Collusion-Safe Proxy Re-Encryption

Haotian Yin[1][0000−0002−5088−6116], Jie Zhang[1][0000−0001−7415−7869], Wanxin Li[1][0000−0002−4482−4445], Yuji Dong[2][0000−0001−6715−7588], Eng Gee Lim[1][0000−0003−0199−7386], Dominik Wojtczak[3][0000−0001−5560−0546]

[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 is a cryptographic scheme that allows a delegator (user i) to delegate its decryption right to a delegatee (user j) via a proxy. A critical concern is the security against collusion between the proxy and the delegatee. In this case, the adversary obtains the secret key of the delegatee, sk_j, and the re-encryption key, $\mathsf{rk}_{i\to j}$. The master secret security is first formalised by Ateniese et al. (NDSS 2005) to capture the secrecy of i's secret key during collusion. This notion was further formalised by Zhou et al. (ASIACRYPT 2023) as the indistinguishability of re-encrypted ciphertext against chosen-message attacks, called collusion safety (CS), which implies the master secret security. In this paper, we demonstrate that a PRE scheme fails to satisfy CS as claimed, and many other schemes also fail to meet this requirement. Then, we propose a generic construction to achieve CS at the cost of doubling the key size from the IND-CPA secure PRE, offering significantly improved generality and efficiency than the existing technique by secret sharing.

Keywords: Proxy re-encryption · Collusion safety · Master secret secrecy.

1 Introduction

Proxy re-encryption (PRE) is a cryptographic primitive that enables a delegator to delegate the decryption right to a delegatee by generating a re-encryption key, while the proxy performing the re-encryption learns nothing about the underlying plaintext. This capability is particularly useful in scenarios such as secure data sharing, delegated access control, and distributed storage systems, where data owners need to grant access to different parties without exposing their keys or re-encrypting the data themselves.

Ateniese et al. [5] were the first to formalise the notion of PRE schemes, along with the standard security requirement of indistinguishability under chosen plaintext attacks (IND-CPA). They also introduced a vital security notion called master secret security (MSS), which ensures the secrecy of the delegator's secret key even in the presence of collusion between the proxy and the delegatee. MSS requires that given the re-encryption key, $\mathsf{rk}_{i\to j}$, and the secret key of the delegatee, sk_j, the adversary is unable to compute the secret key of the delegator, sk_i.

Direction. The scheme proposed by Ateniese et al. [5] is *unidirectional*, meaning the proxy can only re-encrypt ciphertexts from a delegator to a delegatee, but not in the reverse direction, given a specific re-encryption key. This property is useful in scenarios where the delegator does not have access to the delegatee's secret key, as the delegation is one-way. This also avoids the complexity of bidirectional setting [9], where the re-encryption key generation algorithm requires both the delegator's and the delegatee's secret keys. In contrast, unidirectional PRE only requires the delegator's secret key and the delegatee's public key, so only the delegator needs to be *online*. While bidirectional PRE may be suitable in cases where reversing the direction of re-encryption poses no security risk, in general, a bidirectional PRE can be derived from a unidirectional one. The latter is more restrictive and preferable for broader applications. Therefore, this paper focuses exclusively on unidirectional PRE schemes.

Hops. Another important aspect of PRE schemes is the number of times a ciphertext can be re-encrypted. In the pioneering work of Ateniese et al. [5], ciphertexts can be re-encrypted at most once. In their construction, re-encryption maps a ciphertext from one group to another one using a bilinear mapping. This procedure is not invertible (by the assumption of bilinear hard problems, which also implies unidirectionality) and cannot be applied again (single-hop). In contrast, multi-hop PRE schemes allow ciphertexts to be re-encrypted multiple times, potentially an unbounded number of times.

1.1 Master Secret Security

It is important to note that re-encryption allows an MSS adversary to transform the delegator's ciphertext and subsequently decrypt it. This naturally raises the question: if collusion inherently enables decryption, why it remains necessary to protect the delegator's secret key? The answer lies in application scenarios where the system directly accepts and processes re-encrypted ciphertexts that are not subject to further re-encryption. As long as the delegator's secret key remains confidential, these ciphertexts remain secure even in the presence of collusion. In such contexts, MSS enables more fine-grained access control.

This distinction gives rise to two levels of decryption rights: one for decrypting *re-encryptable* ciphertexts and another for decrypting all ciphertexts. Freshly generated ciphertexts are classified as first-level, and re-encrypted ciphertexts as

second-level. In a unidirectional scheme, a re-encryption key can only convert first-level ciphertexts into second-level ones. Consequently, collusion compromises only the security of the delegator's first-level ciphertexts, leaving second-level ciphertexts unaffected.

1.2 Applications

Consider an email forwarding system: Alice delegates her decryption rights to Bob via a proxy while she is on vacation. While most emails can be forwarded (first-level ciphertexts), specific sensitive messages are encrypted as second-level ciphertexts to prevent delegation. For instance, a sensitive but non-urgent email may fall into this category — the sender wants Alice to handle it personally, not her delegatee Bob. Without CS, collusion between Bob and the proxy could expose these messages.

Similar requirements arise in other domains. In healthcare data sharing, a hospital may allow a proxy to forward patient records to another institution. Still, specific confidential reports must remain inaccessible, even in the event of collusion or unauthorised access. Likewise, in enterprise document delegation, an employee may authorise a proxy to share project files with an external contractor while ensuring that confidential documents cannot be decrypted beyond the intended scope. These scenarios illustrate that CS is critical for fine-grained access control in single-hop PRE systems.

1.3 Collusion Safety

Zhou et al. [35] proposed a fine-grained PRE scheme equipped with a stronger security notion known as collusion safety (CS). This notion ensures the indistinguishability of second-level ciphertexts even when the proxy and the delegatee collude. More formally, they defined a security game analogous to IND-CPA, in which the adversary is given multiple first-level/second-level ciphertexts and the re-encryption key, and must distinguish between the second-level ciphertexts of two chosen messages, m_0 or m_1 selected by itself.

It is straightforward to observe that CS implies MSS: if the adversary possesses the delegator's secret key, it can decrypt all ciphertexts and trivially win the CS game. Therefore, CS is stronger than MSS. Intuitively, in a unidirectional PRE scheme, the delegator provides its secret key as input to the re-encryption key generation algorithm, while the delegatee only shares its public key. This is reasonable to protect the delegator's secret key during the re-encryption and against possible collusion. The formal definition of CS is provided in Section 2.

Table 1 presents a survey of various PRE schemes, highlighting their collusion resistance, underlying hardness assumptions, hop count, and re-encryption techniques. We excluded schemes that explicitly state they do not address collusion safety, for example, [3], [25], and [32]. Although FL19 [15] did not explicitly claim CS, its re-encryption key generation algorithm is a variant of Kir14's. Notably, all of these schemes are *single-hop*.

Scheme	Assumption	Hop	Technique	Collusion Resistance
AFGH06 [5]	DBDH	Single-hop	Bilinear pairing	MSS
HRSV07 [20]	SDHI and DL	Single-hop	Obfuscator	MSS
CCV12 [11]	SXDH	Single-hop	Obfuscator	MSS
Kri14 [21]	LWE	Single-hop	Preimage sampling	MSS
FL19 [15]	LWE	Single-hop	Preimage sampling	MSS
DSDR21 [14]	LWE	Single-hop	Encrypting secret key	×
ZLHZ23 [35]	LWE	Single-hop	Preimage sampling	CS
ZWXLW24 [34]	LWE	Single-hop	Encrypting secret key	MSS*

Table 1. PRE schemes and their properties. Items marked with an asterisk (*) have CS with respect to some threshold settings, i.e., the scheme satisfies MSS if the number of corrupted proxies is less than the predefined threshold t. DBDH stands for decisional bilinear Diffie-Hellman. SDHI stands for strong Diffie-Hellman Indistinguishability. DL stands for decision linear. SXDH stands for symmetric external Diffie-Hellman. LWE stands for Learning with Errors.

1.4 Collusion Non-Safe Schemes

DSDR21 [14] is an identity-based PRE scheme that claims to be collusion-resistant. However, we were unable to find any formal proof or definition supporting its resistance against collusion. Therefore, we evaluate its security using the notion of MSS, which CS implies. The scheme's re-encryption key is constructed as follows:

$$\mathsf{rk}_{i\to j} := \begin{bmatrix} \mathbf{R}_1\mathbf{A}_{id_j} & \mathbf{R}_1\mathbf{u} + \mathbf{r}_2 - \mathsf{P2}(\mathbf{x}_{id_i}) \\ \mathbf{0}_{1\times m} & 1 \end{bmatrix} \in \mathbb{Z}_q^{(mk+1)\times(m+1)},$$

where $\mathbf{R}_1$ and $\mathbf{r}_2$ are small noise terms, $\mathbf{A}_{id_j}$ is the public identity matrix of user id_j, $\mathbf{u}$ is the master public key, and $\mathsf{P2}(\mathbf{x}_{id_i})$ is the binary expansion of user id_i's secret key $\mathbf{x}_{id_i}$. Given a binary decomposition of $\mathbf{x}$, $\mathsf{BD}(\mathbf{x})$, it holds that $\mathsf{BD}(\mathbf{x})^t \cdot \mathsf{P2}(\mathbf{y}) = \mathbf{x}^t\mathbf{y}$ [7]. Additionally, the secret key satisfies $\mathbf{A}_{id}\mathbf{x}_{id} = \mathbf{u}$. With access to the secret key $\mathbf{x}_{id_j}$ and the re-encryption key $\mathsf{rk}_{i\to j}$, an adversary can compute

$$\mathsf{rk}_{i\to j} \cdot \begin{bmatrix} \mathbf{x}_{id_j} \\ -1 \end{bmatrix} = \begin{bmatrix} \mathsf{P2}(\mathbf{x}_{id_i}) - \mathbf{r}_2 \\ -1 \end{bmatrix}.$$

Since $\mathbf{r}_2$ is small, the adversary can efficiently recover $\mathbf{x}_{id_i}$ from the above result. Therefore, this scheme fails to satisfy MSS. A detailed analysis is provided in Section 3.

We observe that this "encrypting-secret-key" paradigm also appears in other works, although they did not claim to achieve MSS or CS, e.g., [27,23,33,28]. A similar collusion attack can be conducted against these schemes.

FHE-based PREs. Another widely adopted construction is based on fully homomorphic encryption (FHE), initially proposed by Gentry [18]. In this approach, the re-encryption key generation is simply an encryption of the delegator's secret key under the delegatee's public key. As a result, such schemes do

not satisfy MSS or CS. In essence, these schemes fail to decouple the right to decrypt re-encryptable ciphertexts from the right to decrypt all ciphertexts.

Cohen [12] introduced the indistinguishability against honest re-encryption attacks (IND-HRA) model as a refinement of the IND-CPA model. In the same work, it was demonstrated that if the underlying FHE scheme is *circuit-private*, then the corresponding PRE scheme achieves IND-HRA security. This paradigm was further explored in a subsequent work [13].

It is important to note that IND-HRA is orthogonal to MSS and CS. While IND-HRA requires the indistinguishability of first-level ciphertexts given independently generated first- and second-level ciphertexts, CS focuses on the indistinguishability of second-level ciphertexts in the presence of the corresponding re-encryption key and the delegatee's secret key.

Mitigating Collusion. ZWXLW24 [34] proposed a threshold PRE that mitigates collusion by splitting the re-encryption key. Their construction is proven to be secure as long as the number of colluding proxies remains below a predefined threshold t. While this approach reduces the risk of collusion, it also increases the complexity of system deployment, as it requires at least t proxies to perform re-encryption collaboratively.

In contrast, existing lattice-based constructions using preimage sampling [21,35] can achieve collusion resistance without relying on multiple proxies.

1.5 Main Observation

The formalisation of CS by Zhou et al. [35] defines it as the IND-CPA security of second-level ciphertexts in single-hop settings. More precisely, CS protects second-level ciphertexts that are *not* intended to be decryptable by delegatees. Therefore, CS becomes relevant only when the application system directly accepts second-level ciphertexts. If all ciphertexts generated by delegators or other senders are first-level, and the proxy merely re-encrypts them for valid delegatees, then CS is unnecessary, as all ciphertexts are inherently decryptable by design.

These examples in Section 1.2 highlight a application-level limitation of CS: CS is only meaningful in a *single-hop setting*. If a ciphertext can be re-encrypted multiple times (possibly up to a bounded number of hops), then only the final-hop ciphertext (the one that cannot be re-encrypted further) can be protected. This notion is somewhat unnatural. In addition, if the scheme has an unbounded hop, one could not even define the security of the final hop. As shown in Table 1, CS in single-hop settings appears to be the default assumption among scheme designers. We therefore advocate for explicitly stating the conditions under which CS applies, namely when the system directly processes second-level ciphertexts and the PRE scheme is single-hop; in this case, CS matters.

Generic Construction. Building on the above observations and the constructions for CS and MSS [5,35], we propose a generic method to upgrade a PRE

scheme that lacks CS into one that satisfies it. The core idea is to "double" the key generation process: instead of producing a single key pair $(\mathsf{sk}, \mathsf{pk})$, the algorithm outputs two pairs $((\mathsf{sk}_1, \mathsf{sk}_2), (\mathsf{pk}_1, \mathsf{pk}_2))$.

Considering re-encryption from $(\mathsf{pk}_1, \mathsf{pk}_2)$ to $(\mathsf{pk}'_1, \mathsf{pk}'_2)$, our construction re-encrypts the ciphertext under pk_1 to the one under pk'_2. In this construction, first-level ciphertexts are encrypted under pk_1, while second-level ciphertexts are encrypted under pk_2. As a result, any collusion between the proxy and the delegatee can only compromise the security of pk_1, leaving pk_2 unaffected.

This construction doubles the key length but remains more efficient than approaches that split the re-encryption key and rely on multiple proxies [34]. Such threshold-based schemes incur bandwidth and computational costs that scale linearly with the threshold parameter t.

In comparison to preimage sampling techniques [21,35], our construction is generic and applicable to any IND-CPA-secure PRE schemes.

A formal definition and security proof of our generic construction are provided in Section 4.

2 Preliminaries

This section introduces the definitions and security notions related to PRE.

2.1 Notations

Let $n \in \mathbb{N}$, and define $[n] = \{1, 2, \ldots, n\}$. Let $\lambda \in \mathbb{N}$ be the security parameter. For a finite set S, sampling an element s uniformly from S is denoted by $s \leftarrow \mathsf{S}$. We use $\perp$ to indicate that an algorithm terminates with an error.

An algorithm $\mathcal{A}$ is said to run in probabilistic polynomial time (PPT) if the running time is polynomial in the security parameter λ (possibly implicitly). A function $\mathsf{negl}(\cdot)$ is called negligible if there exists a positive integer N such that for all $n > N$, it holds that $|\mathsf{negl}(n)| < 1/\mathsf{poly}(n)$ for every positive polynomial $\mathsf{poly}(\cdot)$. A probability is overwhelming (in λ) if it is at least $1 - \mathsf{negl}(\lambda)$.

2.2 Proxy Re-Encryption

We begin by defining the syntax of a single-hop PRE scheme, following the formalisation by [5].

Definition 1 (Single-hop PRE scheme). *A single-hop PRE scheme* PRE *consists of a tuple of algorithms* $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{ReKeyGen}, \mathsf{Enc}, \mathsf{ReEnc}, \mathsf{Dec})$*, defined as follows:*

- $\mathsf{Setup}(1^\lambda) \to \mathsf{pp}$*. Given the security parameter* 1^λ*, the setup algorithm outputs public parameters* pp*, which serve as implicit input to the remaining algorithms.*
- $\mathsf{KeyGen} \to (\mathsf{pk}_i, \mathsf{sk}_i)$*. The key generation algorithm outputs a public/secret key for user* i*.*

- $\mathsf{ReKeyGen}(\mathsf{pk}_i, \mathsf{sk}_i, \mathsf{pk}_j) \rightarrow \mathsf{rk}_{i \rightarrow j}$. *Given user i's key pair* $(\mathsf{pk}_i, \mathsf{sk}_i)$ *and user j's public key* pk_j*, this algorithm outputs a re-encryption key* $\mathsf{rk}_{i \rightarrow j}$.
- $\mathsf{Enc}(\ell, \mathsf{pk}_i, \mathsf{m}) \rightarrow \mathsf{ct}_i$. *On input a level indicator* $\ell \in \{1, 2\}$*, a public key* pk_i*, and a message* $\mathsf{m} \in \mathcal{M}$*, the encryption algorithm outputs* ct_i*, a* first-level *ciphertext if* $\ell = 1$ *or* second-level *ciphertext* ct_i *if* $\ell = 2$.
- $\mathsf{ReEnc}(\mathsf{rk}_{i \rightarrow j}, \mathsf{ct}_i) \rightarrow \mathsf{ct}_j / \perp$. *Given a re-encryption key* $\mathsf{rk}_{i \rightarrow j}$ *and a first-level ciphertext* ct_i*, the algorithm outputs a* second-level *ciphertext* ct_j *or the error symbol* $\perp$.
- $\mathsf{Dec}(\ell, \mathsf{sk}_i, \mathsf{ct}_i) \rightarrow \mathsf{m} / \perp$. *Given level indicator* $\ell \in \{1, 2\}$*, a secret key* sk_i*, and a ciphertext* ct_i*, the algorithm outputs a message* $\mathsf{m} \in \mathcal{M}$ *or the error symbol* $\perp$.

Note that this definition of ciphertext follows the formalisation of Zhou et al. [35], where first-level ciphertexts can be re-encrypted (upgraded) to second-level ones.[4]

Definition 2 (Correctness of single-hop PRE). *A proxy re-encryption scheme* PRE *is correct with respect to message space* $\mathcal{M}$ *if for all* $\lambda \in \mathbb{N}, \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$*, and* $\mathsf{m} \in \mathcal{M}$*, such that:*

- *for all* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$:

$$\mathsf{Dec}(\ell, \mathsf{sk}, \mathsf{Enc}(\ell, \mathsf{pk}, \mathsf{m})) = \mathsf{m},$$

  for $\ell \in \{1, 2\}$*, with overwhelming probability.*
- *for all* $(\mathsf{pk}_i, \mathsf{sk}_i), (\mathsf{pk}_j, \mathsf{sk}_j) \leftarrow \mathsf{KeyGen}(\mathsf{pp}), \mathsf{rk}_{i \rightarrow j} \leftarrow \mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j)$:

$$\mathsf{Dec}(2, \mathsf{sk}_j, \mathsf{ReEnc}(\mathsf{rk}_{i \rightarrow j}, \mathsf{Enc}(1, \mathsf{pk}_i, \mathsf{m}))) = \mathsf{m},$$

  with overwhelming probability.

2.3 Collusion Safety

We adopt the definition of CS security game from Zhou et al. [35].

Definition 3 (CS Game). *Let* λ *be the security parameter and* $\mathcal{A}$ *be a PPT machine. The CS game is defined as an interaction between* $\mathcal{A}$ *and a challenger who provides the following oracles, which may be queried in any order, subject to the constraints below:*
Phase 1:

- *Setup: Generate the public parameters* pp *and provide them to* $\mathcal{A}$*. Initialise a counter* $\mathsf{numKeys} \leftarrow 0$*, and two sets:* Hon *(honest users) and* Cor *(corrupted users), both initially empty.*
- *Honest Key Generation: Generate a key pair* $(\mathsf{pk}_{\mathsf{numKeys}}, \mathsf{sk}_{\mathsf{numKeys}}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$*, return* $\mathsf{pk}_{\mathsf{numKeys}}$ *to* $\mathcal{A}$*, add* numKeys *to* Hon*, and increment* numKeys.

[4] This differs from the naming convention used by Ateniese et al. [5], although the definitions are essentially equivalent.

– *Corrupted Key Generation: Generate a key pair* $(\mathsf{pk}_{\mathsf{numKeys}}, \mathsf{sk}_{\mathsf{numKeys}}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$*, return* $(\mathsf{pk}_{\mathsf{numKeys}}, \mathsf{sk}_{\mathsf{numKeys}})$ *to* $\mathcal{A}$*, add* numKeys *to* Cor*, and increment* numKeys.

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* $i = j$*, output* $\perp$*; otherwise, return* $\mathsf{rk}_{i\to j}$.
– *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* $\mathsf{ct}^* \leftarrow \mathsf{Enc}(2, \mathsf{pk}_i, \mathsf{m}_b)$*, and return* ct^**. This oracle may be queried at most once.*

Phase 3:

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

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

$$\mathsf{Adv}^{cs}_{pre}(\mathcal{A}) := |\Pr[\mathsf{win}] - 1/2|\,,$$

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

The definition of MSS can be formalised analogously (see Appendix A for details).

2.4 Chosen Plaintext Attack

Indistinguishability against chosen plaintext attacks (IND-CPA) for PRE is defined as [12,5].

Definition 4 (IND-CPA Game). *Let* λ *be the security parameter and* $\mathcal{A}$ *be a PPT adversary. The IND-CPA game is defined as an interaction between* $\mathcal{A}$ *and a challenger who provides the following oracles, which may be queried in any order, subject to the constraints below:*
Phase 1:

– *Setup: Generate the public parameters* pp *and provide them to* $\mathcal{A}$*. Initialise a counter* $\mathsf{numKeys} \leftarrow 0$*, and two sets:* Hon *(honest users) and* Cor *(corrupted users), both initially empty.*
– *Honest Key Generation: Generate a key pair* $(\mathsf{pk}_{\mathsf{numKeys}}, \mathsf{sk}_{\mathsf{numKeys}}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$*, return* $\mathsf{pk}_{\mathsf{numKeys}}$ *to* $\mathcal{A}$*, add* numKeys *to* Hon*, and increment* numKeys.
– *Corrupted Key Generation: Generate a key pair* $(\mathsf{pk}_{\mathsf{numKeys}}, \mathsf{sk}_{\mathsf{numKeys}}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$*, return* $(\mathsf{pk}_{\mathsf{numKeys}}, \mathsf{sk}_{\mathsf{numKeys}})$ *to* $\mathcal{A}$*, add* numKeys *to* Cor*, and increment* numKeys.

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* $\mathsf{rk}_{i \to j}$.
- *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* $\mathsf{ct}^* \leftarrow \mathsf{Enc}(1, \mathsf{pk}_i, \mathsf{m}_b)$, *and return* ct^*. *This oracle may be queried at most once.*

Phase 3:

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

The IND-CPA advantage of $\mathcal{A}$ *is defined as*

$$\mathsf{Adv}_{pre}^{ind\text{-}cpa}(\mathcal{A}) := |\Pr[\mathsf{win}] - 1/2| \,,$$

where the probability is taken over the randomness of $\mathcal{A}$ *and the oracles in the IND-CPA game.*

3 Collusion Non-Safe Schemes

We identify many PRE schemes that are not collusion-safe and illustrate the main attack idea here.

3.1 PRE from FHE

Cohen [12] proposed the IND-HRA-secure PRE from fully homomorphic encryption (FHE) [18], assuming circuit privacy. This construction defines the re-encryption key as an encryption of the delegator's secret key under the delegatee's public key. Consequently, such a scheme cannot provide CS.

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 CS game, the adversary can obtain the $\mathsf{rk}_{i \to j}$ to some corrupted user j via the oracle $\mathcal{O}_{\mathsf{ReKeyGen}}$, and decrypt the ciphertext $\mathsf{Enc}(\mathsf{pk}_j, \mathsf{sk}_i)$ using sk_j to recover sk_i. Since any ciphertext re-encrypted to the user i is decryptable using sk_i, the adversary can trivially decrypt any challenge ciphertext and win the CS game with overwhelming probability.[5]

[5] This probability refers to the correctness of decryption. Any reasonable decryption algorithm should be overwhelmingly correct (with probability $1 - \mathsf{negl}(\lambda)$ for some negligible polynomial $\mathsf{negl}(\cdot)$).

3.2 PRE from LWE

Dutta et al. [14] proposed an identity-based PRE scheme, claiming it to be collusion-resistant. However, we find that the scheme fails to satisfy collusion safety (CS) and, therefore, fails to achieve the level of security asserted. For background on lattice-based cryptography, refer to Appendix B.

To encode identities, the scheme employs a full-rank difference (FRD) map as introduced in [1]:

$$\mathsf{FRD} : \mathbb{Z}_q^n \to \mathbb{Z}_q^{n\times n}, id \to \mathbf{H}_{id},$$

satisfying the following properties:

- For all $id \in \mathbb{Z}_q^n \setminus \{\mathbf{0}\}$, the matrix $\mathbf{H}_{id}$ is full rank;
- For all distinct $id, id' \in \mathbb{Z}_q^n$, the matrix $\mathbf{H}_{id} - \mathbf{H}_{id'}$ is full rank;
- The map FRD is computable in polynomial time.

Let $k = \lceil \log q \rceil$. To enable re-encryption, their re-encryption key generation uses techniques similar to the key switching [8]:

- $\mathsf{BD}(\mathbf{x} \in \mathbb{Z}_q^n, q)$ decomposes $\mathbf{x}$ into its bit representation. That is, write $\mathbf{x} = \sum_{j=0}^{k-1} 2^j \cdot \mathbf{u}_j$, where each $\mathbf{u}_j \in \mathbb{Z}_2^n$, and output $(\mathbf{u}_0, \mathbf{u}_1, \ldots, \mathbf{u}_{k-1}) \in \mathbb{Z}_2^{nk}$.
- $\mathsf{P2}(\mathbf{x} \in \mathbb{Z}_q^n, q)$ outputs the vector $(2^0 \cdot \mathbf{x}, 2^1 \cdot \mathbf{x}, \ldots, 2^{k-1} \cdot \mathbf{x}) \in \mathbb{Z}_q^{nk}$.

By [6, Lemma 2], it holds that

$$\langle \mathsf{BD}(\mathbf{x}, q), \mathsf{P2}(\mathbf{y}, q) \rangle = \langle \mathbf{x}, \mathbf{y} \rangle$$

for equal-length vectors $\mathbf{x}$ and $\mathbf{y}$. For simplicity, we omit the parameter q in the remainder of this section.

We briefly outline the algorithms relevant to our analysis. Encryption, decryption, and re-encryption procedures are omitted.

- $\mathsf{Setup}(1^\lambda)$: Sample $\bar{\mathbf{A}} \leftarrow \mathbb{Z}_q^{n\times \bar{m}}, \mathbf{R} \leftarrow D$, and construct $\mathbf{A} := [\bar{\mathbf{A}} \mid -\bar{\mathbf{A}}\mathbf{R}] \in \mathbb{Z}_q^{n\times m}$, where $m := \bar{m} + nk$. Sample $\mathbf{u} \leftarrow \mathbb{Z}_q^n$ uniformly at random. Output the public parameter $\mathsf{pp} = (\mathbf{A}, \mathbf{u})$; the master secret key is $\mathsf{msk} := \mathbf{R}$.
- $\mathsf{Extract}(\mathsf{msk}, id)$: Construct $\mathbf{A}_{id} := [\bar{\mathbf{A}} \mid -\bar{\mathbf{A}}\mathbf{R} + \mathbf{H}_{id}\mathbf{G}] \in \mathbb{Z}_q^{n\times m}$. Sample $\mathbf{x}_{id} \leftarrow D_{\Lambda_{\mathbf{u}}^{\perp}(\mathbf{A}_{id}),s}$ using $\mathsf{SamplePre}$ with trapdoor $\mathbf{R}$ for $\mathbf{A}_{id}$. Output the secret key $\mathsf{sk}_{id} := \mathbf{x}_{id} \in \mathbb{Z}^m$.
- $\mathsf{ReKeyGen}(\mathsf{sk}_{id_i}, id_i, id_j)$: Construct $\mathbf{A}_{id_i}$ and $\mathbf{A}_{id_j}$. Sample $\mathbf{R}_1 \leftarrow \chi^{mk\times n}$ and $\mathbf{r}_2 \leftarrow \chi^{mk}$. Construct the re-encryption key

$$\mathsf{rk}_{i\to j} := \begin{bmatrix} \mathbf{R}_1\mathbf{A}_{id_j} & \mathbf{R}_1\mathbf{u} + \mathbf{r}_2 - \mathsf{P2}(\mathbf{x}_{id_i}) \\ \mathbf{0}_{1\times m} & 1 \end{bmatrix} \in \mathbb{Z}_q^{(mk+1)\times(m+1)}.$$

  Output $\mathsf{rk}_{i\to j}$.

Attack Strategy. In the CS security game, the adversary $\mathcal{A}$ (as defined in Definition 3) selects a challenge user i and two equal-length messages m_0 and m_1. Upon querying the challenge oracle, $\mathcal{A}$ receives the challenge ciphertext ct^*. Then, $\mathcal{A}$ queries the re-encryption key generation oracle for the pair (i, j), where j is a corrupted user, and obtains $\mathsf{rk}_{i\to j}$. The adversary computes:

$$\begin{aligned}\mathsf{rk}_{i\to j}\cdot\begin{bmatrix}\mathbf{x}_{id_j}\\ -1\end{bmatrix} &\equiv \begin{bmatrix}\mathbf{R}_1\mathbf{A}_{id_j} & \mathbf{R}_1\mathbf{u}+\mathbf{r}_2-\mathsf{P2}(\mathbf{x}_{id_i})\\ \mathbf{0}_{1\times m} & 1\end{bmatrix}\cdot\begin{bmatrix}\mathbf{x}_{id_j}\\ -1\end{bmatrix} \pmod q\\ &\equiv \begin{bmatrix}\mathbf{R}_1(\mathbf{A}_{id_j}\cdot\mathbf{x}_{id_j}-\mathbf{u})+\mathsf{P2}(\mathbf{x}_{id_i})-\mathbf{r}_2\\ -1\end{bmatrix} \pmod q\\ &\equiv \begin{bmatrix}\mathsf{P2}(\mathbf{x}_{id_i})-\mathbf{r}_2\\ -1\end{bmatrix} \pmod q,\end{aligned}$$

where the second equation follows from the correctness of $\mathsf{Extract}$ and $\mathsf{SamplePre}$ (see Lemma 3), which ensure that $\mathbf{A}_{id_j}\cdot\mathbf{x}_{id_j}\equiv\mathbf{u} \pmod q$.

The adversary thus obtains $\mathsf{P2}(\mathbf{x}_{id_i})-\mathbf{r}_2$. Given that $\mathbf{x}_{id_i}\sim D_{\Lambda^{\perp}_{\mathbf{u}},s}, \mathbf{r}_2\sim\chi^{mk}$, the goal is to recover $\mathbf{x}_{id_i}$. Recall that $\mathsf{P2}(\mathbf{x})=\mathbf{g}\otimes\mathbf{x}$, where $\mathbf{g}=(1,2,4,\ldots,2^{k-1})$. Therefore:

$$\begin{aligned}\mathsf{P2}(\mathbf{x}_{id_i})-\mathbf{r}_2 &= \mathbf{g}\otimes\mathbf{x}_{id_i}-\mathbf{r}_2\\ &= (\mathbf{g}\cdot(1))\otimes(\mathbf{I}\cdot\mathbf{x}_{id_i})-\mathbf{r}_2\\ &= (\mathbf{g}\otimes\mathbf{I})\cdot((1)\otimes\mathbf{x}_{id_i})-\mathbf{r}_2\\ &= (\mathbf{g}\otimes\mathbf{I})\cdot\mathbf{x}_{id_i}+(-\mathbf{r}_2).\end{aligned}$$

Let $\mathbf{P}:=\mathbf{g}\otimes\mathbf{I}$ and $\mathbf{r}:=-\mathbf{r}_2$. This reduces the problem to a form of "search-LWE" instance: given $\mathbf{P}\cdot\mathbf{x}_{id_i}+\mathbf{r}$ and $\mathbf{P}$, recover $\mathbf{x}_{id_i}$. This can be viewed as k structured LWE instances:

$$\mathbf{v}_j\equiv 2^{j-1}\cdot\mathbf{I}\cdot\mathbf{x}_{id_i}+\mathbf{r}_j \pmod q,\quad \text{for } j=1,\ldots,k$$

where $\mathbf{r}:=(\mathbf{r}_1,\mathbf{r}_2,\ldots,\mathbf{r}_k):=(r_1^{(1)},\ldots,r_1^{(m)},r_2^{(1)},\ldots,r_2^{(m)},\ldots,r_k^{(m)})$. These instances are highly structured, enabling bit-by-bit recovery of $\mathbf{x}_{id_i}$.

Consider each entry of the last instance $\mathbf{v}_k=(v_k^{(1)},v_k^{(2)},\ldots,v_k^{(m)})\in\mathbb{Z}_q^m$. Focusing on $v_k^{(1)}\equiv 2^{k-1}\cdot x^{(1)}+r_k^{(1)} \pmod q$, where $x^{(1)}$ is the first entry of $\mathbf{x}_{id_i}$, we can extract the least significant bit of $x^{(1)}$ from $v_k^{(1)}$. The next bit can be recovered using $v_{k-1}^{(1)}$ and the previously recovered bit. The process continues recursively.

Lemma 1 formalises the bit-by-bit recovery of $x^{(1)}$.

Due to the efficient implementation of trapdoors in [24] must have q being a power of 2 [17], we only consider this situation, i.e., $q=2^{\lceil\log q\rceil}=2^k$.

Lemma 1. *Given $(v_1^{(1)},v_2^{(1)},\ldots,v_k^{(1)})$, there exists an efficient algorithm to recover every bit of $x^{(1)}$ with overwhelming probability when $q=2^k$.*

Proof. Let $t = k$ and write $x^{(1)} \bmod q = \sum_{i=0}^{t-1} 2^i \cdot b_i^{(1)}$, where $b_i^{(1)} \in \{0,1\}$ and $x^{(1)} \bmod q$ denotes the standard representative in $\{0,\ldots,q-1\}$. From the equation:

$$\begin{aligned} v_k^{(1)} &\equiv 2^{k-1} \cdot x^{(1)} + r_k^{(1)} \pmod q \\ &\equiv 2^{k-1} \cdot (2^{t-1} \cdot b_{t-1}^{(1)} + 2^{t-2} \cdot b_{t-2}^{(1)} + \cdots + 2 \cdot b_1^{(1)} + b_0^{(1)}) + r_k^{(1)} \pmod q \\ &\equiv q \cdot (2^{t-2} \cdot b_{t-1}^{(1)} + 2^{t-3} \cdot b_{t-2}^{(1)} + \cdots + b_1^{(1)}) + 2^{k-1} \cdot b_0^{(1)} + r_k^{(1)} \pmod q \\ &\equiv 2^{k-1} \cdot b_0^{(1)} + r_k^{(1)} \pmod q. \end{aligned} \tag{1}$$

We compute:

$$r' := v_k^{(1)} \bmod 2^{k-1} = r_k^{(1)} \bmod 2^{k-1}.$$

Since $r_k^{(1)}$ is a noise from an LWE instance, it follows that $r_k^{(1)} \in (-2^{k-2}, 2^{k-2})$ with overwhelming probability $(1-\epsilon)$ for some negligible function $\epsilon(n)$. We can recover $r_k^{(1)}$ as follows: if $r' \geq 2^{k-2}$, set $r_k^{(1)} := r' - 2^{k-1}$; otherwise, $r_k^{(1)} := r'$.

If $v_k^{(1)} = r_k^{(1)}$ or $v_k^{(1)} = q + r_k^{(1)}$, then $b_0^{(1)} = 0$; otherwise, $b_0^{(1)} = 1$.

To recover $b_1^{(1)}$, use:

$$\begin{aligned} v_{k-1}^{(1)} &\equiv 2^{k-2} \cdot x^{(1)} + r_{k-1}^{(1)} \pmod q \\ &\equiv 2^{k-2} \cdot (2^{t-1} \cdot b_{t-1}^{(1)} + 2^{t-2} \cdot b_{t-2}^{(1)} + \cdots + 2 \cdot b_1^{(1)} + b_0^{(1)}) + r_{k-1}^{(1)} \pmod q \\ &\equiv 2^{k-1} \cdot b_1^{(1)} + 2^{k-2} \cdot b_0^{(1)} + r_{k-1}^{(1)} \pmod q. \end{aligned}$$

Subtracting the known term yields:

$$v_{k-1}^{(1)} - 2^{k-2} \cdot b_0^{(1)} \equiv 2^{k-1} \cdot b_1^{(1)} + r_{k-1}^{(1)} \pmod q,$$

which can be solved similarly to Equation (1). The remaining bits are recovered recursively.

The recovery succeeds with probability $(1-\epsilon)^k$. □

By applying this procedure to all entries of $\mathbf{x}_{id_i}$, the adversary can efficiently distinguish the re-encrypted ciphertext with probability:

$$(1-\epsilon)^{km} \approx e^{-km\cdot\epsilon} = e^{-\log q \cdot \mathsf{poly}(n) \cdot \mathsf{negl}(n)} = e^{-\mathsf{negl}'(n)} \approx 1 - \mathsf{negl}'(n),$$

where $\mathsf{negl}'(n)$ is a negligible function. Therefore, the PRE scheme proposed in [14] does not satisfy CS with overwhelming probability. We provide a demo implementation[6].

[6] https://github.com/coro-afk/cs-pre-attack

Attack Analysis. Each iteration requires $O(1)$ addition, subtraction, and modular operations. The secret key is an m-dimensional vector with k-bit entries, so the total complexity is $O(k \cdot m)$ operations. Using the parameter setting from [24, Fig. 2], let the security parameter $\lambda = 284$, the modulus logarithm $k = 24$, and dimension $m = 13,812$. The total number of iterations is $m \cdot k \approx 3.3 \times 10^5$.

The success probability of the attack depends entirely on the distribution of $\mathbf{r}_2$, which is sampled from χ^{mk} according to the scheme. In the correctness analysis of Dutta et al. [14, Theorem 5], the noise term includes $\mathbf{r}_2^t \cdot \mathsf{BD}(\mathbf{c}_1)$, where $\mathbf{c}_1$ is the first part of the identity-based ciphertext. They showed that the entire noise term is less than $q/4$ with overwhelming probability, ensuring correct decryption, which also implies that our attack succeeds with overwhelming probability. In essence, our attack recovers the secret key through a form of "decryption", and the correctness of the re-encryption guarantees the feasibility of this attack.

Other Similar Schemes. Many PRE schemes adopt a similar re-encryption key generation paradigm that involves encrypting the delegator's secret key. Examples include attribute-based PRE schemes [22,27,29,31,33] and fully homomorphic PRE scheme [32]. Consequently, these schemes also fail to achieve CS. Note that CS/MSS is only applicable in single-hop settings, where the encrypt-secret-key paradigm must be avoided if collusion is a concern. In contrast, this paradigm remains a practical choice for multi-hop PRE schemes [16,28,13], where collusion is not considered.

An adaptively secure PRE [16], enhanced from the selectively secure version [10], also employs a similar paradigm: the delegator's secret key is masked by a ciphertext of message 0 encrypted under the delegatee's public key [16, Construction 2]. This enhancement introduces the *source-hiding* property, which obscures the structure of the re-encryption key by adding extra noise to ciphertexts. However, this feature targets honest delegatees and is orthogonal to collusion safety, where the delegatee and the proxy collude. Therefore, our attack also applies to their scheme; nevertheless, since it is a multi-hop scheme, collusion is outside its security model.

Problem. The core issue with the encrypting-secret-key approach is that it implicitly delegates *all* decryption capabilities. However, in many applications, the delegator intends to delegate *only* the re-encryption capability—meaning that only ciphertexts re-encrypted by the proxy should be decryptable by the delegatee.

Known solutions that achieve CS or MSS follow one of two approaches:

1. Mapping ciphertexts to a distinct space [5], where second-level and first-level ciphertexts reside in different domains, making reverse computation infeasible.

2. Generating re-encryption keys via preimage sampling [19], which reveals no information about the trapdoor. This technique is employed in fine-grained PRE [35], tag-based CCA-secure PRE [15], and its predecessor Kir14 [21], which, to the best of our knowledge, is the first application of preimage sampling in the context of PRE.

A straightforward strategy to mitigate collusion is to split the re-encryption key using a secret sharing scheme [34]. However, the effectiveness of this approach depends heavily on the threshold settings, which can significantly impact the scheme's efficiency. Moreover, the underlying re-encryption mechanism in these schemes still relies on the encrypting-secret-key paradigm.

4 Generic Construction for CS PRE

CS can be achieved through a generic transformation applied to any IND-CPA-secure PRE. The core idea is to generate two distinct key pairs for each user: one for encrypting and decrypting fresh (first-level) ciphertexts, and the other for handling re-encrypted (second-level) ciphertexts. This separation ensures that collusion between the proxy and the delegatee cannot compromise the security of second-level ciphertexts. The intuition behind this approach is that if a system needs to secure two different types of ciphertexts, the most straightforward solution is to encrypt them under separate keys.

4.1 Generic Construction

Let the IND-CPA-secure single-hop PRE scheme be defined as:

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

We construct a new scheme:

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

based on PRE, as follows:

- $\mathsf{Setup}' := \mathsf{Setup}$.
- $\mathsf{KeyGen}'(\mathsf{pp}) \to (\mathsf{pk}_i, \mathsf{sk}_i)$. Run KeyGen twice to obtain two key pairs: $\left(\mathsf{pk}_i^{(1)}, \mathsf{sk}_i^{(1)}\right)$ and $\left(\mathsf{pk}_i^{(2)}, \mathsf{sk}_i^{(2)}\right)$, return

$$(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \left(\left(\mathsf{pk}_i^{(1)}, \mathsf{pk}_i^{(2)}\right), \left(\mathsf{sk}_i^{(1)}, \mathsf{sk}_i^{(2)}\right)\right).$$

- $\mathsf{ReKeyGen}'(\mathsf{pk}_i, \mathsf{sk}_i, \mathsf{pk}_j) \to \mathsf{rk}_{i\to j}$. Parse

$$\left(\left(\mathsf{pk}_i^{(1)}, \mathsf{pk}_i^{(2)}\right), \left(\mathsf{sk}_i^{(1)}, \mathsf{sk}_i^{(2)}\right)\right) \leftarrow (\mathsf{pk}_i, \mathsf{sk}_i), \left(\mathsf{pk}_j^{(1)}, \mathsf{pk}_j^{(2)}\right) \leftarrow \mathsf{pk}_j.$$

  Compute $\mathsf{rk}_{i\to j}^{(1\to 2)} \leftarrow \mathsf{ReKeyGen}\left(\mathsf{pk}_i^{(1)}, \mathsf{sk}_i^{(1)}, \mathsf{pk}_j^{(2)}\right)$, and return $\mathsf{rk}_{i\to j} \leftarrow \mathsf{rk}_{i\to j}^{(1\to 2)}$.

- $\mathsf{Enc}'\left(2, \left(\mathsf{pk}^{(1)}, \mathsf{pk}^{(2)}\right), \mathsf{m}\right) \to \mathsf{ct}_2$. Generate a temporary key pair $(\mathsf{pk}_{temp}, \mathsf{sk}_{temp}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$, re-encryption key $\mathsf{rk}_{temp} \leftarrow \mathsf{ReKeyGen}(\mathsf{pk}_{temp}, \mathsf{sk}_{temp}, \mathsf{pk}^{(2)})$, and ciphertext $\mathsf{ct}_{temp} \leftarrow \mathsf{Enc}(1, \mathsf{pk}_{temp}, \mathsf{m})$. Return the second-level ciphertext $\mathsf{ct}_2 \leftarrow \mathsf{ReEnc}(\mathsf{rk}_{temp}, \mathsf{ct}_{temp})$.
- Define the remaining algorithms as follows:

$$\begin{aligned}
\mathsf{ReEnc}' &:= \mathsf{ReEnc}, \\
\mathsf{Enc}'\left(1, \left(\mathsf{pk}^{(1)}, \mathsf{pk}^{(2)}\right), \cdot\right) &:= \mathsf{Enc}\left(1, \mathsf{pk}^{(1)}, \cdot\right), \\
\mathsf{Dec}'\left(1, \left(\mathsf{sk}^{(1)}, \mathsf{sk}^{(2)}\right), \cdot\right) &:= \mathsf{Dec}\left(1, \mathsf{sk}^{(1)}, \cdot\right), \\
\mathsf{Dec}'\left(2, \left(\mathsf{sk}^{(1)}, \mathsf{sk}^{(2)}\right), \cdot\right) &:= \mathsf{Dec}\left(2, \mathsf{sk}^{(2)}, \cdot\right).
\end{aligned}$$

Remark 1. In the earlier pre-proceedings version of this work, the level-2 encryption algorithm $\mathsf{Enc}'(2, \cdot, \cdot)$ directly invoked the underlying Enc. We modified this to use a temporary key pair and re-encryption. This adjustment ensures that the generated ciphertext distribution matches the simulator's distribution in our reduction, thereby closing a simulation gap in the security proof for collusion safety without requiring any additional assumptions (re-encryption simulatability [12]).

This construction is also applicable to the IND-HRA-secure schemes [12,13,27,31] since IND-HRA security implies IND-CPA security. Thus, these schemes can be upgraded to satisfy CS at the cost of doubling the public and secret key sizes. For the same reason, this construction is also applicable to the IND-CCA-secure PRE schemes.

See applications of the generic construction in Appendix D.

4.2 Security Proof

Theorem 1. *Given an IND-CPA-secure single-hop scheme* PRE*, there exists a single-hop scheme* PRE' *that is both IND-CPA-secure and CS.*

The security of the proposed construction is established via a reduction. The full proof is shown in Appendix C.

Proof sketch: assume there exists a CS adversary $\mathcal{A}$ against our transformed scheme PRE'. We build an IND-CPA adversary $\mathcal{B}$ for the original scheme PRE. $\mathcal{B}$ simulates the CS game perfectly, obtaining second-level keys from the IND-CPA challenger. It locally generates first-level keys, enabling it to answer all re-encryption key generation queries. For the challenge, $\mathcal{B}$ forwards the challenge messages to the IND-CPA challenger, receives a *first-level* ciphertext under a second-level key, and re-encrypts it before returning to $\mathcal{A}$. Any advantage of $\mathcal{A}$ in distinguishing second-level ciphertexts translates to an advantage of $\mathcal{B}$ in the IND-CPA game, contradicting the assumed security of PRE. Hence, PRE' is IND-CPA-secure and CS.

Acknowledgments. The authors thank the reviewers for their valuable comments. This work was partially supported by the XJTLU Research Development Fund under Grant No. RDF-21-02-014, RDF-22-02-106; and XJTLU Teaching Development Fund under Grant No. TDF2223-R25-207. The author Haotian Yin thanks Di Liu for the helpful discussion.

References

1. Agrawal, S., Boneh, D., Boyen, X.: Efficient Lattice (H)IBE in the Standard Model. In: Gilbert, H. (ed.) Advances in Cryptology – EUROCRYPT 2010. pp. 553–572. Springer, Berlin, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13190-5_28
2. Ajtai, M.: Generating hard instances of lattice problems (extended abstract). In: Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing - STOC '96. pp. 99–108. ACM Press, Philadelphia, Pennsylvania, United States (1996). https://doi.org/10.1145/237814.237838, http://portal.acm.org/citation.cfm?doid=237814.237838
3. Aono, Y., Boyen, X., Phong, L.T., Wang, L.: Key-Private Proxy Re-encryption under LWE. In: Paul, G., Vaudenay, S. (eds.) Progress in Cryptology – INDOCRYPT 2013. pp. 1–18. Springer International Publishing, Cham (2013). https://doi.org/10.1007/978-3-319-03515-4_1
4. Applebaum, B., Cash, D., Peikert, C., Sahai, A.: Fast Cryptographic Primitives and Circular-Secure Encryption Based on Hard Learning Problems. In: Halevi, S. (ed.) Advances in Cryptology - CRYPTO 2009. pp. 595–618. Springer, Berlin, Heidelberg (2009). https://doi.org/10.1007/978-3-642-03356-8_35
5. Ateniese, G., Fu, K., Green, M., Hohenberger, S.: Improved proxy re-encryption schemes with applications to secure distributed storage. ACM Trans. Inf. Syst. Secur. **9**(1), 1–30 (Feb 2006). https://doi.org/10.1145/1127345.1127346, https://dl.acm.org/doi/10.1145/1127345.1127346
6. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: Fully Homomorphic Encryption without Bootstrapping (2011), https://eprint.iacr.org/2011/277
7. Brakerski, Z., Vaikuntanathan, V.: Fully Homomorphic Encryption from Ring-LWE and Security for Key Dependent Messages. In: Rogaway, P. (ed.) Advances in Cryptology – CRYPTO 2011. pp. 505–524. Springer, Berlin, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22792-9_29
8. Brakerski, Z., Vaikuntanathan, V.: Efficient Fully Homomorphic Encryption from (Standard) LWE. SIAM J. Comput. **43**(2), 831–871 (Jan 2014). https://doi.org/10.1137/120868669, https://epubs.siam.org/doi/abs/10.1137/120868669
9. 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. CCS '07, Association for Computing Machinery, New York, NY, USA (Oct 2007). https://doi.org/10.1145/1315245.1315269, https://dl.acm.org/doi/10.1145/1315245.1315269
10. Chandran, N., Chase, M., Liu, F.H., Nishimaki, R., Xagawa, K.: Re-encryption, Functional Re-encryption, and Multi-hop Re-encryption: A Framework for Achieving Obfuscation-Based Security and Instantiations from Lattices. In: Krawczyk, H. (ed.) Public-Key Cryptography – PKC 2014. pp. 95–112. Springer, Berlin, Heidelberg (2014). https://doi.org/10.1007/978-3-642-54631-0_6

11. Chandran, N., Chase, M., Vaikuntanathan, V.: Functional Re-encryption and Collusion-Resistant Obfuscation. In: Cramer, R. (ed.) Theory of Cryptography. pp. 404–421. Springer, Berlin, Heidelberg (2012). https://doi.org/10.1007/978-3-642-28914-9_23
12. Cohen, A.: What About Bob? The Inadequacy of CPA Security for Proxy Reencryption. In: Lin, D., Sako, K. (eds.) Public-Key Cryptography – PKC 2019. pp. 287–316. Springer International Publishing, Cham (2019). https://doi.org/10.1007/978-3-030-17259-6_10
13. Cohen, A., Cousins, D.B., Genise, N., Kline, E., Polyakov, Y., RV, S.: HRA-Secure Homomorphic Lattice-Based Proxy Re-Encryption with Tight Security (2024), https://eprint.iacr.org/2024/681
14. Dutta, P., Susilo, W., Duong, D.H., Roy, P.S.: Collusion-resistant identity-based Proxy Re-encryption: Lattice-based constructions in Standard Model. Theoretical Computer Science **871**, 16–29 (Jun 2021). https://doi.org/10.1016/j.tcs.2021.04.008, https://www.sciencedirect.com/science/article/pii/S0304397521002127
15. Fan, X., Liu, F.H.: Proxy Re-Encryption and Re-Signatures from Lattices. In: Deng, R.H., Gauthier-Umaña, V., Ochoa, M., Yung, M. (eds.) Applied Cryptography and Network Security. pp. 363–382. Springer International Publishing, Cham (2019). https://doi.org/10.1007/978-3-030-21568-2_18
16. Fuchsbauer, G., Kamath, C., Klein, K., Pietrzak, K.: Adaptively Secure Proxy Re-encryption. In: Lin, D., Sako, K. (eds.) Public-Key Cryptography – PKC 2019. pp. 317–346. Springer International Publishing, Cham (2019). https://doi.org/10.1007/978-3-030-17259-6_11
17. Genise, N., Micciancio, D.: Faster Gaussian Sampling for Trapdoor Lattices with Arbitrary Modulus. In: Nielsen, J.B., Rijmen, V. (eds.) Advances in Cryptology – EUROCRYPT 2018. pp. 174–203. Springer International Publishing, Cham (2018). https://doi.org/10.1007/978-3-319-78381-9_7
18. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing. pp. 169–178. STOC '09, Association for Computing Machinery, New York, NY, USA (May 2009). https://doi.org/10.1145/1536414.1536440, https://dl.acm.org/doi/10.1145/1536414.1536440
19. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for Hard Lattices and New Cryptographic Constructions (2007), https://eprint.iacr.org/2007/432
20. Hohenberger, S., Rothblum, G.N., shelat, a., Vaikuntanathan, V.: Securely Obfuscating Re-encryption. In: Vadhan, S.P. (ed.) Theory of Cryptography. pp. 233–252. Springer, Berlin, Heidelberg (2007). https://doi.org/10.1007/978-3-540-70936-7_13
21. Kirshanova, E.: Proxy Re-encryption from Lattices. In: Krawczyk, H. (ed.) Public-Key Cryptography – PKC 2014, vol. 8383, pp. 77–94. Springer Berlin Heidelberg, Berlin, Heidelberg (2014). https://doi.org/10.1007/978-3-642-54631-0_5, http://link.springer.com/10.1007/978-3-642-54631-0_5
22. Liang, X., Weng, J., Yang, A., Yao, L., Jiang, Z., Wu, Z.: Attribute-Based Conditional Proxy Re-encryption in the Standard Model Under LWE. In: Computer Security – ESORICS 2021: 26th European Symposium on Research in Computer Security, Darmstadt, Germany, October 4–8, 2021, Proceedings, Part II. pp. 147–168. Springer-Verlag, Berlin, Heidelberg (Oct 2021). https://doi.org/10.1007/978-3-030-88428-4_8, https://doi.org/10.1007/978-3-030-88428-4_8
23. Luo, F., Wang, H., Susilo, W., Yan, X., Zheng, X.: Public Trace-and-Revoke Proxy Re-Encryption for Secure Data Sharing in Clouds. IEEE Transactions on Informa-

tion Forensics and Security **19**, 2919–2934 (2024). https://doi.org/10.1109/TIFS.2024.3357240, https://ieeexplore.ieee.org/abstract/document/10411921

24. Micciancio, D., Peikert, C.: Trapdoors for Lattices: Simpler, Tighter, Faster, Smaller. In: Pointcheval, D., Johansson, T. (eds.) Advances in Cryptology – EUROCRYPT 2012. pp. 700–718. Springer, Berlin, Heidelberg (2012). https://doi.org/10.1007/978-3-642-29011-4_41
25. Polyakov, Y., Rohloff, K., Sahu, G., Vaikuntanathan, V.: Fast Proxy Re-Encryption for Publish/Subscribe Systems. ACM Trans. Priv. Secur. **20**(4), 14:1–14:31 (Sep 2017). https://doi.org/10.1145/3128607, https://dl.acm.org/doi/10.1145/3128607
26. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. J. ACM **56**(6), 34:1–34:40 (Sep 2009). https://doi.org/10.1145/1568318.1568324, https://dl.acm.org/doi/10.1145/1568318.1568324
27. Susilo, W., Dutta, P., Duong, D.H., Roy, P.S.: Lattice-Based HRA-secure Attribute-Based Proxy Re-Encryption in Standard Model. In: Bertino, E., Shulman, H., Waidner, M. (eds.) Computer Security – ESORICS 2021. pp. 169–191. Springer International Publishing, Cham (2021). https://doi.org/10.1007/978-3-030-88428-4_9
28. Wan, X., Wang, Y., Xue, H., Wang, M.: Unbounded Multi-hop Proxy Re-encryption with HRA Security: An LWE-Based Optimization. In: Susilo, W., Pieprzyk, J. (eds.) Information Security and Privacy. pp. 124–144. Springer Nature, Singapore (2025). https://doi.org/10.1007/978-981-96-9098-5_7
29. Yao, L., Weng, J., Wu, P., Li, X., Liu, Y., Lai, J., Yang, G., Deng, R.H.: Conditional Attribute-Based Proxy Re-Encryption: Definitions and Constructions from LWE (2022), https://eprint.iacr.org/2022/637
30. Yin, H., Zhang, J., Li, W., Dong, Y., Lim, E.G., Wojtczak, D.: Revisiting Honest Re-Encryption Attack for Proxy Re-Encryption Schemes (2025), https://eprint.iacr.org/2025/704
31. Zhang, Y., Wang, Y., Wang, M.: Improved AB-CPREs with HRA Security. In: 2023 International Conference on Data Security and Privacy Protection (DSPP). pp. 223–231 (Oct 2023). https://doi.org/10.1109/DSPP58763.2023.10404893, https://ieeexplore.ieee.org/abstract/document/10404893
32. Zhao, F., Wang, H., Weng, J.: Constant-Size Unbounded Multi-hop Fully Homomorphic Proxy Re-encryption from Lattices. In: Garcia-Alfaro, J., Kozik, R., Choraś, M., Katsikas, S. (eds.) Computer Security – ESORICS 2024. pp. 238–258. Springer Nature Switzerland, Cham (2024). https://doi.org/10.1007/978-3-031-70896-1_12
33. Zhao, F., Weng, J., Xie, W., Hou, L., Li, M.: Time-based attribute-based proxy re-encryption with decryption key update. Des. Codes Cryptogr. **92**(12), 4099–4129 (Dec 2024). https://doi.org/10.1007/s10623-024-01467-x, https://doi.org/10.1007/s10623-024-01467-x
34. Zhao, F., Weng, J., Xie, W., Li, M., Weng, J.: HRA-secure attribute-based threshold proxy re-encryption from lattices. Information Sciences **655**, 119900 (Jan 2024). https://doi.org/10.1016/j.ins.2023.119900, https://www.sciencedirect.com/science/article/pii/S0020025523014858
35. Zhou, Y., Liu, S., Han, S., Zhang, H.: Fine-Grained Proxy Re-encryption: Definitions and Constructions from LWE. In: Guo, J., Steinfeld, R. (eds.) Advances in Cryptology – ASIACRYPT 2023. pp. 199–231. Springer Nature, Singapore (2023). https://doi.org/10.1007/978-981-99-8736-8_7

A Master Secret Secrecy

Definition 5 (MSS Game). *Let λ be the security parameter and $\mathcal{A}$ a PPT adversary. The MSS game is defined as an interaction between $\mathcal{A}$ and a challenger who provides the following oracles, which may be invoked in any order, subject to the constraints below:*
Phase 1: (identical to the CS game)

- *Setup: Generate the public parameters and provide them to $\mathcal{A}$. Initialise a counter* $\mathsf{numKeys} := 0$*, and two sets:* Hon *(honest users) and* Cor *(corrupted users), both initially empty.*
- *Honest Key Generation: Generate a key pair* $(\mathsf{pk}_{\mathsf{numKeys}}, \mathsf{sk}_{\mathsf{numKeys}}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$ *and return* $\mathsf{pk}_{\mathsf{numKeys}}$ *to $\mathcal{A}$. Add* $\mathsf{numKeys}$ *to* Hon *and increment* $\mathsf{numKeys}$*.*
- *Corrupted Key Generation: Generate a key pair* $(\mathsf{pk}_{\mathsf{numKeys}}, \mathsf{sk}_{\mathsf{numKeys}}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$ *and return both keys to $\mathcal{A}$. Add* $\mathsf{numKeys}$ *to* Cor *and increment* $\mathsf{numKeys}$*.*

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 $i = j$, output $\perp$; otherwise, return* $\mathsf{rk}_{i\to j}$*.*

Phase 3:

- *Decision: On input* $(i, j, \mathsf{rk}_{i\to j})$*, where $i \in \mathsf{Cor}$ and $j \in \mathsf{Hon}$, choose a message* $\mathsf{m} \in \mathcal{M}$*, compute* $\mathsf{ct}_i \leftarrow \mathsf{Enc}(1, \mathsf{pk}_i, \mathsf{m})$*, and* $\mathsf{ct}_j \leftarrow \mathsf{ReEnc}(\mathsf{rk}_{i\to j}, \mathsf{ct}_i)$*. Return* win *if* $\mathsf{Dec}(2, \mathsf{sk}_j, \mathsf{ct}_j) = \mathsf{m}$*.*

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

$$\mathsf{Adv}^{mss}_{pre}(\mathcal{A}) := \Pr[\mathsf{win}],$$

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

We relax the requirement on the re-encryption key: in the original definition [5], the adversary is required to output the exact re-encryption key. In contrast, considering the re-encryption key generation algorithm could be probabilistic, our definition allows the adversary to output any key that is functionality equivalent, i.e., capable of correctly re-encrypting ciphertexts to the delegatee. This subtle formalisation aligns with the notion of re-encryption key unlearnability [30].

B Lattice Background

Definition 6 (Discrete Gaussian Distribution). *The Gaussian function with parameter s and centre $\mathbf{c} \in \mathbb{R}^n$ is defined as*

$$\rho_{s,\mathbf{c}} : \mathbb{R}^n \to \mathbb{R}, \quad \rho_{s,\mathbf{c}}(\mathbf{x}) := e^{-\pi\|\mathbf{x}-\mathbf{c}\|^2/s^2}.$$

For a countable set $\mathcal{S} \subseteq \mathbb{R}^n$, the discrete Gaussian distribution $D_{\mathcal{S},s,\mathbf{c}}$ parametrised by s and $\mathbf{c}$ is defined as

$$D_{\mathcal{S},s,\mathbf{c}}(x) := \frac{\rho_{s,\mathbf{c}}(\mathbf{x})}{\sum_{\mathbf{x}\in\mathcal{S}} \rho_{s,\mathbf{c}}(\mathbf{x})} \quad \textit{for } \mathbf{x} \in \mathcal{S},$$

and

$$D_{\mathcal{S},s,\mathbf{c}}(\mathbf{x}) := 0 \quad \textit{for } \mathbf{x} \notin \mathcal{S}.$$

Usually, s is omitted when $s = 1$, and $\mathbf{c}$ is omitted if $\mathbf{c} = \mathbf{0}$.

Definition 7 (LWE Assumption [26,4]). *Let $n, m, q \in \mathbb{N}$ and χ be a distribution over $\mathbb{Z}_q$. The $\mathsf{LWE}_{n,q,\chi,m}$-assumption requires that for any PPT adversary $\mathcal{A}$, its advantage function satisfies*

$$\mathsf{Adv}^{\mathsf{LWE}}_{[n,q,\chi,m],\mathcal{A}}(\lambda) := |\Pr[\mathcal{A}(\mathbf{A}, \mathbf{u}) = 1] - \Pr[\mathcal{A}(\mathbf{A}, \mathbf{As} + \mathbf{e}) = 1]| \leq \mathsf{negl}(\lambda),$$

where $\mathbf{A} \leftarrow \mathbb{Z}_q^{m\times n}$, $\mathbf{s} \leftarrow \chi^n$, $\mathbf{e} \leftarrow \chi^m$, and $\mathbf{u} \leftarrow \mathbb{Z}_q^m$.

For $Q \in \mathbb{N}$, the Q-$\mathsf{LWE}_{n,q,\chi,m}$-assumption requires that for any PPT adversary $\mathcal{A}$, its advantage satisfies

$$\mathsf{Adv}^{Q\text{-}\mathsf{LWE}}_{[n,q,\chi,m],\mathcal{A}}(\lambda) := |\Pr[\mathcal{A}(\mathbf{A}, \mathbf{AS} + \mathbf{E}) = 1] - \Pr[\mathcal{A}(\mathbf{A}, \mathbf{U}) = 1]| \leq \mathsf{negl}(\lambda),$$

where $\mathbf{A} \leftarrow \mathbb{Z}_q^{m\times n}$, $\mathbf{S} \leftarrow \chi^{n\times Q}$, $\mathbf{E} \leftarrow \chi^{m\times Q}$, and $\mathbf{U} \leftarrow \mathbb{Z}_q^{m\times Q}$.

A simple hybrid argument shows that $\mathsf{Adv}^{Q\text{-}\mathsf{LWE}}_{[n,q,\chi,m],\mathcal{A}}(\lambda) \leq Q \cdot \mathsf{Adv}^{\mathsf{LWE}}_{[n,q,\chi,m],\mathcal{A}}(\lambda)$.

Lemma 2 (Trapdoor Generation [2,24]). *Let $\bar{\mathbf{A}} \in \mathbb{Z}_q^{n\times\bar{m}}$ and $\mathbf{H}$ be an invertible matrix. There exists an efficient algorithm $\mathsf{TrapGen}$ that takes $\bar{\mathbf{A}}$ and $\mathbf{H}$ (and a distribution D) as input, output the matrix $\mathbf{A} = [\bar{\mathbf{A}} \mid \mathbf{HG} - \bar{\mathbf{A}}\mathbf{R}]$ and its trapdoor $\mathbf{R} \in \mathbb{Z}^{\bar{m}\times w}$ (from distribution D).*

Lemma 3 (Preimage Sampling [24]). *Let $\mathbf{A} \in \mathbb{Z}_q^{m\times n}$ be a matrix with a trapdoor $\mathbf{R}$, $\mathbf{H}$ be the tag of the trapdoor, and $\mathbf{u}$ be a vector. There exists an efficient algorithm $\mathsf{SamplePre}$ that takes $\mathbf{A}, \mathbf{H}, \mathbf{R}$, and $\mathbf{u}$ (and a distribution D) as input, output the vector $\mathbf{x}$ (from distribution D) such that $\mathbf{Ax} = \mathbf{u} \pmod q$.*

C Proof of Theorem 1

Proof. Let $\mathcal{A}$ be a CS adversary (as defined in Definition 3) and PRE' defined as above. We construct a reduction $\mathcal{B}$ that interacts with the IND-CPA challenger (as defined in Definition 4) and simulates the CS game for $\mathcal{A}$:

– Setup: $\mathcal{B}$ invokes its IND-CPA setup oracle, maintain $\mathsf{numKeys}$, Hon, and Cor locally. In addition, initialise a local key-value store Keys to maintain key pairs.

- Honest Key Generation: $\mathcal{B}$ queries the IND-CPA honest key generation oracle to obtain $\mathsf{pk}^{(2)}$. It then locally generates a second key pair $(\mathsf{pk}^{(1)}, \mathsf{sk}^{(1)})$, and sets:
$$\mathsf{pk} \leftarrow \left(\mathsf{pk}^{(1)}, \mathsf{pk}^{(2)}\right), \mathsf{sk} \leftarrow \left(\mathsf{sk}^{(1)}, \perp\right).$$
The index $\mathsf{numKeys}$ is added to the honest set Hon, and the pair $(\mathsf{numKeys}, (\mathsf{pk}, \mathsf{sk}))$ is stored in Keys. Increment $\mathsf{numKeys}$ and return $(\mathsf{numKeys}, \mathsf{pk})$.
- Corrupted Key Generation: Similar to the honest case, but $\mathcal{B}$ queries the IND-CPA corrupted key generation oracle to obtain $(\mathsf{pk}^{(2)}, \mathsf{sk}^{(2)})$, and locally generates $(\mathsf{pk}^{(1)}, \mathsf{sk}^{(1)})$. Set:
$$\mathsf{pk} \leftarrow \left(\mathsf{pk}^{(1)}, \mathsf{pk}^{(2)}\right), \mathsf{sk} \leftarrow \left(\mathsf{sk}^{(1)}, \mathsf{sk}^{(2)}\right).$$
Add $\mathsf{numKeys}$ to the corrupted set Cor, store the pair in Keys, and return $(\mathsf{numKeys}, (\mathsf{pk}, \mathsf{sk}))$.
- $\mathcal{O}_{\mathsf{ReKeyGen}}$: On input (i, j), if $i = j$, return $\perp$. Otherwise, retrieve $(\mathsf{pk}_t, \mathsf{sk}_t)$ for $t \in \{i, j\}$ from Keys and parse:
$$\left(\left(\mathsf{pk}_t^{(1)}, \mathsf{pk}_t^{(2)}\right), \left(\mathsf{sk}_t^{(1)}, \mathsf{sk}_t^{(2)}\right)\right) \leftarrow (\mathsf{pk}_t, \mathsf{sk}_t).$$
Compute the re-encryption key locally:
$$\mathsf{ReKeyGen}\left(\mathsf{pk}_i^{(1)}, \mathsf{sk}_i^{(1)}, \mathsf{pk}_j^{(2)}\right),$$
and returns the result. Note that $\mathcal{B}$ can answer all re-encryption key generation queries (especially from honest keys to corrupted users, without assistance from the IND-CPA challenger) since it generates all $\mathsf{sk}_i^{(1)}$ locally.
- Challenge Oracle: on input $(i, \mathsf{m}_0, \mathsf{m}_1)$ from $\mathcal{A}$, $\mathcal{B}$ rejects this query if $i \notin \mathsf{Hon}$. Otherwise, select honest $j \neq i$, forward the query $(j, \mathsf{m}_0, \mathsf{m}_1)$ to the IND-CPA challenger, and receive the challenge ciphertext $\mathsf{ct}_1^* \leftarrow \mathsf{Enc}\left(1, \mathsf{pk}_j^{(2)}, \mathsf{m}_b\right)$, where b is the unknown bit selected by the IND-CPA challenger. $\mathcal{B}$ *queries* the re-encryption key
$$\mathsf{rk}_{j\to i}^{(2\to 2)} \leftarrow \mathsf{ReKeyGen}\left(\mathsf{pk}_j^{(2)}, \mathsf{sk}_j^{(2)}, \mathsf{pk}_i^{(2)}\right)$$
by sending (j, i) to the IND-CPA challenger. This query is valid since $\mathsf{pk}_i^{(2)}$ and $\mathsf{pk}_j^{(2)}$ are both valid and honest keys registered in the IND-CPA game. It locally computes the ciphertext
$$\mathsf{ct}_2^* \leftarrow \mathsf{ReEnc}\left(\mathsf{rk}_{j\to i}^{(2\to 2)}, \mathsf{ct}_1^*\right),$$
and returns this ciphertext ct_2^* to $\mathcal{A}$. Note that this corresponds to a valid second-level ciphertext in PRE'.
- Decision: $\mathcal{B}$ forwards $\mathcal{A}$'s guess b' to the IND-CPA challenger.

Note that in the $\mathsf{Enc}'(2,\cdot,\cdot)$ algorithm, the generation of a second-level ciphertext relies on a temporary key pair and a subsequent re-encryption. In our simulation, the simulator $\mathcal{B}$ selects the public key $\mathsf{pk}_j^{(2)}$ of another honest user j to act as this temporary public key. Since j is an honest user and its secret key remains uncompromised, this perfectly simulates the ephemeral secret key used in $\mathsf{Enc}'(2,\cdot,\cdot)$. Consequently, the distribution of the challenge ciphertext ct_2^* constructed by $\mathcal{B}$ is identical to the distribution of the second-level challenge ciphertext output by the Enc' in the real CS game. Therefore, the advantage of $\mathcal{B}$ in the IND-CPA game equals the advantage of $\mathcal{A}$ in the CS game, yielding $\Pr[\mathsf{win}_{\mathcal{B}}] = \Pr[\mathsf{win}_{\mathcal{A}}]$. □

D Applications

D.1 PRE from FHE

Let $\mathsf{FHE} := (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ be an FHE scheme. By our generic construction:

- $\mathsf{KeyGen}'(\mathsf{pp}) \to (\mathsf{pk}_i, \mathsf{sk}_i)$. Run KeyGen twice to obtain two key pairs: $\left(\mathsf{pk}_i^{(1)}, \mathsf{sk}_i^{(1)}\right)$ and $\left(\mathsf{pk}_i^{(2)}, \mathsf{sk}_i^{(2)}\right)$, return

$$(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \left(\left(\mathsf{pk}_i^{(1)}, \mathsf{pk}_i^{(2)}\right), \left(\mathsf{sk}_i^{(1)}, \mathsf{sk}_i^{(2)}\right)\right).$$

- $\mathsf{ReKeyGen}'(\mathsf{pk}_i, \mathsf{sk}_i, \mathsf{pk}_j) \to \mathsf{rk}_{i\to j}$. Parse

$$\left(\left(\mathsf{pk}_i^{(1)}, \mathsf{pk}_i^{(2)}\right), \left(\mathsf{sk}_i^{(1)}, \mathsf{sk}_i^{(2)}\right)\right) \leftarrow (\mathsf{pk}_i, \mathsf{sk}_i), \left(\mathsf{pk}_j^{(1)}, \mathsf{pk}_j^{(2)}\right) \leftarrow \mathsf{pk}_j.$$

  Compute $\mathsf{ct}_k \leftarrow \mathsf{Enc}\left(\mathsf{pk}_j^{(2)}, \mathsf{sk}_i^{(1)}\right)$, let

$$\mathsf{rk}_{i\to j} \leftarrow \left(\mathsf{ct}_k, \mathsf{pk}_j^{(2)}\right),$$

  and return $\mathsf{rk}_{i\to j}$.
- $\mathsf{ReEnc}'(\mathsf{rk}_{i\to j}, \mathsf{ct}_i)$: Parse $\left(\mathsf{ct}_k, \mathsf{pk}_j^{(2)}\right) \leftarrow \mathsf{rk}_{i\to j}$, compute $\mathsf{ct}_{i\to j} \leftarrow \mathsf{Enc}\left(\mathsf{pk}_j^{(2)}, \mathsf{ct}_i\right)$, and $\mathsf{ct}_j \leftarrow \mathsf{Eval}\left(\mathsf{pk}_j^{(2)}, \mathsf{Dec}, \mathsf{ct}_k, \mathsf{ct}_{i\to j}\right)$. Return ct_j.

Setup, encryption, and decryption algorithms are defined as generic constructions.

D.2 PRE from LWE

To apply our generic construction on the identity-based PRE proposed by Dutta et al. [14], we use the identity with double the length of the original:

- $\mathsf{Extract}'(\mathsf{msk}, id^{(1)}\|id^{(2)}) \to \mathsf{sk}_{id^{(1)}\|id^{(2)}}$: For $t \in \{1,2\}$, construct $\mathbf{A}_{id^{(t)}} = [\bar{\mathbf{A}} \mid -\bar{\mathbf{A}}\mathbf{R}+\mathbf{H}_{id^{(t)}}\mathbf{G}] \in \mathbb{Z}_q^{n\times m}$, sample $\mathbf{x}_{id^{(t)}} \leftarrow D_{\Lambda^{\perp}_{\mathbf{u}}(\mathbf{A}_{id^{(t)}}),s}$ using $\mathsf{SamplePre}$ with trapdoor $\mathbf{R}$ for $\mathbf{A}^{(t)}_{id}$, and output the secret key $\mathsf{sk}_{id^{(1)}\|id^{(2)}} \leftarrow (\mathbf{x}_{id^{(1)}}, \mathbf{x}_{id^{(2)}}) \in \mathbb{Z}^{m\times 2}$.
- $\mathsf{ReKeyGen}'\left(\mathsf{sk}_{id_i^{(1)}\|id_i^{(2)}}, id_i^{(1)}\|id_i^{(2)}, id_j^{(1)}\|id_j^{(2)}\right)$: Construct $\mathbf{A}_{id_i^{(1)}}$ and $\mathbf{A}_{id_j^{(2)}}$, sample $\mathbf{R}_1 \leftarrow \chi^{mk\times n}$ and $\mathbf{r}_2 \leftarrow \chi^{mk}$, and construct the re-encryption key

$$\mathsf{rk}_{i\to j} = \begin{bmatrix} \mathbf{R}_1\mathbf{A}_{id_j^{(2)}} & \mathbf{R}_1\mathbf{u} + \mathbf{r}_2 - \mathsf{P2}\left(\mathbf{x}_{id_i^{(1)}}\right) \\ \mathbf{0}_{1\times m} & 1 \end{bmatrix} \in \mathbb{Z}_q^{(mk+1)\times(m+1)}.$$

  Output $\mathsf{rk}_{i\to j}$.

Encryption, decryption, and re-encryption algorithms are defined as the generic construction.