}$ knows tsk^*, dk^* and sk^*, $\mathcal{B}$ can answer to re-encryption key generation, re-encryption, and second-level decryption queries perfectly. Furthermore, $\mathcal{B}$ can answer to first-level decryption queries $\widehat{c}$ by its ability to make decryption queries and the knowledge of tsk^* and dk^*.

When $\mathcal{A}$ submits two plaintexts (m_0, m_1) of equal length and a key pair $(\mathsf{sk}_{\mathcal{A}}, \mathsf{pk}_{\mathcal{A}})$ as a challenge query, $\mathcal{B}$ proceeds as follows:

1. Parse $\mathsf{sk}_{\mathcal{A}}$ as $(tsk_{\mathcal{A}}, \widehat{dk}_{\mathcal{A}}, dk_{\mathcal{A}}, sk_{\mathcal{A}})$ and $\mathsf{pk}_{\mathcal{A}}$ as $(tpk_{\mathcal{A}}, \widehat{pk}_{\mathcal{A}}, pk_{\mathcal{A}}, vk_{\mathcal{A}})$.

2. Flip a fair coin $w \in \{0,1\}$, and execute $c^* \leftarrow \mathsf{TEnc}(tpk_{\mathcal{A}}, m_w)$, $(tsk^*_{\mathcal{A}.1}, tsk^*_{\mathcal{A}.2}, tvk^*_{\mathcal{A}}) \leftarrow \mathsf{TSplit}(tsk_{\mathcal{A}})$, $\psi^* \leftarrow \mathsf{TEnc}(pk^*, tsk^*_{\mathcal{A}.1})$, $\sigma^* \leftarrow \mathsf{Sign}(sk^*, \langle \psi^* \| tvk^*_{\mathcal{A}} \| \mathsf{pk}_{\mathcal{A}} \| \mathsf{pk}^* \rangle)$, and $\mu^*_2 \leftarrow \mathsf{TShDec}(tpk_{\mathcal{A}}, tsk^*_{\mathcal{A}.2}, c^*)$.

3. Set $\widehat{M}_0 = \langle \mathsf{pk}_{\mathcal{A}} \| \mathsf{pk}^* \| c^* \| \mu^*_2 \| \psi^* \| tvk^*_{\mathcal{A}} \| \sigma^* \rangle$ and $\widehat{M}_1 = \langle \mathsf{pk}_{\mathcal{A}} \| \mathsf{pk}^* \| 0^{\ell} \| \psi^* \| tvk^*_{\mathcal{A}} \| \sigma^* \rangle$, where $\ell = |c^*| + |\mu^*_2|$.

4. Submit $(\widehat{M}_0, \widehat{M}_1)$ to $\mathcal{B}$'s CCA challenger, and receive $\mathcal{B}$'s challenge ciphertext $\widehat{c}^*$.

5. Return $\widehat{c}^*$ to $\mathcal{A}$ as $\mathcal{A}$'s challenge ciphertext.

When $\mathcal{A}$ asks a re-encryption verification query $(\mathsf{pk}, c, \widehat{c})$, if $\widehat{c} = \widehat{c}^*$, then $\mathcal{B}$ answers with $\perp$ (which is the legitimate answer in Games 2 and 3). Otherwise (i.e. $\widehat{c} \neq \widehat{c}^*$), $\mathcal{B}$ can answer to the re-encryption verification query perfectly as the challenger in Games 2 and 3 does, by forwarding $\widehat{c}$ to $\mathcal{B}$'s challenger as a decryption query and calculating the remaining procedure of $\mathsf{REncVer}$ using tsk^* and dk^*.

Finally, when $\mathcal{A}$ terminates with its guess bit $w' \in \{0,1\}$, $\mathcal{B}$ sets $b' \leftarrow 1$ if $w' = w$, otherwise it sets $b' \leftarrow 0$, and terminates with output b'.

Let b be $\mathcal{B}$'s challenge bit. It is easy to see that depending on $\mathcal{B}$'s challenge bit b, $\mathcal{B}$ simulates Game 2 or Game 3 perfectly for $\mathcal{A}$ so that $\mathcal{A}$'s challenge bit is w. In particular, $\mathcal{B}$ answers to all queries made by $\mathcal{A}$ as should be done in Games 2 and 3 perfectly. Furthermore, $\mathcal{B}$ outputs 1 whenever $\mathcal{A}$ succeeds in guessing w (i.e. $w' = w$ occurs). Therefore, we have $\Pr[b' = 1 | b = 0] = \Pr[\mathsf{Succ}_2]$ and $\Pr[b' = 1 | b = 1] = \Pr[\mathsf{Succ}_3]$, and thus $|\Pr[\mathsf{Succ}_2] - \Pr[\mathsf{Succ}_3]|$ is negligible by the CCA security of the underlying PKE scheme.

It remains to show the upperbound of $|\Pr[\mathsf{Succ}_1] - \Pr[\mathsf{Succ}_2]|$ to be negligible. To see this, note that Game 1 and Game 2 proceed identically unless Query_1 or Query_2 occurs in the corresponding games. Hence, we have

$$|\Pr[\mathsf{Succ}_1] - \Pr[\mathsf{Succ}_2]| \leq \Pr[\mathsf{Query}_1] = \Pr[\mathsf{Query}_2].$$

Furthermore, by the triangle inequality, we have

$$\Pr[\mathsf{Query}_2] \leq |\Pr[\mathsf{Query}_2] - \Pr[\mathsf{Query}_3]| + \Pr[\mathsf{Query}_3]$$

We can show the upperbound of $|\Pr[\mathsf{Query}_2] - \Pr[\mathsf{Query}_3]|$ to be negligible by the CCA security of the building block PKE scheme, with essentially the same way as we did for $|\Pr[\mathsf{Succ}_2] - \Pr[\mathsf{Succ}_3]|$ to be negligible. Specifically, consider the reduction algorithm $\mathcal{B}'$ that runs in the same way as the above $\mathcal{B}$, except that $\mathcal{B}'$ outputs $b' = 1$ only when $\mathcal{A}$ makes a re-encryption verification query that contains c^*. Then, $\mathcal{B}'$ simulates Game 2 and Game 3 perfectly, and thus the probability that $\mathcal{A}$ makes a re-encryption query that contains c^* is exactly the same as the probability that $\mathcal{A}$ does so in the game simulated by $\mathcal{B}'$, i.e. $\Pr[b' = 1|b = 0] = \Pr[\mathsf{Query}_2]$ and $\Pr[b' = 1|b = 1] = \Pr[\mathsf{Query}_3]$, and thus $|\Pr[\mathsf{Query}_2] - \Pr[\mathsf{Query}_3]|$ is negligible due to the CCA security of the underlying PKE scheme.

Finally, we can show that $\Pr[\mathsf{Query}_3]$ is upperbounded to be negligible due to the strong smoothness of the underlying TPKE scheme. To see this, note that in Game 3, the information on c^* is not at all contained in $\mathcal{A}$'s challenge ciphertext $\widehat{c}^*$. Note also that in Game 3, although the key pair $(\mathsf{sk}_{\mathcal{A}}, \mathsf{pk}_{\mathcal{A}})$ is chosen by $\mathcal{A}$, it is required to be a valid key pair (and thus must be in the range of TKG). Moreover, the randomness for generating c^* is honestly chosen by the challenger in Game 3, and thus the probability that c^* is contained in one particular re-encryption verification query is bounded by $\mathsf{Smth}(k)$. By applying the union bound over all of $\mathcal{A}$'s Q queries, the upperbound of $\Pr[\mathsf{Query}_3]$ is $Q \cdot \mathsf{Smth}(k)$, which is negligible.

We have seen that each $|\Pr[\mathsf{Succ}_i] - \Pr[\mathsf{Succ}_{i+1}]|$ is negligible and $\Pr[\mathsf{Succ}_3] = 1/2$, and therefore, $\mathcal{A}$'s first-level CCA advantage is negligible. This completes the proof of Theorem 11. □

4.5 Conclusion

In this chapter, we introduced a new functionality for PRE that we call re-encryption verifiability, proposed a generic construction, and proved its security. In practice, we can reduce the level of trust we have to put on the proxies by achieving the functionality of re-encryption verification. These days, large scale surveillance by NSA becomes public. Therefore, it becomes difficult to trust the third parties completely. The contribution of this research is to solve that problem not by the management and/or operation but by the technology. From the view point of theory, moreover, we got out of the RCCA security and achieve the CCA security. Strictly speaking, however, the security that we achieved in this research is secret-key detectable RCCA. The biggest contribution of this research is that we strictly proved the backward compatibility between security notions. There exists many security notions for PRE and we cannot compare the most of them. In these situations, researchers were not able to judge that which definitions should be used. The result and way of thinking in this chapter is important to escape from this situation.

A proposal of concrete PRE scheme with public verifiability is future work. It will be difficult for our generic construction to achieve the public verifiability because the first-level ciphertext is contained in the second-level ciphertext. If we can construct a PRE scheme with public verifiability, we can avoid the long and difficult soundness proof which is denoted in this chapter.

Chapter 5

Proxy Re-encryption from Indistinguishability Obfuscation

5.1 Introduction

5.1.1 Background and Motivation

Proxy re-encryption (PRE) is an interesting extension of traditional public key encryption (PKE). In addition to the normal operations of PKE, with a dedicated re-encryption key (generated by receiver A), a semi-trusted party called *proxy* can turn a class of ciphertexts destined for user A into those for user B. A remarkable property of PRE is that the proxy carrying out the transform is totally ignorant of the plaintext. PRE was first formalized by Blaze et al. [19] and has received much attention in recent years. There are many models as well as implementations [19, 6, 33, 77, 91, 37, 58, 65, 74], and most of the previous constructions are based on bilinear maps.

We can classify the PRE schemes by its properties. First category is the number of hops. In "single-hop" PRE, once the ciphertext for user A is re-encrypted for user B, that ciphertext cannot be re-encrypted to another user. The PRE scheme that can re-encrypt multiple times is called "multi-hop" PRE. The second category is the direction of re-encryption. If we can re-encrypt the ciphertext from user A to user B, but cannot re-encrypt it from user B to user A by using the re-encryption key, this setting is called "uni-directional". In contrast, if the re-encryption key can re-encrypt both directions, this setting is called "bi-directional".

Since the proposal of the (candidate) construction of multilinear maps [48], many cryptographic primitives based on them have been proposed. Especially, a cryptographic obfuscation that is secure in the meaning of indistinguishability [50] (indistinguishability obfuscation, $i\mathcal{O}$) has received much attention. These days, many researches about $i\mathcal{O}$ (construction, security analysis, and application to the cryptosystems, and so on) have been developed. Since $i\mathcal{O}$ is relatively a new cryptographic tool, it is theoretically important to understand what can be (or cannot be) constructed by using $i\mathcal{O}$. In this chapter, we show the construction of the PRE scheme based on $i\mathcal{O}$. We can see the idea of re-encryption in the construction of attribute-based encryption for polynomial size circuit [55] or bootstrapping techniques for the constructions of fully homomorphic encryption (e.g. [52]). Therefore, the result of this chapter may contribute for constructions of these cryptosystems.

5.1.2 Related Work

Blaze et al. formalized the concept of PRE cryptosystems [19] and proposed the first bidirectional PRE scheme based on ElGamal PKE scheme. Subsequently, Ateniese et al. [6], Canetti and Hohenberger [33], Libert and Vergnaud [77], and Chow et al. [37] proposed different PRE schemes with various properties. Shao and Cao [91] proposed a PRE scheme without pairing. Later, however, Zhang et al. [93] pointed out that it is not secure in the Libert-Vergnaud security model [77]. Subsequently, Matsuda et al. proposed a PRE scheme without pairing [82]. Later, however, Weng et al. [92] pointed out that their scheme is not CCA secure. Hanaoka et al. [58] proposed a new definition of CCA security in PRE and showed a generic construction of uni-directional PRE. Isshiki et al. [65] proposed a CCA secure PRE scheme. Kirshanova [74] proposed a lattice-based PRE scheme.

A research of cryptographic obfuscation has been started by Barak et al. [7]. In [7], they showed that it is impossible to construct the cryptographic obfuscator for arbitrary polynomial size circuit which is secure in the meaning of virtual black box (VBB). Hereafter, it has been showed the results of the VBB obfuscator for specific functions (e.g. point function [107], re-encryption [63], encrypted signature [57], and Hyperplane membership [36]). Garg et al. [50] showed the (candidate) construction of cryptographic obfuscator for arbitrary polynomial size circuit secure in the meaning of indistinguishability. After this proposal, many cryptographic primitives or cryptosystems based on it were proposed. For example, public key encryption and key encapsulation mechanism [98], functional encryption for polynomial size circuit [50, 106], signature [98, 95] and its universal aggregator [61], multiparty key exchange [26], multiparty computation [49], replacing random oracle [64], self-bilinear map [110], and so on.

In summary, to the best of our knowledge, there is no previous work about the construction of PRE scheme based on (standard) $i\mathcal{O}$.

5.1.3 Our Contribution

In this chapter, we show the construction of PRE scheme based on $i\mathcal{O}$ by adding two algorithms to the PKE scheme that was proposed by Sahai and Waters [98]. Although a re-encryption function from one public key to another can be considered as a randomized funcionality on the underlying plaintext and that functionality can be achieved by using probabilistic $i\mathcal{O}$ that was proposed by Canetti et al. [35], we need sub-exponential $i\mathcal{O}$ to construct the probabilistic $i\mathcal{O}$. In our construction, we use not probabilistic $i\mathcal{O}$ but standard $i\mathcal{O}$. Moreover, our scheme has some (good) properties as follows:

Unidirectional re-encryption key: A re-encryption key that can re-encrypt ciphertexts for user i to user j is generated by using a secret key of user i and a public key of use j. Ciphertexts for user i is re-encrypted for user j by using this re-encryption key unidirectionally.

Unbounded multi-hop re-encryption: Our scheme can execute the re-encryption multiple times. That is, we can re-encrypt the ciphertext that has been already re-encrypted. In addition, there is no limitation of the number of the re-encryption.

Constant size ciphertext: The size of the re-encrypted ciphertext does not expand by the re-encryption and it remains constant.

Fast decryption The decryption algorithm is completely the same as the Sahai and Waters' PKE scheme [98] and runs very fast.

To the best of our knowledge, this is the first construction of multi-hop and uni-directional PRE (MUPRE) scheme.

We can prove the CPA security for MUPRE that we define in this chapter. This CPA security, intuitively, is a definition that removes some oracles from Hanaoka et al.'s security definitions for CCA secure single-hop uni-directional PRE [58] and extends them for the multi-hop setting. In our scheme, a message space is limited to $\{0,1\}$ because of the security proof.

Intuition for the construction and security proof In PRE, if the proxy is allowed to decrypt the ciphertext for user i, the functionality of the re-encryption for user j can easily be achieved by a "decrypt-then-encrypt" approach. In this research, we adopt this approach by using the power of $i\mathcal{O}$. That is, we set the re-encryption key as the obfuscated circuit that a secret key of user i and a public key of user j are hardwired, and execute "decrypt-then-encrypt".

In the security proof, it becomes a main problem that how to invalidate the power of re-encryption key generation queries from an adversary. Since the re-encryption key is the obfuscated circuit, we have to consider that the challenge ciphertext may be input to that obfuscated circuit. To solve this problem, we adopt the methodology that we hardwire the key of puncturable PRF to generate a randomness for re-encryption, and puncture that key by using the challenge ciphertext. Thanks to this puncturable property, we can invalidate the re-encryption key from challenge user to another. To execute this operation, simulator has to know the challenge ciphertext before starting the security game between the challenger and an adversary. However, in our scheme, this is achievable because the message space is limited to $\{0,1\}$.

5.1.4 CPA Secure PKE scheme via $i\mathcal{O}$

We recall the IND-CPA secure PKE scheme that was proposed by Sahai and Waters' [98]. Our proposed PRE scheme can be considered as an extension of this PKE scheme. We can prove IND-CPA security of this scheme if the underlying PRG is computationally secure, $i\mathcal{O}$ is an indistinguishability obfuscator, and F is a secure PPRF.

$\mathsf{KG}(1^k)$: First, it chooses a PPRF key $K \leftarrow \{0,1\}^k$. Next, it creates the encryption circuit $\mathcal{C}_{\mathsf{Enc}}$ in Figure 5.1.[1] Then, it returns $(pk, sk) := (i\mathcal{O}(k, \mathcal{C}_{\mathsf{Enc}}), K)$.

$\mathsf{Enc}(pk, m, r)$: It returns $ct \leftarrow i\mathcal{O}(k, \mathcal{C}_{\mathsf{Enc}})(m, r)$.

$\mathsf{Dec}(sk, ct)$: It returns $m = ct_2 \oplus F(K_i, ct_1)$.

[1]The message space of this PKE scheme is not limited to $\{0,1\}$.

```
Encryption Circuit $\mathcal{C}_{\mathsf{Enc}}$:
Input: message $m \in \{0,1\}$, a randomness $r \in \{0,1\}^k$
Hardwire: PPRF key $K$
1. Compute $t := PRG(r)$.
2. Compute $ct := (ct_1, ct_2) = (t, F(K,t) \oplus m)$.
3. Output $ct$.
```

Figure 5.1: Encryption Circuit

5.1.5 Chapter Organization

The remainder of this chapter is organized as follows. In Section 5.2, we introduce the model and the security definition for MUPRE. In Section 5.3, we present our construction of an MUPRE scheme and prove its security. Section 5.4 is the conclusion of this chapter.

5.2 Proxy Re-encryption

In this section, we denote the model of multi-hop uni-directional proxy re-encryption (MUPRE) and define the CPA security for it.

5.2.1 Model

Here, we define the syntax of MUPRE. Formally, a MUPRE scheme consists of the following five algorithms (KG, Enc, RKG, REnc, Dec):

KG This is the key generation algorithm that takes 1^k as input, and outputs a public key pk and a secret key sk. This process is written as $(pk, sk) \leftarrow \mathsf{KG}(1^k)$.

Enc This is the encryption algorithm that takes a public key pk, a plaintext $m \in \mathcal{M}$, and a randomness r as input, and outputs a ciphertext ct. This process is written as $ct \leftarrow \mathsf{Enc}(pk, m, r)$.

RKG This is the re-encryption key generation algorithm that takes a secret key sk_i (of user i), a public key pk_j (of user j), and a randomness R as input, and outputs a re-encryption key $rk_{i\to j}$. This process is written as $rk_{i\to j} \leftarrow \mathsf{RKG}(sk_i, pk_j, R)$.

REnc This is the re-encryption algorithm that takes a ciphertext ct_i (for user i) and a re-encryption key $rk_{i\to j}$ as input, and outputs a ciphertext ct_j (for user j). This ct_j could be the special symbol $\perp$ meaning that $rk_{i\to j}$ or ct_i is invalid. This process is written as ct_j (*or* $\perp$) $\leftarrow \mathsf{REnc}(rk_{i\to j}, ct_i)$.

Dec This is the decryption algorithm that takes a secret key sk and a ciphertext ct as input, and outputs a decryption result m (which could be $\perp$). This process is written as $m \leftarrow \mathsf{Dec}(sk, ct)$.

Correctness. We say that an MUPRE scheme is correct if :

1. For any (sk, pk) output by KG, any message $m \in \mathcal{M}$, and any randomness r, we have: $m = \mathsf{Dec}(sk, \mathsf{Enc}(pk, m, r))$.

2. For all $n > 1$, any key pairs $(pk_1, sk_1) \cdots (pk_n, sk_n)$ output by KG, any message $m \in \mathcal{M}$, any randomness r and $\{R_i\}_{i \in [n]}$, all $i < n$, and any re-encryption keys $rk_{i \to i+1}$ output by $\mathsf{RKG}(sk_i, pk_{i+1}, R_i)$, we have: $m = \mathsf{Dec}(sk_{i+1}, \mathsf{REnc}(rk_{i \to i+1}, \mathsf{Enc}(pk_i, m, r)))$.

5.2.2 Security Definition

Here, we give the formal security definition of MUPRE. In this chapter, we consider the CPA security.

CPA Security of MUPRE. We define the chosen plaintext security for MUPRE with the following CPA-MUPRE game. This CPA-MUPRE security, intuitively, is a definition that removes a re-encryption query and a decryption query from Hanaoka et al.'s security definition for single-hop uni-directional PRE [58] and extends it for the multi-hop setting. This security game is parameterized by an integer $n \in \mathbf{N}$ and is played between the challenger $\mathcal{B}$ and an adversary $\mathcal{A}$:

Setup : Firstly, $\mathcal{B}$ generates honest user's key pairs $(pk_i, sk_i) \leftarrow \mathsf{KG}(1^k)$ for $i \in [n]$, and sets $\mathcal{PK} = \{pk_i\}_{i \in [n]}$. Next, $\mathcal{B}$ generates a challenge user's key pair $(pk^*, sk^*) \leftarrow \mathsf{KG}(1^k)$. Next, $\mathcal{B}$ $\mathcal{B}$ generates two lists $\mathcal{PK}^*_{\mathcal{S}} := \{pk^*\}$ and $\mathcal{PK}^*_{\mathcal{T}} := \{pk^*, \{pk_i\}_{i=1}^n\}$, and gives 1^k and $\mathcal{PK}^* := \mathcal{PK} \cup \{pk^*\}$ to $\mathcal{A}$. After that, $\mathcal{A}$ can adaptively make the following types of queries:

Re-encryption key generation (RKG) query : On input $(pk_i \in \mathcal{PK}^*, pk_j)$, where pk_j is an arbitrary public key of $\mathcal{A}$'s choice (for which $\mathcal{A}$ is not required to reveal the corresponding secret key), $\mathcal{B}$ responds as follows: If $pk_i \in \mathcal{PK}^*_{\mathcal{S}}$ and $pk_j \notin \mathcal{PK}^*_{\mathcal{T}}$, then the challenger responds with $\perp$. Otherwise, the challenger responds with $\mathsf{RKG}(sk_i, pk_j)$. When $pk_i \in \mathcal{PK}^*_{\mathcal{S}}$ and $pk_j \in \mathcal{PK}^*_{\mathcal{T}}$ is queried, $\mathcal{B}$ adds that pk_j to the list $\mathcal{PK}^*_{\mathcal{S}}$. When $pk_i \notin \mathcal{PK}^*_{\mathcal{S}}$ and $pk_i \in \mathcal{PK}^*_{\mathcal{T}}$ and $pk_j \notin \mathcal{PK}^*_{\mathcal{T}}$ is queried, $\mathcal{B}$ removes the pk_i from the list $\mathcal{PK}^*_{\mathcal{T}}$.[2]

Challenge query : This query is asked only once. On input (m_0, m_1) where (m_0, m_1) is a message pair of equal length, $\mathcal{B}$ picks a random bit $b \in \{0, 1\}$ and a randomness r^*, and computes $ct^* \leftarrow \mathsf{Enc}(pk^*, m_b, r^*)$. Then it gives ct^* to $\mathcal{A}$.

Finally, $\mathcal{A}$ outputs its guess b' for b and wins the game if $b = b'$. We define the advantage of $\mathcal{A}$ by $\mathsf{Adv}^{\mathsf{CPA\text{-}MUPRE}}_{(\mathcal{A},n)}(k) = |\Pr[b = b'] - \frac{1}{2}|$.

Definition 8. *We say that an MUPRE scheme is CPA secure, if for any PPT adversary $\mathcal{A}$ and all positive polynomials n, $\mathsf{Adv}^{\mathsf{CPA\text{-}MUPRE}}_{(\mathcal{A},n)}(k)$ is negligible.*

[2]This means that the honest users which are re-encrypted from the challenge user are equally treated as the challenge user. Similarly, the honest users which re-encrypt to the corrupt user are equally treated as the corrupt user.

5.3 Proposed Construction

In this section, we show our construction of the MUPRE scheme via the $i\mathcal{O}$.

5.3.1 Construction

Here, we show the intuition and construction of our scheme.

Intuition. If it is allowed to decrypt the ciphertext that is encrypted to user i (and then re-encrypt it for user j), the functionality of re-encryption can easily be achieved. In our scheme, we directly adopt the above policy by using the power of $i\mathcal{O}$. That is, we set the re-encryption key as the obfuscated "decrypt-then-encrypt" circuit. We realize the PRE scheme by adding two algorithms to Sahai and Waters' CPA secure PKE scheme. In the re-encryption circuit, we need to prepare the randomness to re-encrypt the decrypted plaintext for user j. This randomness is hardwired in the re-encryption circuit. To complete the security proof, we set the PPRF key K in the re-encryption circuit, and make the randomness for re-encryption by using this PPRF key K and the ciphertext which is input to the re-encryption circuit.

Our PRE scheme is constructed as follows. Let PRG is a pseudorandom number generator that maps $\{0,1\}^k$ to $\{0,1\}^{2k}$, F be a puncturable PRF that takes inputs of k bits and outputs 1 bit, and F' be a puncturable PRF that takes inputs of $2k+1$ bits and outputs k bits. In our scheme, the message space is limited to $\{0,1\}$.

$\mathsf{KG}(1^k)$: First, it chooses a PPRF key $K \leftarrow \{0,1\}^k$. Next, it constructs the encryption circuit $\mathcal{C}_{\mathsf{Enc}}$ in Figure 5.2. Then, it returns $(pk, sk) := (i\mathcal{O}(k, \mathcal{C}_{\mathsf{Enc}}), K)$.

```
Encryption Circuit $\mathcal{C}_{\mathsf{Enc}}$:
Input: message $m \in \{0,1\}$, randomness $r \in \{0,1\}^k$
Hardwire: PPRF key $K$
1. Compute $t := PRG(r)$.
2. Compute $ct := (ct_1, ct_2) = (t, F(K,t) \oplus m)$.
3. Output $ct$.
```

Figure 5.2: Encryption Circuit $\mathcal{C}_{\mathsf{Enc}}$

$\mathsf{Enc}(pk, m, r)$: It returns $ct \leftarrow i\mathcal{O}(k, \mathcal{C}_{\mathsf{Enc}})(m, r)$.

$\mathsf{RKG}(sk_i, pk_j, R)$: It constructs the re-encryption circuit $\mathcal{C}_{\mathsf{REnc}}$ in Figure 5.3. Then, it returns $rk_{i\to j} := i\mathcal{O}(k, \mathcal{C}_{\mathsf{REnc}})$.

$\mathsf{REnc}(ct_i, rk_{i\to j})$: It returns $ct_j \leftarrow i\mathcal{O}(k, \mathcal{C}_{\mathsf{REnc}_{i\to j}})(ct_i)$.

$\mathsf{Dec}(sk, ct)$: It returns $m = ct_2 \oplus F(K, ct_1)$.

5.3.2 Security Proof

Here, we show the formal security proof of our PRE scheme.

```
Re-encryption Circuit $\mathcal{C}_{\mathsf{REnc}}$:
Input: $ct_i$ (a ciphertext of user $i$).
Hardwire: $K_i$ (a secret key of user $i$),
          $i\mathcal{O}(k, \mathcal{C}_{\mathsf{Enc}_j})$ (a public key of user $j$),
          $K'$ (a PPRF key).
1. Compute $m = ct_2 \oplus F(K_i, ct_1)$.
2. Compute $r' := F'(K', ct_1 \| ct_2)$.
3. Compute $ct_j :=i\mathcal{O}(k, \mathcal{C}_{\mathsf{Enc}_j})(m, r')$.
4. Output $ct_j = (ct_1, ct_2) = (t', F(K_j, t') \oplus m)$.
   (Here, $t' := PRG(r')$.)
```

Figure 5.3: Re-encryption Circuit $\mathcal{C}_{\mathsf{REnc}}$

Intuition. The security proof of our PRE scheme follows the security proof of the CPA secure PKE scheme in [98] to some extent. However we cannot directly apply it for the proof of our scheme because of the following reason. In the security proof of [98], we puncture the challenge user's key K^*. If we try to apply this methodology directly to our proof, the challenger has to respond the re-encryption key generation queries from an adversary using the punctured key. However, this simulation does not succeed because that punctured key cannot decrypt the challenge ciphertext. To solve this problem, just like other security proof of PRE schemes, we have to invalidate the power of re-encryption key generation queries from an adversary before erasing the information of message from the challenge ciphertext. In our scheme, a randomness is hardwired in the re-encryption circuit to generate a re-encrypted ciphertext. We set a PPRF key K' as this randomness, and puncture this key K' by a challenge ciphertext ct^* to invalidate the power of re-encryption key generation queries. The challenger can determine that a challenge message pair (m_0, m_1) is $(0, 1)$ because we limit the message space to $\{0, 1\}$ in this research. Therefore, the simulator can generate the challenge ciphertext before the re-encryption key generation queries from an adversary and puncture the hardwired key by that challenge ciphertext. However, this is not enough. The simulator cannot generate the re-encryption circuit which correctly works when the input is $(ct_1^*, ct_2 (\neq ct_2^*))$. To solve this problem, we hardwire the outputs (=re-encrypted ciphertext) directly to the re-encryption circuit. After this changes, since we can erase the information of this hardwired outputs by using the power of the PPRF, we succeed to prove the security of our scheme same as the proof of [98].

Theorem 12. *If the $i\mathcal{O}$ is an indistinguishability obfuscator, PRG is computationally secure, F is a secure PPRF, then the proposed MUPRE scheme is CPA secure.*

Let $\mathcal{A}$ be any PPT adversary that attacks the CPA security of the proposed MUPRE scheme. We consider the following sequence of games.

Game 0: This is the CPA-MUPRE game.

Game 1: Same as Game 0, except for the encryption circuit of the challenge user. In the construction of $\mathcal{C}^*_{\mathsf{Enc}}$, it does not compute $t \leftarrow PRG(r)$. It randomly chooses $t^* \leftarrow \{0, 1\}^{2k}$ and uses it instead.

Game 2: Same as Game 1, except for the re-encryption circuit $\mathcal{C}_{\mathsf{REnc}_{*\to j}}$ which re-encrypts from the challenge user to a user j. We change the re-encryption circuit from $\mathcal{C}_{\mathsf{REnc}_{*\to j}}$ to $\mathcal{C}'_{\mathsf{REnc}_{*\to j}}$ as in Fig. 5.4. First, it calculates the outputs when $(ct_1^*, 0)$ and $(ct_1^*, 1)$ are input to the re-encryption circuit $\mathcal{C}_{\mathsf{REnc}_{*\to j}}$. Here, we describe this hardwired outputs (=re-encrypted ciphertexts) as $\widetilde{ct_0^*}$ and $\widetilde{ct_1^*}$, respectively. Next, it computes a punctured key $K^*(t^*) \leftarrow \mathsf{Punc}(K^*, t^*)$ and $K'(t^*\|0, t^*\|1) \leftarrow \mathsf{Punc}(K', \{t^*\|0, t^*\|1\})$. Then, it constructs a re-encryption circuit $\mathcal{C}'_{\mathsf{REnc}_{*\to j}}$ as in Fig.5.4. We execute this game hop q times (q is a number of RKG queries from $\mathcal{A}$) and change all the re-encryption circuits from the challenger user to a user of $\{PK^*\}(\in pk^* \cup \{pk_j\}_{j=1}^n)$.

```
Re-encryption Circuit $\mathcal{C}'_{\mathsf{REnc}_{*\to j}}$:
Input: $ct_i$ (a ciphertext of the challenge user).
Hardwire: Punctured key $K^*(t^*)$,
          $i\mathcal{O}(k, \mathcal{C}_{\mathsf{Enc}_j})$,
          Punctured key $K'(t^*\|0, t^*\|1))$,
          Two ciphertexts $\widetilde{ct_0^*}$ and $\widetilde{ct_1^*}$.
1. If the input ciphertext is $(ct_1^*, 0)$,
   then it outputs $\widetilde{ct_0^*}$, and else if $(ct_1^*, 1)$ is input,
   then outputs $\widetilde{ct_1^*}$.
   After that, it aborts without executing
   any following steps.
2. Compute $m = ct_2 \oplus F(K_i, ct_1)$.
3. Compute $r' := F'(K'(t^*\|0, t^*\|1), ct_1\|ct_2)$.
4. Compute $ct_j := i\mathcal{O}(k, \mathcal{C}_{\mathsf{Enc}_j})(m, r')$.
5. Output $ct_j = (ct_1, ct_2) = (t', F(K_j, t') \oplus m)$.
   Here, we denote $t' := PRG(r')$.
```

Figure 5.4: Re-encryption Circuit $\mathcal{C}'_{\mathsf{REnc}_{*\to j}}$

Game 3: Same as Game 2, except for the second component of the hardwired ciphertexts in the re-encryption circuit $\mathcal{C}'_{\mathsf{REnc}_{*\to j}}$ that re-encrypts from the challenge user to user j. It replaces the second component of the hardwired ciphertexts to a randomness which is randomly chosen from $\{0,1\}$. We execute this game hop $(n+1)$ times and change all re-encryption circuits that re-encrypt from the challenge user to the user of $\{PK^*\}(\in pk^* \cup \{pk_j\}_{j=1}^n)$.

Game 4: Same as Game 3, except for the hardwired key in the encryption circuit of the challenge user. It replaces the hardwired key K^* to the punctured key $K^*(t^*) \leftarrow \mathsf{Punc}(K^*, t^*)$ in the encryption circuit $\mathcal{C}^*_{\mathsf{Enc}}$.

Game 5: Same as Game 4, except for the second component of the challenge ciphertext. It does not choose $b \leftarrow \{0,1\}$ and compute $F(K^*(t^*), t^*) \oplus m_b$. Instead, it randomly chooses $z^* \leftarrow \{0,1\}$ and sets (t^*, z^*) as the challenge ciphertext.

For $i \in [5]$, let W_i be the event that the CPA-MUPRE adversary $\mathcal{A}$ succeeds in guessing the challenge bit b in Game i (i.e. $b' = b$ occurs). The advantage of $\mathcal{A}$ is, by definition, $\mathsf{Adv}^{\mathsf{CPA\text{-}MUPRE}}_{(\mathcal{A},n)}(k) = |\Pr[W_0] - \frac{1}{2}|$. By the triangle inequality, we have:

$$\mathsf{Adv}^{\mathsf{CPA\text{-}MUPRE}}_{(\mathcal{A},n)}(k) \leq \sum_{\alpha=0}^{4} |\Pr[W_\alpha] - \Pr[W_{\alpha+1}]| + |\Pr[W_5] - \frac{1}{2}| \tag{5.1}$$

We complete the proof by upperbounding each term in the right-hand side of the above inequality to be negligible.

Lemma 14. *If the PRG is secure pseudorandom number generator,* $|\Pr[W_0] - \Pr[W_1]|$ *is negligible.*

Proof of Lemma 14. We show that we can construct a PRG adversary $\mathcal{B}$ such that $\mathsf{Adv}^{\mathsf{PRG}}_{\mathcal{B}}(k) = |\Pr[W_0] - \Pr[W_1]|$. The description of $\mathcal{B}$ is as follows:

First, $\mathcal{B}$ is given γ^* from the challenger. Here, this γ^* is $PRG(s)$ ($s \leftarrow \{0,1\}^k$) or randomly chosen $\{0,1\}^{2k}$. $\mathcal{B}$ generates the honest users' key pairs $\{pk_i, sk_i\}_{i\in[n]}$. $\mathcal{B}$ sets $t^* := \gamma^*$, constructs the encryption circuit $\mathcal{C}_{\mathsf{Enc}^*}$, and generates a $pk^* := i\mathcal{O}(\mathcal{C}_{\mathsf{Enc}^*})$. Then, $\mathcal{B}$ gives $\{(pk_i\}_{i\in[n]}$ and pk^* to $\mathcal{A}$.

$\mathcal{B}$ answers to $\mathcal{A}$'s challenge query and RKG queries in exactly the same way as Game 0. Finally, when $\mathcal{A}$ outputs the guess bit $b' \in \{0,1\}$, $\mathcal{B}$ judges that γ^* is $PRG(s)$ if $b' = b$. Otherwise, $\mathcal{B}$ judges that γ^* is $\{0,1\}^{2k}$. Then, $\mathcal{B}$ outputs this result and aborts.

The above completes the description of $\mathcal{B}$. Note that when γ^* is $PRG(s)$, $\mathcal{B}$ completely simulates the Game 0 to $\mathcal{A}$, and when γ^* is $\{0,1\}^{2k}$, $\mathcal{B}$ completely simulates the Game 1 to $\mathcal{A}$. Therefore, if the $\mathsf{Adv}^{\mathsf{PRG}}_{\mathcal{B}}(k)$ is negligible, $|\Pr[W_0] - \Pr[W_1]|$ is also negligible. □

Lemma 15. *If the* $i\mathcal{O}$ *is indistinguishability obfuscator,* $|\Pr[W_1] - \Pr[W_2]|$ *is negligible.*

Proof of Lemma 15. We show that we can construct a $i\mathcal{O}$ adversary $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ such that $\mathsf{Adv}^{i\mathcal{O}}_{(\mathcal{B}_1,\mathcal{B}_2)}(k) = |\Pr[W_1] - \Pr[W_2]|$. The description of $\mathcal{B}$ is as follows:

First, $\mathcal{B}_1$ is given 1^k from the challenger. $\mathcal{B}$ runs $\mathsf{KG}(1^k)$ and generates the challenge user's key pair (pk^*, sk^*) and the honest user's key pairs $\{pk_i, sk_i\}_{i\in[n]}$. Then, $\mathcal{B}_1$ randomly chooses $t^* \in \{0,1\}^{2k}$. $\mathcal{B}$ sets $\sigma := (1^k, pk^*, sk^*, \{pk_i, sk_i\}_{i\in[n]}, t^*)$. After that, $\mathcal{B}_1$ constructs a re-encryption circuit $\mathcal{C}_{\mathsf{REnc}}$ as in Fig.5.3. We denote this circuit as $\mathcal{C}_0$. Next, $\mathcal{B}_1$ inputs $(t^*, 0)$ and $(t^*, 1)$ to this re-encryption circuit $\mathcal{C}_{\mathsf{REnc}}$ and computes the outputs. Then, $\mathcal{B}_1$ hardwires these two outputs to the $\mathcal{C}_{\mathsf{REnc}}$, and rewrite this circuit to output the hardwired value when $(t^*, 0)$ and $(t^*, 1)$ are input, respectively. In addition, $\mathcal{B}_1$ computes the punctured key $K^*(t^*)$ and $K'(S \in \{t^*\|0, t^*\|1\})$, and replaces the PPRF key K^* to $K^*(t^*)$ and K' to $K'(S \in \{t^*\|0, t^*\|1\})$ in re-encryption circuit. We denote this circuit as $\mathcal{C}_1$. Note that the functionality of these two circuits $\mathcal{C}_0$ and $\mathcal{C}_1$ are completely same. $\mathcal{B}_1$ outputs $(\sigma, \mathcal{C}_0, \mathcal{C}_1)$ and aborts.

$\mathcal{B}_2$ is given σ and $i\mathcal{O}(k, \mathcal{C}^*)$ from the challenger. Here, this $i\mathcal{O}(k, \mathcal{C}^*)$ is $i\mathcal{O}(k, \mathcal{C}_0)$ or $i\mathcal{O}(k, \mathcal{C}_1)$. $\mathcal{B}_2$ gets out pk^* and $\{pk_i\}_{i\in[n]}$ from the σ and inputs them to $\mathcal{A}$. $\mathcal{B}_2$ answers to $\mathcal{A}$'s challenge query and RKG queries in exactly the same way as Game 1 other than RKG query from the challenge user to user j. When the re-encryption key from the challenge user to user j is queried from $\mathcal{A}$, $\mathcal{B}_2$ returns the $i\mathcal{O}(k, \mathcal{C}^*)$ to $\mathcal{A}$. Finally, when $\mathcal{A}$ outputs the guess bit $b' \in \{0,1\}$, $\mathcal{B}_2$ judges that $i\mathcal{O}(k, \mathcal{C}^*)$ is $i\mathcal{O}(k, \mathcal{C}_0)$ if $b' = b$. Otherwise, $\mathcal{B}_2$ judges that $i\mathcal{O}$ $(k, \mathcal{C}^*)$ is $i\mathcal{O}(k, \mathcal{C}_1)$. Then, $\mathcal{B}_2$ outputs this result and aborts.

The above completes the description of $\mathcal{B}$. Note that when $i\mathcal{O}(k, \mathcal{C}^*)$ is $i\mathcal{O}(k, \mathcal{C}_0)$, $\mathcal{B}$ completely simulates the Game 1 to $\mathcal{A}$, and when $i\mathcal{O}(k, \mathcal{C}^*)$ is $i\mathcal{O}(k, \mathcal{C}_1)$, $\mathcal{B}$ completely simulates the Game 2 to $\mathcal{A}$. Therefore, if the $\mathsf{Adv}^{i\mathcal{O}}_{(\mathcal{B}_1,\mathcal{B}_2)}(k)$ is negligible, $|\Pr[W_1] - \Pr[W_2]|$ is also negligible. □

Lemma 16. *If the F is secure PPRF, $|\Pr[W_2] - \Pr[W_3]|$ is negligible.*

Proof of Lemma 16. We show that we can construct a PPRF adversary $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ such that $\mathsf{Adv}^{\mathsf{PPRF}}_{(\mathcal{B}_1,\mathcal{B}_2)}(k) = |\Pr[W_2] - \Pr[W_3]|$. The description of $\mathcal{B}$ is as follows:

First, $\mathcal{B}_1$ is given 1^k from the challenger. $\mathcal{B}_1$ randomly chooses $t^* \leftarrow \{0,1\}^{2k}$. $\mathcal{B}_1$ sets the puncture point $S := t^*$, outputs S and state $\sigma := 1^k$, and aborts.

$\mathcal{B}_2$ is given $(\tau, K^*(S), S, Z)$ from the challenger. Here, this Z is $F(K^*, S)$ or $U_{m(k \cdot |S|)}$. First, $\mathcal{B}_2$ sets $sk^* := K^*(S)$. Then, $\mathcal{B}_2$ constructs a re-encryption circuit $\mathcal{C}_{\mathsf{REnc}}$ as in Fig.5.3 by using sk^* and t^*, and sets $pk^* := i\mathcal{O}(\mathcal{C}_{\mathsf{Enc}})$. $\mathcal{B}_2$ generates the honest user's key pairs $\{pk_i, sk_i\}_{i\in[n]}$ and inputs pk^* and $\{pk_i\}_{i\in[n]}$ to $\mathcal{A}$. $\mathcal{B}_2$ answers to $\mathcal{A}$'s challenge query and RKG queries in exactly the same way as Game 2 other than the RKG query from the challenge user to user j. When the RKG query from the challenge user are asked, $\mathcal{B}_2$ responds as follows: First, $\mathcal{B}_2$ computes $y_0 = 0 \oplus Z$ and $y_1 = 1 \oplus Z$. Next, $\mathcal{B}_2$ randomly chooses PPRF key K'. Then, $\mathcal{B}_2$ computes $\widehat{ct_i} = y_i \oplus F(K', t^* \| i)$ for $i = 0$ and $i = 1$. After that, $\mathcal{B}_2$ computes $K'(t^*\|0, t^*\|1) \leftarrow \mathsf{Punc}(K', \{t^*\|0, t^*\|1\})$ and constructs a re-encryption circuit as in Fig.5.3 by using $t^*, K(t^*)$ and $K'(t^*\|0, t^*\|1)$. Differently from the Game 2, $\mathcal{B}$ sets two hardwired ciphertexts as $\widehat{ct_0}$ and $\widehat{ct_1}$, respectively. Then, $\mathcal{B}$ obfuscates this circuit and returns it as a re-encryption key from the challenge user to user j. Finally, when $\mathcal{A}$ outputs the guess bit $b' \in \{0,1\}$, $\mathcal{B}_2$ judges that Z is $F(K^*, S)$ if $b' = b$. Otherwise, $\mathcal{B}_2$ judges that Z is $U_{m(k)\cdot|S|}$. Then, $\mathcal{B}_2$ outputs this result and aborts.

The above completes the description of $\mathcal{B}$. Note that when Z is $F(K^*, S)$, $\mathcal{B}$ completely simulates the Game 2 to $\mathcal{A}$, and when Z is $U_{m(k)\cdot|S|}$, $\mathcal{B}$ completely simulates the Game 3 to $\mathcal{A}$. Therefore, if the $\mathsf{Adv}^{\mathsf{PPRF}}_{(\mathcal{B}_1,\mathcal{B}_2)}(k)$ is negligible, $|\Pr[W_2] - \Pr[W_3]|$ is also negligible. □

Lemma 17. *If the $i\mathcal{O}$ is indistinguishability obfuscator, $|\Pr[W_3] - \Pr[W_4]|$ is negligible.*

Proof of Lemma 17. We show that we can construct a $i\mathcal{O}$ adversary $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ such that $\mathsf{Adv}^{i\mathcal{O}}_{(\mathcal{B}_1,\mathcal{B}_2)}(k) = |\Pr[W_3] - \Pr[W_4]|$. The description of $\mathcal{B}$ is as follows:

First, $\mathcal{B}_1$ is given 1^k from the challenger, $\mathcal{B}$ runs $\mathsf{KG}(1^k)$ and generates the challenge user's key pair (pk^*, sk^*) and the honest user's key pairs $\{pk_i, sk_i\}_{i\in[n]}$. Then, $\mathcal{B}_1$ randomly chooses $t^* \in \{0,1\}^{2k}$ and sets $\sigma := (K^*, t^*)$. After that, $\mathcal{B}_1$ randomly chooses a PPRF key K^* and constructs a encryption circuit as in Fig.5.2 by using K^*. We denote this circuit as $\mathcal{C}_0$. Next, $\mathcal{B}_1$ computes $K(t^*) \leftarrow \mathsf{Punc}(K^*, t^*)$ and constructs a encryption circuit as in Fig.5.2 by using $K^*(t^*)$. We denote this circuit as $\mathcal{C}_1$. Note that the functionality of these two circuits $\mathcal{C}_0$ and $\mathcal{C}_1$ are completely same except for when the t^* is input. However, that t^* is not a image of the PRG with overwhelming probability. $\mathcal{B}_1$ outputs $(\sigma, \mathcal{C}_0, \mathcal{C}_1)$ and aborts.

$\mathcal{B}_2$ is given σ and $i\mathcal{O}(k, \mathcal{C}^*)$ from the challenger. Here, this $i\mathcal{O}(k, \mathcal{C}^*)$ is $i\mathcal{O}(k, \mathcal{C}_0)$ or $i\mathcal{O}(k, \mathcal{C}_1)$. $\mathcal{B}_2$ gets out $\{pk_i\}_{i\in[n]}$ from the σ and sets $pk^* := i\mathcal{O}(k, \mathcal{C}^*)$. Then, $\mathcal{B}_2$ inputs them to $\mathcal{A}$. $\mathcal{B}_2$ answers to $\mathcal{A}$'s challenge query and the RKG queries in exactly the same way as Game 3. Finally, when $\mathcal{A}$ outputs the guess bit $b' \in \{0,1\}$, $\mathcal{B}_2$ judges that $i\mathcal{O}(k, \mathcal{C}^*)$

is $i\mathcal{O}(k, \mathcal{C}_0)$ if $b' = b$. Otherwise, $\mathcal{B}_2$ judges that $i\mathcal{O}$ $(k, \mathcal{C}^*)$ is $i\mathcal{O}(k, \mathcal{C}_1)$. Then, $\mathcal{B}_2$ outputs this result and aborts.

The above completes the description of $\mathcal{B}$. Note that when $i\mathcal{O}(k, \mathcal{C}^*)$ is $i\mathcal{O}(k, \mathcal{C}_0)$, $\mathcal{B}$ completely simulates the Game 3 to $\mathcal{A}$, and when $i\mathcal{O}(k, \mathcal{C}^*)$ is $i\mathcal{O}(k, \mathcal{C}_1)$, $\mathcal{B}$ completely simulates the Game 4 to $\mathcal{A}$. Therefore, if the $\mathsf{Adv}^{i\mathcal{O}}_{(\mathcal{B}_1,\mathcal{B}_2)}(k)$ is negligible, $|\Pr[W_3] - \Pr[W_4]|$ is also negligible. □

Lemma 18. *If the F is secure PPRF, $|\Pr[W_4] - \Pr[W_5]|$ is negligible.*

Proof of Lemma 18. We show that we can construct a PPRF adversary $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ such that $\mathsf{Adv}^{\mathsf{PPRF}}_{(\mathcal{B}_1,\mathcal{B}_2)}(k) = |\Pr[W_4] - \Pr[W_5]|$. The description of $\mathcal{B}$ is as follows:

First, $\mathcal{B}_1$ is given 1^k from the challenger. $\mathcal{B}_1$ randomly chooses $t^* \leftarrow \{0,1\}^{2k}$. $\mathcal{B}_1$ sets the puncture point $S := t^*$, outputs S and state $\sigma := 1^k$, and aborts.

$\mathcal{B}_2$ is given $(\tau, K^*(S), S, Z)$ from the challenger. Here, this Z is $F(K^*, S)$ or $U_{m(k \cdot |S|)}$. First, $\mathcal{B}_2$ sets $sk^* := K^*(S)$. Then, $\mathcal{B}_2$ constructs a encryption circuit $\mathcal{C}_{\mathsf{Enc}}$ as in Fig.5.2 by using sk^* and t^*, and sets $pk^* := i\mathcal{O}(\mathcal{C}_{\mathsf{Enc}})$. $\mathcal{B}_2$ generates the honest users' key pairs $(pk_i, sk_i)_{i \in [n]}$ and inputs pk^* and $\{pk_i\}_{i \in [n]}$ to $\mathcal{A}$. $\mathcal{B}_2$ answers to $\mathcal{A}$'s RKG queries in exactly the same way as Game 4. When $\mathcal{A}$ asks challenge query, $\mathcal{B}_2$ responds as follows: First, $\mathcal{B}_2$ chooses random $b \in \{0,1\}$. Then, $\mathcal{B}_2$ sets $ct^* = (ct_1^*, ct_2^*) := (t^*, Z \oplus m_b)$ and returns ct^* to $\mathcal{A}$. Finally, when $\mathcal{A}$ outputs the guess bit $b' \in \{0,1\}$, $\mathcal{B}_2$ judges that Z is $F(K^*, S)$ if $b' = b$. Otherwise, $\mathcal{B}_2$ judges that Z is $U_{m(k) \cdot |S|}$. Then, $\mathcal{B}_2$ outputs this result and aborts.

The above completes the description of $\mathcal{B}$. Note that when Z is $F(K^*, S)$, $\mathcal{B}$ completely simulates the Game 4 to $\mathcal{A}$, and when Z is $U_{m(k) \cdot |S|}$, $\mathcal{B}$ completely simulates the Game 5 to $\mathcal{A}$. Therefore, if the $\mathsf{Adv}^{\mathsf{PPRF}}_{(\mathcal{B}_1,\mathcal{B}_2)}(k)$ is negligible, $|\Pr[W_4] - \Pr[W_5]|$ is also negligible. □

Lemma 19. $|\Pr[W_5] - \frac{1}{2}| = 0$.

Proof of Lemma 19. In Game 5, the challenge ciphertext contains no information about challenge bit b. Therefore, $|\Pr[W_5] - \frac{1}{2}| = 0$. □

Lemmas 14 to 19 guarantee that the right hand side of the inequation (5.1) is negligible, and thus $\mathcal{A}$ has negligible advantage in the CPA-MUPRE game. Since the negligible upperbound of the advantage can be shown for any CPA-MUPRE adversary $\mathcal{A}$, we conclude that our proposed MUPRE scheme is CPA secure. This completes the proof of Theorem 12.

5.4 Conclusion

In this chapter, we proposed a MUPRE scheme via $i\mathcal{O}$ and proved its CPA security. Achieving the CCA-secure MUPRE is future work. In addition, as the technique which is used in the security proof of this research seems interesting, finding the applications of this technique is also future work.

Although the $i\mathcal{O}$ is theoretically a big hummer, we cannot expect its practicality for a while. In our construction, we need an obfuscator for "Dec-then-ReEnc" circuit and need not a obfuscator for arbitrary circuit. Cryptographic obfuscator for specific circuit may work within the realistic time. In this situation, we can find the practical meaning for the cryptosystems based on $i\mathcal{O}$. We hope the research directions like this in the future.

Chapter 6

Provably Secure Password Reset Protocols

6.1 Introduction

6.1.1 Background and Motivation

User authentication is one of the fundamental research themes not only in theory but also in practice. Although there are some unsolved problems, password-based user authentication systems are widely used in practice because of their usability and some other advantages. One of the unsolved problems is that users may forget their passwords, and this problem cannot be avoided as long as the user is a human. One may expect that we can overcome this problem by alternatives such as graphical password [105], biometrics [68], etc. However, these authentication methods also need backup authentication systems when users are unable to perform the primary authentication due to operational errors, physical problems, etc. To make matters worse, they have their own problems. Their problems in primary authentication may include insufficient entropy of biometric information, wolves/lambs [41], and the initial/operational cost of additional devices. Their problems in backup authentication can be even more complicated. For example, if an officer can attend on users at authentication devices, some of the problems in the primary authentication may be solved promptly (e.g. operational errors) but others may stay still hard (e.g. severe physical problems such as a burn on a finger in biometrics). It should be noted that relying on such an officer costs a lot, and may cause some other problems (e.g. privacy problems).

A potential advantage of password-based authentication is a clear reset scenario in backup authentication: if a password which is valid in the next execution of the primary authentication can be securely delivered to a user who is unable to perform the primary authentication, the user can be rescued by the delivered password. Thus we reach our main research question: *can we design a provably secure protocol for this reset (namely, a provably secure password reset protocol)?*

In existing websites which rely on password-based authentication, backup authentication for reissuing a valid password often uses personal information (e.g. e-mail address, answers to secret questions, etc.) which was provided by the user in the initial registration procedure. Although such backup authentication mechanisms are convenient for users, their security relies on nothing but heuristic evaluation or intuition. Even in the case of a provably secure primary authentication protocol such as PAKE (password-authenticated key exchange) [17, 16, 29, 71], its provable security is meaningless in practice if it is used with a heuristic (and hence, potentially weak) password reset protocol. Bonneau et al. [27], in fact, showed that the security of the secret question (which is a popular backup authentication mechanism) is not enough. In order to prevent the cat-and-mouse game between attacks and heuristic countermeasures completely, not only primary authentication but also backup authentication should be provably secure.

Inspired by the above views, we investigate provably secure password reset protocols in this chapter. Differently from the existing usable security papers regarding backup authentication [113, 70, 67, 94, 100, 101, 96, 69, 59], we follow the provable security approach in cryptography.

We firstly define a model, then provide security definitions of a password reset protocol, and finally show provably secure protocols. In particular, we propose generic constructions of a provably secure password reset protocol based on a pseudorandom function and public key encryption.

The Difficulty of Model Design In a password reset protocol, it is demanded that a user can re-register a refreshed password under the situation that the user does not have his/her password. However, in an authentication system that uses only user's identity and password which are provided by the user in the initial registration procedure, a legitimate user who forgets his/her password and an adversary are indistinguishable because the former does not have a unique information. Therefore, when we consider a password reset protocol, we have to adopt a different model from standard user authentication systems. We will easily be able to construct a provably secure password reset protocol by assuming the existence of a trusted third party (TTP) in the same way as the case of applied cryptosystems such as identity-based encryption (IBE) [102, 24]. However, as the fact that mass surveillance has been carried out by NSA becomes clear now, we would like to avoid assuming such a big brother. A security definition should capture the real world, and be achievable by a practical protocol (so that we need not require an unrealistic assumption such as a secure channel between a user and a server with which a refreshed password can always be securely delivered).

6.1.2 Our Contribution

In this chapter, we consider a provably secure password reset protocol, formalize a model and security definitions, propose a construction, prove its security, and show the efficiency of our protocol via prototype implementation. Because of the difficulty that we point out in Section 6.1.1, we have to consider a different model from standard user authentication systems to construct a provably secure password reset protocol. In this chapter, we propose a model that introduces a key for password reset. In our model, the system generates a reset key in the initial registration procedure. When a user wants to register/reset his/her password, he/she uses this key in the password registration/reset phase. We assume that the reset key is securely stored, and the user does not lose the reset key even if the user forgets his/her password. We discuss the validity of this assumption in Section 6.1.3. In our model, a user can choose both primary and refreshed passwords. Moreover, our model also captures the leakage of a password. In a standard password-based authentication, if an adversary steals a password of a legitimate user and changes the inside states of the authentication system (e.g. password, information for backup authentication, etc.), the legitimate user cannot take back his/her user account. However, in our model, a user who had his/her password stolen and lost the user account can take back it by using his/her reset key which is securely stored. To the best of our knowledge, none of the previous user authentication protocols can realize this property.

First, in this chapter, we formalize a model and security definitions of a password reset protocol. For simplicity, first, we consider security against passive adversaries. In this security definition, an adversary is allowed to get all the information by observing the transcripts between a user and the server. After that (in Section 6.4.2), we also consider security against active adversaries that can mount man-in-the-middle attacks and concurrent attacks by extending the security definitions for passive adversaries. Then we propose a generic construction based on a pseudorandom function and public key encryption. The security that we require for these building blocks is popular one, and a number of concrete schemes that satisfy our requirements have been already known. Therefore, we can construct many efficient and simple concrete password reset protocols from this generic construction. Finally, we evaluate the performance of our protocol by implementing a prototype. The result shows that our protocol has good efficiency, and can be used in practice.

6.1.3 Introduction of Reset Key

In our model, we introduce a special key for password reset, which we call a reset key, and assume that it is securely stored. Indeed, this is relatively a strong assumption. Here, we explain the reasons why we adopt this methodology and how this assumption is reasonable.

1. This assumption is similar to the setting of key-insulated cryptography [43], where a secret key is updatable by a higher-level secret key that is assumed to be securely stored. We emphasize that this is not only widely accepted in public-key cryptography, but its potential feasibility was pointed out in practical security researches (e.g. [80, 51]). One may think that if we allow these assumptions, then we may as well assume that a user does not lose his/her password. However, as considered in key-insulated cryptography, it is (to some extent) reasonable to assume that the user does not lose the reset key because the reset key is only required in the case of emergency and can therefore be securely stored. One may also think that this assumption does not hold in the system where a user is forced to change his/her password frequently. However, the assumption still holds because the user who remembers his/her password can change his/her password in the system without using a reset key.

2. There exist security protocols in the real world which work as (a valiant of) the key-insulated setting. For example, widely used hardware security tokens have a long-term key in itself and generate a one-time password by using it. If this long-time key is revealed, we can compute a one-time password. We can consider that the security of these tokens is assured in the key-insulated model.

3. Our model (to some extent) captures the implementations of password reset protocols used in the real world. In Facebook, for example, information to reset a password is sent to a user's email address registered in the initial registration procedure. Here, the user's password to log-in to the email account can be seen as a reset key. Our model captures an intuition in such real-world protocols, and provides provable security simultaneously.

4. The problem that we have to solve this time is the one that we do not really care in the research of cryptography. For example, in the case of IBE, TTP receives a user's ID (e.g. user's email address) and derives a corresponding secret key. However, the framework of IBE does not offer how to distinguish whether the user really has that ID or not. In order to use IBE in practice, we have to rely on an infrastructure other than IBE. An easygoing way to solve this problem is, for instance, to send a derived secret key to a user's email address. In this example, we can think of the password to log-in to the email account as a reset key. This is almost the same case as the above example of account recovery in Facebook.

Even if we introduce a reset key in the password reset protocol, it does not mean that the construction or security proof becomes trivial. Intuitively, we can achieve the password reset functionality by using a symmetric key encryption (SKE). User regards a reset key as a key of SKE, encrypts a new password by using a reset key, and sends a ciphertext to the server. However, this simple method is vulnerable to a replay attack. Moreover, this SKE-based construction seems to fail to satisfy the security definition which is descibed in Section 6.2.2. We will explain the reason of this in Section 6.3.

6.1.4 Chapter Organization

The remainder of this chapter is organized as follows. In Section 6.2, we introduce the models and security definitions for a password reset protocol. In addition, we explain how this protocol is used in practice. In Section 6.3, we present our generic construction of a password reset protocol and prove its security. In Section 6.4, we discuss the extension of our proposed construction. In Section 6.5, we show the performance evaluation of our protocol. Section 6.6 is the conclusion.

6.2 Password Reset Protocol

In this section, we present the model and the security definitions of a password reset protocol. The proposed model and the security definitions (which are described in Section 6.2.2) are an extension of the existing model and the security definition for identity-based identification [75, 14]. We assume that a reset key is securely distributed to the client in some way. We also assume that the reset key is securely stored, and the client does not lose the reset key even if the client forgets his/her password as assumed in key-insulated cryptography [43]. One may think that if we allow these assumptions, then we may as well assume that a client does not lose his/her password. As we discussed in Section 6.1.3, however, it is (to some extent) reasonable to assume that the client does not lose the reset key.

6.2.1 Model

A password reset protocol consists of the following two PPT algorithms $(\mathsf{SSetup}, \mathsf{RKG})$ and two subprotocols
$\mathsf{PRR}(C_P(\cdot) \leftrightarrow S_P(\cdot))$ and $\mathsf{Auth}(C_A(\cdot) \leftrightarrow S_A(\cdot))$. Let $\mathcal{PW}$ denote the password space.

SSetup This is the server setup algorithm that takes 1^k as input, and outputs a public parameter pp and a secret key sk. This process is written as $(pp, sk) \leftarrow \mathsf{SSetup}(1^k)$.

RKG This is the reset key generation algorithm that takes a secret key sk and a client's identity $ID \in \{0,1\}^*$ as input, and outputs a reset key rk. This process is written as $rk \leftarrow \mathsf{RKG}(sk, ID)$.

PRR This is the interactive protocol for password (re-)registration between the PPT algorithms C_P and S_P. Let the identity of a client that runs the algorithm C_P be ID. The algorithm C_P takes the identity ID, a password $pw \in \mathcal{PW}$, and a reset key rk as input, the algorithm S_P takes the identity ID, a reset key rk, and a secret key sk as input, and then these algorithms interact with each other. As a result of the interaction, C_P and S_P locally output ϕ and pw_s (which could be $\perp$), respectively[1]. This process is written as $(\phi, pw_s) \leftarrow \mathsf{PRR}(C_P(ID, pw, rk) \leftrightarrow S_P(ID, rk, sk))$[2].

Auth This is the interactive authentication protocol between the PPT algorithms C_A and S_A. Let the identity of a client that runs the algorithm C_A be ID. The algorithm C_A takes the identity ID and a password $pw \in \mathcal{PW}$ as input, the algorithm S_A takes the identity ID, pw_s, and a secret key sk as input, and then these algorithms interact with each other. As a result of the interaction, C_A and S_A locally output ϕ and $\top/\perp$, respectively, where $\top$ (resp. $\perp$) means "accept" (resp. "reject"). This process is written as $(\phi, \top/\perp) \leftarrow \mathsf{Auth}(C_A(ID, pw) \leftrightarrow S_A(ID, pw_s, sk))$[2].

Correctness We require the following correctness property for a password reset protocol. For any $(pp, sk) \leftarrow \mathsf{SSetup}(1^k)$, any $pw \in \mathcal{PW}$, any $ID \in \{0,1\}^*$, any $rk \leftarrow \mathsf{RKG}(sk, ID)$, any $(\phi, pw_s) \leftarrow \mathsf{PRR}(C_P(ID, pw, rk) \leftrightarrow S_P(ID, rk, sk))$, we have $(\phi, \top) \leftarrow \mathsf{Auth}(C_A(ID, pw) \leftrightarrow S_A(ID, pw_s, sk))$.

Remark Here, we explain how to use this protocol in practice. First, the server executes SSetup and generates a public parameter pp and a secret key sk. In the initial registration procedure, the server runs RKG and generates a reset key rk for each client. When a client registers an initial password, a client and the server execute PRR interactively, and a client registers an initial password pw. In this PRR protocol, the client executes the algorithm C_P, and the server executes the algorithm S_P. A client authenticates himself/herself by using an ID and a pw in Auth protocol. In this Auth protocol, a client executes the algorithm C_A, and the server executes the algorithm S_A. When a client forgets his/her

[1]This pw_s denotes the information that is supposed to be stored in the server and is used when the user with password pw next time requests authentication of him/her. We do not require $pw_s = pw$ hold in general. (See, e.g. the construction in Section 6.4.)

[2]The algorithms S_P and S_A of our proposed scheme need not use a secret key sk. In general, however, S_P and S_A are allowed to use sk. In our extended password reset protocol in Section 6.4, in fact, we need sk in both of the algorithms S_P and S_A.

password pw, a client and the server execute PRR interactively, and the client re-registers a refreshed password pw'. In this PRR protocol, the client executes the algorithm C_P, and the server executes the algorithm S_P.

6.2.2 Security Definitions

In this subsection, we give the formal security definitions of a password reset protocol and explain what situations our definitions capture. In this section, we consider passive attacks. Later (in Section 6.4.2), we also consider active attacks. The passive attack captures the situation in which an adversary observes the transactions between a client and the server from outside. In our security definitions, an adversary can get all the information by observing the transactions between a client and the server. Here, we have to consider the two types of security for a password reset protocol. The first one is that an adversary who does not have a correct password cannot pass the authentication. We call it security against impersonation attacks. The second one is that an adversary who does not have a correct reset key cannot (re-)register a password. We call it security against illegal registration attacks.

Let $\mathcal{D}$ be the dictionary of user's password.

Impersonation First, we consider security against impersonation under passive attacks (Imp-PA) for a password reset protocol. This security is defined using the following Imp-PA game which is played by the challenger $\mathcal{B}$ and an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$. First, $\mathcal{B}$ executes $(pp, sk) \leftarrow \mathsf{SSetup}(1^k)$, and generates an empty list L into which tuples of the form $(ID, pw, pw_s, rk, frag_p, frag_r)$ where $frag_p, frag_r \in \{0,1\}$ will be stored. These $frag_p$ and $frag_r$ are used to indicate whether a client with ID is "corrupted" by $\mathcal{A}$ in the sense that either pw or rk is known to $\mathcal{A}$, in which case $\mathcal{A}$ is not allowed to use the ID for its attack. After key generation, $\mathcal{B}$ gives pp to $\mathcal{A}_1$. Then $\mathcal{A}_1$ can adaptively make the following types of queries.

Client create query ($\mathsf{CCreate}$)**:** On input $ID \in \{0,1\}^*$, $\mathcal{B}$ responds as follows. If there exists a tuple of the form $(ID, *, *, *, *, *)$ in the list L, $\mathcal{B}$ does nothing. Otherwise, $\mathcal{B}$ executes $rk \leftarrow \mathsf{RKG}(sk, ID)$, and stores $(ID, \perp, \perp, rk, 0, 1)$ into the list L. If $\mathcal{A}_1$ makes the following queries $(\mathsf{RKR}, \mathsf{PRR}, \mathsf{Auth})$ with an identity ID, then this ID must have appeared as a $\mathsf{CCreate}$ query (and thus be stored in the list L).

Reset key reveal query (RKR)**:** On input ID, $\mathcal{B}$ finds the tuple of the form $(ID, *, *, rk, *, *)$ in the list L, and returns rk to $\mathcal{A}_1$. Then, $\mathcal{B}$ updates the tuple in the list L by $(ID, *, *, rk, *, 0)$.

Password (re-)registration transcript query ($\mathsf{Trans}_{\mathsf{PRR}}$)**:** On input (ID, pw'), $\mathcal{B}$ responds as follows.

1. If $pw' \in \mathcal{PW}$, $\mathcal{B}$ first finds the tuple $(ID, *, *, rk, *, *)$ in the list L, executes $(\phi, pw'_s) \leftarrow \mathsf{PRR}(C_P(ID, pw', rk) \leftrightarrow S_P(ID, rk, sk))$, and returns the transcript $trans_{\mathsf{PRR}}$ of PRR and the result of registration $\top/\perp$ to $\mathcal{A}_1$. Then, $\mathcal{B}$ updates the tuple in the list L by $(ID, pw', pw'_s, *, 0, *)$.

2. If $pw' = \phi$, $\mathcal{B}$ first chooses a random password $pw' \in \mathcal{D}$. Next, $\mathcal{B}$ finds the tuple $(ID, *, *, rk, *, *)$ in the list L, executes $(\phi, pw'_s) \leftarrow \mathsf{PRR}(C_P(ID, pw', rk) \leftrightarrow S_P(ID, rk, sk))$, and returns the transcript $trans_{\mathsf{PRR}}$ of PRR and the result of registration $\top/\bot$ to $\mathcal{A}_1$. Then, $\mathcal{B}$ updates the tuple in the list L by $(ID, pw', pw'_s, *, 1, *)$.

Authentication transcript query ($\mathsf{Trans}_{\mathsf{Auth}}$)**:** On input (ID, pw'), $\mathcal{B}$ responds as follows.

1. If $pw' \in \mathcal{PW}$, $\mathcal{B}$ first finds the tuple $(ID, *, pw_s, *, *, *)$ in the list L, executes $\mathsf{Auth}(C_A(ID, pw') \leftrightarrow S_A(ID, pw_s, sk))$, and returns the transcript $trans_{\mathsf{Auth}}$ of Auth and the result of authentication $\top/\bot$ to $\mathcal{A}_1$.
2. If $pw' = \phi$, $\mathcal{B}$ first finds the tuple $(ID, pw, pw_s, *, *, *)$ in the list L, executes $\mathsf{Auth}(C_A(ID, pw) \leftrightarrow S_A(ID, pw_s, sk))$, and returns the transcript $trans_{\mathsf{Auth}}$ of Auth and the result of authentication $\top/\bot$ to $\mathcal{A}_1$.

Finally, $\mathcal{A}_1$ outputs (ID^*, st). To win the Imp-PA game, the tuple $(ID^*, pw^*, pw^*_s, rk^*, frag^*_p, frag^*_r)$ must exist in the list L and satisfy $frag^*_p = 1$ and $frag^*_r = 1$ (if this is satisfied, we say that ID^* satisfies the "winning precondition"). If these conditions are not satisfied, $\mathcal{B}$ decides that $\mathcal{A}$ has lost the Imp-PA game. Otherwise, $\mathcal{B}$ gives st to $\mathcal{A}_2$. Then $\mathcal{A}_2$ and $\mathcal{B}$ interactively execute $\mathsf{Auth}(\mathcal{A}_2(st) \leftrightarrow S_A(ID^*, pw^*_s, sk))$. During the execution of this Auth protocol, $\mathcal{A}_2$ can adaptively make queries in the same way as $\mathcal{A}_1$. However, $\mathcal{A}_2$ is not allowed to use ID^* in the RKR and $\mathsf{Trans}_{\mathsf{PRR}}$ queries. Finally, $\mathcal{A}$ wins if the S_A's output of Auth is $\top$. We define the advantage of $\mathcal{A}$ by $\mathsf{Adv}^{\mathsf{Imp\text{-}PA}}_{\mathcal{A}}(k) = \Pr[\mathcal{A}\ wins]$.

Definition 9. *Let q_A be the number of* $\mathsf{Trans}_{\mathsf{Auth}}$ *queries by $\mathcal{A}_1$. We say that a password reset protocol is Imp-PA secure if for all PPT adversaries $\mathcal{A}$,* $\mathsf{Adv}^{\mathsf{Imp\text{-}PA}}_{\mathcal{A}}(k) = O(q_A)/|\mathcal{D}| + \varepsilon(k)$.

Illegal Registration Second, we consider security against illegal registration under passive attacks (IR-PA) for a password reset protocol. This security is defined using the following IR-PA game which is played by the challenger $\mathcal{B}$ and an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$. $\mathcal{B}$'s initial procedure and $\mathcal{A}_1$'s queries of this IR-PA game are exactly the same as the Imp-PA game. Finally, $\mathcal{A}_1$ outputs (ID^*, st). To win the IR-PA game, ID^* must satisfy the winning precondition. If these conditions are not satisfied, $\mathcal{B}$ decides that $\mathcal{A}$ has lost the IR-PA game. Otherwise, $\mathcal{B}$ gives st to $\mathcal{A}_2$. Then $\mathcal{A}_2$ and $\mathcal{B}$ interactively execute $\mathsf{PRR}(\mathcal{A}_2(st) \leftrightarrow S_P(ID^*, rk^*, sk))$. During the execution of this PRR protocol, $\mathcal{A}_2$ can adaptively make the queries in the same way as $\mathcal{A}_1$. However, $\mathcal{A}_2$ is not allowed to use ID^* in the RKR and $\mathsf{Trans}_{\mathsf{PRR}}$ queries. Finally, $\mathcal{A}$ wins if S_P's output of PRR is different from $\bot$. We define the advantage of $\mathcal{A}$ by $\mathsf{Adv}^{\mathsf{IR\text{-}PA}}_{\mathcal{A}}(k) = \Pr[\mathcal{A}\ wins]$.

Definition 10. *We say that a password reset protocol is IR-PA secure if for all PPT adversaries $\mathcal{A}$,* $\mathsf{Adv}^{\mathsf{IR\text{-}PA}}_{\mathcal{A}}(k)$ *is negligible.*

Remark Here we explain what situations RKR, $\mathsf{Trans}_{\mathsf{PRR}}$, $\mathsf{Trans}_{\mathsf{Auth}}$ queries allowed for an adversary try to capture. An adversary may register the authentication system as a legitimate user and learn which ID/pw is weak, which is captured by RKR queries. Furthermore, the adversary may eavesdrop the communication between honest users and the server to obtain all the transcripts. This is captured by $\mathsf{Trans}_{\mathsf{PRR}}$ and $\mathsf{Trans}_{\mathsf{Auth}}$ queries. Note that for both types of queries, the challenger behaves differently depending on whether pw' in the adversary's input is ϕ or not. The former case ($pw' = \phi$) captures the situation where the protocols are run by honest users.; The latter case ($pw' \neq \mathcal{PW}$) has different meanings depending on the types of queries. A $\mathsf{Trans}_{\mathsf{PRR}}$ query with $pw' \neq \mathcal{PW}$ captures the situation where the adversary itself tries to learn information from the transcript of the PRR protocol by executing it honestly, using the adversarially chosen password pw'. We may also be able to think of it as capturing the situation where an honest user registers a password that is known to an adversary for some reason (e.g. because of the choice of an easy-to-guess password).; A $\mathsf{Trans}_{\mathsf{Auth}}$ query with $pw' \neq \mathcal{PW}$ captures the case where again the adversary itself tries to learn information from the transcript of Auth protocol by executing it honestly, using the (adversarially chosen) password pw'. It also in some sense captures the situation where an honest user forgets (or mistypes) his/her password.

6.3 Proposed Construction

In this section, we propose a generic construction of a password reset protocol which satisfies the security definitions in Section 6.2.2.

6.3.1 Construction

In this subsection, we show a generic construction of a password reset protocol based on a PRF and PKE.

Intuition Before showing our construction, we explain an intuition of our proposed protocol. In our protocol, we set a reset key $rk := F(K, ID)$. In the password reset procedure, a user encrypts a randomness (which is sent by the server), a reset key, and a password by using a PKE scheme, and sends a ciphertext to the server. The server decrypts the ciphertext and gets a password. The authentication procedure is very similar to the password reset procedure. A user encrypts a randomness (which is sent by the server) and a password by using a PKE scheme, and sends a ciphertext to the server. The server decrypts the ciphertext, and checks whether the decryption result matches the registered password or not. The security of PKE (and the PRF) ensures that the transcripts do not leak the information of the passwords of honest users, and hence an adversary who wants to impersonate an uncorrupted user (with an unknown password) essentially has to guess the password.; In both PRR and Auth protocols, the randomness chosen by the server prevents "replay" attacks.

How Straightforward SKE-based Construction May Fail Suppose we want to reduce the security of a password reset protocol to the security of SKE. Recall that in our Imp-PA security game, an adversary $\mathcal{A}$ is allowed to issue reset key reveal (RKR) queries for ID's that are not the challenge ID^*. This means that the reduction algorithm cannot embed its problem instance regarding SKE into the reset keys of users who will be corrupted. Therefore, the most natural approach for designing the reduction algorithm (that attacks the security of SKE) using an adversary $\mathcal{A}$ against the Imp-PA security of

```
$\mathsf{SSetup}(1^k)$ :
$(pk, dk) \leftarrow \mathsf{PKG}(1^k)$
Choose a random $K \in \{0,1\}^k$
$pp := pk$; $sk := (K, dk)$
return $(pp, sk)$.
```

```
$\mathsf{RKG}(sk, ID)$ :
$(K, dk) \leftarrow sk$
$rk := F(K, ID)$
return $rk$.
```

```
$(\phi, pw_s) \leftarrow \mathsf{PRR}(C_P(ID, pw, rk) \leftrightarrow S_P(ID, rk, sk))$ :
1. $S_P$ chooses a randomness $r \in \{0,1\}^k$ and sends it to $C_P$
2. $C_P$ executes $c \leftarrow \mathsf{PEnc}(pk, ID \| r \| rk \| pw)$ and sends it to $S_P$
3-1. $S_P$ executes $ID' \| r' \| rk' \| pw_s \leftarrow \mathsf{PDec}(dk, c)$
3-2. If $ID' = ID$, $r' = r$, and $rk' = rk$ hold, $S_P$ returns $pw_s$
     else $S_P$ returns $\perp$.
```

```
$(\phi, \top/\perp) \leftarrow \mathsf{Auth}(C_A(ID, pw) \leftrightarrow S_A(ID, pw_s, sk))$ :
1. $S_A$ chooses a randomness $r \in \{0,1\}^k$ and sends it to $C_A$
2. $C_A$ executes $c \leftarrow \mathsf{PEnc}(pk, ID \| r \| pw)$ and sends it to $S_A$
3-1. $S_A$ executes $ID' \| r' \| pw' \leftarrow \mathsf{PDec}(dk, c)$
3-2. If $ID' = ID$, $r' = r$, and $pw' = pw_s$ hold, $S_A$ returns $\top$
     else $S_A$ returns $\perp$.
```

Figure 6.1: The proposed generic construction of a password reset protocol

a considered password reset protocol, will be to guess the index ℓ of $\mathcal{A}$'s $\mathsf{CCreate}$ query (for which $\mathcal{A}$ specifies ID^*) and embed the instance of the reduction algorithm's problem into the challenge user's ID^*. However, once we do this, the number of CCreate queries appears in the numerator of the formula of the advantage, namely, what we will be able to show is something like $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Imp\text{-}PA}} \leq O(q_C q_A)/|\mathcal{D}| + \varepsilon(k)$, where q_C is the number of $\mathcal{A}$'s $\mathsf{CCreate}$ queries. However, Imp-PA security requires that $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Imp\text{-}PA}}$ is upper-bounded by $O(q_A)/|\mathcal{D}| + \varepsilon(k)$, and thus the straightforward approach using SKE is not sufficient for proving Imp-PA security.

Although there could exist a way to avoid this problem by developing a new proof strategy, we try to solve this problem by using PKE scheme.

Our Construction Now, we formally describe the password reset protocol. Let $F : \{0,1\}^k \times \{0,1\}^* \rightarrow \{0,1\}^k$ be a pseudorandom function and $(\mathsf{PKG}, \mathsf{PEnc}, \mathsf{PDec})$ be a PKE scheme. Using these as building blocks, our password reset protocol is constructed as in Fig. 6.1.

6.3.2 Security Proof

In this subsection, we show security proofs of the proposed password reset protocol in Fig. 6.1.

Theorem 13. *If F is a PRF and the PKE scheme is mIND-CCA secure*[3]*, then the proposed password reset protocol in Fig. 6.1 satisfies Imp-PA security.*

Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be an Imp-PA adversary of the password reset protocol, q_A be the number of $\mathsf{Trans}_{\mathsf{Auth}}$ queries by $\mathcal{A}_1$. Here, q_A is a polynomial of the security parameter k. Consider the following sequence of games.

Game 0. This is exactly the Imp-PA game.

Game 1. This game proceeds in the same way as Game 0, except that the first messages r of S_A picked in the executions of PRR (in the response to $\mathcal{A}$'s $\mathsf{Trans}_{\mathsf{PRR}}$ query) and Auth (either in the response to $\mathcal{A}$'s $\mathsf{Trans}_{\mathsf{Auth}}$ query or in the challenge phase), are picked from $\{0,1\}^k \backslash \{r\text{'s that are already used}\}$, so that they are all distinct and never collide. For notational convenience, in this and subsequent games, we introduce the list $\mathcal{R}$ that is used to store r's that are used in the response to the $\mathsf{Trans}_{\mathsf{PRR}}$ query, $\mathsf{Trans}_{\mathsf{Auth}}$ query, and in the execution of Auth in the challenge phase, and we make the challenger choose r uniformly at random from $\{0,1\}^k \backslash \mathcal{R}$ every time it needs to choose r for PRR and Auth, and put the used r into the list $\mathcal{R}$.

Game 2. This game proceeds in the same way as Game 1, except that if $\mathcal{A}_1$ issues RKG queries on ID, then instead of using the result of $F(K, ID)$, $\mathcal{B}$ picks rk uniformly at random from the range of F, and uses it as the reset key corresponding to ID.

Game 3. This game proceeds in the same way as Game 2, except for the following two points.

- If $\mathcal{A}_1$ issues a $\mathsf{Trans}_{\mathsf{PRR}}$ query on ID, then instead of using the result of $\mathsf{PEnc}(pk, ID\|r\|rk\|pw)$, $\mathcal{B}$ executes $c \leftarrow \mathsf{PEnc}(pk, ID\|r\|0^{|rk|+|pw|})$, and uses r and c as the $trans_{\mathsf{PRR}}$ corresponding to ID.
- If $\mathcal{A}_1$ issues a $\mathsf{Trans}_{\mathsf{Auth}}$ query on ID, then instead of using the result of $\mathsf{PEnc}(pk, ID\|r\|pw)$, $\mathcal{B}$ executes $c \leftarrow \mathsf{PEnc}(pk, ID\|r\|0^{|pw|})$, and uses r and c as the $trans_{\mathsf{Auth}}$ corresponding to ID. In addition, when $\mathcal{B}$ calculates the output of S_P (i.e. $\top/\bot$) only by checking $pw = pw_s$ where pw_s is the value found in the tuple corresponding to ID in the list L, without running PDec.

For $i \in \{0,1,2,3\}$, we define the event W_i as the event that $\mathcal{A}$ wins in Game i. The advantage of $\mathcal{A}$ is, by definition, $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Imp\text{-}PA}}(k) = \Pr[W_0]$. We complete the proof by using the following inequality, and the upper bounds in the terms in the right hand side are shown in Lemmas 20 to 23.

$$\Pr[W_0] \leq \sum_{i=0}^{2} |\Pr[W_i] - \Pr[W_{i+1}]| + \Pr[W_3] \tag{6.1}$$

Lemma 20. $|\Pr[W_0] - \Pr[W_1]|$ *is negligible.*

Proof. The difference $|\Pr[W_0] - \Pr[W_1]|$ can be upperbounded by the statistical distance between the distributions of r's used in PRR (in the responses to $\mathcal{A}$'s $\mathsf{Trans}_{\mathsf{PRR}}$ queries) and

[3]Strictly speaking, multi-challenge 1-bounded CCA secure PKE [38] is enough for the security proof.

Auth (in the responses to $\mathcal{A}$'s $\mathsf{Trans}_{\mathsf{Auth}}$ queries and in Auth the challenge phase) in Game 0 and those in Game 1. Since the number of r's in the games is at most $(q_P + q_A + 1)$, the statistical distance between the distributions is at most $(q_P + q_A + 1)^2/2^k$. □

Lemma 21. *If F is a PRF, $|\Pr[W_1] - \Pr[W_2]|$ is negligible.*

Proof. We show that we can construct an adversary $\mathcal{B}$ against the PRF F. The description of $\mathcal{B}$ is as follows:

First, the challenger chooses a key $K \in \{0,1\}^k$ and the challenge bit $\{0,1\}$ uniformly at random (which are both unknown to $\mathcal{B}$). $\mathcal{B}$ executes $(pk, dk) \leftarrow \mathsf{PKG}(1^k)$ and generates an empty list L which will be used to store tuples of the form $(ID, pw, pw_s, rk, frag_p, frag_r)$. $\mathcal{B}$ also generates an empty list $\mathcal{R}$. After that, $\mathcal{B}$ gives $pp := pk$ to $\mathcal{A}_1$.

When $\mathcal{A}_1$ makes a $\mathsf{CCreate}$ query ID, $\mathcal{B}$ responds as follows.

1. If there exists a tuple $(ID, *, *, *, *, *)$ in the list L, $\mathcal{B}$ does nothing.

2. Otherwise, $\mathcal{B}$ submits the identity ID to the challenger, and receives rk. This rk is $F(K, ID)$ if $b = 0$ and is a random string in the range of F if $b = 1$. After that, $\mathcal{B}$ stores $(ID, \phi, \phi, rk, 0, 1)$ into the list L.

When $\mathcal{A}_1$ makes a PR query ID, $\mathcal{B}$ finds the tuple $(ID, pw, *, *, *, *)$ in the list L, and returns pw to $\mathcal{A}_1$. Then, $\mathcal{B}$ updates the tuple in the list L by $(ID, *, *, *, 0, *)$.

When $\mathcal{A}_1$ makes a RKR query ID, $\mathcal{B}$ finds the tuple $(ID, *, *, rk, *, *)$ in the list L, and returns rk to $\mathcal{A}_1$. Then, $\mathcal{B}$ updates the tuple in the list L by $(ID, *, *, *, *, 0)$.

When $\mathcal{A}_1$ makes a $\mathsf{Trans}_{\mathsf{PRR}}$ query (ID, pw') where $pw' \in \mathcal{PW} \cup \{\phi\}$, $\mathcal{B}$ responds as follows. First, $\mathcal{B}$ finds the tuple $(ID, *, *, rk, *, *)$ in the list L.

1. If $pw' \in \mathcal{PW}$, $\mathcal{B}$ chooses a randomness $r \in \{0,1\}^k$. Next, $\mathcal{B}$ executes $c \leftarrow \mathsf{PEnc}(pk, ID\|r\|rk\|pw')$ and $ID'\|r'\|rk'\|pw'_s \leftarrow \mathsf{PDec}(dk, c)$. Then, $\mathcal{B}$ returns $trans_{\mathsf{PRR}} := (r, c)$ and the registration result $z := \top$ to $\mathcal{A}_1$. After that, $\mathcal{B}$ updates the tuple in the list L by $(ID, pw', pw'_s, *, 0, *)$ and adds r to the list $\mathcal{R}$.

2. If $pw' = \phi$, $\mathcal{B}$ first chooses a random password $pw' \in \mathcal{D}$ (where $\mathcal{D}$ is the dictionary from which an honest user is assumed to sample his/her password) and a randomness $r \in \{0,1\}^k$. Next, $\mathcal{B}$ executes $c \leftarrow \mathsf{PEnc}(pk, ID\|r\|rk\|pw')$ and $ID'\|r'\|rk'\|pw'_s \leftarrow \mathsf{PDec}(dk, c)$. Then, $\mathcal{B}$ returns $trans_{\mathsf{PRR}} := (r, c)$ and the registration result $z := \top$ to $\mathcal{A}_1$. After that, $\mathcal{B}$ updates the tuple in the list L by $(ID, pw', pw'_s, *, 1, *)$ and adds r to the list $\mathcal{R}$.

When $\mathcal{A}_1$ makes a $\mathsf{Trans}_{\mathsf{Auth}}$ query (ID, pw') where $pw' \in \mathcal{PW} \cup \{\phi\}$, $\mathcal{B}$ responds as follows. First, $\mathcal{B}$ finds the tuple $(ID, pw, pw_s, *, *, *)$ in the list L.

1. If $pw' \in \mathcal{PW}$, $\mathcal{B}$ chooses a randomness r from $\{0,1\}^k \backslash \mathcal{R}$, executes $c \leftarrow \mathsf{PEnc}(pk, ID\| r\|pw')$ and $ID''\|r''\|pw'' \leftarrow \mathsf{PDec}(dk, c)$. Next, $\mathcal{B}$ sets $z := \top$ if $ID'' = ID$, $r'' = r$, and $pw'' = pw_s$ hold. Otherwise, $\mathcal{B}$ sets $z := \perp$. Then, $\mathcal{B}$ returns $trans_{\mathsf{Auth}} := (r, c)$ and the authentication result z to $\mathcal{A}_1$. After that, $\mathcal{B}$ adds r to the list $\mathcal{R}$.

2. If $pw' = \phi$, $\mathcal{B}$ chooses a randomness r from $\{0,1\}^k \backslash \mathcal{R}$, executes $c \leftarrow \mathsf{PEnc}(pk, ID\|r\| pw)$ and $ID''\|r''\|pw'' \leftarrow \mathsf{PDec}(dk, c)$. Next, $\mathcal{B}$ sets $z := \top$ if $ID'' = ID$,$r'' = r$, and $pw'' = pw_s$ hold. Otherwise, $\mathcal{B}$ sets $z := \perp$. Then, $\mathcal{B}$ returns $trans_{\mathsf{Auth}} := (r, c)$ and the authentication result z to $\mathcal{A}_1$. After that, $\mathcal{B}$ adds r to the list $\mathcal{R}$.

Finally, $\mathcal{A}_1$ terminates with (ID^*, st). $\mathcal{B}$ outputs $b' = 1$ and aborts when the identity ID^* does not satisfy the winning precondition. Otherwise, $\mathcal{B}$ chooses r^* uniformly at random from $\{0,1\}^k \backslash \mathcal{R}$, and gives r^* and st to $\mathcal{A}_2$. $\mathcal{B}$ can respond to the queries from $\mathcal{A}_2$ in the same way as $\mathcal{B}$ did for $\mathcal{A}_1$. However, when $\mathcal{A}_2$ submits ID^* as a RKR or $\mathsf{Trans}_{\mathsf{PRR}}$ query, $\mathcal{B}$ returns $\perp$ to $\mathcal{A}_2$. Finally, $\mathcal{A}_2$ terminates with c^*. Next, $\mathcal{B}$ executes $r'^* \| pw'^* \leftarrow \mathsf{PDec}(dk, c^*)$. Then, $\mathcal{B}$ finds the tuple $(ID^*, *, pw_s^*, *, *, *)$ in the list L and checks whether the conditions $r'^* = r^*$ and $pw'^* = pw_s^*$ hold or not. If these conditions hold, $\mathcal{B}$ terminates with $b' = 0$. Otherwise, $\mathcal{B}$ terminates with $b' = 1$.

The above completes the description of $\mathcal{B}$. It is not hard to see that $\mathcal{B}$ perfectly simulates Game 1 for $\mathcal{A}$ when $\mathcal{B}$'s challenge bit $b = 0$, and Game 2 when $b = 1$. When the challenge bit of $\mathcal{B}$ is 0 and $\mathcal{B}$ does not abort before $\mathcal{A}$ terminates, $\mathcal{B}$'S responses to $\mathcal{A}$'s queries are performed in exactly the same way as those in Game 1. In addition, $\mathcal{B}$ outputs 0 only if $\mathcal{B}$ does not abort and $\mathcal{A}_2$ succeeds in outputting a ciphertext $c^* = \mathsf{PEnc}(pk, ID^* \| r^* \| pw_s^*)$. Therefore, $\Pr[b' = 0 | b = 0] = \Pr[W_1]$. On the other hand, when the challenge bit of $\mathcal{B}$ is 1, the response of the challenger of $\mathcal{B}$ is a random string, and this situation is the same as Game 2. With almost the same discussion as above, we have $\Pr[b' = 0 | b = 1] = \Pr[W_2]$. Therefore, $\mathsf{Adv}_{\mathcal{B}}^{\mathsf{PRF}}(k) = |\Pr[b' = b] - 1/2| = (1/2)|\Pr[b' = 0 | b = 0] - \Pr[b' = 0 | b = 1]| = (1/2)|\Pr[W_1] - \Pr[W_2]|$. Using this equality, and recalling the assumption that the underlying F is a PRF, we conclude that $|\Pr[W_1] - \Pr[W_2]|$ is negligible. □ □

Lemma 22. *If the PKE scheme is mIND-CCA secure,* $|\Pr[W_2] - \Pr[W_3]|$ *is negligible.*

Proof. We show that we can construct a multi-challenge IND-CCA adversary $\mathcal{B}$ against the underlying PKE scheme. The description of $\mathcal{B}$ is as follows:

First, the challenger executes $(pk, dk) \leftarrow \mathsf{PKG}(1^k)$ and chooses the challenge bit $b \in \{0,1\}$ uniformly at random. Then, the challenger gives pk to $\mathcal{B}$. $\mathcal{B}$ chooses a random key $K \in \{0,1\}^k$ for PRF F and generates an empty list L which will be used to store tuples of the form $(ID, pw, pw_s, rk, frag_p, frag_r)$. $\mathcal{B}$ also generates an empty list $\mathcal{R}$. After that, $\mathcal{B}$ gives $pp := pk$ to $\mathcal{A}_1$.

When $\mathcal{A}_1$ makes a $\mathsf{CCreate}$ query ID, $\mathcal{B}$ responds as follows.

1. If there exists a tuple $(ID, *, *, *, *, *)$ in the list L, $\mathcal{B}$ does nothing.
2. Otherwise, $\mathcal{B}$ chooses $rk \in \{0,1\}^k$ uniformly at random from the range of F and gives rk to $\mathcal{A}_1$. After that, $\mathcal{B}$ stores the tuple $(ID, \phi, \phi, rk, 0, 1)$ into the list L.

When $\mathcal{A}_1$ makes a PR query ID, $\mathcal{B}$ finds the tuple $(ID, pw, *, *, *, *)$ in the list L, and returns pw to $\mathcal{A}_1$. Then, $\mathcal{B}$ updates the tuple in the list L by $(ID, *, *, *, 0, *)$.

When $\mathcal{A}_1$ makes a RKR query ID, $\mathcal{B}$ finds the tuple $(ID, *, *, rk, *, *)$ in the list L, and returns rk to $\mathcal{A}_1$. Then, $\mathcal{B}$ updates the tuple in the list L by $(ID, *, *, *, *, 0)$.

When $\mathcal{A}_1$ makes a $\mathsf{Trans}_{\mathsf{PRR}}$ query (ID, pw') where $pw' \in \mathcal{PW} \cup \{\phi\}$, $\mathcal{B}$ responds as follows. First, $\mathcal{B}$ finds the tuple $(ID, *, *, rk, *, *)$ in the list L.

1. If $pw' \in \mathcal{PW}$, $\mathcal{B}$ chooses a randomness $r \in \{0,1\}^k$, submits $(ID \| r \| rk \| pw', ID \| r \| 0^{|rk|+|pw'|})$ to the challenger, and receives c. This c is $\mathsf{PEnc}(pk, ID \| r \| rk \| pw')$ if $b = 0$ and is $\mathsf{PEnc}(pk, ID \| r \| 0^{|rk|+|pw'|})$ if $b = 1$. Then, $\mathcal{B}$ returns $trans_{\mathsf{PRR}} := (r, c)$ and

the registration result $z := \top$ to $\mathcal{A}_1$. After that, $\mathcal{B}$ updates the tuple in the list L by $(ID, pw', pw', *, 0, *)$ and adds r to the list $\mathcal{R}$.

2. If $pw' = \phi$, $\mathcal{B}$ chooses a random password $pw' \in \mathcal{D}$ (where $\mathcal{D}$ is the dictionary) and a randomness $r \in \{0,1\}^k$, submits $(ID\|r\|rk\|pw', ID\|r\|0^{|rk|+|pw'|})$ to the challenger, and receives c. This c is $\mathsf{PEnc}(pk, ID\|r\|rk\|pw')$ if $b = 0$ and is $\mathsf{PEnc}(pk, ID\|r\|0^{|rk|+|pw'|})$ if $b = 1$. Then, $\mathcal{B}$ returns $trans_{\mathsf{PRR}} := (r, c)$ and the registration result $z := \top$ to $\mathcal{A}_1$. After that, $\mathcal{B}$ updates the tuple in the list L by $(ID, pw', pw', *, 1, *)$ and adds r to the list $\mathcal{R}$.

When $\mathcal{A}_1$ makes a $\mathsf{Trans}_{\mathsf{Auth}}$ query (ID, pw') where $pw' \in \mathcal{PW} \cup \{\phi\}$, $\mathcal{B}$ responds as follows. First, $\mathcal{B}$ finds the tuple $(ID, pw, pw_s, *, *, *)$ in the list L.

1. If $pw' \in \mathcal{PW}$, $\mathcal{B}$ chooses a randomness r from $\{0,1\}^k \backslash \mathcal{R}$, submits $(ID\|r\|pw', ID\|r\| 0^{|pw'|})$ to the challenger, and receives c. This c is $\mathsf{PEnc}(pk, ID\|r\|pw')$ if $b = 0$ and is $\mathsf{PEnc}(pk, ID\|r\|0^{|pw'|})$ if $b = 1$. Next, $\mathcal{B}$ sets $z := \top$ if $pw' = pw_s$ holds. Otherwise, $\mathcal{B}$ sets $z := \bot$. Then, $\mathcal{B}$ returns $trans_{\mathsf{Auth}} := (r, c)$ and the authentication result z to $\mathcal{A}_1$. After that, $\mathcal{B}$ adds r to the list $\mathcal{R}$.

2. If $pw' = \phi$, $\mathcal{B}$ chooses a randomness r from $\{0,1\}^k \backslash \mathcal{R}$, submits $(ID\|r\|pw, ID\|r\|0^{|pw|})$ as a challenge query to the challenger, and receives c. This c is $\mathsf{PEnc}(pk, ID\|r\|pw)$ if $b = 0$ and is $\mathsf{PEnc}(pk, ID\|r\|0^{|pw|})$ if $b = 1$. Next, $\mathcal{B}$ sets $z := \top$ if $pw = pw_s$ holds. Otherwise, $\mathcal{B}$ sets $z := \bot$. Then, $\mathcal{B}$ returns $trans_{\mathsf{Auth}} := (r, c)$ and the authentication result z to $\mathcal{A}_1$. After that, $\mathcal{B}$ adds r to the list $\mathcal{R}$.

Finally, $\mathcal{A}_1$ terminates with (ID^*, st). $\mathcal{B}$ outputs $b' = 1$ and aborts when the identity ID^* does not satisfy the winning precondition. Otherwise, $\mathcal{B}$ chooses r^* uniformly at random from $\{0,1\}^k \backslash \mathcal{R}$, and gives r^* and st to $\mathcal{A}_2$. $\mathcal{B}$ can respond to the queries from $\mathcal{A}_2$ in the same way as $\mathcal{B}$ did for $\mathcal{A}_1$. However, when $\mathcal{A}_2$ submits ID^* as a RKR or $\mathsf{Trans}_{\mathsf{PRR}}$ query, $\mathcal{B}$ returns $\bot$ to $\mathcal{A}_2$. Finally, $\mathcal{A}_2$ terminates with c^*. If this c^* is one of the ciphertexts used as a response to the $\mathsf{Trans}_{\mathsf{PRR}}$ or $\mathsf{Trans}_{\mathsf{Auth}}$ queries, $\mathcal{B}$ stops the Imp-PA game, decides that $\mathcal{A}$ has lost the Imp-PA game, and terminates with $b' = 1$. Next, $\mathcal{B}$ submits c^* as a decryption query[4] to the challenger, and receives $ID^*\|r'^*\|pw'^*$. Then, $\mathcal{B}$ finds the tuple $(ID^*, *, pw_s^*, *, *, *)$ in the list L and checks whether the conditions $r'^* = r^*$ and $pw'^* = pw_s^*$ hold or not. If these conditions hold, $\mathcal{B}$ terminates with output $b' = 0$. Otherwise, $\mathcal{B}$ terminates with output $b' = 1$.

The above completes the description of $\mathcal{B}$. It is not hard to see that $\mathcal{B}$ perfectly simulates Game 2 when $\mathcal{B}$'s challenge bit $b = 0$, and Game 3 when $b = 1$. When the challenge bit of $\mathcal{B}$ is 0 and $\mathcal{B}$ does not abort before $\mathcal{A}$ terminates, $\mathcal{B}$ responses to $\mathcal{A}$'s queries are distributed identically to those in Game 2. In addition, $\mathcal{B}$ outputs 0 only if $\mathcal{B}$ does not abort and $\mathcal{A}_2$ succeeds in outputting a ciphertext c^* satisfying $\mathsf{PDec}(dk, c^*) = ID^*\|r^*\|pw_s^*$. Therefore, $\Pr[b' = 0 | b = 0] = \Pr[W_2]$. On the other hand, when the challenge bit of $\mathcal{B}$ is 1, $\mathcal{A}_2$ succeeds in outputting a ciphertext c^* satisfying $\mathsf{PDec}(dk, c^*) = ID^*\|r^{\|}0^{|pw_s^*|}$, and this situation is the same as Game 3. With almost the same discussion as above, we have $\Pr[b' = 0 | b = 1] = \Pr[W_3]$. Therefore, $\mathsf{Adv}_{\mathcal{B}}^{\mathsf{mIND\text{-}CCA}}(k) = |\Pr[b' = b] - 1/2| =$

[4]This is the only decryption query that $\mathcal{B}$ submits, which is the reason why 1-bounded CCA security [38] suffices.

$(1/2)|\Pr[b' = 0|b = 0] - \Pr[b' = 0|b = 1]| = (1/2)|\Pr[W_2] - \Pr[W_3]|$. Using this equality, and recalling the assumption that the underlying PKE scheme is mIND-CCA secure, we conclude that $|\Pr[W_2] - \Pr[W_3]|$ is negligible. $\square$

Lemma 23. $\Pr[W_3] \leq (q_A + 1)/|\mathcal{D}|$.

Proof. Note that in Game 3, the "transcript" part in the responses to the $\mathsf{Trans}_{\mathsf{PRR}}$ and $\mathsf{Trans}_{\mathsf{Auth}}$ queries contain no information of pw. However, if an adversary makes a $\mathsf{Trans}_{\mathsf{Auth}}$ query (ID, pw) with $pw \neq \phi$, the "server's output" part (i.e. $\top$ or $\bot$) leaks whether $pw = pw'$. However, other than this, no information about pw leaks. Since pw is chosen randomly from $\mathcal{D}$, the probability that $\mathcal{A}$ wins in Game 3 is at most $(q_A + 1)/|\mathcal{D}|$. $\square$

Lemmas 20 to 23 guarantee that the right hand side of the inequality (6.1) is upper-bounded by $O(q_A)/|\mathcal{D}| + \varepsilon(k)$. This completes the proof of Theorem 13. $\square$

Theorem 14. *If F is a PRF and the PKE scheme is mIND-CCA secure*[5]*, then the proposed password reset protocol in Fig. 6.1 satisfies IR-PA security.*

Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be an Imp-PA adversary of the password reset protocol. Let q_P be the number of $\mathsf{Trans}_{\mathsf{PRR}}$ queries by $\mathcal{A}_1$. Here, q_P is a polynomial of the security parameter. Consider the following sequence of games.

Game 0. This is exactly the IR-PA game.

Game 1. This game proceeds in the same way as Game 0, except that the first messages r of S_P picked in the execution of PRR (either in the response to $\mathcal{A}$'s $\mathsf{Trans}_{\mathsf{PRR}}$ query or in the challenge phase) and Auth (in the response to $\mathcal{A}$'s $\mathsf{Trans}_{\mathsf{Auth}}$ query), are picked from $\{0,1\}^k \backslash \{r\text{'s that are already used }\}$, so that they are all distinct and never collide. For notational convenience, in this and subsequent games, we introduce the list $\mathcal{R}$ that is used to store r's that are used in the response to the $\mathsf{Trans}_{\mathsf{PRR}}$ query or in the execution of PRR in the challenge phase and $\mathsf{Trans}_{\mathsf{Auth}}$ query, and we make the challenger choose r uniformly at random from $\{0,1\}^k \backslash \mathcal{R}$ every time it needs to choose r for PRR and Auth, and put the used r into the list $\mathcal{R}$.

Game 2. This game proceeds in the same way as Game 1, except that if $\mathcal{A}_1$ issues a $\mathsf{Trans}_{\mathsf{PRR}}$ query on ID, then instead of using the result of $F(K, ID)$, $\mathcal{B}$ picks rk uniformly at random from the range of F, and uses it as the reset key corresponding to ID.

Game 3. This game proceeds in the same way as Game 2, except that if $\mathcal{A}_1$ issues a $\mathsf{Trans}_{\mathsf{PRR}}$ query on ID, then instead of using the result of $\mathsf{PEnc}(pk, ID \| r \| rk \| pw)$, $\mathcal{B}$ executes $c \leftarrow \mathsf{PEnc}(pk, ID \| r \| 0^{|rk|+|pw|})$, and uses r and c as the $trans_{\mathsf{PRR}}$ corresponding to ID.

For $i \in \{0,1,2,3\}$, we define the event W_i as the event that $\mathcal{A}$ wins in Game i. The advantage of $\mathcal{A}$ is, by definition, $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{IR\text{-}PA}}(k) = \Pr[W_0]$. We complete the proof by using

[5]Multi-challenge 1-bounded CCA secure PKE is enough for the security proof.

the following inequality, and the upper bounds in the terms in the right hand side are shown in Lemmas 24 to 27.

$$\Pr[W_0] \leq \sum_{i=0}^{2} |\Pr[W_i] - \Pr[W_{i+1}]| + \Pr[W_3] \tag{6.2}$$

Lemma 24. $|\Pr[W_0] - \Pr[W_1]|$ *is negligible.*

Proof. The difference $|\Pr[W_0] - \Pr[W_1]|$ can be upperbounded by the statistical distance between the distributions of r's used in PRR (in the response to $\mathcal{A}$'s $\mathsf{Trans}_{\mathsf{PRR}}$ queries and in the challenge phase) and Auth (in the response to $\mathcal{A}$'s $\mathsf{Trans}_{\mathsf{Auth}}$ queries) used in Game 0 and that in Game 1. Since the number of r's in the games is at most $(q_P + q_A + 1)$, the statistical distance between the distributions is at most $(q_P + q_A + 1)^2/2^k$. □

Lemma 25. *If the F is a PRF,* $|\Pr[W_1] - \Pr[W_2]|$ *is negligible.*

Proof. This proof follows closely to the one of Lemma 21. Therefore, we omit it. □

Lemma 26. *If the PKE scheme is mIND-CCA secure,* $|\Pr[W_2] - \Pr[W_3]|$ *is negligible.*

Proof. We show that we can construct a multi-challenge IND-CCA adversary $\mathcal{B}$ against the underlying PKE scheme. The description of $\mathcal{B}$ is as follows:

First, the challenger executes $(pk, dk) \leftarrow \mathsf{PKG}(1^k)$ and chooses the challenge bit $b \in \{0, 1\}$ uniformly at random. Then, the challenger gives pk to $\mathcal{B}$. $\mathcal{B}$ chooses a random key $K \in \{0, 1\}^k$ for PRF F and generates an empty list L which will be used to store tuples of the form $(ID, pw, pw_s, rk, frag_p, frag_r)$. $\mathcal{B}$ also generates an empty list $\mathcal{R}$. After that, $\mathcal{B}$ gives $pp := pk$ to $\mathcal{A}_1$.

When $\mathcal{A}_1$ makes a $\mathsf{CCreate}$ query ID, $\mathcal{B}$ responds as follows.

1. If there exists a tuple $(ID, *, *, *, *, *)$ in the list L, $\mathcal{B}$ does nothing.

2. Otherwise, $\mathcal{B}$ chooses $rk \in \{0, 1\}^k$ uniformly at random from the range of F and gives rk to $\mathcal{A}_1$. After that, $\mathcal{B}$ stores the tuple $(ID, \phi, \phi, rk, 0, 1)$ into the list L.

When $\mathcal{A}_1$ makes a PR query ID, $\mathcal{B}$ finds the tuple $(ID, pw, *, *, *, *)$ in the list L, and returns pw to $\mathcal{A}_1$. Then, $\mathcal{B}$ updates the tuple in the list L by $(ID, *, *, *, 0, *)$.

When $\mathcal{A}_1$ makes a RKR query ID, $\mathcal{B}$ finds the tuple $(ID, *, *, rk, *, *)$ in the list L, and returns rk to $\mathcal{A}_1$. Then, $\mathcal{B}$ updates the tuple in the list L by $(ID, *, *, *, *, 0)$.

When $\mathcal{A}_1$ makes a $\mathsf{Trans}_{\mathsf{PRR}}$ query (ID, pw') where $pw' \in \mathcal{PW} \cup \{\phi\}$, $\mathcal{B}$ responds as follows. First, $\mathcal{B}$ finds the tuple $(ID, *, *, rk, *, *)$ in the list L.

1. If $pw' \in \mathcal{PW}$, $\mathcal{B}$ chooses a randomness $r \in \{0, 1\}^k \backslash \mathcal{R}$, submits $(ID \| r \| rk \| pw', ID \| r \| 0^{|rk|+|pw'|})$ to the challenger, and receives c. This c is $\mathsf{PEnc}(pk, ID \| r \| rk \| pw')$ if $b = 0$ and is $\mathsf{PEnc}(pk, ID \| r \| 0^{|rk|+|pw'|})$ if $b = 1$. Then, $\mathcal{B}$ returns $trans_{\mathsf{PRR}} := (r, c)$ and the registration result $z := \top$ to $\mathcal{A}_1$. After that, $\mathcal{B}$ updates the tuple in the list L by $(ID, pw', pw', *, 0, *)$ and adds r to the list $\mathcal{R}$.

2. If $pw' = \phi$, $\mathcal{B}$ chooses a random password $pw' \in \mathcal{D}$ (where $\mathcal{D}$ is the dictionary) and a randomness $r \in \{0, 1\}^k \backslash \mathcal{R}$, submits $(ID \| r \| rk \| pw', ID \| r \| 0^{|rk|+|pw'|})$ to the challenger, and receives c. This c is $\mathsf{PEnc}(pk, ID \| r \| rk \| pw')$ if $b = 0$ and is $\mathsf{PEnc}(pk,$

$ID\|r\|0^{|rk|+|pw'|})$ if $b = 1$. Then, $\mathcal{B}$ returns $trans_{\mathsf{PRR}} := (r, c)$ and the registration result $z := \top$ to $\mathcal{A}_1$. After that, $\mathcal{B}$ updates the tuple in the list L by $(ID, pw', pw', *, 1, *)$ and adds r to the list $\mathcal{R}$.

When $\mathcal{A}_1$ makes a $\mathsf{Trans}_{\mathsf{Auth}}$ query (ID, pw') where $pw' \in \mathcal{PW} \cup \{\phi\}$, $\mathcal{B}$ responds as follows. First, $\mathcal{B}$ finds the tuple $(ID, pw, pw_s, *, *, *)$ in the list L.

1. If $pw' \in \mathcal{PW}$, $\mathcal{B}$ chooses a randomness r from $\{0,1\}^k$ and executes $c \leftarrow \mathsf{PEnc}(pk, ID\| r\|pw')$. Next, $\mathcal{B}$ sets $z := \top$ if $pw' = pw_s$ holds. Otherwise, $\mathcal{B}$ sets $z := \bot$. Then, $\mathcal{B}$ returns $trans_{\mathsf{Auth}} := (r, c)$ and the authentication result z to $\mathcal{A}_1$. After that, $\mathcal{B}$ adds r to the list $\mathcal{R}$.

2. If $pw' = \phi$, $\mathcal{B}$ chooses a randomness r from $\{0,1\}^k$ and executes $c \leftarrow \mathsf{PEnc}(pk, ID\|r\| pw)$. Next, $\mathcal{B}$ sets $z := \top$ if $pw = pw_s$ holds. Otherwise, $\mathcal{B}$ sets $z := \bot$. Then, $\mathcal{B}$ returns $trans_{\mathsf{Auth}} := (r, c)$ and the authentication result z to $\mathcal{A}_1$. After that, $\mathcal{B}$ adds r to the list $\mathcal{R}$.

Finally, $\mathcal{A}_1$ terminates with (ID^*, st). $\mathcal{B}$ outputs $b' = 1$ and aborts when the identity ID^* does not satisfy the winning precondition. Otherwise, $\mathcal{B}$ chooses r^* uniformly at random from $\{0,1\}^k \backslash \mathcal{R}$, and gives r^* and st to $\mathcal{A}_2$. $\mathcal{B}$ responds to the queries from $\mathcal{A}_2$ in the same way as $\mathcal{B}$ did for $\mathcal{A}_1$. However, when $\mathcal{A}_2$ submits ID^* as a RKR or $\mathsf{Trans}_{\mathsf{PRR}}$ query, $\mathcal{B}$ returns $\bot$ to $\mathcal{A}_2$. Finally, $\mathcal{A}_2$ terminates with c^*. If this c^* is one of the ciphertexts as a response to the $\mathsf{Trans}_{\mathsf{PRR}}$ queries, $\mathcal{B}$ stops the IR-PA game, decides that $\mathcal{A}$ has lost the IR-PA game, and terminates with $b' = 1$. Next, $\mathcal{B}$ submits c^* as a decryption query[6] to the challenger, and receives $ID'^*\|r'^*\|rk'^*\|pw'^*$. Then, $\mathcal{B}$ finds the tuple $(ID^*, *, pw_s^*, rk^*, *, *)$ in the list L and checks whether the conditions $ID'^* = ID^*$, $r'^* = r^*$, $rk'^* = rk^*$, and $pw'^* = pw_s^*$ hold or not. If these conditions hold, $\mathcal{B}$ terminates with $b' = 0$. Otherwise, $\mathcal{B}$ terminates with $b' = 1$.

The above completes the description of $\mathcal{B}$. It is not hard to see that $\mathcal{B}$ perfectly simulates Game 2 when $\mathcal{B}$'s challenge bit $b = 0$, and Game 3 when $b = 1$. When the challenge bit of $\mathcal{B}$ is 0 and $\mathcal{B}$ does not abort before $\mathcal{A}$ terminates, $\mathcal{B}$ responses to $\mathcal{A}$'s queries are distributed identically to those in Game 2. In addition, $\mathcal{B}$ outputs 0 only if $\mathcal{B}$ does not abort and $\mathcal{A}_2$ succeeds in outputting a ciphertext that is decrypted to the $ID^*\|r^*\|rk^*\|pw_s^*$. Therefore, $\Pr[b' = 0|b = 0] = \Pr[W_2]$. On the other hand, when the challenge bit of $\mathcal{B}$ is 1, $\mathcal{A}_2$ succeeds in outputting a ciphertext that is decrypted to the $ID^*\|r^*\|0^{|rk^*|+|pw_s^*|}$, and this situation is the same as Game 3. With almost the same discussion as above, we have $\Pr[b' = 0|b = 1] = \Pr[W_3]$. Therefore, $\mathsf{Adv}_{\mathcal{B}}^{\mathsf{mIND\text{-}CCA}}(k) = |\Pr[b' = b] - 1/2| = (1/2)|\Pr[b' = 0|b = 0] - \Pr[b' = 0|b = 1]| = (1/2)|\Pr[W_2] - \Pr[W_3]|$. Using this equality, and recalling the assumption that the underlying PKE scheme is mIND-CCA secure, we conclude that $|\Pr[W_2] - \Pr[W_3]|$ is negligible. □

Lemma 27. $\Pr[W_3]$ *is negligible.*

Proof. Note that in Game 3, the "transcript" part in the responses to the $\mathsf{Trans}_{\mathsf{PRR}}$ queries contain no information about rk. Therefore, the probability that $\mathcal{A}$ succeeds the guess of rk (that is, the probability that $\mathcal{A}$ wins in Game 3) is negligible. □

[6]This is the only decryption query that $\mathcal{B}$ submits, which is the reason why 1-bounded CCA security [38] suffices.

Lemmas 24 to 27 guarantee that the right hand side of the inequality (6.2) is negligible, and thus $\mathcal{A}$ has negligible advantage in the IR-PA game. This completes the proof of Theorem 14. $\square$

6.4 Extension

In this section, we discuss two extensions of our password reset protocol.

6.4.1 Password Reset Protocol with Password Salting

In practice, it is recommended not to store the client's raw password into the server. The server stores "processed data" instead of the client's password itself, and uses it for the authentication. Our proposed password reset protocol can be easily extended to a protocol with password salting. In the setup procedure, a server chooses another key K'. In the password re-registration procedure, the server decrypts the password pw from a ciphertext ct, executes $pw_s := F(K', pw)$, and stores pw_s. In the authentication procedure, server checks whether the condition $F(k', pw') = pw_s$ holds or not, and outputs $\top/\bot$. In this scheme, pw_s which is stored in the server collides if different users set the same password. To prevent this situation, we modify the scheme to compute $F(K', ID\|pw)$ instead of $F(K', pw)$. Although secret key size of this scheme becomes bigger compared to the original scheme, we can avoid it by using the domain separation technique. That is, instead of preparing the new key K', the server uses a key K for two purposes by adding the prefix bit. When the server generates a reset key rk, it executes $rk := F(K, 0\|ID)$. When the server generates pw_s, it generates $pw_s := F(K, 1\|ID\|pw)$.

Our password reset protocol with password salting is constructed as in Fig. 6.2. The security proofs for this extended scheme follow closely to the security proofs of original scheme. Therefore, we omit them.

6.4.2 Security against Active Adversary

In Section 6.2.2, we only considered the security against passive attacks. In this section, we give the formal security definitions against active attacks for a password reset protocol by extending the security definitions for passive ones, and show that our proposed protocol in Section **??** (and Section 6.4.1) satisfies them under the same assumptions on the building blocks.

Impersonation First, we consider security against impersonation under active attacks (Imp-AA) for a password reset protocol. Imp-AA security is defined using the following Imp-AA game which is played by the challenger $\mathcal{B}$ and an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$. First, $\mathcal{B}$ executes $(pp, sk) \leftarrow \mathsf{SSetup}(1^k)$, and generates an empty list L into which tuples of the form $(ID, pw, pw_s, rk, frag_p, frag_r)$ where $frag_p, frag_r \in \{0,1\}$ will be stored. These $frag_p$ and $frag_r$ are used to indicate whether a client with ID is "corrupted" by $\mathcal{A}$ in the sense that either pw or rk is known to $\mathcal{A}$, in which case $\mathcal{A}$ is not allowed to use the ID for its attack. Moreover, $\mathcal{B}$ generates two lists L'_P and L'_A into which tuples of the form $(sid, ID, pw, st_c, st_s, trans_{sid})$ will be stored. Here, sid means session ID, st_c/st_s mean the state information in client/server side, and $trans_{sid}$ means a transcript with respect

```
$\mathsf{SSetup}(1^k)$ :
$(pk, dk) \leftarrow \mathsf{PKG}(1^k)$
Choose a random $K \in \{0,1\}^k$
$pp := pk$; $sk := (K, dk)$
return $(pp, sk)$.

$\mathsf{RKG}(sk, ID)$ :
$(K, dk) \leftarrow sk$
$rk := F(K, 0\|ID)$
return $rk$.

$(\phi, pw_s) \leftarrow \mathsf{PRR}(C_P(ID, pw, rk) \leftrightarrow S_P(ID, rk, sk))$ :
$K \leftarrow sk$
1. $S_P$ chooses a randomness $r \in \{0,1\}^k$ and sends it to $C_P$
2. $C_P$ executes $c \leftarrow \mathsf{PEnc}(pk, ID\|r\|rk\|pw)$ and sends it to $S_P$
3-1. $S_P$ executes $ID'\|r'\|rk'\|pw' \leftarrow \mathsf{PDec}(dk, c)$
3-2. $S_P$ executes $pw_s := F(K, 1\|ID'\|pw')$
3-3. If $ID' = ID$, $r' = r$, and $rk' = rk$ hold, $S_P$ returns $pw_s$
      else $S_P$ returns $\perp$.

$(\phi, \top/\perp) \leftarrow \mathsf{Auth}(C_A(ID, pw) \leftrightarrow S_A(ID, pw_s, sk))$ :
$K \leftarrow sk$
1. $S_A$ chooses a randomness $r \in \{0,1\}^k$ and sends it to $C_A$
2. $C_A$ executes $c \leftarrow \mathsf{PEnc}(pk, ID\|r\|pw)$ and sends it to $S_A$
3-1. $S_A$ executes $ID'\|r'\|pw' \leftarrow \mathsf{PDec}(dk, c)$
3-2. If $ID' = ID$, $r' = r$, and $pw_s = F(K, 1\|ID\|pw')$ hold, $S_A$ returns $\top$
      else $S_A$ returns $\perp$.
```

Figure 6.2: The proposed generic construction of a password reset protocol with password salting

to *sid*. After that, $\mathcal{B}$ gives pp to $\mathcal{A}_1$. Then $\mathcal{A}_1$ can adaptively make the following types of queries[7]. We explain the meaning of SSession and Send queries in the paragraph of remark right after the Definition 12.

Client create query (CCreate)**:** This is exactly the same as the CCreate query in the Imp-PA game. If $\mathcal{A}_1$ makes the following queries (RKR, PRR, $\mathsf{SSession}_{\mathsf{PRR}}$, $\mathsf{SSession}_{\mathsf{Auth}}$) with an identity ID, then this ID must have appeared as a CCreate query (and thus be stored in the list L).

Password reveal query (PR)**:** This is exactly the same as the PR query in Imp-PA game.

Reset key reveal query (RKR)**:** This is exactly the same as the RKR query in Imp-PA game.

Start session query for password (re-)registration ($\mathsf{SSession}_{\mathsf{PRR}}$) On input $(ID, pw$

[7]PRR and Auth query are not considered in the following list of queries because $\mathcal{A}$ can perform those functionalities by making $\mathsf{Send}_{\mathsf{PRR}}$ and $\mathsf{Send}_{\mathsf{Auth}}$ queries, respectively.

$\in \mathcal{PW} \cup \{\phi\}$), $\mathcal{B}$ responds as follows. First, $\mathcal{B}$ generates a unique session ID sid and returns it to $\mathcal{A}$. If $pw \in \mathcal{PW}$, $\mathcal{B}$ stores $(sid, ID, pw, \phi, \phi, \phi)$ into the list L'_P and updates the tuple in the list L by $(ID, *, *, *, 0, *)$. Otherwise (that is, $pw = \phi$), $\mathcal{B}$ chooses a password pw' uniformly at random from $\mathcal{D}$, stores $(sid, ID, pw', \phi, \phi, \phi)$ into the list L'_P, and updates the tuple in the list L by $(ID, *, *, *, 1, *)$. When $\mathcal{A}_1$ makes a $\mathsf{Send}_{\mathsf{PRR}}$ query with a session ID sid, then this sid must have been generated by this $\mathsf{SSession}_{\mathsf{PRR}}$ query previously.

Start session query for authentication ($\mathsf{SSession}_{\mathsf{Auth}}$) On input $(ID, pw \in \mathcal{PW} \cup \{\phi\})$, $\mathcal{B}$ responds as follows. If $pw = \mathcal{PW}$, $\mathcal{B}$ generates a unique session ID sid and stores $(sid, ID, pw, \phi, \phi, \phi)$ into the list L'_A. Otherwise (that is, $pw = \phi$), $\mathcal{B}$ chooses a password pw' uniformly at random from $\mathcal{D}$, generates a unique session ID sid, and stores $(sid, ID, pw', \phi, \phi, \phi)$ into the list L'_A. When $\mathcal{A}_1$ makes a $\mathsf{Send}_{\mathsf{Auth}}$ query with a session ID sid, then this sid must have been generated by this $\mathsf{SSession}_{\mathsf{Auth}}$ query previously.

Send query for password (re-)registration ($\mathsf{Send}_{\mathsf{PRR}}$) On input (sid, i, M), $\mathcal{B}$ executes an appropriate response corresponding to i and M by using st_c and st_s in the list L'_P. Then, $\mathcal{B}$ updates the tuples in the lists L and L'_P.

Send query for authentication ($\mathsf{Send}_{\mathsf{Auth}}$) On input (sid, i, M), $\mathcal{B}$ executes an appropriate response corresponding to i and M by using st_c and st_s in the list L'_A. Then, $\mathcal{B}$ updates the tuples in the lists L and L'_A.

Finally, $\mathcal{A}_1$ outputs (ID^*, st). To win the Imp-AA game, the tuple $(ID^*, pw^*, pw^*_s, rk^*, frag^*_p, frag^*_r)$ must exist in the list L and satisfy $frag^*_p = 1$ and $frag^*_r = 1$ (if this is satisfied, we say that ID^* satisfies the "winning precondition"). If these conditions are not satisfied, $\mathcal{B}$ decides that $\mathcal{A}$ has lost the Imp-AA game. Otherwise, $\mathcal{B}$ gives st to $\mathcal{A}_2$. Then $\mathcal{A}_2$ and $\mathcal{B}$ interactively execute $\mathsf{Auth}(\mathcal{A}_2(st) \leftrightarrow S_A(ID^*, pw^*_s, sk))$. During the execution of this Auth protocol, $\mathcal{A}_2$ can adaptively make the queries in the same way as $\mathcal{A}_1$. However, $\mathcal{A}_2$ is not allowed to use ID^* in the PR, RKR, $\mathsf{SSession}_{\mathsf{PRR}}$ and $\mathsf{Send}_{\mathsf{PRR}}$ queries. Finally, $\mathcal{A}$ wins if the S_A's output of Auth is $\top$. We define the advantage of $\mathcal{A}$ by $\mathsf{Adv}^{\mathsf{Imp\text{-}AA}}_{\mathcal{A}}(k) = \Pr[\mathcal{A}\ wins]$.

Definition 11. *Let q_{S_A} be the number of $\mathsf{SSession}_{\mathsf{Auth}}$ queries by $\mathcal{A}_1$. We say that a password reset protocol is Imp-AA secure if for all PPT adversaries $\mathcal{A}$, $\mathsf{Adv}^{\mathsf{Imp\text{-}AA}}_{\mathcal{A}}(k)$ is $O(q_{S_A})/|\mathcal{D}| + \varepsilon(k)$.*

Illegal Registration Second, we consider security against illegal registration under active attacks (IR-AA) for a password reset protocol. This security is defined using the following IR-AA game which is played by the challenger $\mathcal{B}$ and an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$. $\mathcal{B}$'s initial procedure and $\mathcal{A}_1$'s queries of this IR-AA game are exactly the same as the Imp-AA game. Finally, $\mathcal{A}_1$ outputs (ID^*, st). To win the IR-AA game, the tuple $(ID^*, pw^*, pw^*_s, rk^*, frag^*_p, frag^*_r)$ must exist in the list L and satisfy $frag^*_r = 1$. If these conditions are not satisfied, $\mathcal{B}$ decides that $\mathcal{A}$ has lost the IR-AA game. Otherwise, $\mathcal{B}$ gives st to $\mathcal{A}_2$. Then $\mathcal{A}_2$ and $\mathcal{B}$ interactively execute $\mathsf{PRR}(\mathcal{A}_2(st) \leftrightarrow S_P(ID^*, rk^*, sk))$. During the execution of this PRR protocol, $\mathcal{A}_2$ can adaptively make the queries in the same way as $\mathcal{A}_1$. However, $\mathcal{A}_2$ is not allowed to use ID^* in the RKR, $\mathsf{SSession}_{\mathsf{PRR}}$, and $\mathsf{Send}_{\mathsf{PRR}}$ queries. Finally, $\mathcal{A}$ wins if S_P's output of PRR is different from $\perp$. We define that the advantage of $\mathcal{A}$ by $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{IR\text{-}AA}}(k) = \Pr[\mathcal{A}\ wins]$.

Definition 12. *We say that a password reset protocol is IR-AA secure if for all PPT adversaries $\mathcal{A}$, $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{IR\text{-}AA}}(k)$ is negligible.*

Remark

1. First, we explain the meaning of $\mathsf{SSession}(ID, pw)$ and $\mathsf{Send}(sid, i, M)$ queries. These queries are extension of a Send query that we can see in a security model of key exchange protocols. Before querying Send, an adversary $\mathcal{A}$ has to query $\mathsf{SSession}(ID, pw)$ and obtain sid previously. When this query is issued, $\mathcal{B}$ generates the tuple $(sid, ID, pw, *, *, *)$ in the list L'. If $\mathcal{A}$ issues (sid, i, M) as a Send query, $\mathcal{A}$ can obtain the correct execution result of algorithm that is generated by client/server. Here, M is inserted into the $(i+1)$-th message. In PRR algorithm of our proposed construction (in Fig. 6.1), for example, a client receives a randomness r from the server, executes $c \leftarrow \mathsf{PEnc}(pk, ID\|r\|rk\|pw)$, and returns c to the server. In this situation, $\mathcal{A}$ issues $\mathsf{Send}_{\mathsf{PRR}}(sid, 2, M)$ and can obtain $\mathsf{PEnc}(pk, ID\|M\|rk\|pw)$ from $\mathcal{B}$. Since there exists no input message in $\mathsf{Send}_{\mathsf{PRR}}(sid, 1, M)$, this M is always ϕ. In contrast to the key exchange protocol, an attack frag may revive by resetting the password that is randomly chosen by $\mathcal{B}$. Therefore, we introduce not only a $\mathsf{Send}_{\mathsf{PRR}}$ query but also $\mathsf{SSession}_{\mathsf{PRR}}$ query to manage the attack frag. The above example is the case of PRR, the same is true in Auth.

2. We require that Send query is executed in order. For example, an adversary $\mathcal{A}$ is not allowed to query $(sid, 2, M)$ without querying $(sid, 1, M)$, or $(sid, 3, M)$ right after querying $(sid, 1, M)$. This is because executions of algorithms without previous steps do not occur in practice. Moreover, we do not consider an adversary that queries $(sid, 1, M)$ after $(sid, 2, M)$. If the adversary wants to issue these queries, he/she has to issue $\mathsf{SSession}$ queries and re-setup the sid. The formalization that captures the above situation (slightly similar to the notion of resettable security [31]) is future work.

3. In our security definitions against active attacks, we omit the Trans queries which are allowed in the ones of passive attacks. This is because an adversary can realize the functionalities of Trans queries by using $\mathsf{SSession}$ queries and Send queries. The adversary $\mathcal{A}$ that wants to obtain a transcript $trans_{\mathsf{PRR}}$ responds as follows: First, $\mathcal{A}$ issues $\mathsf{CCreate}$ query (ID). Next, $\mathcal{A}$ issues $\mathsf{SSession}_{\mathsf{PRR}}$ query (ID, pw) and obtains sid. Then, $\mathcal{A}$ issues $\mathsf{Send}_{\mathsf{PRR}}$ $(sid, 1, \phi)$ and obtains r, $\mathsf{Send}_{\mathsf{PRR}}$ $(sid, 2, r)$ and obtains c, and $\mathsf{Send}_{\mathsf{PRR}}$ $(sid, 3, c)$ and obtains $\top/\perp$. Here, the tuple $(r, c, \top/\perp)$ is a transcript $trans_{\mathsf{PRR}}$ and a registration result that $\mathcal{A}$ can obtain when he/she issues $\mathsf{Trans}_{\mathsf{PRR}}$ (ID, pw). Although the above example is the case of $\mathsf{Trans}_{\mathsf{PRR}}$, the same is true in $\mathsf{Trans}_{\mathsf{Auth}}$.

Construction and Security Proofs Even if we consider the security against active adversaries, the construction is exactly the same as in Fig. 6.1. Here, we show the security proofs.

Theorem 15. *If F is a PRF and the PKE scheme is mIND-CCA secure*[8]*, then the proposed password reset protocol in Fig. 6.1 satisfies Imp-AA security.*

Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be an Imp-AA adversary of the password reset protocol. Let q_{P_i} and q_{A_i} be the numbers of $\mathsf{Send}_{\mathsf{PRR}}(sid, i, M)$ and $\mathsf{Send}_{\mathsf{Auth}}(sid, i, M)$ queries by $\mathcal{A}_1$, respectively. Here, q_{P_i} and q_{A_i} are polynomials of the security parameter. In our protocol, PRR and Auth are both two pass protocols. Therefore, we only have to consider the following three cases for $\mathsf{Send}_{\mathsf{PRR}}$ queries.

1. $r \leftarrow \mathsf{Send}_{\mathsf{PRR}}(sid, 1, \phi)$: This means that $\mathcal{A}$ gives ϕ to the server and obtains a randomness r.

2. $c \leftarrow \mathsf{Send}_{\mathsf{PRR}}(sid, 2, r)$: This means that $\mathcal{A}$ gives a randomness r to the client and obtains a ciphertext c.

3. $\phi \leftarrow \mathsf{Send}_{\mathsf{PRR}}(sid, 3, c)$: This means that $\mathcal{A}$ gives a ciphertext c to the server and obtains a registration result $\top/\bot$.

In the Imp-AA game, the challenger $\mathcal{B}$ responds as follows:

If $\mathcal{A}$ issues $\mathsf{Send}_{\mathsf{PRR}}(sid, 1, \phi)$, $\mathcal{B}$ chooses a randomness r, returns it to $\mathcal{B}$, and updates the tuple in the list L'_P by $(sid, *, *, *, r, r)$.

If $\mathcal{A}$ issues $\mathsf{Send}_{\mathsf{PRR}}(sid, 2, r)$, $\mathcal{B}$ first finds the tuples $(sid, ID, pw, *, *, *)$ in the list L'_P and $(ID, *, *, rk, *, *)$ in the list L. Then, $\mathcal{B}$ executes $c \leftarrow \mathsf{PEnc}(pk, ID\|r\|rk\|pw)$, returns c to $\mathcal{A}$, and updates the tuples in the list L'_P by $(sid, *, *, *, *, *\|c)$ and in the list L by $(ID, pw, *, *, *, *)$.

If $\mathcal{A}$ issues $\mathsf{Send}_{\mathsf{PRR}}(sid, 3, c)$, $\mathcal{B}$ first finds the tuples $(sid, ID, pw, *, r, *)$ in the list L'_P and $(ID, *, *, rk, *, *)$ in the list L. Then, $\mathcal{B}$ responds as follows: $\mathcal{B}$ executes $ID'\|r'\|rk'\|pw_s \leftarrow \mathsf{PDec}(dk, c)$. If this decryption result is $\bot$, $\mathcal{B}$ returns $\bot$ to $\mathcal{A}$ and does not execute the following steps. Then, $\mathcal{B}$ sets $z := \top$ if $ID = ID'$, $r = r'$, and $rk = rk'$ hold. Otherwise, $\mathcal{B}$ sets $z := \bot$. After that, $\mathcal{B}$ updates the tuples in the list L by $(ID, *, pw_s, *, *, *)$ and returns z to $\mathcal{A}$.

Similar to the cases of $\mathsf{Send}_{\mathsf{PRR}}$ queries, we only have to consider the following three cases for $\mathsf{Send}_{\mathsf{Auth}}$ queries.

1. $r \leftarrow \mathsf{Send}_{\mathsf{Auth}}(sid, 1, \phi)$: This means that $\mathcal{A}$ gives ϕ to the server and obtains a randomness r.

2. $c \leftarrow \mathsf{Send}_{\mathsf{Auth}}(sid, 2, r)$: This means that $\mathcal{A}$ gives a randomness r to the client and obtains a ciphertext c.

3. $\top/\bot \leftarrow \mathsf{Send}_{\mathsf{Auth}}(sid, 3, c)$: This means that $\mathcal{A}$ gives a ciphertext c to the server and obtains an authentication result $\top/\bot$.

[8]In the case of security against passive adversaries, 1-bounded CCA secure PKE is enough for the security proof. In the case of security against active adversaries, however, we need (full) CCA secure PKE for the security proof.

In the Imp-AA game, the challenger $\mathcal{B}$ responds as follows:

If $\mathcal{A}$ issues $\mathsf{Send}_{\mathsf{Auth}}(sid, 1, \phi)$, $\mathcal{B}$ chooses a randomness r, returns it to $\mathcal{B}$, and updates the tuple in the list L'_A by $(sid, *, *, *, r, r)$.

If $\mathcal{A}$ issues $\mathsf{Send}_{\mathsf{Auth}}(sid, 2, r)$, $\mathcal{B}$ first finds the tuples $(sid, ID, pw, *, *, *)$ in the list L'_A. Then, $\mathcal{B}$ executes $c \leftarrow \mathsf{PEnc}(pk, ID\|r\|pw)$, returns c to $\mathcal{A}$, and updates the tuple in the list L'_A by $(sid, *, *, *, *, *\|c)$.

If $\mathcal{A}$ issues $\mathsf{Send}_{\mathsf{Auth}}(sid, 3, c)$, $\mathcal{B}$ first finds the tuples $(sid, ID, pw, *, r, *)$ in the list L'_A and $(ID, *, pw_s, *, *, *)$ in the list L. Then, $\mathcal{B}$ executes $ID'\|r'\|pw'_s \leftarrow \mathsf{PDec}(dk, c)$. If the conditions $ID' = ID$, $r' = r$, and $pw'_s = pw_s$ are satisfied, $\mathcal{B}$ returns $z := \top$ to $\mathcal{A}$. Otherwise, $\mathcal{B}$ returns $z := \bot$ to $\mathcal{A}$. Then, $\mathcal{B}$ updates the tuple in the list L'_A by $(sid, *, *, *, *, *\|z)$.

Consider the following sequence of games.

Game 0. This is exactly the Imp-AA game.

Game 1. This game proceeds in the same way as Game 0, except that the first messages r of S_A picked in the execution of PRR (in the response to $\mathcal{A}$'s $\mathsf{Send}_{\mathsf{PRR}}$ query) and Auth (either in the response to $\mathcal{A}$'s $\mathsf{Send}_{\mathsf{Auth}}$ query, or Auth in the challenge phase), are picked from $\{0,1\}^k \backslash \{r\text{'s that are already used }\}$, so that they are all distinct and never collide. For notational convenience, in this and subsequent games, we introduce the list $\mathcal{R}$ that is used to store r's that are used in the response to a $\mathsf{Send}_{\mathsf{PRR}}$ query, $\mathsf{Send}_{\mathsf{Auth}}$ query, or in the execution of Auth in the challenge phase, and we make the challenger choose r uniformly at random from $\{0,1\}^k \backslash \mathcal{R}$ every time it needs to choose r for $\mathsf{Send}_{\mathsf{PRR}}$ and $\mathsf{Send}_{\mathsf{Auth}}$, and put the used r into the list $\mathcal{R}$.

Game 2. This game is the same as Game 1, except that as in Game 2 considered in the proof of Theorem 13, we replace the output of $F(K, \cdot)$ with a random string.

Game 3. This game proceeds in the same way as Game 2, except that following three points.

- If $\mathcal{A}_1$ issues $\mathsf{Send}_{\mathsf{PRR}}(sid, 2, r)$ queries on ID, $\mathcal{B}$ executes $c^* \leftarrow \mathsf{PEnc}(pk, ID\|r\| 0^{|rk|+|pw|})$ and uses c^* instead of $c \leftarrow \mathsf{PEnc}(pk, ID\|r\|rk\|pw)$.
- If $\mathcal{A}_1$ issues $\mathsf{Send}_{\mathsf{Auth}}(sid, 2, r)$ queries on ID, $\mathcal{B}$ executes $c^{**} \leftarrow \mathsf{PEnc}(pk, ID\|r\| 0^{|pw|})$ and uses c^{**} instead of $c \leftarrow \mathsf{PEnc}(pk, ID\|r\|pw)$.
- When $\mathcal{B}$ calculates the output of $\mathsf{Auth}(sid, 3, c)$ (i.e. $\top/\bot$) only by checking $pw = pw_s$ where pw_s is the value found in the tuple corresponding to ID in the list L, without running PDec.

For $i \in \{0, 1, 2, 3\}$, we define the event W_i as the event that $\mathcal{A}$ wins in Game i. The advantage of $\mathcal{A}$ is, by definition, $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Imp\text{-}AA}}(k) = \Pr[W_0]$. We complete the proof by using the following inequality, and the upper bounds in the terms in the right hand side are shown in Lemmas 28 to 31.

$$\Pr[W_0] \leq \sum_{i=0}^{2} |\Pr[W_i] - \Pr[W_{i+1}]| + \Pr[W_3] \tag{6.3}$$

where the above equality can be derived by the triangle inequality.

Lemma 28. $|\Pr[W_0] - \Pr[W_1]|$ *is negligible.*

Proof. The difference $|\Pr[W_0] - \Pr[W_1]|$ can be upperbounded by the statistical distance between the distributions of r's used in PRR and Auth (in the responses to $\mathcal{A}$'s $\mathsf{Send}_{\mathsf{PRR}}$ queries, $\mathsf{Send}_{\mathsf{Auth}}$ queries, and Auth in the challenge phase) in Game 0 and those in Game 1. Since the number of r's in the games is at most $(q_{P_1} + q_{A_1} + 1)$, the statistical distance between the distributions is at most $(q_{P_1} + q_{A_1} + 1)^2/2^k$. □

Lemma 29. *If F is a PRF,* $|\Pr[W_1] - \Pr[W_2]|$ *is negligible.*

Proof. This proof follows closely to the one of Lemma 21. Therefore, we omit it. The only difference between this Lemma and Lemma 21 is $\mathcal{B}$ has to choose a randomness r from $\{0,1\}^k \backslash \mathcal{R}$ not only for Auth but also for PRR. □

Lemma 30. *If the PKE scheme is mIND-CCA secure,* $|\Pr[W_2] - \Pr[W_3]|$ *is negligible.*

Proof. We show that we can construct a multi-challenge IND-CCA adversary $\mathcal{B}$ against the underlying PKE scheme. The description of $\mathcal{B}$ is as follows:

First, the challenger executes $(pk, dk) \leftarrow \mathsf{PKG}(1^k)$ and chooses a bit $b \in \{0,1\}$. Then, the challenger gives pk to $\mathcal{B}$. $\mathcal{B}$ chooses a random key $K \in \{0,1\}^k$ for PRF F and generates an empty list L which will be used to store tuples of the form $(ID, pw, pw_s, rk, frag_p, frag_r)$. $\mathcal{B}$ also generates two empty lists L'_P and L'_A which will be used to store tuples of the form $(sid, ID, pw, st_c, st_s, trans_{sid})$. Moreover, $\mathcal{B}$ generates an empty list $\mathcal{R}$. After that, $\mathcal{B}$ gives $pp := pk$ to $\mathcal{A}_1$.

When $\mathcal{A}_1$ makes a $\mathsf{CCreate}$ query ID, $\mathcal{B}$ responds as follows.

1. If there exists a tuple $(ID, *, *, *, *, *)$ in the list L, $\mathcal{B}$ does nothing.
2. Otherwise, $\mathcal{B}$ chooses $rk \in \{0,1\}^k$ uniformly at random from the range of F and gives rk to $\mathcal{A}_1$. After that, $\mathcal{B}$ stores the tuple $(ID, \phi, \phi, rk, 0, 1)$ into the list L.

When $\mathcal{A}_1$ makes a PR query ID, $\mathcal{B}$ finds the tuple $(ID, pw, *, *, *, *)$ in the list L, and returns pw to $\mathcal{A}_1$. Then, $\mathcal{B}$ updates the tuple in the list L by $(ID, *, *, *, 0, *)$.

When $\mathcal{A}_1$ makes a RKR query ID, $\mathcal{B}$ finds the tuple $(ID, *, *, rk, *, *)$ in the list L, and returns rk to $\mathcal{A}_1$. Then, $\mathcal{B}$ updates the tuple in the list L by $(ID, *, *, *, *, 0)$.

When $\mathcal{A}_1$ makes a $\mathsf{SSession}_{\mathsf{PRR}}$ query $(ID, pw \in \mathcal{PW} \cup \{\phi\})$, $\mathcal{B}$ responds as follows: First, $\mathcal{B}$ generates a unique sid and returns it to $\mathcal{A}$. If $pw \in \mathcal{PW}$, $\mathcal{B}$ stores $(sid, ID, pw, \phi, \phi, \phi)$ into the list L'_P and updates the tuple in the list L by $(ID, *, *, *, 0, *)$. Otherwise (that is $pw = \phi$), $\mathcal{B}$ chooses pw' uniformly at random from $\mathcal{D}$, stores $(sid, ID, pw', \phi, \phi, \phi)$ into the list L'_P, and updates the tuple in the list L by $(ID, *, *, *, 1, *)$.

When $\mathcal{A}_1$ makes a $\mathsf{SSession}_{\mathsf{Auth}}$ query $(ID, pw \in \mathcal{PW} \cup \{\phi\})$, $\mathcal{B}$ responds as follows: First, $\mathcal{B}$ generates a unique sid and returns it to $\mathcal{A}$. If $pw \in \mathcal{PW}$, $\mathcal{B}$ stores $(sid, ID, pw, \phi, \phi, \phi)$ into the list L'_A. Otherwise, (that is, $pw = \phi$), $\mathcal{B}$ chooses a password pw' uniformly at random from $\mathcal{D}$ and stores $(sid, ID, pw', \phi, \phi, \phi)$ into the list L'_A.

When $\mathcal{A}_1$ makes a $\mathsf{Send}_{\mathsf{PRR}}$ query $(sid, 1, \phi)$, $\mathcal{B}$ responds as follows: $\mathcal{B}$ first chooses a randomness r^* from $\{0,1\}^k \backslash \mathcal{R}$ and returns it to $\mathcal{A}_1$. Then, $\mathcal{B}$ updates the tuple in the list L'_P by $(sid, *, *, *, r^*, r^*)$ and adds r^* to the list $\mathcal{R}$.

When $\mathcal{A}_1$ makes a $\mathsf{Send}_{\mathsf{PRR}}$ query $(sid, 2, M)$, $\mathcal{B}$ responds as follows: First, $\mathcal{B}$ finds the tuples $(ID, *, *, rk, *, *)$ in the list L and $(sid, ID, pw, *, *, *)$ in the list L'_P. Then, $\mathcal{B}$ submits $(ID\|M\|rk\|pw, ID\|M\|0^{|rk|+|pw|})$ to the challenger and receives c^*. This c^* is $\mathsf{PEnc}(pk, ID\|M\|rk\|pw)$ if $b = 0$ and is $\mathsf{PEnc}(pk, ID\|M\|0^{|rk|+|pw|})$ if $b = 1$. Then, $\mathcal{B}$ returns c^* to $\mathcal{A}_1$. After that, $\mathcal{B}$ updates the tuples in the list L'_P by $(sid, *, *, *, *, *\|c^*)$ and L by $(ID, pw, *, *, *, *)$.

When $\mathcal{A}_1$ makes a $\mathsf{Send}_{\mathsf{PRR}}$ query $(sid, 3, M)$, $\mathcal{B}$ responds as follows:

- If $c(= M)$ is not the challenge ciphertext of $\mathcal{B}$, $\mathcal{B}$ finds the tuples $(ID, *, *, rk, *, *)$ in the list L and $(sid, ID, *, *, r, *)$ in the list L'_P. $\mathcal{B}$ submits $c(= M)$ as a decryption query to the challenger and receives $ID'\|r'\|rk'\|pw_s$. Then, $\mathcal{B}$ sets $z := \top$ if the conditions $ID' = ID$, $r' = r$, and $rk' = rk$ hold. Otherwise, $\mathcal{B}$ sets $z := \bot$. After that, $\mathcal{B}$ updates the tuple in the list L by $(ID, *, pw_s, *, *, *)$ and returns z to $\mathcal{A}$.

- If $c(= M)$ is one of the challenge ciphertexts of $\mathcal{B}$, $\mathcal{B}$ cannot submit c as a decryption query to the challenger. Here, we denote the queried c as $\widehat{c}$. First, $\mathcal{B}$ finds the tuples $(sid, ID, *, *, r, *\|c)$ and $(\widehat{sid}, \widehat{ID}, \widehat{pw}, *, \widehat{r}, *\|\widehat{c})$ in the list L'_P, and $(ID, *, *, rk, *, *)$ and $(\widehat{ID}, *, *, \widehat{rk}, *, *)$ in the list L. Then, $\mathcal{B}$ sets $z := \top$ if the conditions $ID = \widehat{ID}$, $r = \widehat{r}$, and $rk = \widehat{rk}$ hold. Otherwise, $\mathcal{B}$ sets $z := \bot$. After that, $\mathcal{B}$ sets $pw_s := \widehat{pw}$, updates the tuples in the list L by $(ID, *, pw_s, *, *, *)$, and returns z to $\mathcal{A}$.

When $\mathcal{A}_1$ makes a $\mathsf{Send}_{\mathsf{Auth}}$ query $(sid, 1, \phi)$, $\mathcal{B}$ responds as follows: $\mathcal{B}$ first chooses a randomness r^* from $\{0,1\}^k \backslash \mathcal{R}$ and returns it to $\mathcal{A}_1$. Then, $\mathcal{B}$ updates the tuple in the list L'_A by $(sid, *, *, *, r^*, r^*)$ and adds r^* to the list $\mathcal{R}$.

When $\mathcal{A}_1$ makes a $\mathsf{Send}_{\mathsf{Auth}}$ query $(sid, 2, M)$, $\mathcal{B}$ responds as follows: First, $\mathcal{B}$ finds the tuple $(sid, ID, pw, *, *, *)$ in the list L'_A. Then, $\mathcal{B}$ submits $(ID\|M\|pw, ID\|M\|0^{|pw|})$ to the challenger and receives c^*. This c^* is $\mathsf{PEnc}(pk, ID\|M\|pw)$ if $b = 0$ and is $\mathsf{PEnc}(pk, ID\|M\| 0^{|pw|})$ if $b = 1$. Then, $\mathcal{B}$ returns c^* to $\mathcal{A}_1$. After that, $\mathcal{B}$ updates the tuple in the list L'_A by $(sid, *, *, *, *, *\|c^*)$.

When $\mathcal{A}_1$ makes a $\mathsf{Send}_{\mathsf{Auth}}$ query $(sid, 3, M)$, $\mathcal{B}$ responds as follows: First, $\mathcal{B}$ finds the tuples $(ID, *, pw_s)$ in the list L and $(sid, ID, *, *, r, *)$ in the list L'_A. Then, $\mathcal{B}$ submits $c(= M)$ as a decryption query to the challenger and receives $ID'\|r'\|pw'_s$. If the condition $pw'_s = pw_s$ holds, $\mathcal{B}$ returns $z := \top$ to $\mathcal{A}_1$. Otherwise, $\mathcal{B}$ returns $z := \bot$ to $\mathcal{A}_1$. After that, $\mathcal{B}$ updates the tuples in the list L'_A by $(sid, *, *, *, *, *\|z)$.

Finally, $\mathcal{A}_1$ terminates with (ID^*, st). $\mathcal{B}$ outputs $b' = 1$ and aborts when the identity ID^* does not satisfy the winning precondition. Otherwise, $\mathcal{B}$ chooses r^* uniformly at random from $\{0,1\}^k \backslash \mathcal{R}$, and gives r^* and st to $\mathcal{A}_2$. $\mathcal{B}$ can respond to the queries from $\mathcal{A}_2$ in the same way as $\mathcal{B}$ did for $\mathcal{A}_1$. However, when $\mathcal{A}_2$ submits ID^* as PR, RKR, $\mathsf{SSession}_{\mathsf{PRR}}$, and $\mathsf{Send}_{\mathsf{PRR}}$ queries, $\mathcal{B}$ returns $\bot$ to $\mathcal{A}_2$. Finally, $\mathcal{A}_2$ terminates with c^*. If this c^* is one of the ciphertexts used as a response to the $\mathsf{Send}_{\mathsf{PRR}}$ or $\mathsf{Send}_{\mathsf{Auth}}$ queries, $\mathcal{B}$ stops the Imp-AA game, decides that $\mathcal{A}$ has lost the Imp-AA game, and terminates with $b' = 1$. Next, $\mathcal{B}$ submits c^* as a decryption query to the challenger, and receives $ID^*\|r'^*\|pw'^*$. Then, $\mathcal{B}$ finds the tuple $(sid, ID^*, pw^*, *, *, *)$ in the list L'_P, checks whether the conditions $r'^* = r^*$ and $pw'^* = pw^*$ hold or not. If these conditions hold, $\mathcal{B}$ terminates with output $b' = 0$. Otherwise, $\mathcal{B}$ terminates with output $b' = 1$.

The above completes the description of $\mathcal{B}$. It is not hard to see that $\mathcal{B}$ perfectly simulates Game 2 when $\mathcal{B}$'s challenge bit $b = 0$, and Game 3 when $b = 1$. When the challenge bit of $\mathcal{B}$ is 0 and $\mathcal{B}$ does not abort before $\mathcal{A}$ terminates, $\mathcal{B}$ responses to $\mathcal{A}$'s queries are distributed identically to those in Game 2. In addition, $\mathcal{B}$ outputs 0 only if $\mathcal{B}$ does not abort and $\mathcal{A}_2$ succeeds in outputting a ciphertext c^* satisfying $\mathsf{PDec}(dk, c^*) = ID^*\|r^*\|pw'^*$. Therefore, $\Pr[b' = 0|b = 0] = \Pr[W_2]$. On the other hand, when the challenge bit of $\mathcal{B}$ is 1, $\mathcal{A}_2$ succeeds in outputting a ciphertext c^* satisfying $\mathsf{PDec}(dk, c^*) = ID^*\|r'^*\|0^{|pw'^*|}$, and this situation is the same as Game 3. With almost the same discussion as above, we have $\Pr[b' = 0|b = 1] = \Pr[W_3]$. Therefore, $\mathsf{Adv}^{\mathsf{mIND\text{-}CCA}}_{\mathcal{B}}(k) = |\Pr[b' = b] - 1/2| = (1/2)|\Pr[b' = 0|b = 0] - \Pr[b' = 0|b = 1]| = (1/2)|\Pr[W_2] - \Pr[W_3]|$. Using this equality, and recalling the assumption that the underlying PKE scheme is mIND-CCA secure, we conclude that $|\Pr[W_2] - \Pr[W_3]|$ is negligible. $\square$

Lemma 31. $\Pr[W_3] \le O(q_{A_3})/|\mathcal{D}| + \varepsilon(k)$.

Proof. Note that in Game 3, the responses to the $\mathsf{Send_{PRR}}$ and $\mathsf{Send_{Auth}}$ queries contain no information of pw. However, if an adversary makes a $\mathsf{Send_{Auth}}(sid, 3, c)$ query, the "server's output" part (i.e. $\top$ or $\bot$) leaks whether $pw = pw_s$ or not. However, other than this, no information about pw leaks. Since pw is chosen randomly from $\mathcal{D}$, the probability that $\mathcal{A}$ wins in Game 3 is at most $(q_{A_3} + 1)/|\mathcal{D}|$. $\square$

Theorem 16. *If F is a PRF and the PKE scheme is mIND-CCA secure*[9]*, then the proposed password reset protocol in Fig. 6.1 satisfies IR-AA security.*

Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be an IR-AA adversary of the password reset protocol. Let q_{P_i} and q_{A_i} be the numbers of $\mathsf{Send_{PRR}}(sid, i, M)$ and $\mathsf{Send_{Auth}}(sid, i, M)$ queries by $\mathcal{A}_1$, respectively. Here, q_{P_i} and q_{A_i} are polynomials of the security parameter. Same as the case of Theorem 15, we only have to consider the three cases for $\mathsf{Send_{PRR}}$ and $\mathsf{Send_{Auth}}$ queries, respectively. Since the response of challenger is completely same as the case of Theorem 15, we omit them. Consider the following sequence of games.

Game 0. This is exactly the IR-AA game.

Game 1. This game proceeds in the same way as Game 0, except that the first messages r of S_A picked in the execution of PRR (in the response to $\mathcal{A}$'s $\mathsf{Send_{PRR}}$ query) and Auth (either in the response to $\mathcal{A}$'s $\mathsf{Send_{Auth}}$ query, or Auth in the challenge phase), are picked from $\{0,1\}^k \backslash \{r\text{'s that are already used }\}$, so that they are all distinct and never collide. As with Game 1 in Theorem 15, we introduce the list $\mathcal{R}$ and use it in the same manner.

Game 2. This game proceeds in the same way as Game 1, except that except that as in Game 2 considered in the proof of Theorem 13, we replace the output of $F(K, \cdot)$ with a random string.

Game 3. This game proceeds in the same way as Game 2, except that If $\mathcal{A}_1$ issues $\mathsf{Send_{PRR}}(sid, 2, r)$ queries on ID, $\mathcal{B}$ executes $c^* \leftarrow \mathsf{PEnc}(pk, ID\|r\|0^{|rk|+|pw|})$ and uses c^* instead of $c \leftarrow \mathsf{PEnc}(pk, ID\|r\|rk\|pw)$.

[9]We need (full) CCA secure PKE for the security proof.

For $i \in \{0,1,2,3\}$, we define the event W_i as the event that $\mathcal{A}$ wins in Game i. The advantage of $\mathcal{A}$ is, by definition, $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{IR\text{-}AA}}(k) = \Pr[W_0]$. We complete the proof by using the following inequality, and the upper bounds in the terms in the right hand side are shown in Lemmas 32 to 35.

$$\Pr[W_0] \leq \sum_{i=0}^{2} |\Pr[W_i] - \Pr[W_{i+1}]| + \Pr[W_3] \tag{6.4}$$

where the above equality can be derived by the triangle inequality.

Lemma 32. $|\Pr[W_0] - \Pr[W_1]|$ *is negligible.*

Proof. This proof follows closely to that of Lemma 28. Therefore, we omit it. □

Lemma 33. *If F is a PRF,* $|\Pr[W_1] - \Pr[W_2]|$ *is negligible.*

Proof. This proof follows closely to that of Lemma 29. Therefore, we omit it. □

Lemma 34. *If the PKE scheme is mIND-CCA secure,* $|\Pr[W_2] - \Pr[W_3]|$ *is negligible.*

Proof. The only difference between the proof of this lemma and Lemma 30 is the $\mathcal{B}$'s response to a $\mathsf{Send_{Auth}}$ query $(sid, 2, M)$. Therefore, we show the response to this query and omit the other parts. More concretely, in Lemma 30, the challenger submits $(ID\|M\|pw$, $ID\|M\|0^{|pw|})$ as a challenge query. In this lemma, however, we do not have to execute this and the challenger responds as follows (This is the same responses as Game 1.):

When $\mathcal{A}_1$ makes a $\mathsf{Send_{Auth}}$ query $(sid, 2, M)$, $\mathcal{B}$ responds as follows: $\mathcal{B}$ first finds the tuples $(sid, ID, pw, *, *, *)$ in the list L'_A. Then, $\mathcal{B}$ executes $c \leftarrow \mathsf{PEnc}(pk, ID\|r\|pw)$, returns c to $\mathcal{A}$, and updates the tuple in the list L'_A by $(sid, *, *, *, *, *\|c)$. □

Lemma 35. $\Pr[W_3]$ *is negligible.*

Proof. Note that in Game 3, the responses to the $\mathsf{Send_{PRR}}$ queries contain no information of rk. Therefore, the probability that $\mathcal{A}$ succeeds the guess of rk (that is, the probability that $\mathcal{A}$ wins in Game 3) is negligible. □

Lemmas 32 to 35 guarantee that the right hand side of the inequality (6.4) is negligible, and thus $\mathcal{A}$ has negligible advantage in the IR-AA game. This completes the proof of Theorem 16. □

6.5 Implementation

To show the practical feasibility and test the performance of our protocol, we implemented a prototype of our protocol (excluding the communication part) in Python. We implemented our first scheme (Fig. 6.1), and all cryptographic operations are performed using the python cryptography toolkit (Pycrypto 2.6.1) over Python 2.7.10. We expect that performance can be improved by using other appropriate libraries (e.g. Number Theory Library [103]).

We implement our scheme on a laptop computer (Windows 7 (64bit), Core i7-M640 2.80GHz, 8GB RAM). Table 6.1 shows the computational costs of our first protocol. We

Table 6.1: Performance of our scheme (computational cost)

Name of algorithms	Execution time [μs]
RKG	2.61
PRR1	2.68
PRR2	3.67×10^3
PRR3	4.67×10^4
Auth1	2.71
Auth2	3.62×10^3
Auth3	4.67×10^4

implement $F(K, x)$ in our construction as $H(K\|x)$ where H is SHA-256. This means that we regard SHA-256 hash function as a random oracle. We adopt RSA-OAEP(2048bit) as a PKE scheme. We omit to calculate the the execution time of SSetup algorithm because this operation is executed only once and does not rely on the participation of users. PRR1(Auth1) denotes the first server side execution, PRR2(Auth2) denotes the first client side execution, and PRR3(Auth3) denotes the second server side execution. We set the length of ID as 10 alphanumeric characters, randomnesses that is used in the first pass of PRR/Auth as 128bit, and passwords that is used in the PRR/Auth as 16 alphanumeric characters. We execute each algorithm 10000 times, and show its average time in Table 6.1.

Although the executions of PRR3 and Auth3 (the dominant part of these algorithms is decryption of RSA-OAEP(2048bit)) take more time than other algorithms, even these two algorithms need less than 50ms. Therefore, we can see that our scheme is suitable for practical use.

6.6 Conclusion

In this chapter, we proposed a model, security definitions, and a construction of a provably secure password reset protocol. Our generic construction is based on a PRF and PKE. We can construct a number of concrete password reset protocols from this generic construction.

Countermeasures against server breach is one of the future works. It is interesting to introduce a server decentralization to the password reset protocol like Camenisch et al.'s decentralized password verification protocol [30]. Proposing another model and security definitions is also a future work. In this chapter, we proposed a model that introduces a reset key and two security definitions (impersonation/illegal registration). However, there may exist a more appropriate model and security definitions depending on the system context. Especially, our security definitions against active adversary does not capture the notions of resettable security [31]. We believe that this research opens a door to rigorous security treatment of password reset protocols and that our proposed model/schemes can be a foundation there.

Chapter 7

Conclusion

In this thesis, we showed the results of provably secure cryptosystems and security protocols. In the following, we summarize the contributions of this thesis, and conclude this thesis with future prospects.

7.1 Summary of Contributions

Provably Secure Applied Cryptosystems: In Chapter 3, we proposed four new constructions of re-splittable TPKE schemes. Our new constructions are based on different cryptographic assumptions (DLIN, DBDH, HDH, and CDH) and there exists a trade-off between the hardness assumptions and efficiency of algorithms. It seems that the re-splittable TPKE is close to the standard TPKE and we can easily extend the standard TPKE schemes to the re-splittable ones. However, this is not true. The property that we can re-distribute decryption key shares is useful in practical cases that we use TPKE in the real world. In practice, we can handle flexible changes of the number of the entities or the threshold by changing the values of t and n. We cannot see this property in the standard TPKE. Moreover, re-splittable TPKE plays an important role in the generic construction of a proxy re-encryption scheme. We can obtain many concrete proxy re-encryption schemes via this constructions of re-splittable TPKE schemes.

In Chapter 4, we proposed a proxy re-encryption scheme with re-encryption verifiability. We introduced a new functionality for PRE that we call re-encryption verifiability, proposed a generic construction, and proved its security. In practice, we can reduce the level of trust we have to put on the proxies by achieving the functionality of re-encryption verification. Not only we added a functionality of re-encryption verifiability, but also we showed the backward compatibility between new security definitions and previous ones. That is, secure VPRE scheme automatically becomes secure PRE scheme. Moreover, we showed the construction of VPRE scheme by extending the previous (standard) PRE scheme and proved its security.

In Chapter 5, we proposed a construction of a multi-hop uni-directional proxy re-encryption scheme. The constructions of multi-hop bi-directional PRE schemes and single-hop uni-directional PRE schemes have already proposed. However, there is no previous construction of multi-hop uni-directional PRE. Therefore, this is a first construction of multi-hop uni-directional PRE scheme. We defined the model and security definitions, constructed a concrete scheme via a cryptographic obfuscator, and proved its CPA security. Since the obfuscator does not work with realistic time, we cannot expect the practicality of this scheme now. On the other hand, the technique which was used in the security proof of this scheme seems somewhat interesting.

Provably Secure Protocols: In Chapter 6, we extended a concept of provable security to the security protocols other than cryptosystems. More concretely, this was a result about backup authentication protocols that we call "password reset protocol". Although the security of password reset protocol has been evaluated in a heuristic manner, we introduce a concept of provable security to this field. First, we defined the model and security definitions. In our security definitions, we considered not only passive attacks but also active attacks. Next, we proposed a generic construction based on pseudorandom function and public key encryption. Then, we showed the reduction security of our protocols. Finally, we implemented a prototype to evaluate the efficiency of our protocol. The results of our experiment showed that our protocol is practical enough. In addition, we can expect the progress of theoretical analysis for password reset protocols based on this results.

7.2 Future Prospects

In this section, we describe the future prospects that can be considered from this thesis.

Security Definitions of Applied Cryptosystems and Protocols. In this thesis, we treated provably secure cryptosystems and protocols. We would like to describe that these two themes and results may give a good effect with each other.

There is no doubt that we can extend the possibility of new information services via improved applied cryptosystems. For example, we can show that the flexible and scalable contents distribution systems via attribute-based cryptosystems [97], encrypted databases via keyword searchable encryption [23] or order preserving encryption [1], file sharing systems via proxy cryptosystems [19], and outsourcing of computation via multi party computation [112] or homomorphic cryptosystems [86, 52] as examples. Secure and efficient applied cryptosystems support these convenient online systems.

On the other hand, there may also exist a contribution for applied cryptosystems (or the theory of cryptosystems) by considering the provably secure systems. When we define the model and security for cryptosystems, we (to some extent) cannot avoid containing the human factor in those models. Therefore, those models have to be discussed by many researchers to win a de-facto standard. Like the example of group signature, however, it is not uncommon to find vulnerable points even in the models and security definitions that have already been widely spread. There exists a possibility to find such insufficient points on the models and security definitions in a more simple way by considering provably secure systems. When we consider provably secure systems, we have to discuss the models and security definitions deeply. That is, we have to consider who may be an adversary, what types of attacks can be considered, and how security should be assured or not be assured in the context of real world. When we prove its security by reducing to the (applied) cryptosystems which are used as building blocks, that security proof may not go well. This failure means the either of following three cases:

1. To begin with, there exists an impossibility for proving the security.
2. We have to need other building blocks.
3. We need other security notions for underlying (applied) cryptosystems.

In the last case, the security definitions of those applied cryptosystems may be insufficient when we use them in practice. This is an example that the research of systems can feedback good effects to the research of cryptosystems. Regret to say, we cannot expect this effect in this thesis because the underlying building blocks which are used in our password reset protocol are very basic ones. When the research of provably secure systems based on applied cryptosystems make progress, however, we can expect the above effect. Moreover, there may exist merits not only new security definitions but also the existing ones. For example, in our security definitions of password reset protocols, internal states of challenger are changed by the password re-registration queries which are issued by the adversary. We have no idea how to treat this situation in the context of resettable security [31] where the adversary can reset the states of simulator. Like this example, it will appear the cases that we have to reconsider security definitions of (applied) cryptosystems by considering the provable security for real world systems. We would like to hope that the research on rigorous security of real world systems will make progress and contribute to both practical and theoretical security research.

The Relation between Reduction Security and Formal Method. Formal methods are usually used to verify the security of cryptosystems and cryptographic protocols. Although formal methods can check the validity of security proof, however, it cannot design a security model so far. That is, how to define security is still human tasks and this means that showing reduction security is still very valuable. As we showed in this thesis, the possibility of containing human errors in security proofs of applied cryptosystems is not low since recent ones are very complicated. In this sense, formal methods are also very valuable.

The stream as follows will be desirable from the viewpoint of quality assurance for cryptosystems, cryptographic protocols, and security products.

1. We define the rigorous security model and prove its security by reduction.
2. We check the validity of that security proof using formal methods.
3. We feedback the verification results to the security definitions and proofs.

We need more developments of formal methods themselves to make an above stream since recent formal methods are still meager. In addition, cryptographers should promote the research such as Chapter 6 in this thesis and pile up the research results of “provably secure heuristically secure protocols”. Such research results will extend the scope of formal methods and contribute the research of formal methods themselves.

How the Research of Provably Secure Protocols Make Progress? In this thesis, we discuss the security of password reset protocols. Password reset protocols are part of the online user authentication systems. It is easy to imagine to consider the provable security of protocols that are used on the Internet in practice. By considering a wider perspective, it may possible to rigorously prove the security or property of softwares. For example, we may be able to construct a "malware-undownloadable web browser" by assuming some security properties for underlying machine learning or specification of programming language. We can consider other large-scale example. We may be able to extend the scope of provable security to the control systems. When we define security models of some systems, we will have to introduce the peculiar situations for that ones. In the case of control systems, for example, "we cannot easily shutdown the systems.", "there exist high incentives to use the legacy systems", and so on.

We would like to pay attention to a hierarchical structure that is existing in this research field. Roughly speaking, there exists a hierarchical structure as follows.

$$[\text{Number Theory}] \to [\text{Cryptosystems}] \to [\text{Protocols}] \to [\text{Systems}]$$

When we consider the provable security of a certain part, it is convenient for us to use the parts (and its security) in one under layer. In our research on password reset protocols, we use some cryptographic tools in a black-box manner. In the concrete example of future direction that is written in above paragraph, we assumed not hardness on number theory or cryptosystems but hardness on machine learning or specification of programming language. Therefore, the following two research directions will be valuable to reach provably secure systems.

1. We should show many provably secure protocols. In applied cryptosystems, there exists hierarchy among them, and cryptosystems with high functionality, which is close to protocols (e.g. functional encryption [97], proxy re-encryption [19], and searchable encryption [23]) have been constructed by combining (slightly) basic applied cryptosystems. Similar to these cases, protocols with high functionality, which is close to systems will be constructed by combining basic protocols. By continuing such research, we will be able to reach the research on provably secure systems.

2. We should seek the new candidate destinations of security reduction. In the systems or protocols, we sometimes have to prove the properties like "proof of possession" or "proof of being a human". These proofs are different from the knowledge proofs, which are often used in a construction of cryptosystems. The possibility for constructing provably secure systems/protocols will extend if we can use the tools (and their security) for such proofs to construct systems/protocols. CAPTCHA(Completely Automated Public Turing test to tell Computers and Humans Apart) [2] is representative example to show the "proof of being a human".[1] Of course, it does not matter the research on other than CAPTCHA. To the best of our knowledge however, there exists no such a research.

[1]Although there exist some attack results against CAPTCHA that is used in practice (e.g. [90]), it may be possible to reduce the security of CAPTCHA to number theory in the future.

In Chapter 6 of this thesis, we showed the provable security results that connect between "Cryptosystems" and "Protocols". By proceeding above types of research, we would like to reach provable security results that connect between "Protocols" and "Systems". In the end, we would like to reach more secure society via these provably secure cryptosystems, protocols, and systems.

Bibliography

[1] R. Agrawal, J. Kiernan, R. Srikant, Y. Xu. Order-Preserving Encryption for Numeric Data. *SIGMOD 2004*, pp.563–574, 2004.

[2] L.v. Ahn, M. Blum, N.J. Hopper, J. Langford. CAPTCHA: Using Hard AI Problems for Security. *EUROCRYPT 2003*, LNCS 2656, pp.294–311, 2003.

[3] A. Akavia, S. Goldwasser, V. Vaikuntanathan. Simultaneous Hardcore Bits and Cryptography against Memory Attacks. *TCC 2009*, LNCS 5444, pp.474–495, 2009.

[4] J.H. An, Y. Dodis, T. Rabin. On the Security of Joint Signature and Encryption. *EUROCRYPT 2002*, LNCS 2332, pp.83–107, 2002.

[5] S. Arita, K. Tsurudome. Construction of Threshold Public-Key Encryptions through Tag-Based Encryptions. *ACNS 2009*, LNCS 5536, pp.186–200, 2009.

[6] G. Ateniese, K. Fu, M. Green, S. Hohenberger. Improved proxy re-encryption schemes with applications to secure distributed storage. *ACM Trans. Inf. Syst. Secur. 9(1)*, pp.1–30, 2006.

[7] B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S.P. Vadhan, K. Yang. On the (im)possibility of obfuscating programs. *J. ACM 59(2)*, pp.6, 2012.

[8] M. Bellare, A. Boldyreva, K. Kurosawa, J. Staddon. Multirecipient Encryption Schemes: How to Save on Bandwidth and Computation Without Sacrificing Security. *IEEE Trans. on IT. 53(11)*, pp.3927–3943, 2007.

[9] M. Bellare, A. Boldyreva, S. Micali. Public Key Encryption in a Multi-user Setting: Security Proofs and Improvements. *EUROCRYPT 2000*, LNCS 1807, pp.259–274, 2000.

[10] M. Bellare, A. Desai, D. Pointcheval, P. Rogaway. Relations Among Notions of Security for Public-Key Encryption Schemes. *CRYPTO 1998*, LNCS 1462, pp.26–45, 1998.

[11] M. Bellare, D. Hofheinz, E. Kiltz. Subtleties in the Definition of IND-CCA: When and How Should Challenge Decryption Be Disallowed? *J. Cryptology*, 28(1), pp.29-48, 2015.

[12] M. Bellare, T. Kohno. A Theoretical Treatment of Related-Key Attacks: RKA-PRPs, RKA-PRFs, and Applications. *EUROCRYPT 2003*, LNCS 2656, pp.491–506, 2003.

[13] M. Bellare, C. Namprempre. Authenticated Encryption: Relations among Notions and Analysis of the Generic Composition Paradigm. *J. Cryptology*, Volume 21, No.4, pp.469–491, 2008.

[14] M. Bellare, C. Namprempre, G. Neven. Security Proofs for Identity-Based Identification and Signature Schemes. *J. Cryptology*, Volume 22, No.1, pp.1–61, 2009.

[15] M. Bellare, G. Neven. Multi-Signatures in the Plain Public-Key Model and a General Forking Lemma. *ACMCCS 2006*, pp.390–399, 2006.

[16] M. Bellare, D. Pointcheval, P. Rogaway. Authenticated Key Exchange Secure Against Dictionary Attacks. *EUROCRYPT 2000*, LNCS1807, pp.139–155, 2000.

[17] S.M. Bellovin, M. Merritt. Encrypted Key Exchange: Password-based Protocols Secure Against Dictionary Attacks. *IEEE Symposium on Security and Privacy 1992*, pp.72–84, 1992.

[18] J. Black, P. Rogaway, T. Shrimpton. Encryption-Scheme Security in the Presence of Key-Dependent Messages. *SAC 1998*, LNCS 2595, pp.62–75, 2002.

[19] M. Blaze, G. Bleumer, M. Strauss. Divertible Protocols and Atomic Proxy Cryptography. *EUROCRYPT 1998*, LNCS 1403, pp.127–144, 1998.

[20] A. Boldyreva. Threshold Signatures, Multisignatures and Blind Signatures Based on the Gap-Diffie-Hellman-Group Signature Scheme. *PKC 2003*, LNCS 2567, pp.31–46, 2003.

[21] D. Boneh, X. Boyen. Efficient Selective-ID Secure Identity Based Encryption without Random Oracles. *EUROCRYPT 2004*, LNCS 3027, pp.223–238, 2004.

[22] D. Boneh, X. Boyen, S. Halevi. Chosen Ciphertext Secure Public Key Threshold Encryption Without Random Oracles. *CT-RSA 2006*, LNCS 3860, pp.226–243, 2006.

[23] D. Boneh, G.D Crescenzo, R. Ostrovsky, G. Persiano. Public Key Encryption with Keyword Search. *EUROCRYPT 2004*, LNCS 3027, pp.506–522, 2004.

[24] D. Boneh, M.K. Franklin. Identity-Based Encryption from the Weil Pairing. *CRYPTO 2001*, LNCS2139, pp.213–229, 2001.

[25] D. Boneh, I. Shparlinski. On the Unpredictability of Bits of the Elliptic Curve Diffie–Hellman Scheme. In *Proc. of CRYPTO 2001*, LNCS 2139, pp.201–212, 2001.

[26] D. Boneh, M. Zhandry. Multiparty Key Exchange, Efficient Traitor Tracing, and More from Indistinguishability Obfuscation. *CRYPTO 2014(1)*, LNCS8616, pp.480–499, 2014.

[27] J. Bonneau, E. Bursztein, I. Caron, R. Jackson, M. Williamson. Secrets, Lies, and Account Recovery: Lessons from the Use of Personal Knowledge Questions at Google. *WWW 2015*, pp.141–150, 2015.

[28] X. Boyen, Q. Mei, B. Waters. Direct Chosen Ciphertext Security from Identity-based Techniques. *ACMCCS 2005*, pp.320–329, 2005.

[29] V. Boyko, P.D. MacKenzie, S. Patel. Provably Secure Password-Authenticated Key Exchange Using Diffie-Hellman. *EUROCRYPT 2000*, LNCS1807, pp.156–171, 2000.

[30] J. Camenisch, A. Lehmann, G. Neven. Optimal Distributed Password Verification. *CCS 2015*, pp.182–194, 2015.

[31] R. Canetti, O. Goldreich, S. Goldwasser, S. Micali. Resettable zero-knowledge (extended abstract). *STOC 2000*, pp.235–244, 2000.

[32] R. Canetti, S. Halevi, J. Katz. Chosen-Ciphertext Security from Identity-Based Encryption. *EUROCRYPT 2004*, LNCS 3027, pp.207–222, 2004.

[33] R. Canetti, S. Hohenberger. Chosen-ciphertext secure proxy re-encryption. *ACMCCS 2007*, pp.185–194, 2007.

[34] R. Canetti, H. Krawczyk, J.B. Nielsen. Relaxing Chosen-Ciphertext Security. *CRYPTO 2003*, LNCS 2729, pp.565-582, 2003.

[35] R. Canetti, H. Lin, S. Tessaro, V. Vaikuntanathan. Obfuscation of Probabilistic Circuits and Applications. *TCC 2015(2)*, LNCS9015, pp.468–497, 2015.

[36] R. Canetti, G.N. Rothblum, M. Varia. Obfuscation of Hyperplane Membership. *TCC 2010*, LNCS5978, pp.72–89, 2010.

[37] S. Chow, J. Weng, Y. Yang, R. Deng. Efficient Unidirectional Proxy Re-Encryption. *AFRICACRYPT 2010*, LNCS 6055, pp.316–332, 2010.

[38] R. Cramer, G. Hanaoka, D. Hofheinz, H. Imai, E. Kiltz, R. Pass, A. Shelat, V. Vaikuntanathan. Bounded CCA2-Secure Encryption. *ASIACRYPT 2007*, LNCS4833, pp.502–518, 2007.

[39] CRYPTREC: Cryptography Research and Evaluation Committees. http://www.cryptrec.go.jp/index.html

[40] Y. Desmedt, Y. Frankel. Threshold cryptosystems. *CRYPTO 1989*, LNCS 435, pp.307–315, 1989.

[41] G.R. Doddington, W. Liggett, A.F. Martin, M.A. Przybocki, D.A. Reynolds. SHEEP, GOATS, LAMBS and WOLVES: a statistical analysis of speaker performance in the NIST 1998 speaker recognition evaluation. *ICSLP 1998*, 1998.

[42] Y. Dodis, J. Katz. Chosen-Ciphertext Security of Multiple Encryption. *TCC 2005*, LNCS 3378, pp.188–209, 2005.

[43] Y. Dodis, J. Katz, S. Xu, M. Yung. Key-Insulated Public Key Cryptosystems. *EUROCRYPT 2002*, LNCS2332, pp.65–82, 2002.

[44] D. Dolev, C. Dwork, M. Naor. Non-malleable Cryptography. *SIAM Journal on Computing*, 30(2), pp.391–437, 2000.

[45] C. Dwork, M. Naor, O. Reingold, L.J. Stockmeyer. Magic Functions. *J. ACM*, 50(6), pp.852–921, 2003.

[46] E. Fujisaki, T. Okamoto. Secure Integration of Asymmetric and Symmetric Encryption Schemes. *CRYPTO 1999*, LNCS 1666, pp.538–554, 1999.

[47] Y. Gan, L. Wang, L. Wang, P. Pan, Y. Yang. Efficient Construction of CCA-Secure Threshold PKE Based on Hashed Diffie-Hellman Assumption. *Comput. J.*, Volume 56, Number 10, pp.1249–1257, 2013.

[48] S. Garg, C. Gentry, S. Halevi. Candidate Multilinear Maps from Ideal Lattices. *EUROCRYPT 2013*, LNCS7881, pp.1–17, 2013.

[49] S. Garg, C. Gentry, S. Halevi, M. Raykova. Two-Round Secure MPC from Indistinguishability Obfuscation. *TCC 2014*, LNCS8349, pp.74–94, 2014.

[50] S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai, B. Waters. Candidate Indistinguishability Obfuscation and Functional Encryption for all Circuits. *FOCS 2013*, pp.40–49, 2013.

[51] R. Geambasu, T. Kohno, A.A. Levy, H.M. Levy. Vanish: Increasing Data Privacy with Self-Destructing Data. *Usenix Security 2009*, pp.299–316, 2009.

[52] C. Gentry. Fully homomorphic encryption using ideal lattices. *STOC 2009*, pp.169–178, 2009.

[53] O. Goldreich, L.A. Levin. A Hard-Core Predicate for all One-Way Functions. In *STOC 1989*, pp.25–32, 1989.

[54] S. Goldwasser, S. Micali. Probabilistic Encryption and How to Play Mental Poker Keeping Secret All Partial Information. *STOC 1982*, pp.365–377, 1982.

[55] S. Gorbunov, V. Vaikuntanathan, H. Wee. Attribute-based encryption for circuits. *STOC 2013*, pp.545–554, 2013.

[56] J. Groth, A. Sahai. Efficient Non-interactive Proof Systems for Bilinear Groups. *EUROCRYPT 2008*, LNCS 4965, pp.415–432, 2008.

[57] S. Hada. Secure Obfuscation for Encrypted Signatures. *EUROCRYPT 2010*, LNCS6110, pp.92–112, 2010.

[58] G. Hanaoka, Y. Kawai, N. Kunihiro, T. Matsuda, J. Weng, R. Zhang, Y. Zhao. Generic Construction of Chosen Ciphertext Secure Proxy Re-Encryption. *CT-RSA 2012*, LNCS 7178, pp.349–364, 2012.

[59] A. Hang, A.D. Luca, M. Richter, M. Smith, H. Hussmann. Where Have You Been? Using Location-Based Security Questions for Fallback Authentication. *SOUPS 2015*, 2015.

[60] A. Herzberg, S. Jarecki, H. Krawczyk, M. Yung. Proactive Secret Sharing Or: How to Cope With Perpetual Leakage. *CRYPTO 1995*, LNCS 963, pp.335–352, 1995.

[61] S. Hohenberger, V. Koppula, B. Waters. Universal Signature Aggregators. *EUROCRYPT 2015(1)*, LNCS 9057, pp.3-34, 2015.

[62] S. Hohenberger, A.B. Lewko, B. Waters. Detecting Dangerous Queries: A New Approach for Chosen Ciphertext Security. *EUROCRYPT 2012*, LNCS 7237, pp.663-681, 2012.

[63] S. Hohenberger, G.N. Rothblum, A. Shelat, V. Vaikuntanathan. Securely Obfuscating Re-Encryption. *J. Cryptology 24(4)*, pp.694–719, 2011.

[64] S. Hohenberger, A. Sahai, B. Waters. Replacing a Random Oracle: Full Domain Hash from Indistinguishability Obfuscation. *EUROCRYPT 2014*, LNCS8441, pp.201–220, 2014.

[65] T. Isshiki, M.H. Nguyen, K. Tanaka. Proxy Re-Encryption in a Stronger Security Model Extended from CT-RSA2012. *CT-RSA 2013*, LNCS 7779, pp.277–292, 2013.

[66] A.A. Ivan, Y. Dodis. Proxy Cryptography Revisited. *NDSS 2003*, 2003.

[67] M. Jakobsson, E. Stolterman, S. Wetzel, L. Yang. Love and Authentication. *CHI 2008*, pp.197–200, 2008.

[68] A.K. Jain, A. Ross, S. Prabhakar. An Introduction to Biometric Recognition. *IEEE Transactions on Circuits and Systems for Video Technology*, Volume 14, No.1, pp.4–20, 2004.

[69] A. Javed, D. Bletgen, F. Kohlar, M. Dürmuth, J. Schwenk. Secure Fallback Authentication and the Trusted Friend Attack. *ICDCSW 2014*, pp.22–28, 2014.

[70] M. Just, D. Aspinall. Personal Choice and Challenge Questions: A Security and Usability Assessment. *SOUPS 2008*, 2008.

[71] J. Katz, V. Vaikuntanathan. Round-Optimal Password-Based Authenticated Key Exchange. *J. Cryptology*, Volume 26, No.4, pp.714–743, 2013.

[72] E. Kiltz. Chosen-Ciphertext Security from Tag-Based Encryption. *TCC 2006*, LNCS 3876, pp.581–600, 2006.

[73] E. Kiltz. Chosen-Ciphertext Secure Key-Encapsulation Based on Gap Hashed Diffie-Hellman. *PKC 2007*, LNCS 4450, pp.282–297, 2007.

[74] E. Kirshanova. Proxy Re-encryption from Lattices. *PKC 2014*, LNCS 8383, pp.77–94, 2014.

[75] K. Kurosawa, K. Heng. From Digital Signature to ID-based Identification/Signature. *PKC 2004*, LNCS2947, po. 248–261, 2004.

[76] J. Lai, R.H. Deng, S. Liu, W. Kou. Efficient CCA-Secure PKE from Identity-Based Techniques. *CT-RSA 2010*, LNCS 5985, pp.132–147, 2010.

[77] B. Libert, D. Vergnaud. Unidirectional Chosen-Ciphertext Secure Proxy Re-Encryption. *PKC 2008*, LNCS 4939, pp.360–379, 2008.

[78] B. Libert, M. Yung. Adaptively Secure Non-interactive Threshold Cryptosystems. *ICALP 2011*, LNCS 6756, pp.588–600, 2011.

[79] B. Libert, M. Yung. Non-interactive CCA-Secure Threshold Cryptosystems with Adaptive Security: New Framework and Constructions. *TCC 2012*, LNCS 7194, pp.75–93, 2011.

[80] D. Ma, G. Tsudik. A New Approach to Secure Logging. *DBSEC 2008*, LNCS5094, pp.48–63, 2008.

[81] M. Mambo, E. Okamoto. Proxy Cryptosystems: Delegation of the Power to Decrypt Ciphertexts. *IEICE Trans on Fundamentals of Electronics, Communications and Computer Sciences*, E80-A(1), pp.54–63, 1997.

[82] T. Matsuda, R. Nishimaki, K. Tanaka. CCA Proxy Re-Encryption without Bilinear Maps in the Standard Model. *PKC 2010*, LNCS 6056, pp.261–278, 2010.

[83] M. Naor, M. Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. *STOC 1990*, pp.427–437, 1990.

[84] S. Ohata, Y. Kawai, T. Matsuda, G. Hanaoka, K. Matsuura. Re-encryption Verifiability: How to Detect Malicious Activities of a Proxy in Proxy Re-encryption. *CT-RSA 2015*, LNCS 9048, pp.410–428, 2015.

[85] S. Ohata, T. Matsuda, G. Hanaoka, K. Matsuura. More Constructions of Re-splittable Threshold Public Key Encryption. *IWSEC 2014*, LNCS 8639, pp.109-118, 2014.

[86] P. Paillier. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. *EUROCRYPT 1999*, LNCS 1592, pp.223–238, 1999.

[87] A. Pinto, B. Poettering, J.C.N. Schuldt. Multi-recipient Encryption, Revisited. *AsiaCCS 2014*, pp.229–238, 2014.

[88] C. Rackoff, D. Simon. Non-Interactive Zero-Knowledge Proof of Knowledge and Chosen Ciphertext Attack. *CRYPTO 1991*, LNCS 576, pp.433–444, 1991.

[89] T. Ristenpart, S. Yilek. The Power of Proofs-of-Possession: Securing Multiparty Signatures against Rogue-Key Attacks. *EUROCRYPT 2007*, LNCS 4515, pp.228–245, 2007.

[90] S. Sano, T. Otsuka, K. Itoyama, H.G. Okuno. HMM-based Attacks on Google's ReCAPTCHA with Continuous Visual and Audio Symbols. *JIP*, No.23, Vol.6, pp.814–826, 2015.

[91] J. Shao, Z. Cao. CCA-Secure Proxy Re-Encryption without Pairings. *PKC 2009*, LNCS 5443, pp.357–376, 2009.

[92] J. Weng, Y. Zhao, G. Hanaoka. On the Security of a Bidirectional Proxy Re-Encryption Scheme from PKC 2010. *PKC 2011*, LNCS 6571, pp.284–295, 2011.

[93] X. Zhang, M. Chen, X. Li. Comments on Shao-Cao's Unidirectional Proxy Re-Encryption Scheme from PKC 2009. *Journal of Information Science and Engineering. 27(3)*, pp.1153–1158, 2011.

[94] A. Rabkin. Personal Knowledge Questions for Fallback Authentication: Security Questions in the Era of Facebook. *SOUPS 2008*, 2008.

[95] K. Ramchen, B. Waters. Fully Secure and Fast Signing from Obfuscation. *CCS 2014*, pp.659–673, 2014.

[96] R.W. Reeder, S. Schechter. When the Password Doesn't Work: Secondary Authentication for Websites. *IEEE Security and Privacy*, Volume 9, No.2, pp.43–49, 2011.

[97] A. Sahai, B. Waters. Fuzzy Identity-Based Encryption. *EUROCRYPT 2005*, LNCS 3494, pp.457–473, 2005.

[98] A. Sahai, B. Waters. How to use indistinguishability obfuscation: deniable encryption, and more. *STOC 2014*, pp.475–484, 2014.

[99] Y. Sakai, K. Emura, J.C.N. Schuldt, G. Hanaoka, and K. Ohta Dynamic Threshold Public-Key Encryption with Decryption Consistency from Static Assumptions. *ACISP 2015*, LNCS 9144, pp.77–92, 2015.

[100] S.E. Schechter, A.J.B. Brush, S. Egelman. It's No Secret. Measuring the Security and Reliability of Authentication via "Secret" Questions. *IEEE Symposium on Security and Privacy 2009*, pp.375–390, 2009.

[101] S. Schechter, R.W. Reeder. 1+1=You: Measuring the Comprehensibility of Metaphors for Configuring Backup Authentication. *SOUPS 2009*, 2009.

[102] A. Shamir. Identity-Based Cryptosystems and Signature Schemes. *CRYPTO 1984*, LNCS196, pp.47–53, 1984.

[103] V. Shoup. NTL: A Library for doing Number Theory, http://www.shoup.net/ntl/

[104] V. Shoup, R. Gennaro. Securing Threshold Cryptosystems against Chosen Ciphertext Attack. *EUROCRYPT 1998*, LNCS 1403, pp.1–16, 1998.

[105] L. Standing, J. Conezio, R.N. Haber. Perception and Memory for Pictures: Single-trial Learning of 2500 Visual Stimuli. *Psychonomic Science*, Volume 19, Issue.2, pp.73–74, 1970.

[106] B. Waters. A Punctured Programming Approach to Adaptively Secure Functional Encryption. *IACR Cryptology ePrint Archive*, 2014:588, 2014.

[107] H. Wee. On obfuscating point functions. *STOC 2005*, pp.523–532, 2005.

[108] H. Wee. Threshold and Revocation Cryptosystems via Extractable Hash Proofs. *EUROCRYPT 2011*, LNCS 6632, pp.589–609, 2011.

[109] H. Wee. Public Key Encryption against Related Key Attacks. *PKC 2012*, LNCS 7293, pp.262–279, 2012.

[110] T. Yamakawa, S. Yamada, G. Hanaoka, N. Kunihiro. Self-bilinear Map on Unknown Order Groups from Indistinguishability Obfuscation and Its Applications. *CRYPTO 2014(2)*, LNCS8617, pp.90–107, 2014.

[111] A.C. Yao. Theory and Applications of Trapdoor Functions (Extended Abstract). *FOCS 1982*, pp.80–91, 1982.

[112] A.C. Yao. How to Generate and Exchange Secrets (Extended Abstract). *FOCS 1986*, pp.162–167, 1986.

[113] M. Zviran, W.J. Haga. User Authentication by Cognitive Passwords: An Empirical Assessment. *JCIT 1990*, pp.137–144, 1990.

Appendix A

List of Publications

Refereed Papers

1. Satsuya Ohata, Kanta Matsuura. Proxy Re-encryption via Indistinguishability Obfuscation. *Security and Communication Networks*, (to appear).

2. Satsuya Ohata, Yutaka Kawai, Takahiro Matsuda, Goichiro Hanaoka, Kanta Matsuura. Re-Encryption Verifiability: How to Detect Malicious Activities of a Proxy in Proxy Re-Encryption. *CT-RSA 2015*, LNCS 9048, pp.408–422, 2015.

3. Goichiro Hanaoka, Satsuya Ohata, Takahiro Matsuda, Koji Nuida, Attrapadung Nattapong. Methodology for designing cryptographic systems with advanced functionality based on a modular approach, -Towards reducing the barrier to introducing newly-designed cryptographic schemes into real world systems-. Synthesiology, vol.7, No.2, pp.93–104, 2014. (The result of this article is not included in this thesis.)

4. Satsuya Ohata, Takahiro Matsuda, Goichiro Hanaoka, Kanta Matsuura. More Constructions of Re-splittable Threshold Public Key Encryption. *IWSEC 2014*, LNCS 8639, pp.99–108, 2014.

Non-refereed Papers

1. 大畑幸矢, 松田隆宏, 松浦幹太. パスワード再発行プロトコルの安全性について. *2016* 年 暗号と情報セキュリティシンポジウム *(SCIS2016)*, 2016.

2. 石坂理人, 大畑幸矢, 松浦幹太. 適応的述語安全な暗号文ポリシー型属性ベース Signcryption の一般的構成. *2016* 年 暗号と情報セキュリティシンポジウム *(SCIS2016)*, 2016. (The result of this article is not included in this thesis.)

3. 大畑幸矢, 松田隆宏, 松浦幹太. 証明可能安全なパスワード再発行プロトコル・改. コンピュータセキュリティシンポジウム *2015 (CSS2015)*, 2015.

4. 大畑幸矢, 松浦幹太. 識別不可性難読化に基づく復号の速い代理暗号化について. *2015* 年 暗号と情報セキュリティシンポジウム *(SCIS2015)*, 2015.

5. 大畑幸矢, 松田隆宏, 松浦幹太. 証明可能安全なパスワード再発行プロトコルについて. コンピュータセキュリティシンポジウム *2014 (CSS2014)*, 2014.

6. 大畑幸矢, 松田隆宏, 花岡悟一郎, 松浦幹太. 閾値公開鍵暗号の鍵再分割可能性について. *2014* 年暗号と情報セキュリティシンポジウム *(SCIS2014)*, 2014.

7. 大畑幸矢, 松田隆宏, 花岡悟一郎, 松浦幹太. 検証可能代理人再暗号化方式の安全性について. *2013* 年 暗号と情報セキュリティシンポジウム *(SCIS2013)*, 2013.

Posters and Preprint Paper

1. Satsuya Ohata, Takahiro Matsuda, Kanta Matsuura. On Rigorous Security of Password Recovery Protocols. *IWSEC2015* , Poster Session, 2015.

2. Satsuya Ohata, Yutaka Kawai, Takahiro Matsuda, Goichiro Hanaoka, Kanta Matsuura. Re-encryption Verifiability: How to Detect Malicious Activities of a Proxy in Proxy Re-encryption. *IACR Cryptology ePrint Archive*, 2015/112, 2015.

3. Satsuya Ohata, Toshiyuki Miyazawa, Tetsutaro Kobayashi, Kanta Matsuura. Design and Development of Multi-purpose Cryptographic Platform toward Wider Deployment of Pairing-Based Cryptography. *IWSEC2011* , Poster Session, 2011. (The result of this poster is not included in this thesis.)

Awards

1. コンピュータセキュリティシンポジウム 2015 (CSS2015) 学生論文賞（2015 年 10 月）

2. 2014 年暗号と情報セキュリティシンポジウム (SCIS2014) SCIS 論文賞（2014 年 5 月）

3. 東京大学大学院情報理工学系研究科 研究科長賞（2013 年 3 月）