Re-encryption Verifiability: How to Detect Malicious Activities of a Proxy in Proxy Re-encryption*

Satsuya Ohata[1,3], Yutaka Kawai[2], Takahiro Matsuda[3], Goichiro Hanaoka[3], and Kanta Matsuura[1]

[1] The University of Tokyo, Tokyo, Japan {satsuya,kanta}@iis.u-tokyo.ac.jp
[2] Mitsubishi Electric, Kanagawa, Japan Kawai.Yutaka@da.MitsubishiElectric.co.jp
[3] National Institute of Advanced Industrial Science and Technology, Ibaraki, Japan
{t-matsuda,hanaoka-goichiro}@aist.go.jp

Abstract. In this paper, we introduce a new functionality for proxy re-encryption (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 illegal activities of the proxy. We formalize the security model for a VPRE scheme, and show that the single-hop uni-directional PRE scheme by Hanaoka et al. (CT-RSA 2012) can be extended to a secure VPRE scheme.

Keywords: Proxy Re-encryption, Re-encryption Verifiability, Soundness.

1 Introduction

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. [8] and has received much attention in recent years. There are many models as well as implementations [8, 3, 10, 20, 28, 12, 15, 17]. The type of PRE we focus on in this paper is "single-hop" and "uni-directional", where a ciphertext[4] 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 to transform for the opposite direction.

In ordinary PRE schemes, a proxy is modeled 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 paper, we study a mechanism that enables us to reduce the level of trust 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 $rk_{A\to B}$ to the cloud manager, and can go off-line; when later B requests the data for the cloud manager, he/she 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?* However, 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

* An extended abstract of this paper appears in the proceedings of CT-RSA 2015

[4] 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 [20, 15], and we will also use the names.

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.

1.1 Our Contribution

In this paper, 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. [15], 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 [15]. Then, we show that the PRE scheme by Hanaoka et al. [15] (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 [14] 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.

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 paper (i.e. verifiability of a dishonest behavior of a proxy) are two very different properties, and one approach cannot be a solution for 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. [15]. 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 [11]), the security models in [15] do not assume the so-called *knowledge-of-secret-key* (KOSK) assumption [9], 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 [4, 25], multi-signature [9, 7, 27]). The KOSK assumption does not reflect the reality quite well, and there are several critiques on this assumption (e.g. in the discussions in [7, 27, 25]). 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. [20, 17]). [5] 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 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. [20, 17]) 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.

1.2 Related Work

We briefly review the related work. Mambo and Okamoto introduced the concept of proxy decryption [21]. Later, Ivan and Dodis [18] proposed a generic construction of proxy cryptography based on sequential multiple encryption. Blaze, Bleumer and Strauss formulated the concept of PRE cryptosystems [8] and proposed the first bidirectional PRE scheme based on the ElGamal PKE scheme. Subsequently, Ateniese et al. [3], Canetti and Hohenberger [10], Libert and Vergnaud [20], and Chow et al. [12], proposed different PRE schemes with various properties. Shao and Cao [28] proposed a PRE scheme without pairings. Later, however, Zhang et al. [30] pointed out that it is not secure in the Libert-Vergnaud security model [20]; that is, it does not provide master key security. Subsequently, Matsuda et al. proposed a PRE scheme without pairings [22], but later, Weng, Zhao, and Hanaoka [29] pointed out that their scheme is not chosen-ciphertext secure. Hanaoka et al. [15] proposed a new definition of CCA security in PRE and showed a generic construction of uni-directional PRE. Isshiki et al. [17] proposed a CCA secure PRE scheme.[6] Kirshanova [19] proposed a lattice-based PRE scheme. To the best of our knowledge, none of the previous works considered the re-encryption verifiability.

[5] 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.

[6] Although it is claimed that their security model is stronger than that of [15], they are actually incomparable. The security model for a transformed ciphertext (first-level ciphertext) in [17] allows an adversary a slightly more flexible challenge query than that of [15]. However, all public keys in the security models of [17] 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 [15].

2 Preliminaries

Basic Notation. $\mathbf{N}$ denotes the set of all natural numbers, and for $n \in \mathbf{N}$, 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 $y \leftarrow \mathcal{A}(x; r)$ denotes that $\mathcal{A}$ computes y as output by taking x as input and using r as randomness. k denotes the security parameter. A function $f(k) : \mathbf{N} \to [0, 1]$ is said to be *negligible* if for all positive polynomials p and all sufficiently large $k \in \mathbf{N}$, we have $f(k) < 1/p(k)$.

2.1 Public Key Encryption

A public key encryption scheme (PKE) consists of the following three algorithms $(\mathsf{PKG}, \mathsf{PEnc}, \mathsf{PDec})$.

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 Ciphertext Security [23, 26, 13]. 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 $\mathsf{Adv}^{\mathtt{CCA\text{-}PKE}}_{\mathcal{A}}(k) = |\Pr[b' = b] - 1/2|$. We say a PKE scheme is *CCA secure*, if for any PPT adversary $\mathcal{A}$, $\mathsf{Adv}^{\mathtt{CCA\text{-}PKE}}_{\mathcal{A}}(k)$ is negligible.

Chosen Ciphertext Security in the Multi-user Setting [5]. 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 [5]. 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 $(pk_1, \ldots, pk_n)$ 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 $\mathsf{Adv}^{\mathtt{CCA\text{-}PKE}}_{(\mathcal{A},n)}(k) = |\Pr[b' = b] - 1/2|$.

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 $\mathcal{A}$, $\mathsf{Adv}^{\mathtt{CCA\text{-}PKE}}_{(\mathcal{A},n)}(k)$ is negligible.

Bellare, Boldyreva, and Micali [5] showed that ordinary CCA security and CCA security in the multi-user setting are polynomially equivalent.

2.2 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 $\bot$ (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 [1]. We recall the definition of strong unforgeability [1] 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 $\mathsf{Adv}^{\mathtt{SUF\text{-}SIG}}_{\mathcal{A}}(k) = \Pr[\mathcal{A} \text{ wins}]$.

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

2.3 Re-splittable Threshold Public Key Encryption

A re-splittable threshold public key encryption (TPKE) scheme was introduced in [15]. 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. The first re-splittable TPKE scheme was proposed by [15]. Recently, Ohata et al. [24] proposed three more schemes. (All schemes so far are based on bilinear maps.)

Formally, 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 secret key tsk and a public key tpk.

TEnc This is the encryption algorithm that takes tpk and a plaintext m as input, outputs a ciphertext c.

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.

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

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 $\bot$. When the output is $\top$ (resp. $\bot$), we say that μ is a valid (resp. invalid) decryption share of the ciphertext c.

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 $\bot$).

For any $k \in \mathbf{N}$, any polynomials t, n such that $0 < t \leq n$, any $(tsk, tpk) \leftarrow \mathsf{TKG}(1^k, n, t)$ and any $(tsk_1, \cdots, tsk_n, tvk) \leftarrow \mathsf{TSplit}(tsk)$, we require the following two correctness properties: (1) For any ciphertext c and index $i \in [n]$, if $\mu = \mathsf{TShDec}(tpk, tsk_i, c)$, then we have $\mathsf{TShVer}(tpk, tvk, c, i, \mu) = \top$. (2) For any plaintext 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 $\mathsf{TCom}(tpk, tvk, c, S) = m$.

Chosen Ciphertext Security. CCA security of a re-splittable TPKE scheme is defined using the following game which is parameterized by two integers t, n with $0 \leq t \leq n$ and is played by the challenger and an adversary $\mathcal{A}$: The challenger first runs $(tsk, tpk) \leftarrow \mathsf{TKG}(1^k, n, t)$ 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 $S = \{s_1, \ldots, s_{t-1}\}$, the challenger runs $(tsk_1, \ldots, tsk_n, tvk) \leftarrow \mathsf{TSplit}(tsk)$ and returns $(tsk_{s_1}, \ldots, tsk_{s_{t-1}}, tvk)$ 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 $\mathsf{Adv}^{\mathtt{CCA\text{-}TPKE}}_{(\mathcal{A},n,t)}(k) = |\Pr[b = b'] - 1/2|$. 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 $0 < t \leq n$, $\mathsf{Adv}^{\mathtt{CCA\text{-}TPKE}}_{(\mathcal{A},n,t)}(k)$ is negligible.

Decryption Consistency. Decryption consistency is defined using the game which is defined in the same way as the CCA game, without the challenge query. 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 if (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, $\mathsf{TShVer}(tpk, tvk, c, i, \mu_{s_i}) = \mathsf{TShVer}(tpk, tvk, c, i, \mu'_{s'_i}) = \top$ 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 $\mathsf{Adv}^{\mathtt{DC\text{-}TPKE}}_{(\mathcal{A},n,t)}(k)$ denote the probability of $\mathcal{A}$ winning in this game. We say that a TPKE scheme has *decryption consistency*, if for any PPT adversary $\mathcal{A}$ and for any polynomials t and n such that $0 < t \leq n$, $\mathsf{Adv}^{\mathtt{DC\text{-}TPKE}}_{(\mathcal{A},n,t)}(k)$ is negligible.

Strong Smoothness. In this paper, we will use the property which we call *strong smoothness*. This is introduced under the name of *γ-uniformity* in [14] for ordinary PKE, and is a stronger version of *smoothness* used in [6]. (We borrow the name from [6] because we believe it describes the property more directly.)

Formally, we say that a re-splittable TPKE scheme has *strong smoothness* if the following quantity

$$\mathsf{Smth}(k) = \max_{\substack{c,m,R,\\ (tpk,tsk)\leftarrow\mathsf{TKG}(1^k;R),}} \Pr_{c'\leftarrow\mathsf{TEnc}(tpk,m)}[c = c']$$

is negligible. Here, R is a randomness used by TKG.

We note that strong smoothness is satisfied if a ciphertext contains an unpredictable component (such as a random group element g^r with a generator g of a cyclic group and a randomness r), and we are not aware of any natural construction of (re-splittable) TPKE based on bilinear maps that does not have strong smoothness. For example, the re-splittable TPKE schemes proposed in [15, 24] have this property unconditionally.

3 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 3.1, we define the syntax of a VPRE scheme. In Section 3.2, based on the definitions given in [15] 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 [15]. Finally, in Section 3.3, we show that a VPRE secure in our definitions is also secure (as an ordinary PRE scheme) in the definitions of [15], and thus our definitions have "backward compatibility" with [15].

3.1 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 1. We remark that as in [15], 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) $\mathsf{REncVer}(\mathsf{pk}_i, \mathsf{sk}_j, c_i, \widehat{c}_j) = \top$.

3.2 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: $\mathsf{Dec}_1(\mathsf{sk}_j, \widehat{c}^*) \neq \perp$.

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

Definition 1 (Soundness of Re-encryption). *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 $\perp$ 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 $\mathsf{REncVer}(\mathsf{pk}_{i^*}, \mathsf{sk}_j, c^*, \widehat{c}) = \top$, then the challenger returns $\perp$ 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 $\mathsf{Adv}^{\texttt{second-VPRE}}_{(\mathcal{A},n)}(k) = |\Pr[b = b'] - 1/2|$.

Definition 2 (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, $\mathsf{Adv}^{\texttt{second-VPRE}}_{(\mathcal{A},n)}(k)$ 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[7], 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 $\mathsf{Adv}^{\texttt{first-VPRE}}_{\mathcal{A}}(k) = |\Pr[b = b'] - 1/2|$.

Definition 3 (First-Level CCA Security). *We say that a VPRE scheme is first-level CCA secure, if for any PPT adversary $\mathcal{A}$, $\mathsf{Adv}^{\texttt{first-VPRE}}_{\mathcal{A}}(k)$ is negligible.*

Difference with the Definitions (for PRE) in [15]. 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 [15, Definition 2].

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 [15, Definition 1] is as follows (where we *emphasize* the difference).

First-level decryption query (of [15]) : 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 $\perp$.)

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

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 take in [15] to avoid such "self-broken" definition considered in [15]. 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 replayable CCA security* of [11] and *detectable PKE* of [16], and we believe these connections to be interesting.

3.3 Implications to the Definitions of [15]

Here, we show a "backward compatibility" of our security definitions. Namely, we show that if a VPRE scheme satisfies security definitions given in Section 3.2, then it is also secure as a (V)PRE under the definitions of [15].

Theorem 1. *If a VPRE scheme is first-level CCA secure in the sense of Definition 3, then the VPRE scheme is first-level CCA secure in the sense of [15, Definition 2].*

This is obvious from the definition. In particular, an adversary in our first-level CCA security definition is only more powerful than that of [15, Definition 2] (our adversary can make $\mathsf{REncVer}$ queries which are not considered in [15]).

Theorem 2. *If a VPRE scheme is second-level CCA secure in the sense of Definition 2 and satisfies soundness (Definition 1), then the VPRE scheme is second-level CCA secure in the sense of [15, Definition 1].*

Proof of Theorem 2 Let $n > 0$ be a polynomial, and $\mathcal{A}$ be any PPT adversary that attacks a VPRE scheme in the sense of [15, Definition 1] and makes $Q > 0$ $\mathsf{Dec_1}$ queries. (Since $\mathcal{A}$ is PPT, Q is polynomial.) Consider the following games.

Game 0. The second-level CCA game of [15, Definition 1].

Game 1. Same as Game 0, except that if $\mathcal{A}$ submits a $\mathsf{Dec_1}$ 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 $\mathsf{Dec_1}$ 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 [15, Definition 1]) is calculated as follows:

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

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 $|\Pr[\mathsf{Succ}_0] - \Pr[\mathsf{Succ}_1]| \leq \Pr[\mathsf{Bad}_0] = \Pr[\mathsf{Bad}_1]$. Then, we show that we can construct a soundness adversary $\mathcal{B}$ such that $\mathsf{Adv}^{\mathtt{SND\text{-}VPRE}}_{(\mathcal{B},n)}(k) \geq (1/Q) \cdot \Pr[\mathsf{Bad}_1]$, 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 2, 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 2). In particular, a second-level CCA adversary (in the sense of Definition 2) can simulate Game 1 perfectly for $\mathcal{A}$, and such adversary has advantage exactly $|\Pr[\mathsf{Succ}_1] - 1/2|$.

This completes the proof of Theorem 2. □

4 A Concrete VPRE Scheme

In this section, we show a concrete VPRE scheme, and prove its security. Specifically, our VPRE scheme is a simple extension of the PRE scheme by Hanaoka et al. (which we denote by HKK^+) [15], 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 [15] 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 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. 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 $\widehat{M} = \bot$ then return $\bot$.
$\langle \mathsf{pk}'_i \| \mathsf{pk}'_j \| c_i \| \mu_2 \| \psi \| tvk_i \| \sigma \rangle \leftarrow \widehat{M}$
If $\mathsf{pk}'_j \neq \mathsf{pk}_j$ then return $\bot$.
$(tpk_i, \widehat{pk}_i, pk_i, vk_i) \leftarrow \mathsf{pk}'_i$
If $\mathsf{SVer}(vk_i, \langle \psi \| tvk_i \| \mathsf{pk}'_i \| \mathsf{pk}'_j \rangle, \sigma) = \bot$ then return $\bot$.
$tsk_{i.1} \leftarrow \mathsf{PDec}(dk_j, \psi)$
If $tsk_{i.1} = \bot$ then return $\bot$.
$\mu_1 \leftarrow \mathsf{TShDec}(tpk_i, tsk_{i.1}, c_i)$
If $\mu_1 = \bot$ then return $\bot$.
If $\mathsf{TShVer}(tpk_i, tvk_i, c_i, 2, \mu_2) = \bot$ then return $\bot$.
$m \leftarrow \mathsf{TCom}(tpk_i, tvk_i, c_i, \{\mu_1, \mu_2\})$
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 $\mathsf{SVer}(vk_i, \langle \psi \| tvk_i \| \mathsf{pk}_i \| \mathsf{pk}_j \rangle, \sigma) = \bot$ then return $\bot$.
$(tpk_j, \widehat{pk}_j, pk_j, vk_j) \leftarrow \mathsf{pk}_j$
$\mu_2 \leftarrow \mathsf{TShDec}(tpk_i, tsk_{i.2}, c_i)$
If $\mu_2 = \bot$ then return $\bot$.
$\widehat{M} \leftarrow \langle \mathsf{pk}_i \| \mathsf{pk}_j \| c_i \| \mu_2 \| \psi \| tvk_i \| \sigma \rangle$
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 $\mu_1 = \bot$ then return $\bot$.
$\mu_2 \leftarrow \mathsf{TShDec}(tpk_i, tsk_{i.2}, c)$
If $\mu_2 = \bot$ then return $\bot$.
$m \leftarrow \mathsf{TCom}(tpk_i, tvk_i, c, \{\mu_1, \mu_2\})$
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 $\widehat{M} = \bot$ then return $\bot$.
$\langle \mathsf{pk}'_i \| \mathsf{pk}'_j \| c_i \| \mu_2 \| \psi \| tvk_i \| \sigma \rangle \leftarrow \widehat{M}$
If $(\mathsf{pk}'_i, \mathsf{pk}'_j) \neq (\mathsf{pk}_i, \mathsf{pk}_j)$ then return $\bot$.
If $\mathsf{SVer}(vk_i, \langle \psi \| tvk_i \| \mathsf{pk}'_i \| \mathsf{pk}'_j \rangle, \sigma) = \bot$ then return $\bot$.
$tsk_{i.1} \leftarrow \mathsf{PDec}(dk_j, \psi)$
If $tsk_{i.1} = \bot$ then return $\bot$.
$\mu_1 \leftarrow \mathsf{TShDec}(tpk_i, tsk_{i.1}, c_i)$
If $\mu_1 = \bot$ then return $\bot$.
If $\mathsf{TShVer}(tpk_i, tvk_i, c_i, 2, \mu_2) = \bot$ then return $\bot$.
If $c'_i = c_i$ then return $\top$ else return $\bot$.

Fig. 1. The VPRE scheme eHKK^+ based on the PRE scheme by Hanaoka et al. [15]. 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 [15].

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

Theorem 3. *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 Appendix A. Here, we show the 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 4. *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 Appendix B. Here, we show the intuition of the proof of second-level CCA security. The proof follows closely to the above proof of soundness, and the original security proof of the HKK^+ scheme [15]. 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 5. *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 Appendix C. Here, we show the intuition of the proof of first-level CCA security. As shown in [15], 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.

The main difference from the proof in [15] is that we need the strong smoothness of the underlying re-splittable TPKE scheme, which was not necessary in the original proof of [15], 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 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 $\mathcal{A}$.

References

1. J.H. An, Y. Dodis, and T. Rabin. On the Security of Joint Signature and Encryption. *EUROCRYPT 2002*, LNCS 2332, pp. 83–107, 2002.
2. S. Arita and K. Tsurudome. Construction of Threshold Public-Key Encryptions through Tag-Based Encryptions. *ACNS 2009*, LNCS 5536, pp. 186–200, 2009.
3. G. Ateniese, K. Fu, M. Green, and S. Hohenberger. Improved proxy re-encryption schemes with applications to secure distributed storage. *ACM Trans. Inf. Syst. Secur. 9(1)*, pp. 1–30, 2006.
4. M. Bellare, A. Boldyreva, K. Kurosawa and 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.
5. M. Bellare, A. Boldyreva, and S. Micali. Public Key Encryption in a Multi-user Setting: Security Proofs and Improvements. *EUROCRYPT 2000*, LNCS 1807, pp. 259–274, 2000.
6. M. Bellare, D. Hofheinz, and 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.
7. M. Bellare, and G. Neven. Multi-Signatures in the Plain Public-Key Model and a General Forking Lemma. *ACMCCS 2006*, pp. 390–399, 2006.
8. M. Blaze, G. Bleumer, and M. Strauss. Divertible Protocols and Atomic Proxy Cryptography. *EUROCRYPT 1998*, LNCS 1403, pp. 127–144, 1998.
9. 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.
10. R. Canetti and S. Hohenberger. Chosen-ciphertext secure proxy re-encryption. *ACMCCS 2007*, pp. 185–194, 2007.
11. R. Canetti, H. Krawczyk, and J.B. Nielsen. Relaxing Chosen-Ciphertext Security. *CRYPTO 2003*, LNCS 2729, pp. 565-582, 2003.
12. S. Chow, J. Weng, Y. Yang, and R. Deng. Efficient Unidirectional Proxy Re-Encryption. *AFRICACRYPT 2010*, LNCS 6055, pp. 316–332, 2010.
13. D. Dolev, C. Dwork, and M. Naor. Non-malleable Cryptography. *SIAM Journal on Computing*, 30(2), pp. 391–437, 2000.
14. E. Fujisaki and T. Okamoto. Secure Integration of Asymmetric and Symmetric Encryption Schemes. *CRYPTO 1999*, LNCS 1666, pp. 538–554, 1999.
15. G. Hanaoka, Y. Kawai, N. Kunihiro, T. Matsuda, J. Weng, R. Zhang, and Y. Zhao. Generic Construction of Chosen Ciphertext Secure Proxy Re-Encryption. *CT-RSA 2012*, LNCS 7178, pp. 349–364, 2012.
16. S. Hohenberger, A.B. Lewko, and B. Waters. Detecting Dangerous Queries: A New Approach for Chosen Ciphertext Security. *EUROCRYPT 2012*, LNCS 7237, pp. 663-681, 2012.
17. T. Isshiki, M.H. Nguyen, and K. Tanaka. Proxy Re-Encryption in a Stronger Security Model Extended from CT-RSA2012. *CT-RSA 2013*, LNCS 7779, pp. 277–292, 2013.
18. A.A. Ivan and Y. Dodis. Proxy Cryptography Revisited. *NDSS 2003*, 2003.
19. E. Kirshanova. Proxy Re-encryption from Lattices. *PKC 2014*, LNCS 8383, pp. 77–94, 2014.
20. B. Libert and D. Vergnaud. Unidirectional Chosen-Ciphertext Secure Proxy Re-Encryption. *PKC 2008*, LNCS 4939, pp. 360–379, 2008.
21. M. Mambo and E. Okamoto. Proxy Cryptosystems: Delegation of the Power to Decrypt Ciphertexts. In *IEICE Trans on Fundamentals of Electronics, Communications and Computer Sciences*, E80-A(1), pp. 54–63, 1997.
22. T. Matsuda, R. Nishimaki, and K. Tanaka. CCA Proxy Re-Encryption without Bilinear Maps in the Standard Model. *PKC 2010*, LNCS 6056, pp. 261–278, 2010.

23. M. Naor and M. Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. *Proc. of STOC 1990*, pp. 427–437, 1990.
24. S. Ohata, T. Matsuda, G. Hanaoka, and K. Matsuura. More Constructions of Re-splittable Threshold Public Key Encryption. *IWSEC 2014*, LNCS 8639, pp.109-118, 2014.
25. A. Pinto, B. Poettering, and J.C.N. Schuldt. Multi-recipient Encryption, Revisited. *AsiaCCS 2014*, pp. 229–238, 2014.
26. C. Rackoff and D. Simon. Non-Interactive Zero-Knowledge Proof of Knowledge and Chosen Ciphertext Attack. *CRYPTO 1991*, LNCS 576, pp. 433–444, 1991.
27. T. Ristenpart, and S. Yilek. The Power of Proofs-of-Possession: Securing Multiparty Signatures against Rogue-Key Attacks. *EUROCRYPT 2007*, LNCS 4515, pp. 228–245, 2007.
28. J. Shao and Z. Cao. CCA-Secure Proxy Re-Encryption without Pairings. *PKC 2009*, LNCS 5443, pp. 357–376, 2009.
29. J. Weng, Y. Zhao, and G. Hanaoka. On the Security of a Bidirectional Proxy Re-Encryption Scheme from PKC 2010. *PKC 2011*, LNCS 6571, pp. 284–295, 2011.
30. X. Zhang, M. Chen, and 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.

A Lemmas and Their Proofs for Theorem 3

Here, we state the proof of soundness in Theorem 3. 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 follows:

$$\begin{array}{l}\mathsf{MiniDec}(tpk, tvk, tsk_1, \mu_2, c) : \\ [\ \mu_1 \leftarrow \mathsf{TShDec}(tpk, tsk_1, c);\ \text{If } \mu_1 = \perp \text{ then return } \perp.; \\ \text{If } \mathsf{TShVer}(tpk, tvk, c, 2, \mu_2) = \perp \text{ then return } \perp.; \\ m \leftarrow \mathsf{TCom}(tpk, tvk, c, \{\mu_1, \mu_2\});\ \text{Return } m.\ ]\end{array}$$

$\mathsf{MiniREncVer}$ 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 follows:

$$\begin{array}{l}\mathsf{MiniREncVer}(tpk, tvk, tsk_1, \mu_2, c, c') : \\ [\ \mu_1 \leftarrow \mathsf{TShDec}(tpk, tsk_1, c);\ \text{If } \mu_1 = \perp \text{ then return } \perp.; \\ \text{If } \mathsf{TShVer}(tpk, tvk, c, 2, \mu_2) = \perp \text{ then return } \perp.; \\ \text{If } c' = c \text{ then return } \top \text{ else return } \perp.\ ]\end{array}$$

(Here, c and c' are supposed to correspond to those that appear in the description of $\mathsf{REncVer}$ in Fig 1.)

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 $\mathsf{rk}_{i^* \to j}$ 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 "$\mathsf{Dec}_1(\mathsf{sk}_j, \widehat{c}^*) \neq m^*$" (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 $i \in \{0, \ldots, 5\}$ 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 $i \in \{0, \ldots, 5\}$, 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:

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

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) $\mathsf{TShDec}(tpk_{i^*}, tsk_{i^*.2}, c) = \mu_2 \neq \bot$ 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 $\mathsf{PDec}(dk_j, \psi) = tsk_{i^*.1} \neq \bot$, 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 $\bot$ 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. □

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 $(\widehat{pk}_1, \ldots, \widehat{pk}_n)$ 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 $\bot$ to $\mathcal{A}$ if $\mu_2 = \bot$.

(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 re-encryption 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).

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. □

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) $\mathsf{PDec}(\widehat{dk}_j, \widehat{c}) = \langle \mathsf{pk}_{i^*} \| \mathsf{pk}_j \| c \| \mu_2 \| \psi \| tvk_{i^*} \| \sigma \rangle \neq \bot$
(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 $|\Pr[\mathsf{Win}_2] - \Pr[\mathsf{Win}_3]| \leq \Pr[\mathsf{Forge}_2] = \Pr[\mathsf{Forge}_3]$.

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 re-encryption key generation query of the form $(\mathsf{pk}_{i^*}, \mathsf{pk}_j)$ or when $\mathcal{A}$ submits a re-encryption 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. □

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 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. $\mathcal{B}$ simulates 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, 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 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 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 $\sigma \leftarrow (sk_{i^*}, \langle \psi \| tvk_{i^*} \| \sigma \rangle)$.
(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, if $\mu_2 = \perp$ then return $\perp$ 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 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 $\mathsf{TShVer}(tpk_{i^*}, tvk_{i^*}, c^*, 2, \mu'_2) = \top$ (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. □

B Proof of Theorem 4

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 A. 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 $\perp$.[8]
  - (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 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).

[8] This case will correspond to the situation where $\mathcal{A}$ asks a 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.

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 $\mathsf{rk}_{i^* \to j}$ 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 $\widehat{M} = \perp$, or $(\mathsf{pk}'_i, \mathsf{pk}'_j) \neq (\mathsf{pk}_i, \mathsf{pk}_j)$, or $\mathsf{SVer}(vk_i, \langle \psi \| tvk_i \| \mathsf{pk}'_i \| \mathsf{pk}'_j \rangle, \sigma) = \perp$.
- (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 $\perp$.[9]

[9] 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.

- (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 $(\mathsf{pk}_j, \widehat{c}, *, *, *, *) \notin L^*_{\mathsf{REnc}}$, $\mathsf{PDec}(\widehat{dk}_j, \widehat{c}) = \langle \mathsf{pk}_{i^*} \| \mathsf{pk}_j \| c \| \mu_2 \| \psi \| tvk_{i^*} \| \sigma \rangle \neq \bot$, $\mathsf{SVer}(vk_{i^*}, \langle \psi \| tvk_{i^*} \| \mathsf{pk}_{i^*} \| \mathsf{pk}_j \rangle, \sigma) = \top$, 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}.

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:

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

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.

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 $\mathsf{Adv}^{\mathtt{CCA\text{-}PKE}}_{(\mathcal{B},n)}(k) = |\Pr[\mathsf{Succ}_1] - \Pr[\mathsf{Succ}_2]|$. 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 $(\widehat{pk}_1, \ldots, \widehat{pk}_n)$ 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 re-encryption query of the form $(\mathsf{pk}_{i^*}, \mathsf{pk}_j, c)$ satisfying $\mathsf{pk}_j \in \mathcal{PK}$ and 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).

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

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

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 $\mathsf{Adv}^{\mathtt{CCA\text{-}PKE}}_{(\mathcal{B},n)}(k) = \frac{1}{2}|\Pr[\mathsf{Succ}_1] - \Pr[\mathsf{Succ}_2]|$, 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) $\mathsf{PDec}(\widehat{dk}_j, \widehat{c}) = \langle \mathsf{pk}_{i^*} \| \mathsf{pk}_j \| c \| \mu_2 \| \psi \| tvk_{i^*} \| \sigma \rangle \neq \perp$
(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

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

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. □

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 $\gamma \in \{0,1\}$. 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

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

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

Lemmas 8 to 13 guarantee that the right hand side of the inequation (1) 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 4. □

C Proof of Theorem 5

Let $\mathcal{A}$ be any PPT adversary that attacks the first-level CCA security of the VPRE scheme eHKK^+ and makes in total Q 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 $\bot$ 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 $\bot$. 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 $\bot$.

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:

$$\mathsf{Adv}^{\mathtt{first\text{-}VPRE}}_{\mathcal{A}}(k) = |\Pr[\mathsf{Succ}_0] - \frac{1}{2}| \leq \sum_{i \in \{0,1,2\}} |\Pr[\mathsf{Succ}_i] - \Pr[\mathsf{Succ}_{i+1}]| + |\Pr[\mathsf{Succ}_3] - \frac{1}{2}|.$$

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

Firstly, note that 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 $|\Pr[\mathsf{Succ}_2] - \Pr[\mathsf{Succ}_3]|$. 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 (see the last paragraph of Section 2.3) 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 5. □