Reducing Honest Re-Encryption Attack to Chosen Ciphertext Attack

Haotian Yin[1], Jie Zhang[1], Wanxin Li[1], Yuji Dong[2], Eng Gee Lim[1], Dominik Wojtczak[3]

[1] School of Advanced Technology,
Xi'an Jiaotong-Liverpool University, Suzhou, 215123, China
haotian.yin23@student.xjtlu.edu.cn,{jie.zhang01,wanxin.li,enggee.lim}@xjtlu.edu.cn
[2] School of Internet of Things,
Xi'an Jiaotong-Liverpool University, Suzhou, 215400, China
yuji.dong02@xjtlu.edu.cn
[3] Department of Computer Science,
University of Liverpool, Liverpool, L69 3BX, UK
d.wojtczak@liverpool.ac.uk

Abstract. Proxy re-encryption (PRE) schemes allow a delegator to designate a proxy to re-encrypt its ciphertext into a ciphertext that the delegatee can decrypt. In contrast, the proxy gains nothing helpful from this transformation. This decryption-power transfer is proper in applications of encrypted email forwarding, key escrow, and publish/subscribe systems.

The security notions for PRE are inherited from the standard public key encryption (PKE) schemes, i.e., indistinguishability under chosen-plaintext attacks (CPA) and security under chosen-ciphertext attacks (CCA). A recently popular notion, indistinguishability under honest re-encryption attacks (HRA), was proposed by Cohen in 2019, indicating that CPA security is insufficient for PRE because some CPA-secure PRE leaks the secret key of the delegator. Many post-quantum secure PRE schemes have recently been designed under the HRA security model.

However, HRA security differs from traditional CCA security, and there is no known reduction between them. The existing results show they appear to be incompatible. This paper aims to bridge those two security notions via reductions. In addition, we found that many existing HRA-secure schemes are vulnerable to collusion. We provide a generic transformation from a CPA-secure PRE to a collusion-resistant and CPA-secure PRE. This transformation also applies to HRA-secure PREs.

Keywords: Honest re-encryption attack · Proxy re-encryption · Collusion resistance.

1 Introduction

A proxy re-encryption (PRE) scheme can be considered an extension of the traditional public key encryption (PKE) scheme. The additional re-encryption

functionality enables a proxy to transform a ciphertext encrypted under a delegator's public key to a decryptable ciphertext for a delegatee, while the proxy gains nothing about the secret or plaintext from the procedure. Bidirectional and unidirectional PREs are defined according to the direction of ciphertext transformation by the re-encryption key. We focus on the unidirectional PREs since the delegator can generate the re-encryption key without the secret key from the delegatee (non-interactive). This kind of scheme admits many interesting applications, such as encrypted email forwarding [3], key escrow [16], distributed file system [2], and publish/subscribe systems [24].

The security of PRE is defined accordingly by PKE. The standard and basic security notion for an encryption scheme is ciphertext indistinguishability, which is often described in a security game against chosen plaintext attacks (IND-CPA) or ciphertext attacks (IND-CCA). We only consider ciphertext indistinguishability in this paper, so we will omit the prefix IND and use CPA or CCA to indicate the corresponding security notion. With the function of re-encryption, the PRE adversary is provided an additional re-encryption key generation oracle in both CPA and CCA games and a decryption oracle for the CCA game.

Cohen [10] found that the current CPA game for PRE [1] fails in capturing adversarial delegatee, who is often named Bob in the literature, corresponding to the question of the beginning of the paper's title, "What about Bob?" To be more precise, in some CPA-secure PREs, Bob can decrypt all ciphertexts encrypted under Alice's public key, the delegator, just given an *honest re-encrypted* ciphertext. This is not desirable for many applications, where the delegation should be handled by Alice and the proxy, not the silent recipient, Bob! We can think of this error as violating the principle of least privilege. The re-encrypted ciphertext should give Bob only the message decrypted from it, and should be nothing else. Cohen named this attack an honest re-encryption attack (HRA) and found that a published work [24] is vulnerable to HRA and studied a property, re-encryption simulatability, for upgrading CPA-secure PRE to an HRA-secure one.

1.1 Research Questions

HRA security is becoming a new standard for PRE schemes. We found many lattice-based PRE schemes [11, 17, 27, 30] are constructed and proved secure in the HRA game. However, there is no existing work addressing the reduction from HRA to CCA, although the CCA game is seemingly stronger. The initial study shows that those two security notions are probably incompatible [10]. Therefore, this paper aims to address the following question:

Can we reduce the HRA to CCA for PRE?

Single-hop PREs are sufficient for many applications [2, 23, 29, 31], and collusion resistance is a desirable security property for protecting the first-level ciphertext (that cannot be re-encrypted more). We also found that some PRE schemes proved HRA-secure suffers collusion [10,11,26] in the single-hop setting, where a proxy and a delegatee conspire to compute the delegator's secret (secret

key or messages that are unwilling to disclose). Therefore, this paper tries to work out the second question:

How to generically construct collusion-resist HRA single-hop PRE?

1.2 Related Works

Nuñez, Agudo, and Lopez [21] defined a framework for systematically classifying PRE schemes according to the oracle access period in the security model. More specifically, compared with basic CPA security, "full" CCA security provides decryption and re-encryption oracles for the adversary. Those two oracles are accessible before the challenge or always accessible, according to the specifications of the security game. They use two subscripts to indicate the time when these two oracles are accessible. For example, $\mathsf{CCA}_{1,2}$ means the decryption oracle is accessible before the challenge (only phase 1), and the re-encryption is always accessible (both phase 1 and phase 2); therefore, $\mathsf{CCA}_{2,2}$ represents the full CCA game (all oracles are accessible in all phases), and $\mathsf{CCA}_{0,0}$ represents the basic CPA game (no re-encryption and decryption oracles). One of their results is that the decryption oracle is simulatable by the re-encryption oracle in *multi-hop* PREs, i.e., $\mathsf{CCA}_{0,2}$ is equivalent to $\mathsf{CCA}_{2,2}$. However, this framework was designed before the HRA security model; it tells nothing about the relationship between HRA and CCA security.

Contrary to some functions of proxy application scenarios, a CPA secure PRE might enable the delegatee to obtain the secret key of the delegator. Cohen [9, 10] proposed an HRA security game for capturing this potential threat from the curious delegatee. HRA security is not needed when the delegation mode is "all or nothing", i.e., once the delegator agrees to transfer the decryption rights, all ciphertexts should be decryptable by the designated delegatee. But if this authorization needs to be *really* re-encrypted by the proxy, HRA plays a role. This characteristic, protecting *un*-re-encrypted ciphertext, is similar to the notion of master secret security [2]. However, master secret security describes the security of the delegator's secret key after collusion between proxy and delegatee, so this notion is parallel to an honest re-encryption attack.

Since HRA is a popular model for PRE, it is desirable to construct the generic transformation from HRA to CCA. To the best of our knowledge, although the re-encryption oracle in HRA is similar to the CCA's, except the HRA game forbids adversarial re-encryption, there is no generic transformation toward CCA security. Recently, Sato and Shikata [25] studied the problem and proposed two generic transformations from a CPA-secure PRE to a *bounded* CCA-secure one. They borrowed the design idea of the first CCA-secure PRE [7], which further stemmed from the concept of CCA-secure identity-based encryption [6]. They applied their transformation to a CPA-secure PRE based on CRYSTALS-Kyber [5] (Kyber's IND-CPA-secure encryption, not the Fujisaki-Okamoto [14] transformed one). Before their generic transformation, Nuñez, Agudo, and Lopez [22] studied the same transformation from CPA to CCA, but they only got a weaker form from $\mathsf{CCA}_{0,1}$ to $\mathsf{CCA}_{2,1}$. They also investigated

the infeasibility of directly applying a generic transformation from traditional PKE (such as the Fujisaki-Okamoto transform) on PRE schemes.

Many PRE schemes [11, 13, 19, 26, 27, 29, 30] are proposed under the HRA security model. Even though the post-quantum and bounded CCA-secure PRE from generic transformations is proposed [25], no analysis is conducted on the relationship between the HRA and CCA security. Is the CCA security really incompatible with the HRA security as Cohen initially studied [9]? Do we also need to prove that a CCA-secure PRE is also HRA-secure? Our answers are both negative.

Zhao et al. [28] are concerned with collusion due to the semi-trust third-party proxy. But their PRE is multi-hop; to the best of our knowledge, we believe that collusion in the context of PRE (between the proxy and the delegatee) is not applicable to multi-hop scenarios. The collusion resistance was first defined by Ateniese et al. [2]. We found some HRA-secure single-hop PRE schemes [10, 11, 26] are vulnerable to collusion attacks.

1.3 Contribution

For the first research question, we will discuss the solution in two cases: single-hop PRE and multi-hop PRE. The number of re-encryptions a scheme can perform will affect whether the game simulation can be run flawlessly and the types of ciphertext that can be queried (to prevent trivial attacks). We found that the HRA game can be reduced to the CCA game losslessly since the definition for the derivative of challenge ciphertext is equivalent in a single-hop setting. We just need to simulate the re-encryption in the multi-hop security game to finish the reduction. For the second research question, we will design a generic construction to provide collusion resistance based on the idea of [2]. The contribution can be concluded as follows:

- Reducing HRA security to CCA security perfectly.
- Constructing a generic transformation for CPA/HRA secure PRE with collusion resistance at the cost of doubling the key length.

1.4 Road map

The rest of this paper is organized by the following sections: Section 2 gives the preliminary knowledge; Section 3 reduces HRA to CCA security; Section 4 constructs the generic transformation for providing collusion resistance; Section 5 provides some future works.

2 Preliminaries

This section introduces the PRE definitions and security notions.

2.1 Proxy Re-Encryption

We first adapt the definition for multi-hop PRE schemes [7].

Definition 1 (Multi-hop PRE scheme). *A multi-hop proxy re-encryption scheme* mhPRE *is a tuple of algorithms* $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{ReKeyGen}, \mathsf{Enc}, \mathsf{ReEnc}, \mathsf{Dec})$:

- $\mathsf{Setup} \to \mathsf{pp}$. *The setup algorithm outputs a public configuration/parameter* pp*, which serves as an implicit input of the following algorithms.*
- $\mathsf{KeyGen} \to (\mathsf{pk}_i, \mathsf{sk}_i)$. *The key generation algorithm outputs a pair of public and secret keys* $(\mathsf{pk}_i, \mathsf{sk}_i)$ *for user* i.
- $\mathsf{ReKeyGen}(\mathsf{pk}_i, \mathsf{sk}_i, \mathsf{pk}_j) \to \mathsf{rk}_{i\to j}$. *On input the pair of public and secret keys* $(\mathsf{pk}_i, \mathsf{sk}_i)$ *for user* i *and the public key* pk_j *for user* j*, the re-encryption key generation algorithm outputs a re-encryption key* $\mathsf{rk}_{i\to j}$.
- $\mathsf{Enc}(\mathsf{pk}_i, \mathsf{m}) \to \mathsf{ct}_i$. *On input a public key* pk_i *and a message* $\mathsf{m} \in \mathcal{M}$*, the encryption algorithm outputs a ciphertext* ct_i.
- $\mathsf{ReEnc}(\mathsf{rk}_{i\to j}, \mathsf{ct}_i) \to \mathsf{ct}_j$. *On input a re-encryption key from* i *to* j $\mathsf{rk}_{i\to j}$ *and a ciphertext* ct_i*, the re-encryption algorithm outputs a ciphertext* ct_j *or the error symbol* $\perp$.
- $\mathsf{Dec}(\mathsf{sk}_j, \mathsf{ct}_j) \to \mathsf{m}$. *Given a secret key* sk_j *and a ciphertext* ct_j*, the decryption algorithm outputs a message* $\mathsf{m} \in \mathcal{M}$ *or the error symbol* $\perp$.

Based on the above multi-hop setting, we define the single-hop PRE [2] by introducing additional encryption and decryption algorithms to restrict the re-encryption depth.

Definition 2 (Single-hop PRE scheme). *A single-hop proxy re-encryption scheme* shPRE *is a tuple of algorithms* $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{ReKeyGen}, \mathsf{Enc}_t, \mathsf{ReEnc}, \mathsf{Dec}_t)$ *where* $t \in \{1, 2\}$*:*

- $\mathsf{Setup} \to \mathsf{pp}$. *Same as the multi-hop setting.*
- $\mathsf{KeyGen} \to (\mathsf{pk}_i, \mathsf{sk}_i)$. *Same as the multi-hop setting.*
- $\mathsf{ReKeyGen}(\mathsf{pk}_i, \mathsf{sk}_i, \mathsf{pk}_j) \to \mathsf{rk}_{i\to j}$. *Same as the multi-hop setting.*
- $\mathsf{Enc}_1(\mathsf{pk}_i, \mathsf{m}) \to \mathsf{ct}_i$. *On input a public key* pk_i *and a message* $\mathsf{m} \in \mathcal{M}$*, the encryption algorithm outputs a* first-level *ciphertext* ct_i*. This ciphertext* cannot *be re-encrypted.*
- $\mathsf{Enc}_2(\mathsf{pk}_i, \mathsf{m}) \to \mathsf{ct}_i$. *On input a public key* pk_i *and a message* $\mathsf{m} \in \mathcal{M}$*, the encryption algorithm outputs a* second-level *ciphertext* ct_i*. This ciphertext* can *be re-encrypted to a* first-level *ciphertext.*
- $\mathsf{ReEnc}(\mathsf{rk}_{i\to j}, \mathsf{ct}_i) \to \mathsf{ct}_j$. *On input a re-encryption key from* i *to* j $\mathsf{rk}_{i\to j}$ *and a* second-level *ciphertext* ct_i*, the re-encryption algorithm outputs a* first-level *ciphertext* ct_j *or the error symbol* $\perp$.
- $\mathsf{Dec}_1(\mathsf{sk}_j, \mathsf{ct}_j) \to \mathsf{m}$. *Given a secret key* sk_j *and a* first-level *ciphertext* ct_j*, the decryption algorithm outputs a message* $\mathsf{m} \in \mathcal{M}$ *or the error symbol* $\perp$.
- $\mathsf{Dec}_2(\mathsf{sk}_j, \mathsf{ct}_j) \to \mathsf{m}$. *Given a secret key* sk_j *and a* second-level *ciphertext* ct_j*, the decryption algorithm outputs a message* $\mathsf{m} \in \mathcal{M}$ *or the error symbol* $\perp$.

Note that our definition of ciphertext follows the initial definition of Ateniese et al. [2] (second-level ciphertext can be re-encrypted to a first-level one), different from Zhou et al. [31] (opposite to ours). This is just a difference in naming, they are essentially equivalent definitions.

2.2 Security Notions

The main differences between models are the definition of trivial attacks and oracles. Trivial attacks are captured by the rejection condition of oracles, e.g., a decryption query to the challenge ciphertext should be rejected by the decryption oracle. Stronger models usually equip more powerful and prolific oracles, which provide the adversary with more information and fewer restrictions.

We first describe the security game for Honest Re-encryption attacks [10].

Definition 3 (Security Game for HRA). *Let λ be the security parameter and $\mathcal{A}$ be an oracle Turing machine.* HRA *game consists of an execution of $\mathcal{A}$ with the following oracles:*
Phase 1:

- *Setup: The public parameters are generated and given to $\mathcal{A}$. A counter* $\mathsf{numKeys}$ *is initialized to 0, and the sets* Hon *(of honest, uncorrupted indices) and* Cor *(of corrupted indices) are initialized to be empty. Additionally, the following are initialized: A counter* numCt *to 0, a key-value store $\mathcal{C}$ to empty, and a set* Deriv *to be empty. This oracle is executed first and only once.*
- *Honest Key Generation: Obtain a new key pair* $(\mathsf{pk}_{\mathsf{numKeys}}, \mathsf{sk}_{\mathsf{numKeys}}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$ *and give* $\mathsf{pk}_{\mathsf{numKeys}}$ *to $\mathcal{A}$. The current value of* $\mathsf{numKeys}$ *is added to* Hon *and* $\mathsf{numKeys}$ *is incremented.*
- *Corrupted Key Generation: Obtain a new key pair* $(\mathsf{pk}_{\mathsf{numKeys}}, \mathsf{sk}_{\mathsf{numKeys}}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$ *and give the key pair to $\mathcal{A}$. The current value of* $\mathsf{numKeys}$ *is added to* Cor *and* $\mathsf{numKeys}$ *is incremented.*

Phase 2: For each pair $i, j \leq \mathsf{numKeys}$, compute the re-encryption key $\mathsf{rk}_{i\to j} \leftarrow \mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j)$.

- *Re-encryption Key Generation* $\mathcal{O}_{\mathsf{ReKeyGen}}$ *: On input (i, j) where $i, j \leq \mathsf{numKeys}$, if $i = j$ or if $i \in \mathsf{Hon}$ and $j \in \mathsf{Cor}$, output $\perp$. Otherwise return the re-encryption key* $\mathsf{rk}_{i\to j}$.
- *Encryption* $\mathcal{O}_{\mathsf{Enc}}$*: On input (i, m), where $i \leq \mathsf{numKeys}$, compute* $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}_i, \mathsf{m})$ *and increment* numCt*. Store the value* ct *in $\mathcal{C}$ with key (i, numCt). Return $(\mathsf{numCt}, \mathsf{ct})$. This oracle takes extra identifier for specifying the encryption algorithm in single-hop settings.*
- *Challenge Oracle: On input $(i, \mathsf{m}_0, \mathsf{m}_1)$ where $i \in \mathsf{Hon}$ and $\mathsf{m}_0, \mathsf{m}_1 \in \mathcal{M}$, sample a bit $b \leftarrow \{0, 1\}$ uniformly at random, compute the challenge ciphertext* $\mathsf{ct}^* \leftarrow \mathsf{Enc}(\mathsf{pk}_i, \mathsf{m}_b)$ *(use* Enc_2 *for single-hop PREs), and increment* numCt*. Add* numCt *to the set* Deriv*. Store the value* ct^* *in $\mathcal{C}$ with key (i, numCt). Return $(\mathsf{numCt}, \mathsf{ct}^*)$. This oracle can only be queried once.*
- *Re-encryption* $\mathcal{O}_{\mathsf{ReEnc}}$*: On input (i, j, k) where $i, j \leq \mathsf{numKeys}$ and $k \leq \mathsf{numCt}$, if $j \in \mathsf{Cor}$ and $k \in \mathsf{Deriv}$ return $\perp$. If there is no value in $\mathcal{C}$ with key (i, k), return $\perp$. Otherwise, let ct_i be that value in $\mathcal{C}$, let $\mathsf{ct}_j \leftarrow \mathsf{ReEnc}(\mathsf{rk}_{i\to j}, \mathsf{ct}_i)$, and increment* numCt*. Store the value ct_j in $\mathcal{C}$ with key (j, numCt). If $k \in \mathsf{Deriv}$, add* numCt *to the set* Deriv*. Return $(\mathsf{numCt}, \mathsf{ct}_j)$.*

Phase 3:

– *Decision: On input a bit* $b' \in \{0,1\}$*, return* win *if* $b = b'$.

The HRA *advantage of* $\mathcal{A}$ *is defined as*

$$\mathsf{Adv}^{\mathcal{A}}_{\mathsf{hra}}(\lambda) = \Pr[\mathsf{win}],$$

where the probability is over the randomness of $\mathcal{A}$ *and the oracles in* HRA *game.*

To facilitate reduction, we modify the formalization of CCA security model [7]. We dropped the decryption oracle since the reduction does need this oracle (in [21]'s nomenclature, our CCA model is $\mathsf{CCA}_{0,2}$). The definition of derivatives of the challenge is adapted in that we use the indices of users and ciphertexts but not their value. In addition, we use a set Deriv to store the derivatives of the challenge just like HRA game.

Definition 4 (Security Game for CCA). *Let* λ *be the security parameter and* $\mathcal{A}$ *be an oracle Turing machine.* CCA *game consists of an execution of* $\mathcal{A}$ *with the following oracles, which can be invoked in any order, subject to the constraints below:*
Phase 1:

– *Setup: The public parameters are generated and given to* $\mathcal{A}$*. Counters* $\mathsf{numKeys}$ *and* numCt *are initialized to 0, the sets* Hon*,* Cor *and* Deriv *are initialized to be empty, and key-value stores* Deriv*,* ReKeys*, and* $\mathcal{C}$ *to be empty. This oracle is executed first and only once.*
– *Honest key generation: Obtain a new key pair* $(\mathsf{pk}_{\mathsf{numKeys}}, \mathsf{sk}_{\mathsf{numKeys}}) \leftarrow \mathsf{KG}(\mathsf{pp})$ *and give* $\mathsf{pk}_{\mathsf{numKeys}}$ *to* $\mathcal{A}$*. The current value of* $\mathsf{numKeys}$ *is added to* Hon *and* $\mathsf{numKeys}$ *is incremented.*
– *Corrupted key generation: Obtain a new key pair* $(\mathsf{pk}_{\mathsf{numKeys}}, \mathsf{sk}_{\mathsf{numKeys}}) \leftarrow \mathsf{KG}(\mathsf{pp})$ *and given to* $\mathcal{A}$*. The current value of* $\mathsf{numKeys}$ *is added to* Cor *and* $\mathsf{numKeys}$ *is incremented.*

Phase 2:

– *Re-encryption Key Generation* $\mathcal{O}_{\mathsf{ReKeyGen}}$ *: On input* (i,j) *where* $i,j \leq \mathsf{numKeys}$*, if* $i = j$ *or if* $i \in \mathsf{Hon}$ *and* $j \in \mathsf{Cor}$*, output* $\perp$*. If* $\mathcal{O}_{\mathsf{R}}$ *has not been executed on input* (i,j)*, i.e.,* $(i,j) \notin \mathsf{ReKeys}$*, compute and store* $\mathsf{rk}_{i\to j} \leftarrow \mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j)$*. Output the re-encryption key* $\mathsf{rk}_{i\to j}$*. Add* $((i,j), \mathsf{rk}_{i\to j})$ *to* ReKeys*.*
– *Challenge Oracle: On input* $(i, \mathsf{m}_0, \mathsf{m}_1)$ *where* $i \in \mathsf{Hon}$ *and* $\mathsf{m}_0, \mathsf{m}_1 \in \mathcal{M}$*, sample a bit* $b \leftarrow \{0,1\}$ *uniformly at random, compute the challenge ciphertext* $\mathsf{ct}^* \leftarrow \mathsf{Enc}(\mathsf{pk}_i, \mathsf{m}_b)$ *(use* Enc_2 *for single-hop PREs), and increment* numCt*. Add* (i, numCt) *to set* Deriv*. Store the value* ct^* *in* $\mathcal{C}$ *with key* (i, numCt)*. Return* $(\mathsf{numCt}, \mathsf{ct}^*)$*. This oracle can only be queried once.*
– *Re-encryption* $\mathcal{O}_{\mathsf{ReEnc}}$*: On input* (i,j,k) *where* $i,j \leq \mathsf{numKeys}$ *and* $k \leq \mathsf{numCt}$*. If* $j \in \mathsf{Cor}$ *and* $(i,k) \in \mathsf{Deriv}$*, return* $\perp$*. Let* $\mathsf{ct}_j \leftarrow \mathsf{ReEnc}(\mathsf{rk}_{i\to j}, \mathsf{ct}_i)$*, and increment* numCt*. If*

$$(i,k) \in \mathsf{Deriv}, \tag{1}$$

or

$$(i, j) \in \mathsf{ReKeys} \wedge \mathsf{Dec}(j, \mathsf{ct}_j) \in \{\mathsf{m}_0, \mathsf{m}_1\} \wedge (i, \star) \in \mathsf{Deriv}, \tag{2}$$

add (j, numCt) *to* Deriv. *Return* $(\mathsf{numCt}, \mathsf{ct}_j)$.

Phase 3:

– *Decision: On input a bit* $b' \in \{0, 1\}$, *return* win *if* $b = b'$.

The CCA *advantage of* $\mathcal{A}$ *is defined as*

$$\mathsf{Adv}^{\mathcal{A}}_{\mathsf{cca}}(\lambda) = \Pr[\mathsf{win}],$$

where the probability is over the randomness of $\mathcal{A}$ *and the oracles in* CCA *game.*

Remark 1 (Subtle about the formalization of HRA and CCA games). In the CCA game, we use key-value store Deriv, in which we only store the ciphertext identifier in the HRA game. The reason is that the definition of derivatives requires both the ciphertext and the owner of the encryption key. This makes sense because a wrong-paired re-encryption query should give no advantage to the adversary (given legal challenge (i^*, k^*), query $\mathsf{ReEnc}(i, j, k^*)$, where $i \neq i^*$). This kind of query is forbidden in the HRA game because only owner-ciphertext pairs that are generated honestly from Enc oracle might be accepted by ReEnc. This difference does not affect our reduction from HRA to CCA security.

Remark 2 (Phases of CCA game). We define the CCA game in phases similar to the HRA game to provide a convenient reduction. The original CCA definition has no phases. However, this modification does not weaken our definition since both models have selective corruption (our model has selective "key generation"). The only difference between the two models is whether the adversary must query key generations *before* other queries. Since the adversary runs within probabilistic polynomial time, we can prepare polynomially many key pairs to simulate the "adaptive key generation". Or, we can remove the predestined number of honest and corrupted users by a simple machine in Appendix A. Also, this paper does not discuss the adaptive corruption [13, 17].

The adapted definition[4] for derivatives of the challenge $(\mathsf{pk}^*, \mathsf{ct}^*)$ is defined inductively as the following rules:

1. $(\mathsf{pk}^*, \mathsf{ct}^*)$ is a derivative of itself.
2. If $(\mathsf{pk}, \mathsf{ct})$ is a derivative of $(\mathsf{pk}^*, \mathsf{ct}^*)$ and $(\mathsf{pk}', \mathsf{ct}')$ is a derivative of $(\mathsf{pk}, \mathsf{ct})$, then $(\mathsf{pk}', \mathsf{ct}')$ is a derivative of $(\mathsf{pk}^*, \mathsf{ct}^*)$.
3. If $\mathcal{A}$ has issued a re-encryption query $(\mathsf{pk}, \mathsf{pk}', \mathsf{ct})$ and obtained a ciphertext (ct') as a response, then $(\mathsf{pk}', \mathsf{ct}')$ is a derivative of $(\mathsf{pk}, \mathsf{ct})$.
4. If $\mathcal{A}$ has issued a re-encryption key generation query $(\mathsf{pk}, \mathsf{pk}')$, and $\mathsf{Dec}(\mathsf{pk}', \mathsf{ct}') \in \{\mathsf{m}_0, \mathsf{m}_1\}$, then $(\mathsf{pk}', \mathsf{ct}')$ is a derivative of $(\mathsf{pk}, \mathsf{ct})$ fo all ct.

[4] In original paper [7], the derivatives of the challenge are defined for bidirectional PRE.

Next, we will show that our adaptation and formalization are proper, i.e., they are equivalent. We claim the above inductive definition restricts the same queries as the CCA game in the Definition 4. For each rule,

- Rule 1: This is equivalent when we add the current ciphertext identifier and challenge ciphertext to Deriv in the challenge oracle;
- Rule 2&3: We add the derivative pairs to Deriv that satisfy the assertion (1);
- Rule 4: This is the most complex one. We first check if the corresponding re-encryption key is queried ($(i,j) \in \mathsf{ReKeys}$), then if the decryption is one of the challenge messages ($(j, \mathsf{ct}_j) \in \{\mathsf{m}_0, \mathsf{m}_1\}$), and if any delegator's ciphertext is a derivative of the challenge ($(i, \mathsf{ct}_i) \in \mathsf{Deriv}$), then we put (j, numCt) in Deriv. This is exactly what the conjunction (2) expresses.

3 Reduction from HRA to CCA Security

We use two lemmas to build the main theorem. For simplicity, we use the notation $\mathsf{G}.\mathcal{O}$ to indicate the oracle $\mathcal{O}$ of game G.

3.1 Reduction for Single-Hop PRE

Lemma 1 ($\mathsf{CCA} \Rightarrow \mathsf{HRA}$ for Single-Hop PRE). *For a single-hop PRE, the security defined by the* CCA *game implies the security defined by the* HRA *game.*

Proof. Suppose there is an HRA adversary $\mathcal{A}$ such that $\mathsf{Adv}^{\mathcal{A}}_{\mathsf{hra}}(\lambda) \geq 1/\mathsf{poly}(\lambda)$ for some polynomial function $\mathsf{poly}(\lambda)$, we will show that there exists an adversary $\mathcal{B}$ who wins the CCA game with same probability.

- On setup oracle query, $\mathcal{B}$ calls its CCA setup oracle, and maintains $\mathsf{numKeys}$, Hon, Cor, numCt, Deriv, and $\mathcal{C}$ for $\mathcal{A}$.
- On honest, corrupted, and re-encryption key generation oracles queries, $\mathcal{B}$ can perfectly simulate them by calling its own CCA oracles.
- On encryption oracle queries, $\mathcal{B}$ computes the ciphertext locally by and behaves just like oracle $\mathsf{HRA}.\mathcal{O}_{\mathsf{Enc}}$.
- On challenge oracle query, $\mathcal{B}$ transfers the input of $\mathcal{A}$ to its own challenge oracle and returns what the challenge ciphertext from the query with the identifier numCt maintained by $\mathcal{B}$.
- On re-encryption oracle with query (i, j, k), $\mathcal{B}$ checks if the key pairs of user i and j are generated and if the ciphertext is generated via the encryption oracle $\mathsf{HRA}.\mathcal{O}_{\mathsf{Enc}}$ by checking local store $\mathcal{C}$. We claim that $\mathsf{CCA}.\mathcal{O}_{\mathsf{ReEnc}}$ accepts the query if $\mathsf{HRA}.\mathcal{O}_{\mathsf{ReEnc}}$ accepts the query. Next, we check the condition where those two re-encryption oracles would reject the query. The single-hop setting makes the two assertions (1)(2) *redundant* since the ciphertext cannot be re-encrypted more than once. Therefore, the ciphertext in Deriv that is being re-encrypted could only be the challenge ciphertext. So, there is no need to check if the input ciphertext is a derivative of the challenge except for the *challenge ciphertext itself*. The oracle $\mathsf{CCA}.\mathcal{O}_{\mathsf{ReEnc}}$ will reject query

(i, j, k) where (i, k) is the challenge and $j \in \mathsf{Cor}$. The oracle $\mathsf{HRA}.\mathcal{O}_{\mathsf{ReEnc}}$ will reject any query (i, j, k) such that $j \in \mathsf{Cor} \wedge k \in \mathsf{HRA.Deriv}$ or $(i, k) \notin \mathcal{C}$. Let the challenge ciphertext identifier be k^*, and the challenge user is i^*; their acceptance is shown in Table 1.

Table 1. Acceptance between re-encryption oracles of HRA and CCA

	i	j	k	$\mathsf{CCA}.\mathcal{O}_{\mathsf{ReEnc}}$	$\mathsf{HRA}.\mathcal{O}_{\mathsf{ReEnc}}$
Case 1	$= i^*$	$\in \mathsf{Cor}$	$= k^*$	Reject	Reject
Case 2	$\in \mathsf{Hon}$	$\in \mathsf{Cor}$	$= k^*$	Accept	Reject
Case 3	$\in \mathsf{Cor}$	$\in \mathsf{Cor}$	$= k^*$	Accept	Reject

The above three cases show that the $\mathsf{CCA}.\mathcal{O}_{\mathsf{ReEnc}}$ accepts more queries than $\mathsf{HRA}.\mathcal{O}_{\mathsf{ReEnc}}$ when the ciphertext is the challenge, where only Case 1 both oracles reject because the query is identified as an illegal re-encryption of the challenge's derivative (see Remark 1 for Case 2 and Case 3). Therefore, $\mathcal{B}$ can use its own $\mathsf{CCA}.\mathcal{O}_{\mathsf{ReEnc}}$ to perfectly simulate $\mathsf{HRA}.\mathcal{O}_{\mathsf{ReEnc}}$.

– On the decision oracle, $\mathcal{B}$ transfers the input of $\mathcal{A}$ to its own decision oracle.

In conclusion, $\mathcal{B}$ perfectly simulates the HRA security game for $\mathcal{A}$, i.e., $\mathsf{Adv}^{\mathcal{B}}_{\mathsf{cca}} \geq \mathsf{Adv}^{\mathcal{A}}_{\mathsf{hra}}$. □

This result shows that, in a single-hop setting, although CCA has a seemingly over-inclusive definition of derivatives [9, Appendix B.2], the actual restriction is the inverse (in Table 1, $\mathsf{HRA}.\mathcal{O}_{\mathsf{ReEnc}}$ rejects more queries). This result partially answers the unsolved question of whether CCA security implies HRA security: in single-hop setting, CCA security actually implies HRA, without *re-encryption simulatability* (a reduction chain from the facts that CCA ⇒ CPA, and a CPA-secure and re-encryption simulatable PRE is HRA secure [10, Theorem 5]).

3.2 Reduction for Multi-Hop PRE

In Canetti and Hohenberger's original CCA security game [7], the re-encryption oracle rejects all ciphertexts that could possibly be re-encrypted from the challenge. The HRA game uses Enc oracle records the honest encrypted ciphertext that can be re-encrypted, no matter whether the encrypted message is the challenge message or not. This difference also stops the initial reduction by Cohen [9, Appendix B.2].

In our reduction, we found that those rejected ciphertexts by CCA are simulatable.

Lemma 2 (CCA ⇒ HRA in Multi-Hop Setting). *For a multi-hop PRE, the security defined by the* CCA *game implies the security defined by the* HRA *game.*

Proof. This reduction is analogous to Lemma 1 for single-hop setting. $\mathcal{B}$ will simulate a HRA security game for adversary $\mathcal{A}$ with winning probability greater

than $1/\mathsf{poly}(\lambda)$ for some polynomial poly. We omit the simulation of setup, key generation, encryption, challenge, and decision oracles, cause those oracles are simulatable by $\mathcal{B}$ with its CCA oracles.

– On re-encryption oracle with query (i, j, k), $\mathcal{B}$ rejects with $\perp$ if $j \in \mathsf{Cor}$ and $k \in \mathsf{Deriv}$. If $(i, k) \notin \mathcal{C}$, $\mathcal{B}$ also rejects and return $\perp$. Otherwise, $\mathcal{B}$ find the value ct_i with indices k. We only care about the situation when $\mathsf{HRA}.\mathcal{O}_{\mathsf{ReEnc}}$ accepts while $\mathsf{CCA}.\mathcal{O}_{\mathsf{ReEnc}}$ rejects. The second assertion (2) of $\mathsf{CCA}.\mathcal{O}_{\mathsf{ReEnc}}$ bring *extra* ciphertexts into CCA.Deriv compared with HRA.Deriv. More specifically, $\mathcal{A}$ can re-encrypt other encryptions of the challenge messages as long as those encryptions were generated independently from the challenge ciphertext in HRA game. Fig. 1 shows the difference between those two oracles' acceptance.

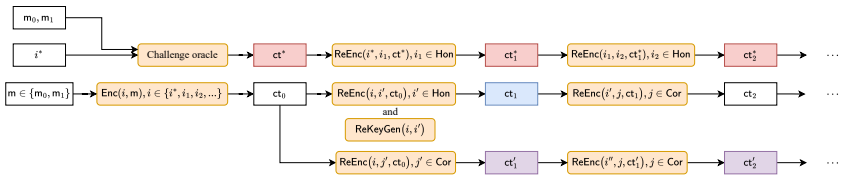

Fig. 1. Acceptance example of $\mathsf{CCA}.\mathcal{O}_{\mathsf{ReEnc}}$ and $\mathsf{HRA}.\mathcal{O}_{\mathsf{ReEnc}}$. In particular, denotes the oracles; denotes the ciphertexts that are in Deriv for both $\mathsf{CCA}.\mathcal{O}_{\mathsf{ReEnc}}$ and $\mathsf{HRA}.\mathcal{O}_{\mathsf{ReEnc}}$; denotes the ciphertexts that are in CCA.Deriv but not HRA.Deriv; denotes the simulated ciphertexts

In this figure, ct_0 is a ciphertext of a challenge *message* that some user of Deriv honestly encrypts; ct_1 is a derivative in CCA.Deriv because its generation (from ct_0 to ct_1) satisfies the assertion (2); therefore, ct_2 can only be generated via HRA.ReEnc, i.e., the query (i', j, ct_1) will be rejected by oracle CCA.ReEnc. We can simulate ct_2 by ct_2', which can be legally generated by oracle CCA.ReEnc.

This simulation of re-encryption will work because $\mathsf{ReKeyGen}(i, j')$ will be rejected since the game rejects re-encryption key generation from an honest user to a corrupted user; therefore, this promises the output of the re-encryption oracle, ct_1', will not be recognized as a derivative of the challenge ciphertext.

Therefore, a CCA adversary $\mathcal{B}$ can successfully simulate the HRA security game for $\mathcal{A}$ perfectly. □

Remark 3 (Indistinguishability of the simulated ciphertext). An implicit assumption in the proof is that the ciphertext under the same public key and the same message is indistinguishable. We omitted to claim it explicitly since it is naturally captured by the basic security of PRE. More precisely, user j (or the adversary $\mathcal{A}$) is given $\mathsf{ct}_2' = \mathsf{ct}_2$ if the re-encryption algorithm is deterministic; otherwise, given the randomness of ct_2', user j (or the adversary $\mathcal{A}$) still cannot distinguish

if ct'_2 is re-encrypted from ct_1 or ct'_1 without the randomness of those two ciphertexts (and/or the randomness of the re-encryption algorithm). This can be formalized as a simulation algorithm $\mathsf{ReEncSim}$ with only public information (no secret key and no message) as inputs. A much weaker assumption compared to *re-encryption simulatability* [10].

This proof shows some connection to replayable CCA [8], see Appendix B.

3.3 Summary of Reductions

Now, we can claim our main theorem.

Theorem 1 ($\mathsf{CCA} \Rightarrow \mathsf{HRA}$)**.** *A CCA-secure PRE is also HRA-secure.*

In summary, we obtain the diagram for the relations between security models in Figure. 2.

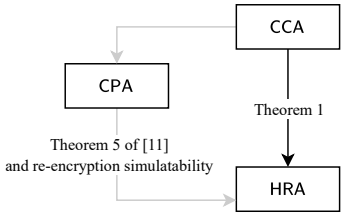

Fig. 2. Summary of our reduction. Our new theorem provides a new reduction that replaces the original reductions from Cohen [10] indicated by gray arrows

4 Collusion Resistant PRE

Collusion resistance (CR) is an entirely different security notion from HRA security. It provides protection for the ciphertext against the adversary who obtained both the delegatee's secret key and the re-encryption key. Honest re-encryption attack security ensures the delegatee (adversarial) can only decrypt the re-encrypted ciphertext.

4.1 Definition

We adapt the definition of collusion-safety from [31] to a selective corruption model.

Definition 5 (Security Game for CR). *Let λ be the security parameter and $\mathcal{A}$ be an oracle Turing machine.* CR *game consists of an execution of $\mathcal{A}$ with the following oracles:*

- *Setup: The public parameters are generated and given to $\mathcal{A}$. A counter* numKeys *is initialized to 0, and the sets* Hon *and* Cor *are initialized to be empty. This oracle is executed first and only once.*
- *Honest Key Generation: Obtain a new key pair* $(\mathsf{pk}_{\mathsf{numKeys}}, \mathsf{sk}_{\mathsf{numKeys}}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$ *and give* $\mathsf{pk}_{\mathsf{numKeys}}$ *to $\mathcal{A}$. The current value of* numKeys *is added to* Hon*, and* numKeys *is incremented.*
- *Corrupted Key Generation: Obtain a new key pair* $(\mathsf{pk}_{\mathsf{numKeys}}, \mathsf{sk}_{\mathsf{numKeys}}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$ *and give the key pair to $\mathcal{A}$. The current value of* numKeys *is added to* Cor*, and* numKeys *is incremented.*
- *Re-encryption Key Generation $\mathcal{O}_{\mathsf{ReKeyGen}}$: On input (i, j) where $i, j \leq$* numKeys*, if $i = j$, output $\perp$. Otherwise return the re-encryption key* $\mathsf{rk}_{i\to j} \leftarrow \mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j)$.
- *Challenge Oracle: On input $(i, \mathsf{m}_0, \mathsf{m}_1)$ where $i \in$* Hon *and $\mathsf{m}_0, \mathsf{m}_1 \in \mathcal{M}$, sample a bit $b \leftarrow \{0, 1\}$ uniformly at random, compute the challenge ciphertext* $\mathsf{ct}^* \leftarrow \mathsf{Enc}_1(\mathsf{pk}_i, \mathsf{m}_b)$*. Return* ct^**. This oracle can only be queried once.*
- *Decision: On input a bit $b' \in \{0, 1\}$, return* win *if $b = b'$.*

The CR *advantage of $\mathcal{A}$ is defined as*

$$\mathsf{Adv}_{\mathsf{cr}}^{\mathcal{A}} = \Pr[\mathsf{win}],$$

where the probability is over the randomness of $\mathcal{A}$ and the oracles in CR *game.*

Remark 4 (Leveled ciphertexts). The challenge oracle returns a first-level ciphertext (cannot be re-encrypted) in the CR game, and all re-encryption key generations are *queryable* to $\mathcal{A}$. This game captures the situation when a corrupted user conspires with the proxy, the first-level ciphertext is still indistinguishable.

4.2 Collusion Attacks

Next, we will show that those schemes are not collusion-resistant. In order to grasp the main idea and facilitate presentation, we ignore the details of the scheme design and settings, such as parameters and dimensions.

PRE from FHE Cohen [10] proved the PRE from fully homomorphic encryption (FHE) [15] with property circuit privacy.

Let $\mathsf{FHE} := (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ be an FHE scheme, the re-encryption key generation algorithm is defined as

$$\mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j) \to \mathsf{rk}_{i\to j} := \mathsf{Enc}(\mathsf{pk}_j, \mathsf{sk}_i) \| \mathsf{pk}_j.$$

In a CR game, the adversary can get the $\mathsf{rk}_{i\to j}$ to some corrupted user j via $\mathcal{O}_{\mathsf{ReKeyGen}}$ and decrypt the ciphertext $\mathsf{Enc}(\mathsf{pk}_j, \mathsf{sk}_i)$ with secret key sk_j to get sk_i. Since this is all secret user i has, all its ciphertexts are decryptable to the adversary.

PRE from LWE Susilo et al. [26] designed an HRA-secure and attribute-based PRE from the learning with errors (LWE) assumption. Different from the other trapdoor-based schemes [12,30,31] (sampling short matrix using the trapdoor), their re-encryption key are a kind of encryption of the secret key.

We briefly introduce the trapdoor generators and related algorithms here. Let $\mathbf{G}$ be the gadget matrix and $\mathbf{T_G}$ be the basis of $\Lambda_q^{\perp}(\mathbf{G})$ [20]:

1. Randomized trapdoor generation algorithm $\mathsf{TrapGen}(n, m, q) \to (\mathbf{A}, \mathbf{T_A})$, where $\mathbf{A}$ is a close-to-uniform matrix in $\mathbb{Z}_q^{n\times m}$, and $\mathbf{T_A}$ is a basis of $\Lambda_q^{\perp}(\mathbf{A})$.
2. Trapdoor delegation algorithm $\mathsf{ExtendRight}(\mathbf{A}, \mathbf{T_A}, \mathbf{B}) \to \mathbf{T}_{(\mathbf{A}|\mathbf{B})}$, where $\mathbf{B}$ is full-rank matrix, and $\mathbf{T}_{(\mathbf{A}|\mathbf{B})}$ is a of $\Lambda_q^{\perp}(\mathbf{A}|\mathbf{B})$.
3. Trapdoor generation from the gadget matrix $\mathsf{ExtendLeft}(\mathbf{A}, \mathbf{G}, \mathbf{T_G}, \mathbf{R}) \to \mathbf{T_M}$, where $\mathbf{M} = (\mathbf{A}|\mathbf{G} + \mathbf{AR})$, $\mathbf{T_M}$ is a basis of $\Lambda_q^{\perp}(\mathbf{M})$.
4. Sampling algorithm $\mathsf{SampleD}(\mathbf{A}, \mathbf{T_A}, \mathbf{U}) \to \mathbf{X}$, where $\mathbf{X}$ satisfies $\mathbf{AX} = \mathbf{U}$.
5. Re-randomizing trapdoor algorithm $\mathsf{RandBasis}(\mathbf{A}, \mathbf{T_A}) \to \mathbf{T'_A}$, where $\mathbf{T'_A}$ a basis of $\Lambda_q^{\perp}(\mathbf{A})$.

For $x \in \mathbb{Z}_q, \mathbf{B} \in \mathbb{Z}_q^{n\times m}, \mathbf{s} \in \mathbb{Z}_q^n$, define

$$E_{\mathbf{s}}(x, \mathbf{B}) := \{(x\mathbf{G} + \mathbf{B})^{\top}\mathbf{s} + \mathbf{e} \in \mathbb{Z}_q^m\}.$$

Let $\mathcal{F} = \{f : \mathbb{Z}_q^{\ell} \to \mathbb{Z}_q\}$ be the key policy function family, $\mathbf{x} = (x_1, x_2, \ldots, x_\ell)$ be the attribute vector, the ciphertext are decryptable using sk_f such that $f(\mathbf{x}) = 0$. The evaluation algorithm for FHE [4] are defined as:

1. $\mathsf{Eval}_{pk}(f, \{\mathbf{B}_i\}_{i=1}^{\ell}) \to \mathbf{B}_f$, where $f \in \mathcal{F}$.
2. $\mathsf{Eval}_{ct}(f, \{x_i, \mathbf{B}_i, \mathbf{ct}_i\}_{i=1}^{\ell}) \to \mathbf{ct}_f$, where $\mathbf{ct}_i \in E_{\mathbf{s}}(x_i, \mathbf{B}_i), \mathbf{ct}_f \in E_{\mathbf{s}}(f(\mathbf{x}), \mathbf{B}_f), \mathbf{B}_f \leftarrow \mathsf{Eval}_{pk}(f, \{\mathbf{B}_i\}_{i=1}^{\ell}), \mathbf{x} = (x_1, \ldots, x_\ell)$.

Let $\mathsf{P2}$ be the power-of-two algorithm takes a matrix $\mathbf{s} \in \mathbb{Z}_q^n$, outputs $(1, 2, \ldots, 2^{\ell})^{\top} \otimes \mathbf{s}$. PRE algorithms $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{ReKeyGen})$ are constructed as follows [26]:

- $\mathsf{Setup}(\lambda, \ell) \to \mathsf{pp}$. Generate a matrix and its trapdoor $(\mathbf{A}_0, \mathbf{T}_{\mathbf{A}_0}) \leftarrow \mathsf{TrapGen}(n, m, q)$; choose $\ell + 1$ random matrices $\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_\ell, \mathbf{U}$; output the public parameters $\mathsf{pp} := \{\mathbf{A}_0, \mathbf{A}_1, \ldots, \mathbf{A}_\ell, \mathbf{U}, \mathbf{G}\}$ and the master key $\mathsf{msk} := \mathbf{T}_{\mathbf{A}_0}$.
- $\mathsf{KeyGen}(\mathsf{msk}, f) \to \mathsf{sk}_f$. Evaluate $\mathbf{A}_f \leftarrow \mathsf{Eval}_{pk}(f, \{\mathbf{A}_i\}_{i=1}^{\ell})$. Compute $\mathbf{T}_{(\mathbf{A}_0|\mathbf{A}_f)} \leftarrow \mathsf{ExtendRight}(\mathbf{A}_0, \mathbf{A}_f, \mathbf{T}_{\mathbf{A}_0})$. Sample $\mathbf{R}_f \leftarrow \mathsf{SampleD}\left([\mathbf{A}_0|\mathbf{A}_f], \mathbf{T}_{(\mathbf{A}_0|\mathbf{A}_f)}, \mathbf{U}\right)$. Output the secret key $\mathsf{sk}_f := \mathbf{R}_f$ for the policy f.
- $\mathsf{ReKeyGen}(\mathsf{sk}_f, f, g) \to \mathsf{rk}_{f\to g}$. Let $\mathbf{R}_f \leftarrow \mathsf{sk}_f$, select an attribute set $\mathbf{y} = (y_1, \ldots, y_\ell)$ such that $g(\mathbf{y}) = 0$. Construct $\mathbf{H_y} := [\mathbf{A}_0 \mid y_1\mathbf{G} + \mathbf{A}_1 \mid \cdots \mid y_\ell\mathbf{G} + \mathbf{A}_\ell]$. Choose uniformly random matrices $\mathbf{R}_1, \mathbf{R}_2, \mathbf{R}_3$. Construct the re-encryption key $\mathsf{rk}_{f\to g} := \begin{bmatrix} (\mathbf{R}_1\mathbf{H_y} + \mathbf{R}_2) & (\mathbf{R}_1\mathbf{U} + \mathbf{R}_3 - \mathsf{P2}(\mathbf{R}_f)) \\ \mathbf{0} & \mathbf{I} \end{bmatrix}$. Output $(\mathsf{rk}_{f\to g}, \mathbf{y})$.

Given $(\mathsf{rk}_{f\to g}, \mathbf{y})$ and $\mathbf{R}_g$, the adversary can "decrypt" the $\mathbf{R}_f$ accordingly. Let the i-th row of $\mathbf{R}_1\mathbf{H_y} + \mathbf{R}_2$ be $\mathbf{r}_{1,i}^{\top}\mathbf{H_y} + \mathbf{r}_{2,i}^{\top}$, transpose it to $\mathbf{H_y}^{\top}\mathbf{r}_{1,1} + \mathbf{r}_{2,1}$, and parse the transpose to $(\mathbf{ct}_{in}, \mathbf{ct}_1, \ldots, \mathbf{ct}_\ell)$. Evaluate $\mathbf{ct}_g \leftarrow \mathsf{Eval}_{ct}(g, \{y_t, \mathbf{A}_t, \mathbf{ct}_t\}_{t=1}^{\ell})$.

Let $\mathbf{ct}'_g \leftarrow [\mathbf{ct}_{in} \mid \mathbf{ct}_g]$. For the i-th row of $\mathbf{R}_1\mathbf{U} + \mathbf{R}_3 - \mathsf{P2}(\mathbf{R}_f)$, let it's transpose be $\mathbf{ct}_{out}$, compute $\mathbf{r}_{f,i} \leftarrow \mathbf{ct}_{out} - \mathbf{R}_g^\top \mathbf{ct}'_g$. Combine all $\mathbf{r}_{f,i}$'s, the adverary obtain $-\mathsf{P2}(\mathbf{R}_f)^\top$, and computes $\mathsf{P2}^{-1}\left(\left(\mathsf{P2}(\mathbf{R}_f)^\top\right)^\top\right)$ to get the full secret key $\mathbf{R}_f$ for policy f (with some rounding for canceling the noise).

PRE from RLWE Cohen et al. [11] designed a homomorphic PRE based on BGV homomorphic encryption. Their re-encryption relies on the key-switching mechanism. The re-encryption key an encryption of the delegator's secret key:

$$\mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j) := \mathsf{Enc}(\mathsf{pk}_j, \mathcal{PW}(\mathsf{sk}_i)).$$

Therefore, the adversary can decrypt and invert the digit decomposition function $\mathcal{PW}$ to get i's secret key.

4.3 Generic Construction

CR can be achieved by a generic construction inspired by [2]. We generate two pairs of keys, one for generating/decrypting the first-level ciphertext and the other one for generating/decrypting the second-level ciphertext. This method isolates the ciphertext from the first-level and second-level, rendering the collusion attack infeasible.

Theorem 2. *Given a CPA-secure single-hop PRE scheme* PRE*, there is a CR and CPA-secure single-hop PRE scheme* PRE′*.*

Proof. Let the CPA-secure single-hop PRE scheme be

$$\mathsf{PRE} := (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{ReKeyGen}, \mathsf{Enc}_1, \mathsf{Enc}_2, \mathsf{ReEnc}, \mathsf{Dec}_1, \mathsf{Dec}_2).$$

We construct the scheme

$$\mathsf{PRE}' := (\mathsf{Setup}', \mathsf{KeyGen}', \mathsf{ReKeyGen}', \mathsf{Enc}'_1, \mathsf{Enc}'_2, \mathsf{ReEnc}', \mathsf{Dec}'_1, \mathsf{Dec}'_2)$$

from PRE:

- $\mathsf{Setup}' := \mathsf{Setup}$.
- $\mathsf{KeyGen}' \rightarrow (\mathsf{pk}_i, \mathsf{sk}_i)$. Run twice KeyGen to get two key pairs $(\mathsf{pk}_i^{(1)}, \mathsf{sk}_i^{(1)}), (\mathsf{pk}_i^{(2)}, \mathsf{sk}_i^{(2)})$, return $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \left((\mathsf{pk}_i^{(1)}, \mathsf{pk}_i^{(2)}), (\mathsf{sk}_i^{(1)}, \mathsf{sk}_i^{(2)})\right)$.
- $\mathsf{ReKeyGen}'(\mathsf{pk}_i, \mathsf{sk}_i, \mathsf{pk}_j) \rightarrow \mathsf{rk}_{i\rightarrow j}$. Parse $\left((\mathsf{pk}_i^{(1)}, \mathsf{pk}_i^{(2)}), (\mathsf{sk}_i^{(1)}, \mathsf{sk}_i^{(2)})\right) \leftarrow (\mathsf{pk}_i, \mathsf{sk}_i)$, $(\mathsf{pk}_j^{(1)}, \mathsf{pk}_j^{(2)}) \leftarrow \mathsf{pk}_j$. Compute $\mathsf{rk}_{i\rightarrow j}^{(2\rightarrow 1)} \leftarrow \mathsf{ReKeyGen}(\mathsf{pk}_i^{(2)}, \mathsf{sk}_i^{(2)}, \mathsf{pk}_j^{(1)})$, and return $\mathsf{rk}_{i\rightarrow j} \leftarrow \mathsf{rk}_{i\rightarrow j}^{(2\rightarrow 1)}$.
- $\mathsf{ReEnc}' := \mathsf{ReEnc}, \mathsf{Enc}'_1 := \mathsf{Enc}_1, \mathsf{Enc}'_2 := \mathsf{Enc}_2, \mathsf{Dec}'_1 := \mathsf{Dec}_1, \mathsf{Dec}'_2 := \mathsf{Dec}_2$.

It follows immediately that, PRE′ is CPA-secure if PRE is CPA-secure.

Next, we reduce the security of CPA to the security of CR. Given a CR adversary $\mathcal{A}$, CPA adversary $\mathcal{B}$ simulates a CR game for it:

- Setup: The same as $\mathcal{B}$'s own CPA setup oracle. $\mathcal{B}$ maintains a local key-value store Keys for key pairs.
- Honest Key Generation: $\mathcal{B}$ calls its CPA honest key generation oracle and gets pk'. Then, $\mathcal{B}$ generates another key pair $(\mathsf{pk}'', \mathsf{sk}'')$ by itself and set $\mathsf{pk} \leftarrow (\mathsf{pk}', \mathsf{pk}''), \mathsf{sk} \leftarrow (\perp, \mathsf{sk}'')$. The current value of $\mathsf{numKeys}$ is added to Hon, $(\mathsf{numKeys}, (\mathsf{pk}, \mathsf{sk}))$ is added to Keys, and $\mathsf{numKeys}$ is incremented. Return $(\mathsf{numKeys}, \mathsf{pk})$.
- Corrupted Key Generation: Similar to honest key generation. $\mathcal{B}$ calls its CPA corrupted key generation oracle and gets $(\mathsf{pk}', \mathsf{sk}')$ and generates $(\mathsf{pk}'', \mathsf{sk}'')$ by itself. Set $\mathsf{pk} \leftarrow (\mathsf{pk}', \mathsf{pk}''), \mathsf{sk} \leftarrow (\mathsf{sk}', \mathsf{sk}'')$. The current value of $\mathsf{numKeys}$ is added to Cor, $(\mathsf{numKeys}, (\mathsf{pk}, \mathsf{sk}))$ is added to Keys. Return $(\mathsf{numKeys}, (\mathsf{pk}, \mathsf{sk}))$.
- $\mathcal{O}_{\mathsf{ReKeyGen}}$: On input (i, j) where $i, j \leq \mathsf{numKeys}$, if $i = j$, output $\perp$. Otherwise, find key pairs of i and j in Keys and parse the keys $\left((\mathsf{pk}_t^{(1)}, \mathsf{pk}_t^{(2)}), (\mathsf{sk}_t^{(1)}, \mathsf{sk}_t^{(2)})\right) \leftarrow (\mathsf{pk}_t, \mathsf{sk}_t)$, where $t \in \{i, j\}$. $\mathcal{B}$ computes the re-encryption key locally by $\mathsf{ReKeyGen}(\mathsf{pk}_i^{(2)}, \mathsf{sk}_i^{(2)}, \mathsf{pk}_j^{(1)})$ and returns the result.
- Challenge Oracle: $\mathcal{B}$ relays the input from $\mathcal{A}$ and the response from CPA's challenge oracle.

Once $\mathcal{A}$ calls the decision oracle, $\mathcal{B}$ outputs what $\mathcal{A}$ inputs. The simulation is perfect. □

Remark 5. CR security is only meaningful for single-hop settings. If the PRE scheme is multi-hop, there is no second encryption algorithm since defining the final delegation of ciphertext is hard. In the email forwarding system [3], where the delegator's ciphertexts are generated by the *sender*. So, this protection enables the sender to decide whether the email sent can be re-encrypted/forwarded to another user.

This construction applies to the HRA-secure schemes [10, 11, 26] since HRA security implies CPA security, rendering them with CR at the cost of doubling the public and secret key sizes.

5 Future Works

The existing generic construction from CPA-secure PRE to CCA-secure PRE [25] is relatively inefficient in both key size and computation, and it is proved under the bounded CCA model. It's unknown if we can build a generic construction from HRA-secure PRE to CCA-secure PRE, although the honest re-encryption oracle in the HRA game [10] is the same as CCA-secure PRE except that the ciphertext can be adaptively chosen [7]. We must somehow enable the ciphertext's self-proved validity or be honestly constructed to provide a CCA secure PRE, where the proxy can reject the invalid ciphertext for re-encryption [22]. Also, there is no CCA secure and post-quantum secure PRE scheme (CCA1-secure PRE of [18] has a flaw in its proof; CCA-secure PRE in [12] uses a non-standard, tag-based CCA security model). Constructing a CCA and post-quantum secure, efficient PRE is a promising future work.

Acknowledgments. This work was partially supported by the XJTLU Research Development Fund under Grant No. RDF-21-02-014; and XJTLU Teaching Development Fund under Grant No. TDF2223-R25-207.

References

1. Ateniese, G., Benson, K., Hohenberger, S.: Key-private proxy re-encryption. In: Topics in Cryptology–CT-RSA 2009: The Cryptographers' Track at the RSA Conference 2009, San Francisco, CA, USA, April 20-24, 2009. Proceedings. pp. 279–294. Springer (2009)
2. Ateniese, G., Fu, K., Green, M., Hohenberger, S.: Improved proxy re-encryption schemes with applications to secure distributed storage. ACM Transactions on Information and System Security (TISSEC) **9**(1), 1–30 (2006)
3. Blaze, M., Bleumer, G., Strauss, M.: Divertible protocols and atomic proxy cryptography. In: International conference on the theory and applications of cryptographic techniques. pp. 127–144. Springer (1998)
4. Boneh, D., Gentry, C., Gorbunov, S., Halevi, S., Nikolaenko, V., Segev, G., Vaikuntanathan, V., Vinayagamurthy, D.: Fully key-homomorphic encryption, arithmetic circuit abe and compact garbled circuits. In: Advances in Cryptology–EUROCRYPT 2014: 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings 33. pp. 533–556. Springer (2014)
5. Bos, J., Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schanck, J.M., Schwabe, P., Seiler, G., Stehlé, D.: Crystals-kyber: a cca-secure module-lattice-based kem. In: 2018 IEEE European Symposium on Security and Privacy (EuroS&P). pp. 353–367. IEEE (2018)
6. Canetti, R., Halevi, S., Katz, J.: Chosen-ciphertext security from identity-based encryption. In: Advances in Cryptology-EUROCRYPT 2004: International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004. Proceedings 23. pp. 207–222. Springer (2004)
7. Canetti, R., Hohenberger, S.: Chosen-ciphertext secure proxy re-encryption. In: Proceedings of the 14th ACM conference on Computer and communications security. pp. 185–194 (2007)
8. Canetti, R., Krawczyk, H., Nielsen, J.B.: Relaxing chosen-ciphertext security. In: Advances in Cryptology-CRYPTO 2003: 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003. Proceedings 23. pp. 565–582. Springer (2003)
9. Cohen, A.: What about bob? the inadequacy of CPA security for proxy reencryption. Cryptology ePrint Archive, Paper 2017/785 (2017), `https://eprint.iacr.org/2017/785`
10. Cohen, A.: What about bob? the inadequacy of cpa security for proxy reencryption. In: IACR International Workshop on Public Key Cryptography. pp. 287–316. Springer (2019)
11. Cohen, A., Cousins, D.B., Genise, N., Kline, E., Polyakov, Y., RV, S.: Hra-secure homomorphic lattice-based proxy re-encryption with tight security. Cryptology ePrint Archive (2024)
12. Fan, X., Liu, F.H.: Proxy re-encryption and re-signatures from lattices. In: Applied Cryptography and Network Security: 17th International Conference, ACNS 2019, Bogota, Colombia, June 5–7, 2019, Proceedings 17. pp. 363–382. Springer (2019)

13. Fuchsbauer, G., Kamath, C., Klein, K., Pietrzak, K.: Adaptively secure proxy re-encryption. In: IACR International Workshop on Public Key Cryptography. pp. 317–346. Springer (2019)
14. Fujisaki, E., Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. Journal of cryptology **26**, 80–101 (2013)
15. Gentry, C.: A fully homomorphic encryption scheme. Stanford university (2009)
16. Ivan, A.A., Dodis, Y.: Proxy cryptography revisited. In: NDSS. Citeseer (2003)
17. Jafargholi, Z., Kamath, C., Klein, K., Komargodski, I., Pietrzak, K., Wichs, D.: Be adaptive, avoid overcommitting. In: Annual International Cryptology Conference. pp. 133–163. Springer (2017)
18. Kirshanova, E.: Proxy re-encryption from lattices. In: Public-Key Cryptography–PKC 2014: 17th International Conference on Practice and Theory in Public-Key Cryptography, Buenos Aires, Argentina, March 26-28, 2014. Proceedings 17. pp. 77–94. Springer (2014)
19. Miao, P., Patranabis, S., Watson, G.: Unidirectional updatable encryption and proxy re-encryption from ddh. In: IACR International Conference on Public-Key Cryptography. pp. 368–398. Springer (2023)
20. Micciancio, D., Peikert, C.: Trapdoors for lattices: Simpler, tighter, faster, smaller. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 700–718. Springer (2012)
21. Nunez, D., Agudo, I., Lopez, J.: A parametric family of attack models for proxy re-encryption. In: 2015 IEEE 28th Computer Security Foundations Symposium. pp. 290–301. IEEE (2015)
22. Nunez, D., Agudo, I., Lopez, J.: On the application of generic cca-secure transformations to proxy re-encryption. Security and Communication Networks **9**(12), 1769–1785 (2016)
23. Paul, A., Selvi, S.S.D., Rangan, C.P.: Efficient attribute-based proxy re-encryption with constant size ciphertexts. In: Progress in Cryptology–INDOCRYPT 2020: 21st International Conference on Cryptology in India, Bangalore, India, December 13–16, 2020, Proceedings 21. pp. 644–665. Springer (2020)
24. Polyakov, Y., Rohloff, K., Sahu, G., Vaikuntanathan, V.: Fast proxy re-encryption for publish/subscribe systems. ACM Transactions on Privacy and Security (TOPS) **20**(4), 1–31 (2017)
25. Sato, S., Shikata, J.: Bounded cca secure proxy re-encryption based on kyber. Cryptology ePrint Archive (2024)
26. Susilo, W., Dutta, P., Duong, D.H., Roy, P.S.: Lattice-based hra-secure attribute-based proxy re-encryption in standard model. In: European Symposium on Research in Computer Security. pp. 169–191. Springer (2021)
27. Wan, X., Wang, Y., Xue, H., Wang, M.: Unbounded multi-hop proxy re-encryption with HRA security: An LWE-based optimization. Cryptology ePrint Archive, Paper 2025/656 (2025), `https://eprint.iacr.org/2025/656`
28. Zhao, F., Wang, H., Weng, J.: Constant-size unbounded multi-hop fully homomorphic proxy re-encryption from lattices. In: European Symposium on Research in Computer Security. pp. 238–258. Springer (2024)
29. Zhao, F., Weng, J., Xie, W., Li, M., Weng, J.: Hra-secure attribute-based threshold proxy re-encryption from lattices. Information Sciences **655**, 119900 (2024)
30. Zhou, Y., Liu, S., Han, S.: Multi-hop fine-grained proxy re-encryption. In: IACR International Conference on Public-Key Cryptography. pp. 161–192. Springer (2024)
31. Zhou, Y., Liu, S., Han, S., Zhang, H.: Fine-grained proxy re-encryption: Definitions and constructions from lwe. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 199–231. Springer (2023)

A Transform Machine for Adaptive Key Generation

We show how to construct a $\mathcal{B}$ with two-phase CCA games to simulate an un-phased CCA game to any adversary $\mathcal{A}$ by replicating the state of CCA. More specifically, $\mathcal{B}$ maintains two copies of a phased CCA machine, CCA_0 and CCA_1, with the same state and randomness in the beginning. $\mathcal{B}$ will simulate an un-phased CCA game upCCA that is identical to CCA, but the adversary is allowed to query user key generation oracles adaptively. Then, $\mathcal{B}$ starts HRA_0 machine and $\mathcal{A}$ to proceed the reduction. When $\mathcal{B}$ receives the first non-key generation query, it copies the state of CCA_0 to CCA_1, CCA_0 enters phase 2, and deals with the query on CCA_0. Afterward, when the next key generation query is received, $\mathcal{B}$ generates this key on CCA_1 and keeps dealing with key generation queries on this machine. When the next non-key generation query is received, $\mathcal{B}$ copies the state of CCA_1 to CCA_0 (the previous state will be overwritten), executes all non-key generation queries on CCA_1, and keeps dealing with key generation queries on this machine. When the next key generation query is received, $\mathcal{B}$ switches to CCA_0 and proceeds with a similar operation. We assume the oracles are with independent randomness as input, rendering this switching feasible. The Fig. 3 shows the state machine transition process.

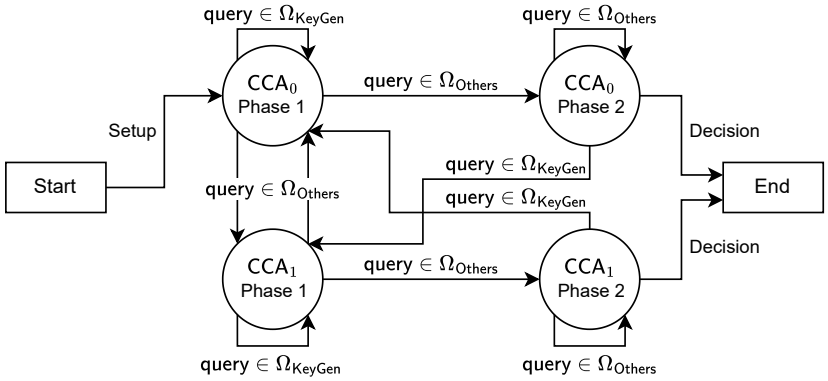

Fig. 3. Simulation of un-phased upCCA with two-phased CCA. In particular, query denotes the type of oracle query from a upCCA adversary, Ω_{KeyGen} is a set comprised of honest and corrupted key generation oracle, and $\Omega_{\mathsf{Others}} = \{\mathcal{O}_{\mathsf{ReKeyGen}}, \mathcal{O}_{\mathsf{ReEnc}}, \mathcal{O}_{\mathsf{Challenge}}\}$. Only one machine is active at a time, and every query in Ω_{Others} are loaded in a ordered local store $\mathcal{S}_{\mathsf{Others}}$. The replication between CCA_0 Phase 1 and CCA_1 Phase 1 is triggered by a query $\mathsf{query} \in \Omega_{\mathsf{Others}}$; the replica will proceed to the next phase and execute queries that were previously loaded in $\mathcal{S}_{\mathsf{Others}}$, and finally execute the trigger query. When any machine is in phase 2, any key generation query will cause it to terminate and continue key generation in phase 1 on another machine

B Replayable CCA

The definition of the CCA game for PRE [7] was inspired by the relaxation of CCA security for PKE [8], so-called *replayable* CCA. In a relaxed CCA game, the decryption oracle rejects any ciphertext decryption if the obtained plaintext is in $\{\mathsf{m}_0, \mathsf{m}_1\}$. This captures the intuition that the ability to generate different ciphertexts that decrypt to the same values as a given ciphertext should not help the adversary win the game. In this aspect, we know that relaxed CCA is strictly weaker than CCA. However, relaxed CCA security is adequate for most typical encryption applications, e.g., non-interactive secure communication, key exchange and authenticators, and hybrid encryption. The main reason behind the usage is that the security of those applications stems from the secrecy of the *plaintext* but not the *ciphertext* strings. For the common applications of HRA-secure PRE, e.g., encrypted email forwarding, allowing multi-ciphertext generation on the same message harms nothing if the encryption algorithm is probabilistic (a ciphertext from a deterministic encryption algorithm is distinguishable in even CPA game). This is more of a model difference and does not reflect a real weakness or attack. Our proof also confirms that this relaxation does not pose a real threat because they can be transformed into each other through a perfect simulation.