Bounded CCA2 Secure Proxy Re-encryption Based on Kyber*

Shingo Sato Junji Shikata

January 30, 2025

Abstract

Proxy re-encryption (PRE) allows a semi-honest party (called a proxy) to convert ciphertexts under a public key into ciphertexts under another public key. Due to this functionality, there are various applications such as encrypted email forwarding, key escrow, and secure distributed file systems. On the other hand, post-quantum cryptography (PQC) is one of the most important research areas. However, there is no post-quantum PRE scheme with security against adaptive chosen ciphertext attacks (denoted by $\mathsf{CCA2}$ security) while many PRE schemes have been proposed so far.

In this paper, we propose a bounded $\mathsf{CCA2}$ secure PRE scheme based on CRYSTALS-Kyber (Kyber, for short) which is a selected algorithm in the NIST PQC competition. To this end, we present generic constructions of bounded $\mathsf{CCA2}$ secure PRE. Our generic constructions start from PRE with a variant of security against chosen plaintext attacks (denoted by CPA security) and a new PRE's property introduced in this paper. In order to instantiate our generic constructions, we present a Kyber-based PRE scheme with the required property. As a result, we can construct a bounded $\mathsf{CCA2}$ secure PRE scheme from Kyber.

1 Introduction

1.1 Background and Related Work

The notion of proxy re-encryption (PRE) was introduced in [5], and PRE is public key encryption (PKE) which allows a semi-honest party (called a proxy) to convert an encryption of a message under a public key into another encryption of the same message under another public key. That is, a user Alice with a public key pk_A can generate a re-encryption key $\mathsf{rk}_{A\to B}$ converting ciphertexts under pk_A into ciphertexts under a public key pk_B of another user Bob and give $\mathsf{rk}_{A\to B}$ to a proxy. Then, this proxy can transform ciphertexts under pk_A into ciphertexts under pk_B, without knowledge of underlying messages. Security of PRE ensures confidentiality of messages even though the adversary has several re-encryption keys.

Due to PRE's functionality, there are various applications such as encrypted email forwarding [5], key escrow [21], secure distributed file systems [3], and secure publish-subscribe system [26]. Hence, there are many PRE schemes such as schemes based on the (computational or decisional) Diffi-Hellman assumption (e.g., [5, 7, 11, 25]), pairing-based schemes (e.g., [2, 3, 8, 23]), and obfuscation-based schemes [9, 10].

In particular, we focus on post-quantum PRE because post-quantum cryptography (PQC) is one of the most important research areas, and there are also many researches on selected algorithms and candidates in the NIST (National Institute of Standards and Technology) PQC standardization process (e.g., [1, 15, 19, 24, 28]), due to advancement of quantum computers. As post-quantum PRE,

*Yokohama National University, Japan. sato-shingo-zk@ynu.ac.jp, shikata-junji-rb@ynu.ac.jp

several lattice-based PRE schemes have been proposed so far (e.g., [9,16,26,31,32]). However, there is no post-quantum PRE scheme with security against adaptive chosen ciphertext attacks (denoted by CCA2 security) which was formalized in [8]. To the best of our knowledge, all the existing lattice-based schemes satisfy *security against chosen plaintext attacks, non-adaptive chosen ciphertext attacks, or honest re-encryption attacks* (denoted by CPA, CCA1, or HRA security, respectively). CPA and CCA1 security are strictly weaker than CCA2 security and may be insufficient in PRE's applications, as discussed in [12]. Additionally, the relation between CCA2 and HRA security is not known. Achieving CCA2 security is important because CCA2 security of PRE is one of the most desirable security notions and provides a wide range of applications.

Related Work. Blaze, Bleumer, and Strauss introduced the notion of PRE and proposed a PRE scheme based on the DDH assumption [5]. This scheme is bidirectional, multi-hop, and CPA secure. Ateniese, Fu, Green, and Hohenberger presented the first (single-hop) unidirectional PRE scheme with CPA security by using bilinear maps [4]. Since Canetti and Hohenberger [8] formalized CCA2 security for PRE, CCA2 secure PRE schemes have been proposed in [7, 8, 11, 23].

As post-quantum PRE, there are only lattice-based PRE schemes. Lattice-based PRE schemes with CPA security have been proposed in [9, 17, 22, 26, 32]. Fan and Liu [16] gave tag-based PRE schemes based on the learning with errors (LWE) assumption and these achieve CCA1 security. On the other hand, Cohen [12] introduced the notion of HRA security and showed that one of practical lattice-based (CPA secure) PRE schemes of [26] was insecure in the HRA security model. Furthermore, Fuchsbauer et al. [17] formalized adaptive CPA and HRA security notions and proposed adaptive HRA secure schemes based on the PRE schemes of [9,18]. Zhou, Liu, and Han [31] presented a LWE-based construction of HRA secure fine-grained PRE scheme whose notion was introduced in [32].

From the above, all the existing post-quantum PRE schemes do not satisfy CCA2 security.

1.2 Our Contribution

Our goal is to propose a bounded CCA2 secure post-quantum PRE scheme with compact ciphertexts. In the game of bounded CCA2 security of PRE, the numbers of queries which the adversary can issue to the decryption and re-encryption oracles are at most a-priori parameters t_d and t_r, respectively. Although bounded CCA2 security is a weak variant of CCA2 security, there are practical applications where PKE's bounded CCA2 security introduced in [13] is sufficient (e.g., see [13, 20, 29, 30]). Compact ciphertexts for bounded CCA2 secure PRE mean that ciphertext size is independent of the parameters t_d, t_r.

To achieve our goal, we propose a generic construction of bounded CCA2 secure PRE. This construction is based on the generic construction of bounded CCA2 secure PKE [13], and its building blocks are PRE with a variant of CPA security and one-time signatures (OTSs). To achieve compact ciphertexts, we require the underlying PRE to satisfy an additional property. Moreover, we present a lattice-based PRE scheme with this additional property, so that we can instantiate our generic construction. Details on our contribution are as follows:

- We formalize a notion of bounded CCA2 security for single-hop unidirectional PRE. This formalization is based on the definition of bounded CCA2 security for PKE [13]. As mentioned before, the maximum numbers of queries which the adversary can issue to the decryption and re-encryption oracles are bounded by a-priori parameters t_d and t_r, respectively.

- As a new property of PRE, we introduce re-encrytption key homomorphism in order to construct the objective bounded CCA2 secure PRE scheme. Due to this property, we also formalize a new security notion: RKH-CPA security, which is a variant of CPA security.

- We propose a generic construction of bounded CCA2 secure (single-hop unidirectional) PRE with compact ciphertexts. This is based on the bounded CCA2 secure PKE scheme [13]. The building blocks of our scheme are re-encryption key homomorphic PRE with RKH-CPA security and strongly unforgeable OTS. An overview of this construction appears in Section 1.3.

- In order to instantiate our generic construction, we present a RKH-CPA secure PRE scheme with re-encryption key homomorphism, from CRYSTALS-Kyber (Kyber, for short) [6] which is a selected PKE algorithm in the NIST PQC competition. This Kyber-based scheme can convert original Kyber ciphertexts under a public key into Kyber ciphertexts under another public key. We have chosen Kyber since this is intended to be used widely as one of standard PQC algorithms.

As a result, we can obtain a bounded CCA2 secure post-quantum PRE scheme with compact ciphertexts by applying our generic construction to the Kyber-based PRE scheme. Furthermore, our Kyber-based PRE is simple and practical because this scheme is constructed just by adding the re-encryption key generation and re-encryption algorithms to the original Kyber algorithms (i.e., Kyber's key generation, encryption, and decryption algorithms). Hence, the resulting bounded CCA2 secure PRE scheme is also constructed simply.

1.3 Technical Overview

We explain an overview of technical aspects of constructing a bounded CCA2 secure PRE with compact ciphertexts.

Bounded CCA2 secure PRE from CPA secure PRE. To construct bounded CCA2 secure PRE with compact ciphertexts, we first consider a basic generic construction of bounded CCA2 secure PRE B-PRE. This construction is based on the generic construction of bounded CCA2 secure PKE [13], so that B-PRE achieves bounded CCA2 security. Furthermore, B-PRE is constructed from CPA secure PRE and OTS in order to achieve the re-encryption functionality, while the generic construction [13] starts from CPA secure PKE and OTS.

More concretely, a public key pk and a secret key sk of B-PRE consist of u public keys $\mathsf{pk}'_1, \ldots, \mathsf{pk}'_u$ and u secret keys $\mathsf{sk}'_1, \ldots, \mathsf{sk}'_u$ of the underlying PRE, respectively, where a positive integer u is a parameter of a cover-free famiy [1]. Then, for a user $i \in \{A, B\}$, let $\mathsf{pk}_i = (\mathsf{pk}'_{i,1}, \ldots, \mathsf{pk}'_{i,u})$ and $\mathsf{sk}_i = (\mathsf{sk}'_{i,1}, \ldots, \mathsf{sk}'_{i,u})$ denote the user i's public key and secret key, respectively. A ciphertext ct_A under pk_A consists of $(\mathsf{vk}_A, \mathsf{ct}'_{\mathsf{vk}_A}, \sigma_A)$, where vk_A is a verification key of the underlying OTS, $\mathsf{ct}'_{\mathsf{vk}_A} = (\mathsf{ct}'_1, \ldots, \mathsf{ct}'_v)$ is a tuple of v ciphertexts of the underlying PRE, and σ_A is an OTS signature on $\mathsf{ct}'_{\mathsf{vk}_A}$. Here, $\mathsf{ct}'_{\mathsf{vk}_A} = (\mathsf{ct}'_1, \ldots, \mathsf{ct}'_v)$ is a ciphertext associated with vk_A, as follows: Let $\alpha_1, \ldots, \alpha_v \in \{1, \ldots, u\}$ be indices determined by vk_A and a cover-free family, and ct'_i is a PRE ciphertext under pk'_{α_i} for each $i \in \{1, \ldots, v\}$. Then the correctness of the ciphertext ct_A is ensured in the same way as the PKE scheme of [13].

We consider converting ct_A into a ciphertext $\mathsf{ct}_B = (\mathsf{vk}_B, \mathsf{ct}'_{\mathsf{vk}_B}, \sigma_B)$ under pk_B. A re-encryption key of B-PRE consists of u^2 re-encryption keys $\mathsf{rk}'_{(A,1)\to(B,1)}, \ldots, \mathsf{rk}'_{(A,u)\to(B,u)}$ of the underlying PRE. Notice that each re-encryption $\mathsf{rk}_{(A,i)\to(B,j)}$ transforms a ciphertext under $\mathsf{pk}_{(A,i)}$ into a ciphertext under $\mathsf{pk}_{(B,j)}$ (where $i \in [u]$ and $j \in [u]$). When re-encrypting a ciphertext ct_A, it is possible to generate a tuple of PRE ciphertexts $\mathsf{ct}'_{\mathsf{vk}_B} = (\mathsf{ct}'_{B,1}, \ldots, \mathsf{ct}'_{B,v})$ on another OTS verification key vk_B since it is possible to convert $\mathsf{ct}'_{A,i}$ into a ciphertext $\mathsf{ct}'_{B,i}$ under pk'_{B,β_i} by using a re-encryption key $\mathsf{rk}_{(A,\alpha_i)\to(B,\beta_i)}$, where the indices $\beta_1, \ldots, \beta_v \in \{1, \ldots, u\}$ are determined by vk_B in the same way as

[1] For simplicity, we employ disjunct matrices in our PRE, instead of cover-free families. Notice that the notion of such matrices is identical to that of cover-free families.

the indices $\alpha_1, \ldots, \alpha_v$. Because a new verification/signing key-pair $(\mathsf{vk}_B, \mathsf{sigk}_B)$ of OTS is generated in the re-encryption procedure, we can generate a signature σ_B on $\mathsf{ct}'_{\mathsf{vk}_B} = (\mathsf{ct}'_{B,1}, \ldots, \mathsf{ct}'_{B,v})$ by using sigk_B. Since each $\mathsf{ct}_{B,i}$ is a valid ciphertext of the underlying PRE, the correctness of the transformed ciphertext $\mathsf{ct}_B = (\mathsf{vk}_B, \mathsf{ct}'_{\mathsf{vk}_B}, \sigma_B)$ is also ensured.

Hence, this basic scheme B-PRE achieves the re-encryption functionality. In Appendix A, this concrete construction and its security proof are given.

Bounded CCA2 secure PRE with Compact Ciphertexts. We explain an overview of our bounded CCA2 secure PRE C-PRE with compact ciphertexts, and this construction is based on the basic scheme B-PRE. The main difference between these schemes is how to create $\mathsf{ct}'_{\mathsf{vk}_A}$ when generating a ciphertext $\mathsf{ct}_A = (\mathsf{vk}_A, \mathsf{ct}'_{\mathsf{vk}_A}, \sigma_A)$ under pk_A. More concretely, $\mathsf{ct}'_{\mathsf{vk}_A}$ is an encryption of a message under a compressed public key $\mathsf{pk}'_{\mathsf{vk}_A} = \sum_{i\in\{1,\ldots,v\}} \mathsf{pk}'_{\alpha_i}$. In order to ensure the correctness of this encryption, we require the underlying PRE to satisfy an additional property: secret-key to public-key homomorphism, which was formalized in [27].

Problem of Re-encryption of Our Scheme. One may think that the secret-key to public-key homomorphism is sufficient to construct the objective PRE scheme; however, we cannot ensure the correctness of re-encrypted ciphertexts just by requiring secret-key to public-key homomorphism. A re-encryption key of B-PRE consists of $\mathsf{rk}'_{(A,1)\to(B,1)}, \ldots, \mathsf{rk}'_{(A,u)\to(B,u)}$. When re-encrypting $\mathsf{ct}_A = (\mathsf{vk}_A, \mathsf{ct}'_{\mathsf{vk}_A}, \sigma_A)$ by using re-encryption keys $\mathsf{rk}'_{(A,\alpha_1)\to(B,\beta_1)}, \ldots, \mathsf{rk}'_{(A,\alpha_v)\to(B,\beta_v)}$ in the same way as B-PRE, there are no ciphertexts $\mathsf{ct}_{A,\alpha_1}, \ldots, \mathsf{ct}_{A,\alpha_v}$ since $\mathsf{ct}'_{\mathsf{vk}_A}$ is a single ciphertext under the public key $\mathsf{pk}'_{\mathsf{vk}_A}$.

In order to resolve this, we introduce a new property of PRE called re-encryption key homomorphism. This property guarantees the homomorphic evaluation of $\mathsf{rk}'_{(A,\alpha_1)\to(B,\beta_1)}, \ldots, \mathsf{rk}'_{(A,\alpha_v)\to(B,\beta_v)}$. Intuitively, we can convert $\mathsf{ct}'_{\mathsf{vk}_A}$ under $\mathsf{pk}'_{\mathsf{vk}_A}$ into a single ciphertext $\mathsf{ct}'_{\mathsf{vk}_B}$ under $\mathsf{pk}'_{\mathsf{vk}_B} = \sum_{i\in\{1,\ldots,v\}} \mathsf{pk}'_{\beta_i}$ by using a re-encryption key $\mathsf{rk}'_{\mathsf{vk}_A\to\mathsf{vk}_B} = \sum_{i\in\{1,\ldots,v\}} \mathsf{rk}'_{(A,\alpha_i)\to(B,\beta_i)}$. Due to the introduced property, it is possible to ensure the correctness of the transformed ciphertext $\mathsf{ct}_B = (\mathsf{vk}_B, \mathsf{ct}'_{\mathsf{vk}_B}, \sigma_B)$. Furthermore, we need to consider a new security notion due to re-encryption key homomorphism, as the remaining problem. We formalize RKH-CPA security for PRE with this homomorphic property. This security is defined in the same as CPA security except that the adversary can access homomorphic re-encryption key generation oracle which returns re-encryption keys $\{\mathsf{rk}'_{(A,i)\to(B,j)}\}_{i\in\{1,\ldots,u\},j\in\{1,\ldots,u\}}$ such that the homomorphic computation such as $\sum_{i\in\{1,\ldots,v\}} \mathsf{rk}'_{(A,\alpha_i)\to(B,\beta_i)}$ is possible. Finally, we give a security proof for C-PRE by assuming the RKH-CPA security of the underlying PRE.

2 Preliminaries

Throughout this paper, we use the following notation: For a positive integer n, let $[n] := \{1, \ldots, n\}$. For n values $x_1, \ldots, x_n$ and a subset $\mathcal{I} \subseteq [n]$, let $(x_i)_{i\in\mathcal{I}}$ be a sequence and $\{x_i\}_{i\in\mathcal{I}}$ be a set of values whose indices are included in $\mathcal{I}$. For a value v, let $|v|$ be the bit-length of v. If a function $f : \mathbb{N} \to \mathbb{R}$ satisfies $f(\lambda) = o(\lambda^{-c})$ for any constant $c > 0$ and sufficiently large $\lambda \in \mathbb{N}$, then f is said to be negligible in λ and denoted by $f(\lambda) \leq \mathsf{negl}(\lambda)$. A probability is an overwhelming probability if it is at least $1 - \mathsf{negl}(\lambda)$. "Probabilistic polynomial-time" is abbreviated as PPT. For a positive integer λ, let $\mathsf{poly}(\lambda)$ be a universal polynomial of λ.

Matrices and vectors. For consistency, we use capital bold letters for matrices, non-capital letters for scalars, and bold letters for (column) vectors. For a (binary) matrix $\boldsymbol{M} \in \{0,1\}^{u\times n}$, we use the standard notation $\boldsymbol{M} = (m_{i,j})$. For a n-dimensional vector $\boldsymbol{v}$, v_i is the i-th entry, namely $\boldsymbol{v} = (v_1, \ldots, v_n)^\top$. For a binary matrix $\boldsymbol{x} \in \{0,1\}^n$, let $\mathsf{supp}(\boldsymbol{x}) := \{i \in [n] \mid x_i = 1\}$. For a binary matrix $\boldsymbol{M} = (m_{i,j}) \in \{0,1\}^{u\times n}$ and a binary vector $\boldsymbol{x} \in \{0,1\}^n$, the binary vector

$\boldsymbol{y} = \boldsymbol{M} \odot \boldsymbol{x} \in \{0,1\}^u$ is defined as $\forall i \in [u], y_i = \bigvee_{j \in [n] \text{ s.t. } m_{i,j}=1} x_j$, where $\bigvee$ is the bitwise-OR. For a binary matrix $\boldsymbol{M} = (m_{i,j}) \in \{0,1\}^{u \times n}$ and $c \in [n]$, let $\phi_{\boldsymbol{M}}(c) := \{i \in [u] \mid m_{i,c} = 1\}$.

Rings and distributions. Let $R := \mathbb{Z}[X]/(X^N + 1)$ and $R_q := \mathbb{Z}_q[X]/(X^N + 1)$, where $N = 2^{N'}$ such that $X^N + 1$ is the $2^{N'-1}$-th cyclotomic polynomial. For a set S, $s \xleftarrow{\$} S$ means that an element $s \in S$ is chosen uniformly at random. For a probability distribution D, $d \leftarrow D$ denotes that d is drawn from the distribution D. Following [6], we describe the definition of the central binomial distribution B_η for a positive integer η, as follows: B_η chooses $\{(a_i, b_i)\}_{i \in [\eta]} \xleftarrow{\$} (\{0,1\} \times \{0,1\})^\eta$ and outputs $\sum_{i=1}^{\eta}(a_i - b_i)$. Here $v \leftarrow \beta_\eta$ denotes that $v \in R$ is drawn from a distribution β_η where each of its coefficients is chosen according to B_η. In the same way as this, $\boldsymbol{v} \leftarrow \beta_\eta^k$ means that a k-dimensional vector $\boldsymbol{v} \in R^k$ is chosen from β_η^k.

Furthermore, we describe definitions of cryptographic primitives and computational assumptions used in our schemes.

2.1 Proxy Re-encryption

Following [2], we describe the syntax of (single-hop) unidirectional proxy re-encryption (PRE), as follows:

Definition 1 (Unidirectional PRE)**.** *For a security parameter λ, let $\mathcal{M} = \mathcal{M}(\lambda)$ be a message space. A (single-hop) unidirectional PRE scheme consists of six polynomial-time algorithms* (Setup, KeyGen, Enc, Dec, ReKeyGen, ReEnc)*:*

- $\mathsf{Setup}(1^\lambda) \to \mathsf{pp}$*: The randomized algorithm* Setup *takes as input a security parameter 1^λ and outputs a public parameter* pp*.*
- $\mathsf{KeyGen}(\mathsf{pp}) \to (\mathsf{pk}, \mathsf{sk})$*: The randomized algorithm* KeyGen *takes as input a public parameter* pp *and outputs a public key* pk *and a secret key* sk*. Here, both* pk *and* sk *implicitly include the public parameter* pp*.*
- $\mathsf{Enc}(\mathsf{pk}, \mathsf{m}) \to \mathsf{ct}$*: The randomized algorithm* Enc *takes as input a pubic key* pk *and a message $\mathsf{m} \in \mathcal{M}$, and outputs a ciphertext* ct*.*
- $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) \to \mathsf{m}/\perp$*: The deterministic algorithm* Dec *takes as input a secret key* sk *and a ciphertext* ct*, and outputs a message* m *or the rejection symbol $\perp$.*
- $\mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j) \to \mathsf{rk}_{i \to j}$*: The randomized or deterministic algorithm takes as input a secrete key sk_i and a public key pk_j, and outputs a re-encryption key $\mathsf{rk}_{i \to j}$.*
- $\mathsf{ReEnc}(\mathsf{rk}_{i \to j}, \mathsf{ct}_i) \to \mathsf{ct}_j$*: The randomized or deterministic algorithm* ReEnc *takes as input a re-encryption key $\mathsf{rk}_{i \to j}$ and a ciphertext ct_i, and outputs a new ciphertext ct_j.*

For simplicity, we suppose that a public parameter pp *is implicitly contained in the inputs of the algorithms* Enc, Dec, ReKeyGen, ReEnc*.*

Definition 2 (Correctness)**.** *A single-hop unidirectional PRE scheme* (Setup, KeyGen, Enc, Dec, ReKeyGen, ReEnc) *is said to be* correct *if, for every $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$, the following holds:*

Encryption Correctness. *For every $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$ and every $\mathsf{m} \in \mathcal{M}$, it holds that $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) = \mathsf{m}$ with overwhelming probability, where $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathsf{m})$.*

Re-encryption Correctness. *For every* $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KeyGen}(\mathsf{pp}), (\mathsf{pk}_j, \mathsf{sk}_j) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$*, every* $\mathsf{m} \in \mathcal{M}$*, and every* $\mathsf{rk}_{i\to j} \leftarrow \mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j)$*, it holds that* $\mathsf{Dec}(\mathsf{sk}_j, \mathsf{ct}_j) = \mathsf{m}$ *with overwhelming probability, where* $\mathsf{ct}_j \leftarrow \mathsf{ReEnc}(\mathsf{rk}_{i\to j}, \mathsf{ct}_i)$ *and* $\mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{pk}_i, \mathsf{m})$.

Following [2, 8], we describe definitions of oracles in security games of PRE.

Definition 3. *An adversary against a PRE scheme* $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{ReKeyGen}, \mathsf{ReEnc})$ *may be given access to the following oracles in a security game of PRE:*

- *Key Generation Oracle* $\mathtt{O.KeyGen}(n, \mathcal{U}_{\mathtt{Corrupt}})$*: Given a key generation query* $(n, \mathcal{U}_{\mathtt{Corrupt}})$ *such that* n *is a positive integer and* $\mathcal{U}_{\mathtt{Corrupt}}$ *is a subset of* $[n]$*, the oracle* $\mathtt{O.KeyGen}$ *computes* $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$ *for every* $i \in [n]$ *and returns* $(\{\mathsf{pk}_i\}_{i\in[n]}, \{\mathsf{sk}_i\}_{i\in\mathcal{U}_{\mathtt{Corrupt}}})$.
- *Re-Encryption Key Generation Oracle* $\mathtt{O.ReKeyGen}(i, j)$*: Given a re-encryption key query* $(i, j) \in [n] \times [n]$*, the oracle* $\mathtt{O.ReKeyGen}$ *returns* $\perp$ *if* $i \in \mathcal{U}_{\mathtt{Honest}} \wedge j \in \mathcal{U}_{\mathtt{Corrupt}}$ *or* $i = j$ *holds; otherwise, this oracle does the following:*
  - *If* $\mathsf{T}_{\mathsf{rk}}[i, j] = \mathsf{rk}_{i\to j}$*,* $\mathtt{O.ReKeyGen}$ *returns* $\mathsf{rk}_{i\to j}$*, where* T_{rk} *is the list of re-encryption key query-response pairs.*
  - *If* $\mathsf{T}_{\mathsf{rk}}[i, j] = \emptyset$*, it returns* $\mathsf{rk}_{i\to j} \leftarrow \mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j)$ *and sets* $\mathsf{T}_{\mathsf{rk}}[i, j] \leftarrow \mathsf{rk}_{i\to j}$.
- *Challenge Oracle* $\mathtt{O.Challenge}_b(i^*, \mathsf{m}_0^*, \mathsf{m}_1^*)$*: Given a challenge query* $(i^*, \mathsf{m}_0^*, \mathsf{m}_1^*)$ *(where* $i^* \in [n]$ *and* $(\mathsf{m}_0, \mathsf{m}_1) \in \mathcal{M} \times \mathcal{M}$*), the oracle* $\mathtt{O.Challenge}_b$ *with* $b \in \{0, 1\}$ *returns* $\perp$ *if* $i^* \in \mathcal{U}_{\mathtt{Corrupt}}$ *or* $|\mathsf{m}_0^*| \neq |\mathsf{m}_1^*|$ *holds, and returns* $\mathsf{ct}^* \leftarrow \mathsf{Enc}(\mathsf{pk}_i, \mathsf{m}_b^*)$ *otherwise.*
- *Decryption Oracle* $\mathtt{O.Dec}(i, \mathsf{ct}_i)$*: Given a decryption query* (i, ct_i)*, the oracle* $\mathtt{O.Dec}$ *returns* $\perp$ *if* (i, ct_i) *is a* derivative *of* (i^*, ct^*) *(Definition 4), and returns* $\mathsf{Dec}(\mathsf{sk}_i, \mathsf{ct}_i)$ *otherwise.*
- *Re-Encryption Oracle* $\mathtt{O.ReEnc}(i, j, \mathsf{ct}_i)$*: Given a re-encryption query* (i, j, ct_i)*, the oracle* $\mathtt{O.ReEnc}$ *returns* $\perp$ *if* $j \in \mathcal{U}_{\mathtt{Corrupt}}$ *and* (i, ct_i) *is a* derivative *of* (i^*, ct^*) *(Definition 4); otherwise, this oracle does the following:*
  - *If* $\mathsf{T}_{\mathsf{rk}}[i, j] = \mathsf{rk}_{i\to j}$*,* $\mathtt{O.ReEnc}$ *returns* $\mathsf{ct}_j \leftarrow \mathsf{ReEnc}(\mathsf{rk}_{i\to j}, \mathsf{ct}_i)$.
  - *If* $\mathsf{T}_{\mathsf{rk}}[i, j] = \emptyset$*, it computes* $\mathsf{rk}_{i\to j} \leftarrow \mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j)$*, returns* $\mathsf{ct}_j \leftarrow \mathsf{ReEnc}(\mathsf{rk}_{i\to j}, \mathsf{ct}_i)$*, and sets* $\mathsf{T}_{\mathsf{rk}}[i, j] \leftarrow \mathsf{rk}_{i\to j}$.

Additionally, we describe the definition of derivatives of single-hop unidirectional PRE ciphertexts in a CCA2 game, by following [8]:

Definition 4 (Derivatives in CCA2 security [8])**.** *Let* $\mathsf{PRE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{ReKeyGen}, \mathsf{ReEnc})$ *be a single-hop unidirectional PRE scheme. Suppose that the challenge ciphertext* ct^* *under a public key* pk_{i^*} *is defined in a security game of PRE.* Derivative*s of* (i^*, ct^*) *are defined as follows:*

- (i^*, ct^*) *is a* derivative *of itself.*
- *If the adversary against* PRE *has queried the re-encryption oracle* $\mathtt{O.ReEnc}$ *on input* (i, i', ct_i) *and obtained the response* $\mathsf{ct}_{i'}$*, then* $(i', \mathsf{ct}_{i'})$ *is a* derivative *of* (i, ct_i).
- *If the adversary against* PRE *has queried the re-encryption key generation oracle* $\mathtt{O.ReKeyGen}$ *on input* (i, i')*, and* $\mathsf{Dec}(\mathsf{pk}_{i'}, \mathsf{ct}_{i'}) \in \{\mathsf{m}_0^*, \mathsf{m}_1^*\}$*, then* $(i', \mathsf{ct}_{i'})$ *is a* derivative *of* (i, ct_i).

As a new security notion of PRE, we formalize a bounded variant of *security against adaptive chosen ciphertext attacks* (denoted by bounded CCA2 security) by following [8, 13].

Definition 5 (Bounded CCA2 security)**.** *Let t_d, t_r be positive integers. A single-hop unidirectional PRE scheme* $\mathsf{PRE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{ReKeyGen}, \mathsf{ReEnc})$ *is* (t_d, t_r)-CCA2 *secure if for any PPT adversary* $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2)$ *against* PRE*, its advantage* $\mathsf{Adv}^{(t_d,t_r)\text{-cca2}}_{\mathsf{PRE},\mathcal{A}}(\lambda) := \left|\Pr[\mathsf{Expt}^{(t_d,t_r)\text{cca2}}_{\mathsf{PRE},\mathcal{A}}(\lambda) = 1] - 1/2\right|$ *is negligible in* λ*, where the experiment* $\mathsf{Expt}^{(t_d,t_r)\text{cca2}}_{\mathsf{PRE},\mathcal{A}}(\lambda)$ *is defined as follows:*

$\underline{\mathsf{Expt}^{(t_d,t_r)\text{-cca2}}_{\mathsf{PRE},\mathcal{A}}(\lambda)}$:
Generate $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$;
Set $\mathsf{T}_{\mathsf{rk}} \leftarrow \emptyset$;
$(n, \mathcal{U}_{\mathtt{Corrupt}}, \mathsf{state}_0) \leftarrow \mathcal{A}_0(\lambda, \mathsf{pp})$;
Run $(\{\mathsf{pk}_i\}_{i\in[n]}, \{\mathsf{sk}_i\}_{i\in\mathcal{U}_{\mathtt{Corrupt}}}) \leftarrow \mathtt{O.KeyGen}(n, \mathcal{U}_{\mathtt{Corrupt}})$;
$(i^*, \mathsf{m}^*_0, \mathsf{m}^*_1, \mathsf{state}_1) \leftarrow \mathcal{A}_1^{\mathtt{O.ReKeyGen},\mathtt{O.Dec}\mathtt{O.ReEnc}}(\mathsf{state}_0, \{\mathsf{pk}_i\}_{i\in[n]}, \{\mathsf{sk}_i\}_{i\in\mathcal{U}_{\mathtt{Corrupt}}})$;
Sample $b \xleftarrow{\$} \{0.1\}$;
Run $\mathsf{ct}^* \leftarrow \mathtt{O.Challenge}_b(i^*, \mathsf{m}^*_0, \mathsf{m}^*_1)$;
$b' \leftarrow \mathcal{A}_2^{\mathtt{O.ReKeyGen},\mathtt{O.Dec},\mathtt{O.ReEnc}}(\mathsf{state}_1, \mathsf{ct}^*)$;
Return 1 *if* $b = b'$*; otherwise, return* 0,

where $\mathcal{A}$ *is allowed to qury at most* t_d *queries to* `O.Dec` *and at most* t_r *queries to* `O.ReEnc`*, and* $(\mathsf{state}_0, \mathsf{state}_1)$ *is internal state information.*

2.2 One-Time Signatures

We describe the syntax of one-time signatures (OTSs), as follows.

Definition 6 (OTS)**.** *For a security parameter* λ*, let* $\mathcal{M} = \mathcal{M}(\lambda)$ *be a message space. An OTS scheme consists of three polynomial-time algorithms* $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Vrfy})$*:*

- $\mathsf{KeyGen}(1^\lambda) \to (\mathsf{vk}, \mathsf{sigk})$*: The randomized algorithm* KeyGen *takes as input a security parameter* 1^λ *and outputs a verification key* vk *and a signing key* sigk*.*
- $\mathsf{Sign}(\mathsf{sigk}, \mathsf{m}) \to \sigma$*: The randomized or deterministic algorithm* Sign *takes as input a signing key* sigk *and a message* $\mathsf{m} \in \mathcal{M}$*, and outputs a signature* σ*.*
- $\mathsf{Vrfy}(\mathsf{vk}, \mathsf{m}, \sigma) \to \top/\bot$*: The deterministic algorithm* Vrfy *takes as input a verification key* vk*, a message* $\mathsf{m} \in \mathcal{M}$*, and a signature* σ*, and it outputs* $\top$ *(accept) or* $\bot$ *(reject).*

Definition 7 (Correctness)**.** *An OTS scheme* $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Vrfy})$ *is said to be* correct *if for every* $(\mathsf{vk}, \mathsf{sigk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$ *and every* $\mathsf{m} \in \mathcal{M}$*, it holds that* $\mathsf{Vrfy}(\mathsf{vk}, \mathsf{m}, \sigma) = \top$ *with overwhelming probability, where* $\sigma \leftarrow \mathsf{Sign}(\mathsf{sigk}, \mathsf{m})$*.*

As a security notion of OTSs, we describe the definition of *strong unforgeability*, as follows:

Definition 8 (Strong unforgeability)**.** *An OTS scheme* $\mathsf{OTS} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Vrfy})$ *is* strongly unforgeable *if for any PPT adversary against* OTS*, its advantage* $\mathsf{Adv}^{\text{suf-ot}}_{\mathsf{OTS},\mathcal{A}}(\lambda) = \Pr[\mathcal{A}\ \textit{wins}]$ *is negligible in* λ*, where* $[\mathcal{A}\ \textit{wins}]$ *is the event that* $\mathcal{A}$ *wins in the following security game between a challenger and* $\mathcal{A}$*:*

Setup. *The challenger generates* $(\mathsf{vk}, \mathsf{sigk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$*, sets* $\mathcal{L} \leftarrow \emptyset$*, and gives* vk *to* $\mathcal{A}$*.*

Queries. $\mathcal{A}$ *is allowed to access the signing oracle* `O.Sign`*, where* `O.Sign` *on input a signing query* $\mathsf{m} \in \mathcal{M}$ *returns* $\bot$ *if* $\mathcal{L} \neq \emptyset$*; otherwise it returns* $\sigma \leftarrow \mathsf{Sign}(\mathsf{sigk}, \mathsf{m})$ *and sets* $\mathcal{L} \leftarrow \mathcal{L} \cup \{(\mathsf{m}, \sigma)\}$*.*

Forgery. $\mathcal{A}$ *outputs a forgery* (m^*, σ^*)*.* $\mathcal{A}$ *wins if it holds that* $(\mathsf{m}^*, \sigma^*) \notin \mathcal{L}$ *and* $\mathsf{Vrfy}(\mathsf{vk}, \mathsf{m}^*, \sigma^*) = \top$*.*

2.3 Module-Learning with Errors (Module-LWE)

Following [6], we describe the definition of the Hermite normal form (HNF) variant of the module-learning with errors (MLWE) assumption, as follows:

Definition 9 (MLWE assumption)**.** *For a security parameter λ, let $n = n(\lambda), k = k(\lambda), \eta = \eta(\lambda)$ denote positive integers. The module-LWE problem is to distinguish between uniform samples $(\boldsymbol{a}_i, b_i) \in R_q^k \times R_q$ from m samples $(\boldsymbol{a}_i, b_i) \in R_q^k \times R_q$ for $i \in [m]$, where $\boldsymbol{a}_i \xleftarrow{\$} R_q^k$, $\boldsymbol{s} \xleftarrow{\$} \beta_\eta^k$, and $e_i \xleftarrow{\$} \beta_\eta$ are samples (uniformly) at random, and $b_i = \boldsymbol{a}_i^T \boldsymbol{s} + e_i$.*

The module-LWE assumption $\mathsf{MLWE}_{m,k,\eta}$ holds if for any PPT algorithm $\mathcal{A}$ solving the module-LWE problem, its advantage

$$\mathsf{Adv}^{\mathrm{mlwe}}_{m,k,\eta}(\mathcal{A}) := \left| \Pr\left[b' = 1 \;\middle|\; \begin{array}{l} \boldsymbol{A} \xleftarrow{\$} R_q^{m\times k}; (\boldsymbol{s}, \boldsymbol{e}) \leftarrow \beta_\eta^k \times \beta_\eta^m; \\ \boldsymbol{b} = \boldsymbol{A}\boldsymbol{s} + \boldsymbol{e}; b' \leftarrow \mathcal{A}(\boldsymbol{A}, \boldsymbol{b}) \end{array} \right] \right.$$
$$\left. - \Pr\left[b' = 1 \;\middle|\; \boldsymbol{A} \xleftarrow{\$} R_q^{m\times k}; \boldsymbol{b} \xleftarrow{\$} R_q^m; b' \leftarrow \mathcal{A}(\boldsymbol{A}, \boldsymbol{b}) \right] \right|$$

is negligible in λ.

2.4 Disjunct Matrices

Following [14], we describe the definition of disjunct matrices. Notice that the notion of disjunct matrices is identical to that of cover-free families.

Definition 10 (t-disjunct matrices)**.** *Let $\bar{n}, u$ be positive integers. A binary matrix $\boldsymbol{M} = (m_{i,j}) \in \{0,1\}^{u\times\bar{n}}$ is t-disjunct if for every distinct $s_1, \ldots, s_t \in [\bar{n}]$ and every $j \in [\bar{n}]\backslash\{s_1, \ldots, s_t\}$, there exists a row $q \in [u]$ such that $m_{q,j} = 1$ and $\forall j' \in \{s_1, \ldots, s_t\}$, $m_{q,j'} = 0$.*

Without loss of generality, we suppose that the hamming weight of all column vectors of a t-disjunct matrix $\boldsymbol{M}$ is some positive integer v.

For t-disjunct matrices, u and v are bounded by $u = \Omega(t^2 \log n)$ and concrete constructions with order-optimal values of u and $v = O(t \log n)$ were proposed (e.g., see [14]).

3 Bounded CCA2 secure PRE with Compact Ciphertexts

In this section, we propose a generic construction of bounded CCA2 secure PRE with compact ciphertexts. To achieve this, we formalize re-encryption key homomorphism and a security notion associated with this property. Then, we propose a generic construction starting from re-encryption key homomorphic PRE with our formalized security and strongly unforgeable OTS, and give a security proof for this construction.

3.1 Re-encryption Key Homomorphism of PRE

In order to construct a bounded CCA2 secure PRE with compact ciphertexts, we introduce *re-encryption key homomorphism* as a new property of PRE. This property is inspired by the *secret-to-public key homomorphism* defined in [27].

Definition 11 (Re-encryption key homomorphism)**.** *Let* $\mathsf{PRE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{ReKeyGen}, \mathsf{ReEnc})$ *be a PRE scheme with the secret key space $\mathcal{K}_{\mathsf{sk}} = \mathcal{K}_{\mathsf{sk}}(\lambda)$ and the public key space $\mathcal{K}_{\mathsf{pk}} = \mathcal{K}_{\mathsf{pk}}(\lambda)$ for a security parameter λ and a public parameter $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$. The PRE scheme* PRE *is said to be* re-encryption key homomorphic *if there exist the following map $\mu : \mathcal{K}_{\mathsf{sk}} \to \mathcal{K}_{\mathsf{pk}}$ and polynomial-time algorithms* $(\mathsf{HReKeyGen}, \mathsf{ReKeyEval})$*:*

- *Every* $(\mathsf{pk}, \mathsf{sk})$ *generated by* KeyGen *satisfies* $\mathsf{pk} = \mu(\mathsf{sk})$*;*
- μ *is a homomorphism: i.e., for all* $\mathsf{sk}, \mathsf{sk}' \in \mathcal{K}_{sk}$*, it holds that* $\mu(\mathsf{sk} + \mathsf{sk}') = \mu(\mathsf{sk}) \cdot \mu(\mathsf{sk}')$*;*
- $\mathsf{HReKeyGen}((\mathsf{sk}_{A_i})_{i\in[u]}, (\mathsf{pk}_{B_j})_{j\in[u]}) \to (\mathsf{rk}_{A_i\to B_j})_{i\in[u],j\in[u]}$*: The randomized algorithm* $\mathsf{HReKeyGen}$ *takes as input* u *secret keys* $(\mathsf{sk}_{A_i})_{i\in[u]}$ *and* u *public keys* $(\mathsf{pk}_{B_j})_{j\in[u]}$ *(where* $\forall i \in [u], \forall j \in [u] : A_i \neq B_j$*), and outputs* u *re-encryption keys* $(\mathsf{rk}_{A_i\to B_j})_{i\in[u],j\in[u]}$*.*
- $\mathsf{ReKeyEval}((\mathsf{rk}_{A_i\to B_i})_{i\in[u]}) \to \mathsf{rk}_{A\to B}$*: The deterministic or randomized algorithm* $\mathsf{ReKeyEval}$ *takes as input* u *re-encryption keys* $(\mathsf{rk}_{A_i\to B_i})_{i\in[u]}$ *and outputs a new re-encryption key* $\mathsf{rk}_{A\to B}$*.*
- *For every* $n = \mathsf{poly}(\lambda)$*,* $u = \mathsf{poly}(\lambda)$*, every* $\mathsf{pp} \leftarrow \mathsf{Setup}(\lambda)$*, every* $\{(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KeyGen}(\mathsf{pp})\}_{i\in[n]}$*, every* $(\mathsf{rk}_{A_i\to B_j})_{i\in[u],j\in[u]} \leftarrow \mathsf{HReKeyGen}((\mathsf{sk}_{A_i})_{i\in[u]}, (\mathsf{pk}_{B_j})_{j\in[u]})$ *(where* $\forall i \in [u], \forall j \in [u] : A_i \neq B_j \wedge A_i \in [n] \wedge B_j \in [n]$*), every* $\mathsf{rk}_{A\to B} \leftarrow \mathsf{ReKeyEval}((\mathsf{rk}_{A_i\to B_i})_{i\in[u]})$*, and every* $\mathsf{m} \in \mathcal{M}$*, it holds that* $\mathsf{Dec}(\mathsf{sk}_B, \mathsf{ct}_B) = \mathsf{m}$ *with overwhelming probability, where* $\mathsf{ct}_B \leftarrow \mathsf{ReEnc}(\mathsf{rk}_{A\to B}, \mathsf{ct}_A)$*,* $\mathsf{ct}_A \leftarrow \mathsf{Enc}(\mathsf{pk}_A, \mathsf{m})$*,* $\mathsf{pk}_A = \mu(\mathsf{pk}_{A_1}, \ldots, \mathsf{pk}_{A_u})$*, and* $\mathsf{sk}_B = \mu(\mathsf{sk}_{B_1}, \ldots, \mathsf{sk}_{B_u})$*.*

Due to the property above, we need to introduce *security against chosen plaintext attacks with re-encryption key homomorphism* (denoted by $\mathsf{RKH\text{-}CPA}$ security) as a new security notion, since the two algorithms $\mathsf{ReKeyGen}$ and $\mathsf{HReKeyGen}$ are not compatible. This security is defined in the same way as the definition of CPA security (Definition 13) except that the adversary is given access to the homomorphic re-encryption key generation oracle $\mathtt{O.HReKeyGen}$, instead of $\mathtt{O.ReKeyGen}$. We formalize $\mathsf{RKH\text{-}CPA}$ security, as follows:

Definition 12 ($\mathsf{RKH\text{-}CPA}$ security)**.** *A PRE scheme* $\mathsf{PRE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{ReKeyGen}, \mathsf{ReEnc}, \mathsf{HReKeyGen}, \mathsf{ReKeyEval})$ *with* $\mathsf{key\ homomorphism}$ *is* $\mathsf{RKH\text{-}CPA}$ *secure if for any PPT adversary* $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2)$ *against* PRE*, its advantage* $\mathsf{Adv}^{\text{rkh-cpa}}_{\mathsf{PRE},\mathcal{A}}(\lambda) := \left|\Pr[\mathsf{Expt}^{\text{rkh-cpa}}_{\mathsf{PRE},\mathcal{A}}(\lambda) = 1] - 1/2\right|$ *is negligible in* λ*, where the experiment* $\mathsf{Expt}^{\text{rkh-cpa}}_{\mathsf{PRE},\mathcal{A}}(\lambda)$ *is defined as follows:*

$\underline{\mathsf{Expt}^{\text{rkh-cpa}}_{\mathsf{PRE},\mathcal{A}}(\lambda):}$
Generate $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$;
$(n, \mathcal{U}_{\mathtt{Corrupt}}, \mathsf{state}_0) \leftarrow \mathcal{A}_0(\lambda, \mathsf{pp})$;
Run $(\{\mathsf{pk}_i\}_{i\in[n]}, \{\mathsf{sk}_i\}_{i\in\mathcal{U}_{\mathtt{Corrupt}}}) \leftarrow \mathtt{O.KeyGen}(n, \mathcal{U}_{\mathtt{Corrupt}})$;
$(i^*, \mathsf{m}^*_0, \mathsf{m}^*_1, \mathsf{state}_1) \leftarrow \mathcal{A}_1^{\mathtt{O.HReKeyGen}}(\mathsf{state}_0, \{\mathsf{pk}_i\}_{i\in[n]}, \{\mathsf{sk}_i\}_{i\in\mathcal{U}_{\mathtt{Corrupt}}})$;
Sample $b \xleftarrow{\$} \{0.1\}$ *and run* $\mathsf{ct}^* \leftarrow \mathtt{O.Challenge}_b(i^*, \mathsf{m}^*_0, \mathsf{m}^*_1)$;
$b' \leftarrow \mathcal{A}_2^{\mathtt{O.HReKeyGen}}(\mathsf{state}_1, \mathsf{ct}^*)$;
Return 1 *if* $b = b'$*; otherwise, return* 0*,*

where $(\mathsf{state}_0, \mathsf{state}_1)$ *is internal state information, and the oracle* $\mathtt{O.HReKeyGen}$ *is defined as follows:*

- *Homomorphic re-encryption key generation oracle* $\mathtt{O.HReKeyGen}$ *given a homomorphic re-encryption key query* $((A_i)_{i\in[u]}, (B_j)_{j\in[u]})$ *such that* $\forall i \in [u] : A_i \in [n] \wedge B_i \in [n]$ *returns* $\perp$ *if* $A_i \in \mathcal{U}_{\mathtt{Honest}} \wedge B_j \in \mathcal{U}_{\mathtt{Corrupt}}$ *or* $A_i = B_j$ *holds for some* $(i, j) \in [u] \times [u]$ *(where* $\mathcal{U}_{\mathtt{Honest}} = [n] \backslash \mathcal{U}_{\mathtt{Corrupt}}$*), and returns* $(\mathsf{rk}_{A_i\to B_j})_{i\in[u],j\in[u]} \leftarrow \mathsf{HReKeyGen}((\mathsf{sk}_{A_i})_{i\in[u]}, (\mathsf{pk}_{B_j})_{j\in[u]})$ *otherwise.*

Regarding the relation between CPA security and $\mathsf{RKH\text{-}CPA}$ security, it is clear that $\mathsf{RKH\text{-}CPA}$ security implies CPA security. However, it seems that CPA security does not necessarily imply $\mathsf{RKH\text{-}CPA}$ security. This is because $\mathtt{O.HReKeyGen}$ in the $\mathsf{RKH\text{-}CPA}$ game needs to return re-encryption keys such that homomorphic evaluation is possible, while $\mathtt{O.ReKeyGen}$ in the CPA game does not necessarily return such re-encryption keys.

3.2 Construction from Re-encryption Key Homomorphic PRE

We give a generic construction of bounded CCA2 secure PRE scheme C-PRE with compact ciphertexts. As described before, this is constructed from RKH-CPA secure PRE and strongly unforgeable OTS. To achieve compact ciphertexts, we require the underlying PRE scheme to be re-encryption key homomorphic (Definition 11).

In the proposed scheme C-PRE, we employ the following cryptographic primitives:

- a re-encryption key homomorphic PRE scheme $\mathsf{PRE}' = (\mathsf{PRE}'.\mathsf{Setup}, \mathsf{PRE}'.\mathsf{KeyGen}, \mathsf{PRE}'.\mathsf{Enc}, \mathsf{PRE}'.\mathsf{Dec}, \mathsf{PRE}'.\mathsf{ReKeyGen}, \mathsf{PRE}'.\mathsf{ReEnc})$ with two PPT algorithms $\mathsf{PRE}'.\mathsf{HReKeyGen}, \mathsf{PRE}'.\mathsf{ReKeyEval}$; and
- a strongly unforgeable OTS scheme $\mathsf{OTS} = (\mathsf{OTS}.\mathsf{KeyGen}, \mathsf{OTS}.\mathsf{Sign}, \mathsf{OTS}.\mathsf{Vrfy})$.

The proposed PRE scheme $\mathsf{C\text{-}PRE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{ReKeyGen}, \mathsf{ReEnc})$ is constructed as follows:

- $\mathsf{Setup}(1^\lambda) \to \mathsf{pp}$:
  - Generate $\mathsf{pp}' \leftarrow \mathsf{PRE}'.\mathsf{Setup}(\mathsf{pp})$.
  - Let $\mathcal{M} = \mathcal{M}(\lambda)$ be the message space, which is the same as that space of PRE'.
  - Let $\bar{n} = \bar{n}(\lambda), u = u(\lambda)$ be positive integers, and let $[\bar{n}]$ be the verification key space of OTS [2].
  - Let $\boldsymbol{M} = (m_{i,j}) \in \{0,1\}^{u \times \bar{n}}$ be a t-disjunct matrix.

  Output $\mathsf{pp} = (\mathsf{pp}', \bar{n}, u, \boldsymbol{M})$.
- $\mathsf{KeyGen}(\mathsf{pp}) \to (\mathsf{pk}, \mathsf{sk})$: Parse $\mathsf{pp} = (\mathsf{pp}', \bar{n}, u, \boldsymbol{M})$ and generate $(\mathsf{pk}'_i, \mathsf{sk}'_i) \leftarrow \mathsf{PRE}'.\mathsf{KeyGen}(\mathsf{pp}')$ for $i \in [u]$. Output $\mathsf{pk} = (\mathsf{pk}'_i)_{i\in[u]}$ and $\mathsf{sk} = (\mathsf{sk}'_i)_{i\in[u]}$.
- $\mathsf{Enc}(\mathsf{pk}, \mathsf{m}) \to \mathsf{ct}$:
  1. Parse $\mathsf{pk} = (\mathsf{pk}'_i)_{i\in[n]}$.
  2. Generate $(\mathsf{vk}, \mathsf{sigk}) \leftarrow \mathsf{OTS}.\mathsf{KeyGen}(1^\lambda)$.
  3. Compute $\{\tau_1, \ldots, \tau_v\} \leftarrow \phi_{\boldsymbol{M}}(\mathsf{vk})$, where all $\tau_1, \ldots, \tau_v \in [u]$ are distinct.
  4. Compute $\mathsf{pk}'_{\mathsf{vk}} \leftarrow \prod_{i\in[v]} \mathsf{pk}'_{\tau_i}$.
  5. Compute $\mathsf{ct}'_{\mathsf{vk}} \leftarrow \mathsf{PRE}'.\mathsf{Enc}(\mathsf{pk}'_{\mathsf{vk}}, \mathsf{m})$.
  6. Compute $\sigma \leftarrow \mathsf{OTS}.\mathsf{Sign}(\mathsf{sigk}, \mathsf{ct}'_{\mathsf{vk}})$.
  7. Output $\mathsf{ct} = (\mathsf{vk}, \mathsf{ct}'_{\mathsf{vk}}, \sigma)$.
- $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) \to \mathsf{m}/\perp$:
  1. Parse $\mathsf{sk} = (\mathsf{sk}'_i)_{i\in[n]}$ and $\mathsf{ct} = (\mathsf{vk}, \mathsf{ct}'_{\mathsf{vk}}, \sigma)$.
  2. Output $\perp$ if $\mathsf{OTS}.\mathsf{Vrfy}(\mathsf{vk}, \mathsf{ct}'_{\mathsf{vk}}, \sigma) = \perp$ holds.
  3. Compute $\mathsf{sk}'_{\mathsf{vk}} \leftarrow \prod_{i\in[v]} \mathsf{sk}'_{\tau_i}$, where $\{\tau_1, \ldots, \tau_v\} \leftarrow \phi_{\boldsymbol{M}}(\mathsf{vk})$.
  4. Output $\mathsf{m}' \leftarrow \mathsf{PRE}'.\mathsf{Dec}(\mathsf{sk}'_{\mathsf{vk}}, \mathsf{ct}'_{\mathsf{vk}})$.

[2] By using a collision resistant hash function, we can compress the verification key size of OTS into the space $[\bar{n}]$ in order to reduce the public key size of C-PRE.

- $\mathsf{ReKeyGen}(\mathsf{sk}_A, \mathsf{pk}_B) \to \mathsf{rk}_{A\to B}$:
  1. Parse $\mathsf{sk}_A = (\mathsf{sk}'_{A,i})_{i\in[u]}$ and $\mathsf{pk}_B = (\mathsf{pk}'_{B,i})_{i\in[u]}$.
  2. Compute $(\mathsf{rk}_{(A,i)\to(B,j)})_{i\in[u],j\in[u]} \leftarrow \mathsf{PRE}'.\mathsf{HReKeyGen}((\mathsf{sk}'_{A,i})_{i\in[u]}, (\mathsf{pk}'_{B,j})_{j\in[u]})$.
  3. Output $\mathsf{rk}_{A\to B} = (\mathsf{rk}_{(A,i)\to(B,j)})_{i\in[u],j\in[u]}$.
- $\mathsf{ReEnc}(\mathsf{rk}_{A\to B}, \mathsf{ct}_A) \to \mathsf{ct}_B$:
  1. Parse $\mathsf{rk} = (\mathsf{rk}_{(A,i)\to(B,j)})_{i\in[u],j\in[u]}$ and $\mathsf{ct}_A = (\mathsf{vk}_A, \mathsf{ct}'_{\mathsf{vk}_A}, \sigma_A)$.
  2. Output $\perp$ if $\mathsf{OTS.Vrfy}(\mathsf{vk}_A, \mathsf{ct}'_{\mathsf{vk}_A}, \sigma_A) = \perp$ holds.
  3. Generate $(\mathsf{vk}_B, \mathsf{sigk}_B) \leftarrow \mathsf{OTS.KeyGen}(1^\lambda)$.
  4. Compute $\{\alpha_1, \ldots, \alpha_v\} \leftarrow \phi_{\boldsymbol{M}}(\mathsf{vk}_A)$ and $\{\beta_1, \ldots, \beta_v\} \leftarrow \phi_{\boldsymbol{M}}(\mathsf{vk}_B)$.
  5. Compute $\mathsf{rk}_{\mathsf{vk}_A\to\mathsf{vk}_B} \leftarrow \mathsf{ReKeyEval}((\mathsf{rk}_{(A,\alpha_i)\to(B,\beta_i)})_{i\in[v]})$.
  6. Compute $\mathsf{ct}'_{\mathsf{vk}_B} \leftarrow \mathsf{PRE}'.\mathsf{ReEnc}(\mathsf{rk}_{\mathsf{vk}_A\to\mathsf{vk}_B}, \mathsf{ct}'_{\mathsf{vk}_A})$.
  7. Compute $\sigma_B \leftarrow \mathsf{OTS.Sign}(\mathsf{sigk}_B, \mathsf{ct}'_{\mathsf{vk}_B})$.
  8. Output $\mathsf{ct}_B = (\mathsf{vk}_B, \mathsf{ct}'_{\mathsf{vk}_B}, \sigma_B)$.

Due to the correctness of PRE', OTS and the re-encryption key homomorphism of PRE', the correctness of C-PRE holds. Proposition 1 shows this correctness, and we omit the proof of this proposition becuae this is proved clearly.

Proposition 1 (Correctness of C-PRE)**.** *If the PRE scheme* PRE' *is* correct *and* re-encryption key-homomorphic*, and the OTS scheme* OTS *is* correct*, then the resulting PRE scheme* C-PRE *is* correct*.*

3.3 Security Proof

The following theorem shows the bounded CCA security of C-PRE:

Theorem 1 (Security of C-PRE)**.** *Suppose that the matrix* $\boldsymbol{M} \in \{0,1\}^{u\times n}$ *is a* t*-disjunct matrix and* n_h *is a number of honest users in* (t,t)-CCA2 *game. If the PRE scheme* PRE' *is* RKH-CPA *secure, and the OTS scheme* OTS *is* strongly unforgeable*, then the resulting PRE scheme* C-PRE *is* (t,t)-CCA2 *secure.*

Particularly, if there exists a PPT algorithm $\mathcal{A}$ *against a* (t,t)-CCA2 *secure PRE scheme* C-PRE*, then there exists a PPT algorithm* $\mathcal{B}$ *against a* RKH-CPA *secure PRE scheme* PRE' *and a PPT algorithm* $\mathcal{F}$ *against* strongly unforgeable *OTS scheme* OTS*, such that*

$$\mathsf{Adv}^{(t,t)\text{-cca2}}_{\text{C-PRE},\mathcal{A}}(\lambda) \le n_h u^2 \cdot \mathsf{Adv}^{\text{kh-cpa}}_{\mathsf{PRE}',\mathcal{B}}(\lambda) + \mathsf{Adv}^{\text{suf}}_{\mathsf{OTS},\mathcal{F}}(\lambda).$$

Proof. Let $\mathcal{A}$ be a PPT adversary against the PRE scheme C-PRE. Let $\mathsf{ct}^* = (\mathsf{vk}^*, \mathsf{ct}'^*_{\mathsf{vk}^*}, \sigma^*)$ denote the challenge ciphertext. In order to prove Theorem 1, we consider security games $\mathsf{Game}_0, \mathsf{Game}_1$. For $i \in \{0,1\}$, let W_i be the event that the experiment in Game_i outputs 1.

$\underline{\mathsf{Game}_0}$: This game is the same as (t,t)-CCA2 game. Then, we have $\mathsf{Adv}^{(t,t)\text{-cca2}}_{\mathsf{PRE},\mathcal{A}}(\lambda) = |\Pr[W_0] - 1/2|$.

$\underline{\mathsf{Game}_1}$: This game is the same as Game_0 except for the following procedures of the decryption oracle `O.Dec` and the re-encryption oracle `O.ReEnc`: At the beginning of the game, the experiment generates $(\mathsf{vk}^*, \mathsf{sigk}^*) \leftarrow \mathsf{OTS.KeyGen}(1^\lambda)$. For a decryption query (i, ct_i) (resp. a re-encryption query (i, j, ct_i)) (where $\mathsf{ct}_i = (\mathsf{vk}_i, \mathsf{ct}'_{\mathsf{vk}_i}, \sigma_i)$), the experiment checks whether it holds that $\mathsf{vk}_i = \mathsf{vk}^*$,

$\mathsf{ct}_i \neq \mathsf{ct}^*$, and $\mathsf{OTS.Vrfy}(\mathsf{vk}_i, \mathsf{ct}'_{\mathsf{vk}_i}, \sigma_i) = \top$. If so, this experiment aborts; otherwise, it returns the result of $\mathtt{O.Dec}(i, \mathsf{ct}_i)$ (resp., $\mathtt{O.ReEnc}(i, j, \mathsf{ct}_i)$).

Let Bad be the event that $\mathcal{A}$ issues a decryption or re-encryption query on $\mathsf{ct}_i = (\mathsf{vk}_i, \mathsf{ct}'_{\mathsf{vk}_i}, \sigma_i)$ such that $\mathsf{vk}_i = \mathsf{vk}^*$, $\mathsf{ct}_i \neq \mathsf{ct}^*$, and $\mathsf{OTS.Vrfy}(\mathsf{vk}_i, \mathsf{ct}'_{\mathsf{vk}_i}, \sigma_i) = \top$. Then, Game_0 and Game_1 are identical unless Bad occurs. Hence, we construct a PPT algorithm $\mathcal{F}$ breaking the strongly unforgeable OTS scheme OTS so that we bound the probability $\Pr[\mathsf{Bad}]$. On input a verification key vk^* of OTS, $\mathcal{F}$ generates $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$ and gives pp to $\mathcal{A}$. Given $(n, \mathcal{U}_{\mathtt{Corrupt}})$, $\mathcal{F}$ generates $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$ for every $i \in [n]$, and returns $(\{\mathsf{pk}_i\}_{i\in[n]}, \{\mathsf{sk}_i\}_{i\in\mathcal{U}_{\mathtt{Corrupt}}})$. By using the generated key-pairs, this algorithm simulates the oracles $\mathtt{O.ReKeyGen}$, $\mathtt{O.Dec}$, and $\mathtt{O.ReEnc}$ except for the following: For a decryption query (i.e., a re-encryption query) on $\mathsf{ct}_i = (\mathsf{vk}_i, \mathsf{ct}'_{\mathsf{vk}_i}, \sigma_i)$, $\mathcal{F}$ aborts and outputs $(\mathsf{ct}'_{\mathsf{vk}_i}, \sigma_i)$ as a forgery in the strong unforgeability game, if it holds that $\mathsf{vk}_i = \mathsf{vk}^*$, $\mathsf{ct}_i \neq \mathsf{ct}^*$, and $\mathsf{OTS.Vrfy}(\mathsf{vk}_i, \mathsf{ct}'_{\mathsf{vk}_i}, \sigma_i) = \top$ (i.e., Bad occurs); otherwise, this algorithm checks whether (i, ct_i) is a derivative of the challenge ciphertext (i^*, ct^*) if (i^*, ct^*) is defined. If so, it returns $\perp$. Otherwise it returns $\mathsf{m}' \leftarrow \mathsf{Dec}(\mathsf{sk}_i, \mathsf{ct}_i)$ (resp., $\mathsf{ct}_j \leftarrow \mathsf{ReEnc}(\mathsf{rk}_{i\to j}, \mathsf{ct}_i)$).

Additionally, when $\mathcal{A}$ submits $(i^*, \mathsf{m}^*_0, \mathsf{m}^*_1)$, $\mathcal{F}$ chooses $b \xleftarrow{\$} \{0,1\}$ and computes $\mathsf{ct}'^*_{\mathsf{vk}^*}$ by following the procedure of $\mathsf{Enc}(\mathsf{pk}_{i^*}, \mathsf{m}^*_b)$. Then, this algorithm issues $\mathsf{ct}'^*_{\mathsf{vk}^*}$ to the signing oracle $\mathtt{O.Sign}$ in the strong unforgeability game and obtains σ^*. $\mathcal{F}$ returns the challenge ciphertext $\mathsf{ct}^* = (\mathsf{vk}^*, \mathsf{ct}'^*_{\mathsf{vk}^*}, \sigma^*)$. Finally, when $\mathcal{A}$ outputs $b' \in \{0,1\}$ and Bad has not occurred, $\mathcal{F}$ halts and aborts.

It is clear that the output of $\mathcal{F}$ is a valid forgery in the strong unforgeability game if Bad occurs. Additionally, $\mathcal{F}$ completely simulates the oracles in the (t,t)-CCA game since it has all key-pairs. Hence, the probability $\Pr[\mathsf{Bad}]$ is at most the advantage $\mathsf{Adv}^{\mathrm{suf}}_{\mathsf{OTS},\mathcal{F}}(\lambda)$ of $\mathcal{F}$, and we have $|\Pr[W_0] - \Pr[W_1]| \leq \mathsf{Adv}^{\mathrm{suf}}_{\mathsf{OTS},\mathcal{F}}(\lambda)$.

In order to bound the winning probability of $\mathcal{A}$ in Game_1, we construct a PPT algorithm $\mathcal{B}$ against the RKH-CPA security of PRE', as follows: On input pp' in the RKH-CPA game, $\mathcal{B}$ generates $(\mathsf{vk}^*, \mathsf{sigk}^*) \leftarrow \mathsf{KeyGen}(1^\lambda)$, sets $\mathsf{pp} = (\mathsf{pp}', \bar{n}, u, \boldsymbol{M})$, and gives pp to $\mathcal{A}$. When $\mathcal{A}$ submits the key generation query $(n, \mathcal{U}_{\mathtt{Corrupt}})$, $\mathcal{B}$ chooses $i^* \xleftarrow{\$} [n_h], j^* \xleftarrow{\$} \phi_{\boldsymbol{M}}(\mathsf{vk}^*)$, and obtains $(\{\mathsf{pk}'_{i,j}\}_{i\in[n],j\in[u]}, \{\mathsf{sk}'_{i,j}\}_{(i,j)\in[n]\times[u]\setminus\{(i^*,j^*)\}})$ by querying the key generation oracle in the RKH-CPA game. Here, for simplicity, we suppose that $(i,j) \in [n]\times[u]$ represents a user-index in the RKH-CPA game and let $\mathcal{U}_{\mathtt{Honest}} := [n]\setminus\mathcal{U}_{\mathtt{Corrupt}}$. Then $\mathcal{B}$ sets $\mathsf{pk}'_{i^*,j^*} := \mathsf{pk}'_{i^*,j^*} \cdot \left(\sum_{j\in\phi_{\boldsymbol{M}}(\mathsf{vk}^*)\setminus\{j^*\}} (\mathsf{pk}'_{i^*,j})^{-1}\right)$ and returns $(\{\mathsf{pk}_i\}_{i\in[n]}, \{\mathsf{sk}_i\}_{i\in\mathcal{U}_{\mathtt{Corrupt}}})$, where let $\mathsf{pk}_i := (\mathsf{pk}_{i,j})_{j\in[u]}$ for every $i \in [n]$, let $\mathsf{sk}_i := (\mathsf{sk}_{i,j})_{i\in[u]}$ for every $i \in [n]\setminus\{i^*\}$, and let $\mathsf{sk}_{i^*} := (\mathsf{sk}_{i^*,j})_{j\in[u]\setminus\{j^*\}}$. Additionally, $\mathcal{B}$ simulates the oracles $\mathtt{O.ReKeyGen}$, $\mathtt{O.Dec}, \mathtt{O.ReEnc}, \mathtt{O.Challenge}_b$, as follows:

- $\mathtt{O.ReKeyGen}(A, B)$: $\mathcal{B}$ checks whether $A \in \mathcal{U}_{\mathtt{Honest}} \wedge B \in \mathcal{U}_{\mathtt{Corrupt}}$ holds. If so, this algorithms returns $\perp$; otherwise, it obtains $(\mathsf{rk}_{(A,i)\to(B,j)})_{i\in[u],j\in[u]}$ by issuing $((A,i)_{i\in[u]}, (B,j)_{j\in[u]})$ to the oracle $\mathtt{O.HReKeyGen}$ in the RKH-CPA game, returns $\mathsf{rk}_{A\to B} = (\mathsf{rk}_{(A,i)\to(B,j)})_{i\in[u],j\in[u]}$, and sets $\mathsf{T}_{\mathsf{rk}}[A,B] \leftarrow \mathsf{rk}_{A\to B}$.
- $\mathtt{O.Dec}(A, \mathsf{ct}_A)$: For $\mathsf{ct}_A = (\mathsf{vk}_A, \mathsf{ct}'_A, \sigma_A)$, $\mathcal{B}$ returns $\perp$ if the challenge ciphertext is defined and ct_A is its derivative. Otherwise, this algorithm does the following:
  1. Abort and output a random bit if $A = i^* \wedge j^* \in \phi_{\boldsymbol{M}}(\mathsf{vk}_A)$ holds.
  2. Return $\perp$ if it holds that $\mathsf{vk}_A = \mathsf{vk}^*$, $\mathsf{ct}_A = \mathsf{ct}^*$, and $\mathsf{OTS.Vrfy}(\mathsf{vk}_A, \mathsf{ct}'_{\mathsf{vk}_A}, \sigma_A) = \top$.
  3. Return $\perp$ if $\mathsf{OTS.Vrfy}(\mathsf{vk}_A, \mathsf{ct}'_{\mathsf{vk}_A}, \sigma_A) = \perp$ holds.
  4. Compute $\mathsf{sk}_{\mathsf{vk}_A} \leftarrow \sum_{i\in[v]} \mathsf{sk}'_{\alpha_i}$, where $\{\alpha_1, \ldots, \alpha_v\} = \phi_{\boldsymbol{M}_A}(\mathsf{vk}_A)$.
  5. Return $\mathsf{m}' \leftarrow \mathsf{PRE}'.\mathsf{Dec}(\mathsf{sk}_{\mathsf{vk}_A}, \mathsf{ct}'_{\mathsf{vk}_A})$.

- $\mathtt{O.ReEnc}(A, B, \mathsf{ct}_A)$: For $\mathsf{ct}_A = (\mathsf{vk}_A, \mathsf{ct}'_{\mathsf{vk}_A}, \sigma_A)$, $\mathcal{B}$ returns $\perp$ if $B \in \mathcal{U}_{\mathtt{Corrupt}}$ holds and ct_A is a derivative of the challenge ciphertext. Otherwise, this algorithm does the following:
  1. Abort and output a random bit if $A = i^* \wedge j^* \in \phi_{\boldsymbol{M}}(\mathsf{vk}_A)$ holds.
  2. Return $\perp$ if it holds that $\mathsf{vk}_A = \mathsf{vk}^*$, $\mathsf{ct}_A = \mathsf{ct}^*$, and $\mathsf{OTS.Vrfy}(\mathsf{vk}_A, \mathsf{ct}'_A, \sigma_A) = \top$.
  3. Abort and output a random bit if $A = i^* \wedge j^* \in \phi_{\boldsymbol{M}}(\mathsf{vk}_B)$.
  4. Return $\perp$ if $\mathsf{OTS.Vrfy}(\mathsf{vk}_A, \mathsf{ct}'_{\mathsf{vk}_A}, \sigma_A) = \perp$ holds.
  5. Generate $(\mathsf{vk}_B, \mathsf{sigk}_B) \leftarrow \mathsf{OTS.KeyGen}(1^\lambda)$.
  6. Compute $\{\alpha_1, \ldots, \alpha_v\} \leftarrow \phi_{\boldsymbol{M}}(\mathsf{vk}_A)$, $\{\beta_1, \ldots, \beta_v\} \leftarrow \phi_{\boldsymbol{M}}(\mathsf{vk}_B)$.
  7. If $\mathsf{T}_{\mathsf{rk}}[A, B] = \emptyset$, compute $(\mathsf{rk}_{(A,i)\to(B,j)})_{i\in[u],j\in[u]} \leftarrow \mathsf{HReKeyGen}((\mathsf{sk}_{A,i})_{i\in[u]}, (\mathsf{pk}_{B,j})_{j\in[u]})$. If $\mathsf{T}_{\mathsf{rk}}[A, B] = \mathsf{rk}_{A\to B}$, parse $\mathsf{rk}_{A\to B} = (\mathsf{rk}_{(A,i)\to(B,j)})_{i\in[u],j\in[u]}$.
  8. Compute $\mathsf{rk}_{\mathsf{vk}_A\to\mathsf{vk}_B} \leftarrow \sum_{i\in[v]} \mathsf{rk}_{(A,\alpha_i)\to(B,\beta_i)}$.
  9. Compute $\mathsf{ct}'_{\mathsf{vk}_B} \leftarrow \mathsf{ReEnc}(\mathsf{rk}_{\mathsf{vk}_A\to\mathsf{vk}_B}, \mathsf{ct}'_{\mathsf{vk}_A})$.
  10. Compute $\sigma_B \leftarrow \mathsf{OTS.Sign}(\mathsf{sigk}_B, \mathsf{ct}'_{\mathsf{vk}_B})$.
  11. Return $\mathsf{ct}_B = (\mathsf{vk}_B, \mathsf{ct}'_{\mathsf{vk}_B}, \sigma_B)$.
- $\mathtt{O.Challenge}_b(i', \mathsf{m}^*_0, \mathsf{m}^*_1)$: $\mathcal{B}$ does the following:
  1. Abort and output a random bit if $i^* \neq i'$ holds.
  2. Obtain the ciphertext $\mathsf{ct}'^*_{\mathsf{vk}^*}$ by issuing $((i^*, j^*), \mathsf{m}^*_0, \mathsf{m}^*_1)$ to the challenge oracle in the RKH-CPA game.
  3. Compute $\sigma^* \leftarrow \mathsf{OTS.Sign}(\mathsf{sigk}^*, \mathsf{ct}'^*_{\mathsf{vk}^*})$.
  4. Return $\mathsf{ct}^* = (\mathsf{vk}^*, \mathsf{ct}'^*_{\mathsf{vk}^*}, \sigma^*)$.

Finally, when $\mathcal{A}$ outputs $b' \in \{0, 1\}$, $\mathcal{B}$ also outputs b'.

We analyze the algorithm $\mathcal{B}$. $\mathcal{B}$ simulates the environment of $\mathcal{A}$ unless $\mathcal{B}$ aborts in the simulation of the oracles $\mathtt{O.ReEnc}, \mathtt{O.Dec}, \mathtt{O.Challenge}_b$. To estimate the winning probability of $\mathcal{B}$, we define Abort as the event that this algorithm aborts in the simulation above (namely, $A = i^* \wedge j^* \in \phi_{\boldsymbol{M}}(\mathsf{vk}_A)$ holds in the oracle $\mathtt{O.Dec}$ or $\mathtt{O.ReEnc}$, or $i^* \neq i'$ holds in the oracle $\mathtt{O.Challenge}$). Additionally, let $W_{\mathcal{B}}$ denote the event that $\mathcal{B}$ outputs a bit $b' \in \{0, 1\}$ such that $b = b'$. Then, $\Pr[W_{\mathcal{B}} \mid \mathsf{Abort}] = 1/2$ and $\Pr[\neg\mathsf{Abort}] \geq 1/(n_h u^2)$ hold since Abort can occur in the simulation of the oracles $\mathtt{O.Dec}, \mathtt{O.ReEnc}, \mathtt{O.Challenge}_b$. Hence, we have

$$\begin{aligned}
\Pr[W_{\mathcal{B}}] &= \Pr[\mathsf{Abort}] \cdot \Pr[W_{\mathcal{B}} \mid \mathsf{Abort}] + \Pr[\neg\mathsf{Abort}] \cdot \Pr[W_{\mathcal{B}} \mid \neg\mathsf{Abort}] \\
&\geq \frac{1}{2}\left(1 - \frac{1}{n_h u^2}\right) + \frac{1}{n_h u^2} \cdot \Pr[W_{\mathcal{B}} \mid \neg\mathsf{Abort}] \\
&= \frac{1}{2} + \frac{1}{n_h u^2}\left(\Pr[W_{\mathcal{B}} \mid \neg\mathsf{Abort}] - \frac{1}{2}\right).
\end{aligned}$$

Since the $\mathcal{A}$'s advantage $\varepsilon_{\mathcal{A}}$ in Game_1 is equivalent to $|\Pr[W_{\mathcal{B}} \mid \neg\mathsf{Abort}] - 1/2|$, the $\mathcal{B}$'s advantage $\varepsilon_{\mathcal{B}} = |\Pr[W_{\mathcal{B}}] - 1/2|$ is at least $\varepsilon_{\mathcal{A}}/(n_h u^2)$.

From the above discussion, we obtain

$$\mathsf{Adv}^{(t,t)\text{-cca2}}_{\mathsf{C\text{-}PRE},\mathcal{A}}(\lambda) \leq n_h u^2 \cdot \mathsf{Adv}^{\text{kh-cpa}}_{\mathsf{PRE}',\mathcal{B}}(\lambda) + \mathsf{Adv}^{\text{suf}}_{\mathsf{OTS},\mathcal{F}}(\lambda),$$

and complete the proof. □

4 Re-Encryption Key Homomorphic PRE from Kyber

In order to instantiate our generic construction with compact ciphertexts, we give a Kyber-based PRE scheme K-PRE with re-encryption key homomorphism and prove that K-PRE is RKH-CPA secure. Concretely, the algorithms Setup, KeyGen, Enc, Dec of K-PRE are the same as those of Kyber [6], and then we add the algorithms ReKeyGen, ReEnc, HReKeyGen, ReKeyEval in order to guarantee the re-encryption functionality and re-encryption key homomorphic property of PRE.

To construct the PRE scheme K-PRE, we employ the following functions:

- The compression functions used in Kyber [6]:
  - $\mathsf{Compress}_q$: For $x \in \mathbb{Z}_q$ and $d \in \mathbb{Z}$, the compression function $\mathsf{Compress}_q$ with a parameter $q \in \mathbb{Z}$ is defined as $\mathsf{Compress}_q(x, d) := \lceil (2^d/q) \cdot x \rfloor \mod 2^d$.
  - $\mathsf{Decompress}_q$: For $x \in \mathbb{Z}_q$ and $d \in \mathbb{Z}$, the compression function $\mathsf{Decompress}_q$ with a parameter $q \in \mathbb{Z}$ is defined as $\mathsf{Decompress}_q(x, d) := \lceil (q/2^d) \cdot x \rfloor$.
- The bit-decomposition algorithm BitDecomp given a vector $x \in \mathbb{Z}_q^N$ decomposes x into its bit representation.
- The powers-of-two algorithm Powersof2 with $\ell = \lceil \log q \rceil$, on input a (column) vector $s \in \mathbb{Z}_q^N$, outputs $(1, 2, \ldots, 2^\ell)^\top \otimes s = (s, 2s, \ldots, 2^{\ell-1}s) \in \mathbb{Z}_q^{N\ell}$, where $\otimes$ is the standard tensor product.

We describe the PRE scheme K-PRE = (Setup, KeyGen, Enc, Dec, ReKeyGen, ReEnc) with (HReKeyGen, ReKeyEval), as follows:

- $\mathsf{Setup}(1^\lambda) \to \mathsf{pp}$:
  - Let $\mathcal{M} = \{0,1\}^\mu$ be the message space, where $\mu = \mu(\lambda)$ is a positive integer.
  - For positive integers $N = N(\lambda), N' = N'(\lambda)$ and a prime q, R and R_q are defined as $R := \mathbb{Z}[X]/(X^N + 1)$ and $R_q := \mathbb{Z}_q[X]/(X^N + 1)$, respectively, where $N = 2^{N'-1}$ such that $X^N + 1$ is the $2^{N'}$-th cyclotomic polynomial (e.g., $N = 256, N' = 9$).
  - Let $\ell := \lceil \log q \rceil$.
  - For some positive integer η, β_η is a distribution where each coefficient of a sample is generated from B_η.
  - Let k, d_t, d_u, d_v be positive integers.
  - Sample $\boldsymbol{A} \xleftarrow{\$} R_q^{k \times k}$.

  Output $\mathsf{pp} = (\lambda, \mu, N, N', q, \ell, \eta, k, d_t, d_u, d_v, \boldsymbol{A})$.
- $\mathsf{KeyGen}(\mathsf{pp}) \to (\mathsf{pk}, \mathsf{sk})$:
  1. Parse $\mathsf{pp} = (\lambda, \mu, N, N', q, \ell, \eta, k, d_t, d_u, d_v, \boldsymbol{A})$.
  2. Sample $(\boldsymbol{s}, \hat{\boldsymbol{s}}) \leftarrow \beta_\eta^k \times \beta_\eta^k$ and $(\boldsymbol{e}, \hat{\boldsymbol{e}}) \leftarrow \beta_\eta^k \times \beta_\eta^k$.
  3. Compute $\boldsymbol{t} \leftarrow \mathsf{Compress}_q(\boldsymbol{A}\boldsymbol{s} + \boldsymbol{e}, d_t)$ and $\hat{\boldsymbol{t}} \leftarrow \mathsf{Compress}_q(\boldsymbol{A}\hat{\boldsymbol{s}} + \hat{\boldsymbol{e}}, d_t)$.
  4. Output $\mathsf{pk} = (\boldsymbol{t}, \hat{\boldsymbol{t}})$ and $\mathsf{sk} = (\boldsymbol{s}, \hat{\boldsymbol{s}})$.
- $\mathsf{Enc}(\mathsf{pk}, \mathsf{m}) \to \mathsf{ct}$:
  1. Parse $\mathsf{pk} = (\boldsymbol{t}, \hat{\boldsymbol{t}})$.

2. Compute $\boldsymbol{t} \leftarrow \mathsf{Decompress}_q(\boldsymbol{t}, d_t)$.
3. Sample $(\boldsymbol{r}, \boldsymbol{e}_1, e_2) \leftarrow \beta_\eta^k \times \beta_\eta^k \times \beta_\eta$.
4. Compute $\boldsymbol{u} \leftarrow \mathsf{Compress}_q(\boldsymbol{A}^\top \boldsymbol{r} + \boldsymbol{e}_1^\top, d_u)$.
5. Compute $v \leftarrow \mathsf{Compress}_q(\boldsymbol{t}^\top \boldsymbol{r} + e_2 + \lceil \frac{q}{2} \rfloor \cdot \mathsf{m}, d_v)$.
6. Output $\mathsf{ct} = (\boldsymbol{u}, v)$.

- $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) \to \mathsf{m}$:

1. Parse $\mathsf{sk} = (\boldsymbol{s}, \hat{\boldsymbol{s}})$ and $\mathsf{ct} = (\boldsymbol{u}, v)$.
2. Compute $\boldsymbol{u} \leftarrow \mathsf{Decompress}_q(\boldsymbol{u}, d_u)$ and $v \leftarrow \mathsf{Decompress}_q(v, d_v)$.
3. Output $\mathsf{m} \leftarrow \mathsf{Compress}_q(v - \boldsymbol{s}^\top \boldsymbol{u}, 1)$.

- $\mathsf{ReKeyGen}(\mathsf{sk}_A, \mathsf{pk}_B) \to \mathsf{rk}_{A \to B}$:

1. Parse $\mathsf{sk}_A = (\boldsymbol{s}_A, \hat{\boldsymbol{s}}_A)$ and $\mathsf{pk}_B = (\boldsymbol{t}_B, \hat{\boldsymbol{t}}_B)$.
2. Compute $\hat{\boldsymbol{t}}_B \leftarrow \mathsf{Decompress}_q(\hat{\boldsymbol{t}}_B, d_t)$.
3. Choose $\boldsymbol{R}_{A\to B,1}, \boldsymbol{R}_{A\to B,2} \leftarrow \beta_\eta^{k \times k\ell}$ and $\boldsymbol{r}_{A\to B,3} \leftarrow \beta_\eta^k$.
4. Compute $\boldsymbol{U}_{A\to B} \leftarrow \boldsymbol{A}^\top \boldsymbol{R}_{A\to B,1} + \boldsymbol{R}_{A\to B,2} \in R_q^{k \times k\ell}$.
5. Compute $\boldsymbol{v}_{A\to B} \leftarrow \hat{\boldsymbol{t}}_B^\top \boldsymbol{R}_{A\to B,1} + \boldsymbol{r}_{A\to B,3}^\top - \mathsf{Powersof2}(\boldsymbol{s}_A^\top) \in R_q^{k\ell}$.
6. Output $\mathsf{rk}_{A\to B} = (\boldsymbol{U}_{A\to B}, \boldsymbol{v}_{A\to B})$.

- $\mathsf{ReEnc}(\mathsf{rk}_{A\to B}, \mathsf{ct}_A) \to \mathsf{ct}_B$:

1. Parse $\mathsf{rk}_{A\to B} = (\boldsymbol{U}_{A\to B}, \boldsymbol{v}_{A\to B})$ and $\mathsf{ct}_A = (\boldsymbol{u}_A, v_A)$.
2. Compute $\boldsymbol{u}_A \leftarrow \mathsf{Decompress}_q(\boldsymbol{u}_A, d_u)$ and $v_A \leftarrow \mathsf{Decompress}_q(v_A, d_v)$.
3. Compute $\boldsymbol{u}_B \leftarrow \boldsymbol{U}_{A\to B} \cdot \mathsf{BitDecomp}(\boldsymbol{u}_A)$.
4. Compute $v_B \leftarrow v_A + \boldsymbol{v}_{A\to B}^\top \cdot \mathsf{BitDecomp}(\boldsymbol{u}_A)$.
5. Compute $\boldsymbol{u}_B \leftarrow \mathsf{Compress}_q(\boldsymbol{u}_B, d_u)$ and $v_B \leftarrow \mathsf{Compress}_q(v_B, d_v)$.
6. Output $\mathsf{ct}_B = (\boldsymbol{u}_B, v_B)$.

- $\mathsf{HReKeyGen}((\mathsf{sk}_{A_i})_{i\in[u]}, (\mathsf{pk}_{B_j})_{j\in[u]}) \to (\mathsf{rk}_{A_i\to B_j})_{i\in[u],j\in[u]}$:

1. Parse $\mathsf{sk}_{A_i} = (\boldsymbol{s}_{A_i}, \hat{\boldsymbol{s}}_{A_i})$ and $\mathsf{pk}_{B_j} = (\boldsymbol{t}_{B_j}, \hat{\boldsymbol{t}}_{B_j})$ for every $i \in [u]$ and $j \in [u]$.
2. Compute $\hat{\boldsymbol{t}}_{B_j} \leftarrow \mathsf{Decompress}_q(\hat{\boldsymbol{t}}_{B_j}, d_t)$ for every $j \in [u]$.
3. Choose $\boldsymbol{R}_{A\to B,1}, \boldsymbol{R}_{A\to B,2} \leftarrow \beta_\eta^{k \times k\ell}$.
4. Choose $\boldsymbol{r}_{A_i\to B_j} \leftarrow \beta_\eta^{k\times 1}$ for every $i \in [u]$ and $j \in [u]$.
5. Compute $\boldsymbol{U}_{A\to B} \leftarrow \boldsymbol{A}^\top \boldsymbol{R}_{A\to B,1} + \boldsymbol{R}_{A\to B,2}$.
6. Compute $\boldsymbol{v}_{A_i\to B_j} \leftarrow \hat{\boldsymbol{t}}_{B_j}^\top \boldsymbol{R}_{A\to B,1} + \boldsymbol{r}_{A_i\to B_j} - \mathsf{Powersof2}(\boldsymbol{s}_{A_i}^\top)$ for every $i \in [u]$ and every $j \in [u]$.
7. Output $(\mathsf{rk}_{A_i\to B_j})_{i\in[u],j\in[u]}$, where $\mathsf{rk}_{A_i\to B_j} = (\boldsymbol{U}_{A\to B}, \boldsymbol{v}_{A_i\to B_j})$.

- $\mathsf{ReKeyEval}((\mathsf{rk}_{A_i\to B_i})_{i\in[u]}) \to \mathsf{rk}_{A\to B}$:

1. Parse $\mathsf{rk}_{A_i\to B_i} = (\boldsymbol{U}_{A\to B}, \boldsymbol{v}_{A_i\to B_i})$ for every $i \in [u]$.

2. Compute $\boldsymbol{v}_{A\to B} \leftarrow \sum_{i\in[u]} \boldsymbol{v}_{A_i\to B_i} \in R_q^{k\ell}$.
3. Output $\mathsf{rk}_{A\to B} = (\boldsymbol{U}_{A\to B}, \boldsymbol{v}_{A\to B})$.

The scheme is correct and re-encryption key homomorphic with overwhelming probability. The formal propositions and the proofs of these properties appear in Appendix B. We give informal propositions, as follows:

Proposition 2 (Correctness of K-PRE (informal))**.** *The proposed PRE scheme* K-PRE *is* correct *with overwhelming probability, under a suitable parameter setting by the algorithm* Setup*.*

Proposition 3 (Re-encryption key homomorphism of K-PRE (informal))**.** *The proposed PRE scheme* K-PRE *is* re-encrytpion key homomorphic *with overwhelming probability, under a suitable parameter setting by the algorithm* Setup*.*

Furthermore, the following theorem shows the security of K-PRE:

Theorem 2 (RKH-CPA security of K-PRE)**.** *If the* $\mathsf{MLWE}_{k+1,k\ell,\eta}$ *assumption holds, the proposed PRE scheme* K-PRE *is* RKH-CPA *secure.*

In particular, if there exists a PPT adversary $\mathcal{A}$ *against the* RKH-CPA *security of* K-PRE*, then there exists a PPT algorithm* $\mathcal{B}$ *against the* $\mathsf{MLWE}_{k+1,k\ell,\eta}$ *problem, such that*

$$\mathsf{Adv}^{\text{rkh-cpa}}_{\text{K-PRE},\mathcal{A}}(\lambda) \le n_h(q_{rk}k+3)\cdot \mathsf{Adv}^{\text{mlwe}}_{k+1,k\ell,\eta}(\mathcal{B}),$$

where n_h *is the number of honest users, and* q_{rk} *is the maximum number of queries issued to the re-encryption key generation oracle.*

Proof. Let $\mathcal{A}$ denote a PPT adversary against the RKH-CPA security of the PRE scheme K-PRE. Let n $(= n_h + n_c)$ be the total number of users whose key-pairs are generated in the RKH-CPA game, where n_h and n_c are the numbers of honest users and corrupted users, respectively. Let q_{rk} be the maximum number of queries issued to the `O.HReKeyGen` oracle. The challenge ciphertext under the public key of the user $i^* \in [n]$ is denoted by $\mathsf{ct}^* = (\boldsymbol{u}^*, v^*)$. In order to prove Theorem 2, we consider security games $\mathsf{Game}_0, (\mathsf{Game}_1^{(\kappa)})_{\kappa\in[n_h]}, (\mathsf{Game}_2^{(\kappa)})_{\kappa\in[n_h]}, (\mathsf{Game}_3^{(\kappa)})_{\kappa\in[n_h]}, \mathsf{Game}_4$. For $i \in [3]$ and $\kappa \in [n_h]$, let $W_i^{(\kappa)}$ be the events that the experiment in $\mathsf{Game}_i^{(\kappa)}$ outputs 1 (i.e., $b = b'$ holds for the output $b' \in \{0,1\}$ of $\mathcal{A}$). Let W_0 and W_4 denote the events that the experiments in Game_0 and Game_4 output 1, respectively.

$\underline{\mathsf{Game}_0}$: This game is the original RKH-CPA game. Then, we have $\mathsf{Adv}^{\text{rkh-cpa}}_{\mathsf{PRE},\mathcal{A}}(\lambda) = |\Pr[W_0] - 1/2|$.

Let $\mathsf{Game}_1^{(0)}$ be the same game as Game_0. Here, without loss of generality, the index $\tau_\kappa \in \mathcal{U}_{\mathtt{Honest}}$ of an honest user is denoted by $\kappa = \tau_\kappa$ for every $\kappa \in [n_h]$. For every $\kappa \in [n_h]$, we consider a security game $\mathsf{Game}_1^{(\kappa)}$.

$\underline{\mathsf{Game}_1^{(\kappa)}}$: This game is the same as $\mathsf{Game}_1^{(\kappa-1)}$ except that $\hat{\boldsymbol{t}}_\kappa = \mathsf{Compress}_q(\boldsymbol{A}\hat{\boldsymbol{s}}_\kappa + \hat{\boldsymbol{e}}_\kappa, d_t)$ is replaced by a uniformly random $\hat{\boldsymbol{t}}_\kappa = \mathsf{Compress}_q(\hat{\boldsymbol{b}}_\kappa, d_t)$ (where $\hat{\boldsymbol{b}}_\kappa \in R_q^k$ is chosen uniformly at random), when generating the public key $\mathsf{pk}_\kappa = (\boldsymbol{t}_\kappa, \hat{\boldsymbol{t}}_\kappa)$ of the honest user κ.

Assuming the existence of $\mathcal{A}$, there exists a PPT algorithm $\mathcal{B}_1^{(\kappa)}$ against the $\mathsf{MLWE}_{k,k,\eta}$ problem, because the secret value $\hat{\boldsymbol{s}}_\kappa$ is not necessary to simulate the environments of $\mathcal{A}$ in both $\mathsf{Game}_1^{(\kappa-1)}$ and $\mathsf{Game}_1^{(\kappa)}$. By using $\mathcal{A}$'s output, $\mathcal{B}_1^{(\kappa)}$ can distinguish between a $\mathsf{MLWE}_{k,k,\eta}$ sample and a uniformly random one, in the straightforward way. Hence, we have $\left|\Pr[W_1^{(\kappa-1)}] - \Pr[W_1^{(\kappa)}]\right| \le \mathsf{Adv}^{\text{mlwe}}_{k,k,\eta}(\mathcal{B}_1^{(\kappa)})$ for every $\kappa \in [n_h]$.

Here, we define $\mathsf{Game}_2^{(0)}$ as the same game as $\mathsf{Game}_1^{(n_h)}$, and consider the security game $\mathsf{Game}_2^{(\kappa)}$ for every $\kappa \in \mathcal{U}_{\mathtt{Honest}}$.

$\underline{\mathsf{Game}_2^{(\kappa)}}$: This game is the same as $\mathsf{Game}_2^{(\kappa-1)}$ except that, on input a homomorphic re-encryption key query $((A_i)_{i\in[u]}, (B_j)_{j\in[u]})$ such that $\kappa = A_i$ for some $i \in [u]$, the oracle $\mathtt{O.HReKeyGen}$ generates a uniformly random $\boldsymbol{U}_{A\to B} \in R_q^{k\times k\ell}$ and a uniformly random $\boldsymbol{v}_{A_i\to B_j} \in R_q^{k\ell}$ for every $i \in [u]$ and $j \in [u]$, instead of $\boldsymbol{U}_{A\to B} \leftarrow \boldsymbol{A}^\top \boldsymbol{R}_{A\to B,1} + \boldsymbol{R}_{A\to B,2}$ and $\boldsymbol{v}_{A_i\to B_j} \leftarrow \hat{\boldsymbol{t}}_{B_j}^\top \boldsymbol{R}_{A\to B,1} + \boldsymbol{r}_{A_i\to B_j} - \mathsf{Powersof2}(\boldsymbol{s}_{A_i}^\top)$.

For each $m \in [k]$, there exists a PPT algorithm $\mathcal{B}_2^{(m,\kappa)}$ distinguishing whether the m-th row of $[\boldsymbol{U}_{A\to B} \| \boldsymbol{v}_{A_1\to B_1} \| \cdots \| \boldsymbol{v}_{A_u\to B_u}]$ is an $\mathsf{MLWE}_{1,k\ell,\eta}$ sample or uniformly random sample. Namely, there exists a PPT algorithm $\mathcal{B}_2^{(m,\kappa)}$ solving the $\mathsf{MLWE}_{1,k\ell,\eta}$ problem, by using $\mathcal{A}$. In addition, the total number of queries issued to the $\mathtt{O.HReKeyGen}$ oracle is at most q_{rk}. Hence, we have $\left|\Pr[W_2^{(\kappa-1)}] - \Pr[W_2^{(\kappa)}]\right| \leq q_{rk}k \cdot \mathsf{Adv}^{\mathrm{mlwe}}_{k,k\ell,\eta}(\mathcal{B}_2^{(\kappa)})$ by letting $\mathcal{B}_2^{(\kappa)}$ be the PPT algorithm against the $\mathsf{MLWE}_{1,k\ell,\eta}$ assumption, such that $\mathsf{Adv}^{\mathrm{mlwe}}_{1,k\ell,\eta}(\mathcal{B}_2^{(m,\kappa)}) \leq \mathsf{Adv}^{\mathrm{mlwe}}_{k,k\ell,\eta}(\mathcal{B}_2^{(\kappa)})$ for all $m \in [k]$.

Here, let $\mathsf{Game}_3^{(0)}$ be the same game as $\mathsf{Game}_2^{(n_h)}$, and we define the security game $\mathsf{Game}_3^{(\kappa)}$ for every $\kappa \in \mathcal{U}_{\mathtt{Honest}}$.

$\underline{\mathsf{Game}_3^{(\kappa)}}$: This game is the same as $\mathsf{Game}_3^{(\kappa-1)}$ except that $\boldsymbol{t}_\kappa = \mathsf{Compress}_q(\boldsymbol{A}^\top \boldsymbol{s}_\kappa + \boldsymbol{e}_\kappa, d_t)$ is replaced by a uniformly random $\boldsymbol{t}_\kappa = \mathsf{Compress}_q(\boldsymbol{b}_\kappa, d_t)$ (where $\boldsymbol{b}_\kappa \in R_q^k$ is chosen uniformly at random), when generating the public key $\mathsf{pk}_\kappa = (\boldsymbol{t}_\kappa, \hat{\boldsymbol{t}}_\kappa)$ of the honest user $\kappa \in \mathcal{U}_{\mathtt{Honest}}$.

There exists a PPT algorithm $\mathcal{B}_3^{(\kappa)}$ against the $\mathsf{MLWE}_{k,k,\eta}$ problem. Since $\boldsymbol{s}_\kappa$ is not used in both $\mathsf{Game}_3^{(\kappa-1)}$ and $\mathsf{Game}_3^{(\kappa)}$, it is possible to simulate the views of $\mathcal{A}$ in the two games and construct $\mathcal{B}_3^{(\kappa)}$. Hence, we have $\left|\Pr[W_3^{(\kappa-1)}] - \Pr[W_3^{(\kappa)}]\right| \leq \mathsf{Adv}^{\mathrm{mlwe}}_{k,k,\eta}(\mathcal{B}_3^{(\kappa)})$.

$\underline{\mathsf{Game}_4}$: This game is the same as $\mathsf{Game}_3^{(n_h)}$ except that the challenge ciphertext $\mathsf{ct}^* = (\boldsymbol{u}^*, v^*) \leftarrow \mathsf{Enc}(\mathsf{pk}_{i^*}, \mathsf{m}_b^*)$ is replaced by a uniformly random $(\boldsymbol{u}^*, v^*) \xleftarrow{\$} R_q^k \times R_q$.

The secret key $(\boldsymbol{s}_\kappa, \hat{\boldsymbol{s}}_\kappa)$ for every $\kappa \in [n_h]$ is not used in both $\mathsf{Game}_3^{(n_h)}$ and Game_4. Thus, in these games, it is possible to simulate the environments of $\mathcal{A}$ without using that secret key, and construct a PPT algorithm $\mathcal{B}_4^{(i^*)}$ against the $\mathsf{MLWE}_{k,k,\eta}$ problem. Hence, we have $\left|\Pr[W_3^{(n_h)}] - \Pr[W_4]\right| \leq n_h \cdot \mathsf{Adv}^{\mathrm{mlwe}}_{k+1,k,\eta}(\mathcal{B}_4^{(i^*)})$. Furthermore, $\Pr[W_4] = 1/2$ holds since the challenge ciphertext ct^* is independent of $b \in \{0,1\}$ in Game_4.

Let $\mathcal{B}$ be a PPT algorithm against the $\mathsf{MLWE}_{k+1,k\ell,\eta}$ problem, such that it holds that $\mathsf{Adv}^{\mathrm{mlwe}}_{k+1,k\ell,\eta}(\mathcal{B}_i^{(\kappa)}) < \mathsf{Adv}^{\mathrm{mlwe}}_{k+1,k\ell,\eta}(\mathcal{B})$ for all $i \in [4]$ and all $\kappa \in \mathcal{U}_{\mathtt{Honest}}$. From the discussion above, we obtain

$$\begin{aligned}
\mathsf{Adv}^{\mathrm{rkh\text{-}cpa}}_{\mathsf{K\text{-}PRE},\mathcal{A}}(\lambda) \leq{}& \sum_{i=1}^{3}\sum_{\kappa=1}^{n_h} \left|\Pr[W_i^{(\kappa-1)}] - \Pr[W_i^{(\kappa)}]\right| \\
&+ \left|\Pr[W_3^{(n_h)}] - \Pr[W_4]\right| + \left|\Pr[W_4] - \frac{1}{2}\right| \\
\leq{}& n_h(q_{rk}k + 3) \cdot \mathsf{Adv}^{\mathrm{mlwe}}_{k+1,k\ell,\eta}(\mathcal{B}).
\end{aligned}$$

This completes the proof. □

5 Conclusion

We aimed at constructing a bounded CCA2 secure post-quantum PRE scheme with compact ciphertexts. To this end, we formalized the notions of re-encryption key homomorphism and RKH-CPA

security for PRE, and proposed a generic construction of bounded CCA2 secure PRE with compact ciphertexts, which starts from re-encryption key homomorphic PRE with RKH-CPA security and strongly unforgeable OTS. To instantiate this generic construction, we presented a Kyber-based re-encryption key homomorphic PRE scheme with RKH-CPA security.

As a result, we can construct a bounded CCA2 secure post-quantum PRE scheme with compact ciphertexts by applying the generic construction to our Kyber-based PRE. The resulting bounded CCA2 PRE scheme from Kyber is simple because our Kyber-based PRE is constructed without changing the original Kyber's key generation, encryption, and decryption algorithms.

Although we just discussed single-hop PRE schemes, we can consider a multi-hop variant of our PRE schemes C-PRE, K-PRE as an extension of these schemes.

Acknowledgements. This research was conducted under a contract of "Research and development on new generation cryptography for secure wireless communication services" among "Research and Development for Expansion of Radio Wave Resources (JPJ000254)", which was supported by the Ministry of Internal Affairs and Communications, Japan.

References

[1] J. B. Almeida, S. A. Olmos, M. Barbosa, G. Barthe, F. Dupressoir, B. Grégoire, V. Laporte, J. Léchenet, C. Low, T. Oliveira, H. Pacheco, M. Quaresma, P. Schwabe, and P. Strub. Formally verifying kyber - episode V: machine-checked IND-CCA security and correctness of ML-KEM in easycrypt. In *CRYPTO (2)*, volume 14921 of *LNCS*, pages 384–421. Springer, 2024.

[2] G. Ateniese, K. Benson, and S. Hohenberger. Key-private proxy re-encryption. In *CT-RSA*, volume 5473 of *LNCS*, pages 279–294. Springer, 2009.

[3] G. Ateniese, K. Fu, M. Green, and S. Hohenberger. Improved proxy re-encryption schemes with applications to secure distributed storage. In *NDSS*. The Internet Society, 2005.

[4] G. Ateniese, K. Fu, M. Green, and S. Hohenberger. Improved proxy re-encryption schemes with applications to secure distributed storage. *ACM Trans. Inf. Syst. Secur.*, 9(1):1–30, 2006.

[5] M. Blaze, G. Bleumer, and M. Strauss. Divertible protocols and atomic proxy cryptography. In *EUROCRYPT*, volume 1403 of *LNCS*, pages 127–144. Springer, 1998.

[6] J. W. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanck, P. Schwabe, G. Seiler, and D. Stehlé. CRYSTALS - kyber: A cca-secure module-lattice-based KEM. In *EuroS&P*, pages 353–367. IEEE, 2018.

[7] S. Canard, J. Devigne, and F. Laguillaumie. Improving the security of an efficient unidirectional proxy re-encryption scheme. *J. Internet Serv. Inf. Secur.*, 1(2/3):140–160, 2011.

[8] R. Canetti and S. Hohenberger. Chosen-ciphertext secure proxy re-encryption. In *CCS*, pages 185–194. ACM, 2007.

[9] N. Chandran, M. Chase, F. Liu, R. Nishimaki, and K. Xagawa. Re-encryption, functional re-encryption, and multi-hop re-encryption: A framework for achieving obfuscation-based security and instantiations from lattices. In *Public Key Cryptography*, volume 8383 of *LNCS*, pages 95–112. Springer, 2014.

[10] N. Chandran, M. Chase, and V. Vaikuntanathan. Functional re-encryption and collusion-resistant obfuscation. In *TCC*, volume 7194 of *LNCS*, pages 404–421. Springer, 2012.

[11] S. S. M. Chow, J. Weng, Y. Yang, and R. H. Deng. Efficient unidirectional proxy re-encryption. In *AFRICACRYPT*, volume 6055 of *LNCS*, pages 316–332. Springer, 2010.

[12] A. Cohen. What about bob? the inadequacy of CPA security for proxy reencryption. In *Public Key Cryptography (2)*, volume 11443 of *LNCS*, pages 287–316. Springer, 2019.

[13] R. Cramer, G. Hanaoka, D. Hofheinz, H. Imai, E. Kiltz, R. Pass, A. Shelat, and V. Vaikuntanathan. Bounded cca2-secure encryption. In *ASIACRYPT*, volume 4833 of *LNCS*, pages 502–518. Springer, 2007.

[14] D.-Z. Du and F. K. Hwang. *Combinatorial Group Testing and Its Applications (2nd Edition)*, volume 12 of *Series on Applied Mathematics*. World Scientific, 2000.

[15] J. Duman, K. Hövelmanns, E. Kiltz, V. Lyubashevsky, and G. Seiler. Faster lattice-based kems via a generic fujisaki-okamoto transform using prefix hashing. In *CCS*, pages 2722–2737. ACM, 2021.

[16] X. Fan and F. Liu. Proxy re-encryption and re-signatures from lattices. In *ACNS*, volume 11464 of *LNCS*, pages 363–382. Springer, 2019.

[17] G. Fuchsbauer, C. Kamath, K. Klein, and K. Pietrzak. Adaptively secure proxy re-encryption. In *Public Key Cryptography (2)*, volume 11443 of *LNCS*, pages 317–346. Springer, 2019.

[18] C. Gentry. Fully homomorphic encryption using ideal lattices. In *STOC*, pages 169–178. ACM, 2009.

[19] P. Grubbs, V. Maram, and K. G. Paterson. Anonymous, robust post-quantum public key encryption. In *EUROCRYPT (3)*, volume 13277 of *LNCS*, pages 402–432. Springer, 2022.

[20] L. Huguenin-Dumittan and S. Vaudenay. On ind-qcca security in the ROM and its applications - CPA security is sufficient for TLS 1.3. In *EUROCRYPT (3)*, volume 13277 of *LNCS*, pages 613–642. Springer, 2022.

[21] A. Ivan and Y. Dodis. Proxy cryptography revisited. In *NDSS*. The Internet Society, 2003.

[22] X. Liang, J. Weng, A. Yang, L. Yao, Z. Jiang, and Z. Wu. Attribute-based conditional proxy re-encryption in the standard model under LWE. In *ESORICS (2)*, volume 12973 of *LNCS*, pages 147–168. Springer, 2021.

[23] B. Libert and D. Vergnaud. Unidirectional chosen-ciphertext secure proxy re-encryption. In *Public Key Cryptography*, volume 4939 of *LNCS*, pages 360–379. Springer, 2008.

[24] V. Maram and K. Xagawa. Post-quantum anonymity of kyber. In *Public Key Cryptography (1)*, volume 13940 of *LNCS*, pages 3–35. Springer, 2023.

[25] P. Miao, S. Patranabis, and G. J. Watson. Unidirectional updatable encryption and proxy re-encryption from DDH. In *Public Key Cryptography (2)*, volume 13941 of *LNCS*, pages 368–398. Springer, 2023.

[26] Y. Polyakov, K. Rohloff, G. Sahu, and V. Vaikuntanathan. Fast proxy re-encryption for publish/subscribe systems. *ACM Trans. Priv. Secur.*, 20(4):14:1–14:31, 2017.

[27] S. Tessaro and D. A. Wilson. Bounded-collusion identity-based encryption from semantically-secure public-key encryption: Generic constructions with short ciphertexts. In *Public Key Cryptography*, volume 8383 of *LNCS*, pages 257–274. Springer, 2014.

[28] K. Xagawa. Anonymity of NIST PQC round 3 kems. In *EUROCRYPT (3)*, volume 13277 of *LNCS*, pages 551–581. Springer, 2022.

[29] K. Yoneyama. Compact authenticated key exchange from bounded cca-secure KEM. *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.*, 98-A(1):132–143, 2015.

[30] B. Zhou, H. Jiang, and Y. Zhao. Cpa-secure kems are also sufficient for post-quantum TLS 1.3. In *ASIACRYPT (3)*, volume 15486 of *LNCS*, pages 433–464. Springer, 2024.

[31] Y. Zhou, S. Liu, and S. Han. Multi-hop fine-grained proxy re-encryption. In *Public Key Cryptography (4)*, volume 14604 of *LNCS*, pages 161–192. Springer, 2024.

[32] Y. Zhou, S. Liu, S. Han, and H. Zhang. Fine-grained proxy re-encryption: Definitions and constructions from LWE. In *ASIACRYPT (6)*, volume 14443 of *LNCS*, pages 199–231. Springer, 2023.

A Bounded CCA secure PRE from CPA secure PRE

In this section, we propose a generic construction of bounded CCA secure PRE, which starts from any CPA secure PRE and strongly unforgeable OTS, and then give a security proof for this construction.

A.1 Building Blocks

We describe the definitions of CPA security and all-or-nothing transforms, which are used to construct the objective PRE scheme.

Following [2], we describe the definitions of *security against chosen plaintext attacks* (denoted by CPA security) for PRE.

Definition 13 (CPA security)**.** *A PRE scheme* $\mathsf{PRE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{ReKeyGen}, \mathsf{ReEnc})$ *is* CPA *secure if for any PPT adversary* $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2)$ *against* PRE*, its advantage* $\mathsf{Adv}^{\mathrm{cpa}}_{\mathsf{PRE},\mathcal{A}}(\lambda) := \left|\Pr[\mathsf{Expt}^{\mathrm{cpa}}_{\mathsf{PRE},\mathcal{A}}(\lambda) = 1] - 1/2\right|$ *is negligible in* λ*, where the experiment* $\mathsf{Expt}^{\mathrm{cpa}}_{\mathsf{PRE},\mathcal{A}}(\lambda)$ *is defined as follows:*

$\underline{\mathsf{Expt}^{\mathrm{cpa}}_{\mathsf{PRE},\mathcal{A}}(\lambda)}$:

Generate $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$;
$(n, \mathcal{U}_{\mathtt{Corrupt}}, \mathsf{state}_0) \leftarrow \mathcal{A}_0(\lambda, \mathsf{pp})$;
Run $(\{\mathsf{pk}_i\}_{i\in[n]}, \{\mathsf{sk}_i\}_{i\in\mathcal{U}_{\mathtt{Corrupt}}}) \leftarrow \mathtt{O.KeyGen}(n, \mathcal{U}_{\mathtt{Corrupt}})$;
$(i^*, \mathsf{m}^*_0, \mathsf{m}^*_1, \mathsf{state}_1) \leftarrow \mathcal{A}_1^{\mathtt{O.ReKeyGen}}(\mathsf{state}_0, \{\mathsf{pk}_i\}_{i\in[n]}, \{\mathsf{sk}_i\}_{i\in\mathcal{U}_{\mathtt{Corrupt}}})$;
Sample $b \xleftarrow{\$} \{0.1\}$ *and run* $\mathsf{ct}^* \leftarrow \mathtt{O.Challenge}_b(i^*, \mathsf{m}^*_0, \mathsf{m}^*_1)$;
$b' \leftarrow \mathcal{A}_2^{\mathtt{O.ReKeyGen}}(\mathsf{state}_1, \mathsf{ct}^*)$;
Return 1 *if* $b = b'$*; otherwise, return* 0,

where $(\mathsf{state}_0, \mathsf{state}_1)$ *is internal state information.*

All-or-Nothing Transform. An all-or-nothing transform (AONT) splits a message X into v secret shares $x_1, \ldots, x_v$ and a public share z and recovers X from the shares $(x_1, \ldots, x_v, z)$. Thus, we can regard an AONT as v-out-of-v secret sharing. We describe the definition of AONTs, as follows:

Definition 14 (AONT)**.** *An efficient randomized algorithm* Trans *is* $(\mu, \bar{\mu}, v)$-AONT *if the following conditions hold:*

1. *Given* $X \in \{0,1\}^{\mu}$*,* Trans *outputs* $v+1$ *blocks* $(x_1, \ldots, x_v, z) \in (\{0,1\}^{\bar{\mu}})^{v+1}$*, where for* $i \in [v]$*,* x_i *is a secret share, and* z *is a public share.*

2. *There exists an efficient inverse function* Inverse *which, on input* $(x_1, \ldots, x_v, z) \in (\{0,1\}^{\bar{\mu}})^{v+1}$*, outputs* $X \in \{0,1\}^{\mu}$*.*

3. Trans *satisfies* indistinguishability*, as follows: For any PPT algorithm* $\mathcal{A}$ *against* Trans*, its advantage*

$$\mathsf{Adv}^{\mathrm{ind}}_{\mathsf{Trans},\mathcal{A}}(\lambda) := \left| \Pr\left[b = b' \mid b \xleftarrow{\$} \{0,1\}; b' \leftarrow \mathcal{A}^{\mathtt{0.LR}}(1^{\lambda}) \right] - \frac{1}{2} \right|$$

is negligible in λ*, where* 0.LR *is the* left-or-right *oracle which, on input* $(j, X_0, X_1) \in [v] \times (\{0,1\}^{\mu})^2$*, returns* $(x_1, \ldots, x_{j-1}, x_{j+1}, \ldots, x_v, z)$*.*

A.2 Construction from CPA secure PRE

We present a bounded CCA secure PRE scheme B-PRE which is based on a generic construction [13] of bounded CCA secure PKE. To ensure the re-encryption functionality, we use CPA secure PRE while the PKE scheme of [13] uses CPA secure PKE. For simplicity, we employ disjunct matrices while the bound CCA secure PKE [13] uses cover-free families. Notice that the notion of disjunct matrices is identical to that of cover-free families.

In order to construct the proposed PRE scheme, we use the following building blocks:

- a CPA secure PRE scheme $\mathsf{PRE}' = (\mathsf{PRE}'.\mathsf{Setup}, \mathsf{PRE}'.\mathsf{KeyGen}, \mathsf{PRE}'.\mathsf{Enc}, \mathsf{PRE}'.\mathsf{Dec}, \mathsf{PRE}'.\mathsf{ReKeyGen}, \mathsf{PRE}'.\mathsf{ReEnc})$ with the message space $\mathcal{M}_{\mathsf{PRE}'} = \{0,1\}^{\bar{\mu}}$, where $\bar{\mu} = \bar{\mu}(\lambda)$ is a positive integer for a security parameter λ;
- a strongly unforgeable OTS scheme $\mathsf{OTS} = (\mathsf{OTS}.\mathsf{KeyGen}, \mathsf{OTS}.\mathsf{Sign}, \mathsf{OTS}.\mathsf{Vrfy})$; and
- a $(\mu, \bar{\mu}, v)$-AONT Trans with an efficient inverse function Inverse, where $\mu = \mu(\lambda)$ and $v = v(\lambda)$ are positive integers for a security parameter λ.

The proposed PRE scheme $\mathsf{B\text{-}PRE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{ReKeyGen}, \mathsf{ReEnc})$ is constructed as follows:

- $\mathsf{Setup}(1^{\lambda}) \to \mathsf{pp}$:
  - Generate $\mathsf{pp}' \leftarrow \mathsf{PRE}'.\mathsf{Setup}(\mathsf{pp})$.
  - Let $\mu = \mu(\lambda)$, $\bar{\mu} = \bar{\mu}(\lambda)$, and $v = v(\lambda)$ be positive integers.
  - Let $\mathcal{M} = \{0,1\}^{\mu}$ be the message space.
  - Let $\bar{n} = \bar{n}(\lambda), u = u(\lambda)$ be positive integers, and let $[\bar{n}]$ be the verification key-space of OTS.
  - Let $\boldsymbol{M} = (m_{i,j}) \in \{0,1\}^{u \times \bar{n}}$ be a t-disjunct matrix, where the hamming weight of each column vector is v.

  Output $\mathsf{pp} = (\mathsf{pp}', \mu, \bar{\mu}, v, \bar{n}, u, \boldsymbol{M})$.

- $\mathsf{KeyGen}(\mathsf{pp}) \to (\mathsf{pk}, \mathsf{sk})$: Parse $\mathsf{pp} = (\mathsf{pp}', \mu, \bar{\mu}, v, \bar{n}, u, \boldsymbol{M})$ and generate $(\mathsf{pk}'_i, \mathsf{sk}'_i) \leftarrow \mathsf{PRE}'.\mathsf{KeyGen}(\mathsf{pp}')$ for $i \in [u]$. Output $\mathsf{pk} = (\mathsf{pk}'_i)_{i\in[u]}$ and $\mathsf{sk} = (\mathsf{sk}'_i)_{i\in[u]}$.
- $\mathsf{Enc}(\mathsf{pk}, \mathsf{m}) \to \mathsf{ct}$:
  1. Parse $\mathsf{pk} = (\mathsf{pk}'_i)_{i\in[u]}$.
  2. Generate $(\mathsf{vk}, \mathsf{sigk}) \leftarrow \mathsf{OTS.KeyGen}(1^\lambda)$.
  3. Compute $(x_1, \ldots, x_v, z) \leftarrow \mathsf{Trans}(\mathsf{m})$.
  4. Compute $\{\tau_1, \ldots, \tau_v\} \leftarrow \phi_{\boldsymbol{M}}(\mathsf{vk})$, where all $\tau_1, \ldots, \tau_v \in [u]$ are distinct.
  5. Compute $\mathsf{ct}'_i \leftarrow \mathsf{PRE}'.\mathsf{Enc}(\mathsf{pk}_{\tau_i}, x_i)$ for every $i \in [v]$.
  6. Compute $\sigma \leftarrow \mathsf{OTS.Sign}(\mathsf{sigk}, (\mathsf{ct}'_1 \parallel \cdots \parallel \mathsf{ct}'_v \parallel z))$.
  7. Output $\mathsf{ct} = (\mathsf{vk}, (\mathsf{ct}'_i)_{i\in[v]}, z, \sigma)$.
- $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) \to \mathsf{m}/\perp$:
  1. Parse $\mathsf{sk} = (\mathsf{sk}'_i)_{i\in[u]}$ and $\mathsf{ct} = (\mathsf{vk}, (\mathsf{ct}'_i)_{i\in[v]}, z, \sigma)$.
  2. Output $\perp$ if $\mathsf{OTS.Vrfy}(\mathsf{vk}, (\mathsf{ct}'_1 \parallel \cdots \parallel \mathsf{ct}'_v \parallel z), \sigma) = \perp$ holds.
  3. Compute $\{\tau_1, \ldots, \tau_v\} \leftarrow \phi_{\boldsymbol{M}}(\mathsf{vk})$.
  4. Compute $x'_i \leftarrow \mathsf{PRE}'.\mathsf{Dec}(\mathsf{sk}'_{\tau_i}, \mathsf{ct}'_i)$ for every $i \in [v]$.
  5. Output $\mathsf{m}' \leftarrow \mathsf{Inverse}(x'_1, \ldots, x'_v, z)$ if $x'_i \neq \perp$ holds for every $i \in [v]$; otherwise, output $\perp$.
- $\mathsf{ReKeyGen}(\mathsf{sk}_A, \mathsf{pk}_B) \to \mathsf{rk}_{A\to B}$:
  1. Parse $\mathsf{sk}_A = (\mathsf{sk}'_{A,i})_{i\in[u]}$ and $\mathsf{pk}_B = (\mathsf{pk}'_{B,i})_{i\in[u]}$.
  2. For $i \in [u]$ and $j \in [u]$, compute $\mathsf{rk}_{(A,i)\to(B,j)} \leftarrow \mathsf{PRE}'.\mathsf{ReKeyGen}(\mathsf{sk}'_{A,i}, \mathsf{pk}'_{B,j})$.
  3. Output $\mathsf{rk}_{A\to B} = (\mathsf{rk}_{(A,i)\to(B,j)})_{i\in[u],j\in[u]}$.
- $\mathsf{ReEnc}(\mathsf{rk}_{A\to B}, \mathsf{ct}_A) \to \mathsf{ct}_B$:
  1. Parse $\mathsf{rk}_{A\to B} = (\mathsf{rk}_{(A,i)\to(B,j)})_{i\in[u],j\in[u]}$ and $\mathsf{ct}_A = (\mathsf{vk}_A, (\mathsf{ct}'_{A,i})_{i\in[v]}, z, \sigma_A)$.
  2. Output $\perp$ if $\mathsf{OTS.Vrfy}(\mathsf{vk}_A, (\mathsf{ct}'_{A,1} \parallel \cdots \parallel \mathsf{ct}'_{A,v} \parallel z), \sigma_A) = \perp$ holds.
  3. Generate $(\mathsf{vk}_B, \mathsf{sigk}_B) \leftarrow \mathsf{OTS.KeyGen}(1^\lambda)$.
  4. Compute $\{\alpha_1, \ldots, \alpha_v\} \leftarrow \phi_{\boldsymbol{M}}(\mathsf{vk}_A)$ and $\{\beta_1, \ldots, \beta_v\} \leftarrow \phi_{\boldsymbol{M}}(\mathsf{vk}_B)$.
  5. For every $i \in [v]$, compute $\mathsf{ct}'_{B,i} \leftarrow \mathsf{PRE}'.\mathsf{ReEnc}(\mathsf{rk}_{(A,\alpha_i)\to(B,\beta_i)}, \mathsf{ct}'_{A,i})$.
  6. Compute $\sigma_B \leftarrow \mathsf{OTS.Sign}(\mathsf{sigk}_B, (\mathsf{ct}'_{B,1} \parallel \cdots \parallel \mathsf{ct}'_{B,v} \parallel z))$.
  7. Output $\mathsf{ct}_B = (\mathsf{vk}_B, (\mathsf{ct}'_{B,i})_{i\in[v]}, z, \sigma_B)$.

Proposition 4 shows that the correctness of PRE follows that of the underlying primitives PRE', OTS, and $(\mathsf{Trans}, \mathsf{Inverse})$. This proposition can be proved clearly. Hence, we omit this proof.

Proposition 4 (Correctness of B-PRE)**.** *If the PRE scheme* PRE'*, the OTS scheme* OTS *are* correct*, and* Trans *with* Inverse *is* $(\mu, \bar{\mu}, v)$-AONT*, then the resulting PRE scheme* B-PRE *is* correct*.*

A.3 Security Proof

Theorem 3 shows the bounded CCA2 security (i.e., (t,t)-CCA2 security) of the proposed PRE scheme B-PRE.

Theorem 3 ((t,t)-CCA2 security of PRE)**.** *Suppose that the matrix $\boldsymbol{M} \in \{0,1\}^{u \times \bar{n}}$ is a t-disjunct matrix, and n_h is the number of honest users in the (t,t)-CPA2 game. If the PRE scheme PRE' is* CPA *secure, the OTS scheme* OTS *is* strongly unforgeable*, and the algorithm* Trans *is $(\mu, \bar{\mu}, v)$-AONT, then the resulting PRE scheme* B-PRE *is (t,t)-CCA2 secure.*

Particularlly, if there exists a PPT algorithm $\mathcal{A}$ against the (t,t)-CCA2 secure PRE B-PRE*, then there exist PPT adversaries $\mathcal{B}_1$ against the* CPA *secure PRE* PRE'*, $\mathcal{F}$ against the* strongly unforgeable *OTS* OTS*, and $\mathcal{B}_2$ against $(\mu, \bar{\mu}, v)$-AONT* Trans *such that*

$$\mathsf{Adv}^{(t,t)\text{-cca2}}_{\mathsf{B\text{-}PRE},\mathcal{A}}(\lambda) \leq n_h u^2 \cdot \mathsf{Adv}^{\mathrm{cpa}}_{\mathcal{B}_1,\mathsf{PRE}'}(\lambda) + \mathsf{Adv}^{\mathrm{suf}}_{\mathsf{OTS},\mathcal{F}}(\lambda) + n_h u^2 \cdot \mathsf{Adv}^{\mathrm{ind}}_{\mathcal{B}_2,\mathsf{AONT}}(\lambda).$$

Proof. Let $\mathcal{A}$ be a PPT adversary against the PRE scheme B-PRE. Let $\mathsf{ct}^* = (\mathsf{vk}^*, (\mathsf{ct}'^*_i)_{i\in[v]}, z^*, \sigma^*)$ denote the challenge ciphertext under pk_{i^*}.

For each $i \in \{0,1\}$, we consider a security game Game_i and define W_i as the event that the experiment in Game_i outputs 1, in order to prove Theorem 3.

$\underline{\mathsf{Game}_0}$: This game is the same as the (t,t)-CCA2 security game. Then, we have $\mathsf{Adv}^{(t,t)\text{-cca2}}_{\mathsf{PRE},\mathcal{A}}(\lambda) = |\Pr[W_0] - 1/2|$.

$\underline{\mathsf{Game}_1}$: This game is the same as Game_0 except for the following procedures of the decryption oracle $\mathtt{O.Dec}$ and the re-encryption oracle $\mathtt{O.ReEnc}$: For a decryption or re-encryption query on $\mathsf{ct}_i = (\mathsf{vk}_i, (\mathsf{ct}'_{i,j})_{j\in[v]}, z_i, \sigma_i)$, the oracle $\mathtt{O.Dec}$ or $\mathtt{O.ReEnc}$ checks whether it holds that $\mathsf{vk}_i = \mathsf{vk}^*$, $\mathsf{ct}_i \neq \mathsf{ct}^*$, and $\mathsf{OTS.Vrfy}(\mathsf{vk}_i, (\mathsf{ct}'_{i,1}\| \cdots \|\mathsf{ct}'_{i,v}\|z_i), \sigma_i) = \top$. If so, the experiment aborts; otherwise, $\mathtt{O.Dec}$ computes $\mathsf{m}' \leftarrow \mathsf{Dec}(\mathsf{sk}_i, \mathsf{ct}_i)$ and returns $\mathsf{m}' \in \mathcal{M} \cup \{\perp\}$.

Let Bad be the event that $\mathcal{A}$ issues a decryption or re-encryption query on ct_i such that $\mathsf{vk}_i = \mathsf{vk}^*$, $\mathsf{ct}_i \neq \mathsf{ct}^*$, and $\mathsf{OTS.Vrfy}(\mathsf{vk}_i, (\mathsf{ct}'_{i,1}\| \cdots \|\mathsf{ct}'_{i,v}\|z_i), \sigma_i)) = \top$. Then, Game_0 and Game_1 are identical unless Bad occurs. Hence, we bound the probability that Bad occurs. In order to estimate this upper bound, we construct a PPT algorithm $\mathcal{F}$ breaking the strongly unforgeable OTS scheme OTS. On input a verification key vk^* of OTS, $\mathcal{F}$ generates $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$ and gives pp to $\mathcal{A}$. When $\mathcal{A}$ submits $(n, \mathcal{U}_{\mathtt{Corrupt}})$, $\mathcal{F}$ generates $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$ for every $i \in [n]$ and returns $(\{\mathsf{pk}_i\}_{i\in[n]}, \{\mathsf{sk}_i\}_{i\in\mathcal{U}_{\mathtt{Corrupt}}})$. By using these generated key-pairs, $\mathcal{F}$ simulates the oracle $\mathtt{O.ReKeyGen}$. Additionally, the oracle $\mathtt{O.Dec}$ (resp., $\mathtt{O.ReEnc}$) is simulated as follows: For a decryption query (i, ct_i) (resp., (i, j, ct_i)) (where $\mathsf{ct}_i = (\mathsf{vk}_i, (\mathsf{ct}'_{i,j})_{j\in[v]}, z_i, \sigma_i)$), $\mathcal{F}$ aborts and outputs a forgery $((\mathsf{ct}'_{i,1}\| \cdots \|\mathsf{ct}'_{i,v}\|z_i), \sigma_i)$ in the strong unforgeability game of OTS, if it holds that $\mathsf{vk}_i = \mathsf{vk}^*$, $\mathsf{ct}_i \neq \mathsf{ct}^*$, and $\mathsf{OTS.Vrfy}(\mathsf{vk}_i, (\mathsf{ct}'_{i,1}\| \cdots \|\mathsf{ct}'_{i,v}\|z_i), \sigma_i)) = \top$ (i.e., Bad occurs); otherwise, the algorithm computes $\mathsf{m}' \leftarrow \mathsf{Dec}(\mathsf{sk}_i, \mathsf{ct}_i)$ and returns $\mathsf{m}' \in \mathcal{M} \cup \{\perp\}$ (resp., computes $\mathsf{ct}_j \leftarrow \mathsf{ReEnc}(\mathsf{rk}_{i\to j}, \mathsf{ct}_i)$ and returns ct_j).

Furthermore, when $\mathcal{A}$ submits a challenge query $(i^*, \mathsf{m}^*_0, \mathsf{m}^*_1)$, $\mathcal{F}$ chooses $b \xleftarrow{\$} \{0,1\}$ computes $((\mathsf{ct}'^*_i)_{i\in[v]}, z^*)$ by following the procedure of $\mathsf{Enc}(\mathsf{pk}_{i^*}, \mathsf{m}^*_b)$. Then, this algorithm issues $(\mathsf{ct}'^*_1\| \cdots \|\mathsf{ct}'^*_v\|z^*)$ to the signing oracle in the strong unforgeability game and obtains σ^*. And then, $\mathcal{F}$ returns the challenge ciphertext $\mathsf{ct}^* = (\mathsf{vk}^*, (\mathsf{ct}'^*_i)_{i\in[v]}, z^*, \sigma^*)$. Finally, when $\mathcal{A}$ outputs $b' \in \{0,1\}$ and Bad has not occurred, $\mathcal{F}$ halts and outputs 0.

We analyze the algorithm $\mathcal{F}$ against OTS. It is clear that the output of $\mathcal{F}$ is a valid forgery in the strong unforgeability game if Bad occurs. Unless this event happens, $\mathcal{F}$ completely simulates the oracles in the (t,t)-CCA2 game by using all key-pairs. Hence, the probability $\Pr[\mathsf{Bad}]$ is at most the advantage $\mathsf{Adv}^{\mathrm{suf}}_{\mathsf{OTS},\mathcal{F}}(\lambda)$ of $\mathcal{F}$, and we have $|\Pr[W_0] - \Pr[W_1]| \leq \mathsf{Adv}^{\mathrm{suf}}_{\mathsf{OTS},\mathcal{F}}(\lambda)$.

In order to bound the winning probability of $\mathcal{A}$ in Game_1, we consider the following experiment $\mathcal{B}$: At the beginning of the (t,t)-CCA2 game, $\mathcal{B}$ gives $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$ to $\mathcal{A}$ and simulates $(\{\mathsf{pk}_i\}_{i\in[n]}, \{\mathsf{sk}_i\}_{i\in\mathcal{U}_{\mathtt{Corrupt}}}) \leftarrow \mathtt{O.KeyGen}(n, \mathcal{U}_{\mathtt{Corrupt}})$. And then, $\mathcal{B}$ generates $(\mathsf{vk}^*, \mathsf{sigk}^*) \leftarrow \mathsf{OTS.KeyGen}(1^\lambda)$, chooses indices $i^* \xleftarrow{\$} [n_h]$, $j^* \xleftarrow{\$} \phi_{\boldsymbol{M}}(\mathsf{vk}^*)$, and simulates the environment of $\mathcal{A}$ except for the following: The experiment aborts and outputs a random bit if $\mathcal{A}$ issues

- a decryption or re-encryption query on $(i^*, (\mathsf{vk}_{i^*}, (\mathsf{ct}'_{i^*,j})_{j\in[v]}, z_{i^*}, \tau_{i^*}))$ such that $j^* \in \phi_{\boldsymbol{M}}(\mathsf{vk}_{i^*})$; or
- a challenge query $(i', \mathsf{m}_0^*, \mathsf{m}_1^*)$ such that $i^* \neq i'$.

Finally, when $\mathcal{A}$ outputs $b' \in \{0,1\}$, $\mathcal{B}$ also outputs b'.

For the event $W_{\mathcal{B}}$ that $b = b'$ holds in the experiment $\mathcal{B}$, we estimate the probability $\Pr[W_{\mathcal{B}}]$. Let Abort be the event that $\mathcal{B}$ aborts in the simulation of the decryption or re-encryption oracle. Notice that $\Pr[W_{\mathcal{B}} \mid \mathsf{Abort}] = 1/2$. Due to the t-disjunct property of $\boldsymbol{M}$, it holds that $\Pr[\neg\mathsf{Abort}] \geq 1/(n_h u^2)$. Then, we have

$$\begin{aligned}
\Pr[W_{\mathcal{B}}] &= \Pr[\mathsf{Abort}] \cdot \Pr[W_{\mathcal{B}} \mid \mathsf{Abort}] + \Pr[\neg\mathsf{Abort}] \cdot \Pr[W_{\mathcal{B}} \mid \neg\mathsf{Abort}] \\
&\geq \frac{1}{2}\left(1 - \frac{1}{n_h u^2}\right) + \frac{1}{n_h u^2} \cdot \Pr[W_{\mathcal{B}} \mid \neg\mathsf{Abort}] \\
&= \frac{1}{2} + \frac{1}{n_h u^2}\left(\Pr[W_{\mathcal{B}} \mid \neg\mathsf{Abort}] - \frac{1}{2}\right).
\end{aligned}$$

The $\mathcal{A}$'s advantage $\varepsilon_{\mathcal{A}}$ in Game_1 is equivalent to $|\Pr[W_{\mathcal{B}} \mid \neg\mathsf{Abort}] - 1/2|$. Hence, the $\mathcal{B}$'s advantage $\varepsilon_{\mathcal{B}} = |\Pr[W_{\mathcal{B}}] - 1/2|$ is at least $\varepsilon_{\mathcal{A}}/(n_h u^2)$. Here, let $\phi_{\boldsymbol{M}}(\mathsf{vk}^*) := \{\tau_1^*, \ldots, \tau_v^*\}$ and $\tau_{k^*}^* := j^*$ (where $\tau_1^*, \ldots, \tau_v^* \in [u]$ and $k^* \in [v]$). In order to bound $\varepsilon_{\mathcal{B}}$, we change the environment of $\mathcal{A}$. In this modified environment, the j^*-th share $x_{j^*}^*$ generated by Trans is replaced with the all-zero string $0^{|x_{j^*}^*|}$, when producing the challenge ciphertext. The probability $\Pr[W_{\mathcal{B}}]$ is defined as $p^{(0)}$, and the probability that $W_{\mathcal{B}}$ occurs in the modified environment is defined as $p^{(1)}$. Then we have $\varepsilon_{\mathcal{B}} \leq |p^{(0)} - p^{(1)}| + |p^{(1)} - 1/2|$.

In order to bound $|p^{(0)} - p^{(1)}|$, we construct a PPT algorithm $\mathcal{B}_1$ against the CPA security of PRE', as follows: On input the public parameter pp' in the CPA game, $\mathcal{B}_1$ generates pp by following the algorithm Setup and gives pp to $\mathcal{A}$. When $\mathcal{A}$ submits the key generation query $(n, \mathcal{U}_{\mathtt{Corrupt}})$, $\mathcal{B}_1$ generates $(\mathsf{vk}^*, \mathsf{sigk}^*) \leftarrow \mathsf{OTS.KeyGen}(1^\lambda)$, chooses $i^* \xleftarrow{\$} [n], j^* \xleftarrow{\$} \phi_{\boldsymbol{M}}(\mathsf{vk}^*)$, and obtains $(\{\mathsf{pk}'_{i,j}\}_{(i,j)\in[n]\times[u]}, \{\mathsf{sk}'_{i,j}\}_{(i,j)\in[n]\times[u]\setminus\{(i^*,j^*)\}})$ by issuing a key generation query $(nu, \mathcal{U}'_{\mathtt{Corrupt}})$ such that $\mathcal{U}'_{\mathtt{Corrupt}} = \{(i,j)\}\setminus\{(i^*,j^*)\}$, in the CPA game. Here, for simplicity, we suppose that $(i,j) \in [n]\times[u]$ represents a user in the CPA game. Then $\mathcal{B}_1$ returns $(\{\mathsf{pk}_i\}_{i\in[n]}, \{\mathsf{sk}_i\}_{i\in\mathcal{U}_{\mathtt{Corrupt}}})$, where let $\mathsf{pk}_i := (\mathsf{pk}'_{i,j})_{j\in[u]}$ for every $i \in [n]$, let $\mathsf{sk}_i := (\mathsf{sk}'_{i,j})_{j\in[u]}$ for every $i \in [n]\setminus\{i^*\}$, and let $\mathsf{sk}_{i^*} := (\mathsf{sk}'_{i^*,j})_{j\in[u]\setminus\{j^*\}}$. Furthermore, $\mathcal{B}_1$ simulates the oracles $\mathtt{O.ReKeyGen}, \mathtt{O.Dec}, \mathtt{O.ReEnc}, \mathtt{O.Challenge}_b$, as follows:

- $\mathtt{O.ReKeyGen}(A, B)$: If $(A \in \mathcal{U}_{\mathtt{Honest}} \wedge B \in \mathcal{U}_{\mathtt{Corrupt}})$ or $A = B$ holds, $\mathcal{B}_1$ returns $\perp$; otherwise it does the following:
  - (Case $A = i^*$): Obtain $\mathsf{rk}_{(i^*,j^*)\to(B,j)}$ by issuing $((i^*,j^*),(B,j))$ to the re-encryption key generation oracle in the CPA game, for every $j \in [u]$. For every $i \in [u]\setminus\{j^*\}$ and every $j \in [u]$, compute $\mathsf{rk}_{(i^*,i)\to(B,j)} \leftarrow \mathsf{PRE}'.\mathsf{ReKeyGen}(\mathsf{sk}'_{i^*,i}, \mathsf{pk}'_{B,j})$.
  - (Case $A \neq i^*$): Compute $\mathsf{rk}_{(A,i)\to(B,j)} \leftarrow \mathsf{PRE}'.\mathsf{ReKeyGen}(\mathsf{sk}'_{A,i}, \mathsf{pk}'_{B,j})$ for every $i \in [u]$ and every $j \in [u]$.

Finally, $\mathcal{B}_1$ returns $\mathsf{rk}_{A\to B} = (\mathsf{rk}_{(A,i)\to(B,j)})_{i\in[u],j\in[u]}$.

- $\mathtt{O.Dec}(A, \mathsf{ct}_A)$: $\mathcal{B}_1$ parses $\mathsf{ct}_A = (\mathsf{vk}_A, (\mathsf{ct}'_{A,i})_{i\in[v]}, z, \sigma_A)$ and does the following:
  1. Return $\perp$ if (A, ct_A) is a $\mathsf{derivative}$ of (i^*, ct^*).
  2. Abort and output a random bit if $A = i^* \wedge j^* \in \phi_{\boldsymbol{M}}(\mathsf{vk}_A)$ holds.
  3. Return $\perp$ if it holds that $\mathsf{vk}_A = \mathsf{vk}^*$, $\mathsf{ct}_A \neq \mathsf{ct}^*$ and $\mathsf{OTS.Vrfy}(\mathsf{vk}_A, (\mathsf{ct}'_{A,1}\| \cdots \|\mathsf{ct}'_{A,v}\|z), \sigma_A) = \top$.
  4. Return $\perp$ if $\mathsf{OTS.Vrfy}(\mathsf{vk}_A, (\mathsf{ct}'_{A,1}\| \cdots \|\mathsf{ct}'_{A,v}\|z), \sigma_A) = \perp$ holds.
  5. Compute $x'_i \leftarrow \mathsf{PRE'.Dec}(\mathsf{sk}'_{A,i}, \mathsf{ct}'_{A,i})$ for every $i \in [v]$,.
  6. Return $\mathsf{m}' \leftarrow \mathsf{Inverse}(x'_1, \ldots, x'_v, z)$ if $x'_i \neq \perp$ holds for every $i \in [v]$; otherwise return $\perp$.
- $\mathtt{O.ReEnc}(A, B, \mathsf{ct}_A)$: $\mathcal{B}_1$ parses $\mathsf{ct}_A = (\mathsf{vk}_A, (\mathsf{ct}'_{A,i})_{i\in[v]}, z, \sigma_A)$ and does the following:
  1. Return $\perp$ if $B \in \mathcal{U}_{\mathtt{Corrupt}}$ holds and (A, ct_A) is a $\mathsf{derivative}$ of (i^*, ct^*).
  2. Abort and output a random bit if $A = i^* \wedge j^* \in \phi_{\boldsymbol{M}}(\mathsf{vk}_A)$ holds.
  3. Return $\perp$ if it holds that $\mathsf{vk}_A = \mathsf{vk}^*$, $\mathsf{ct}_A \neq \mathsf{ct}^*$ and $\mathsf{OTS.Vrfy}(\mathsf{vk}_A, (\mathsf{ct}'_{A,1}\| \cdots \|\mathsf{ct}'_{A,v}\|z), \sigma_A) = \top$.
  4. Return $\perp$ if $\mathsf{OTS.Vrfy}(\mathsf{vk}_A, (\mathsf{ct}'_{A,1}\| \cdots \|\mathsf{ct}'_{A,v}\|z), \sigma_A) = \perp$ holds.
  5. Generate $(\mathsf{vk}_B, \mathsf{sigk}_B) \leftarrow \mathsf{OTS.KeyGen}(1^\lambda)$.
  6. Compute $\mathsf{rk}_{(A,i)\to(B,j)} \leftarrow \mathsf{PRE'.ReKeyGen}(\mathsf{sk}'_{A,i}, \mathsf{pk}'_{B,j})$ for every $i \in [u]$ and $j \in [u]$.
  7. Let $\{\alpha_1, \ldots, \alpha_v\} \leftarrow \phi_{\boldsymbol{M}}(\mathsf{vk}_A)$ and $\{\beta_1, \ldots, \beta_v\} \leftarrow \phi_{\boldsymbol{M}}(\mathsf{vk}_B)$.
  8. Compute $\mathsf{ct}'_{B,i} \leftarrow \mathsf{PRE'.ReEnc}(\mathsf{rk}_{(A,\alpha_i)\to(B,\beta_i)}, \mathsf{ct}'_{A,i})$ for every $i \in [v]$.
  9. Compute $\sigma_B \leftarrow \mathsf{OTS.Sign}(\mathsf{sigk}_B, (\mathsf{ct}'_{B,1}\| \cdots \|\mathsf{ct}'_{B,v}\|z))$.
  10. Return $\mathsf{ct}_B = (\mathsf{vk}_B, (\mathsf{ct}'_{B,i})_{i\in[v]}, z, \sigma_B)$.
- $\mathtt{O.Challenge}_b(i', \mathsf{m}^*_0, \mathsf{m}^*_1)$: $\mathcal{B}_1$ does the following:
  1. Abort and output a random bit if $i^* \neq i'$.
  2. Let $\{\tau^*_1, \ldots, \tau^*_v\} \leftarrow \phi_{\boldsymbol{M}}(\mathsf{vk}^*)$.
  3. Compute $(x^*_1, \ldots, x^*_v, z^*) \leftarrow \mathsf{Trans}(\mathsf{m}^*_b)$.
  4. Obtain $\mathsf{ct}'^*_{j^*}$ by submitting a challenge query $(x^*_{j^*}, 0^{\bar{\mu}})$ to the CPA game.
  5. For every $j \in [v]\backslash\{j^*\}$, then compute $\mathsf{ct}'^*_j \leftarrow \mathsf{PRE'.Enc}(\mathsf{pk}'_{\tau^*_j}, x^*_j)$.
  6. Compute $\sigma^* \leftarrow \mathsf{OTS.Sign}(\mathsf{sigk}^*, (\mathsf{ct}'^*_1\| \cdots \|\mathsf{ct}'^*_v\|z^*))$.
  7. Return $\mathsf{ct}^* = (\mathsf{vk}^*, (\mathsf{ct}'^*_i)_{i\in[v]}, z^*, \sigma^*)$.

When $\mathcal{A}$ finally outputs the guessing bit $b' \in \{0,1\}$, $\mathcal{B}_1$ outputs 1 if $b = b'$; otherwise, it outputs 0.

We analyze the algorithm $\mathcal{B}_1$. Unless $\mathcal{A}$ issues a decryption query or re-encryption query on $(A, (\mathsf{vk}_A, (\mathsf{ct}'_{A,i})_{i\in[v]}, z_A, \sigma_A))$ such that $A = i^* \wedge j^* \in \phi_{\boldsymbol{M}}(\mathsf{vk}_A)$, $\mathcal{B}_1$ can simulate the oracles $\mathtt{O.Dec}$ and $\mathtt{O.ReEnc}$. The t-disjunct property of $\boldsymbol{M}$ ensures that $\mathcal{A}$ cannot issue such a query. Additionally, $\mathcal{B}_1$ wins the CPA game by employing $\mathcal{A}$'s output, in the straightforward way. Hence, we have $\left|p^{(0)} - p^{(1)}\right| \leq \mathsf{Adv}^{\mathrm{cpa}}_{\mathsf{PRE'},\mathcal{B}_1}(\lambda)$.

In order to bound the winning probability in the modified environment (i.e., $|p^{(1)} - 1/2|$), we construct a PPT algorithm $\mathcal{B}_2$ against $(\mu, \bar{\mu}, v)$-AONT Trans. By using $\mathcal{A}$, we construct $\mathcal{B}_2$ given the

oracle $\mathtt{O.LR}$ in the indistinguishability game of AONT: At the beginning of the (t,t)-CCA game, $\mathcal{B}_2$ gives $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$ to $\mathcal{A}$. When $\mathcal{A}$ issues $(n, \mathcal{U}_{\mathtt{Corrupt}})$, $\mathcal{B}_2$ generates $(\mathsf{vk}^*, \mathsf{sigk}^*) \leftarrow \mathsf{OTS.KeyGen}(1^\lambda)$, chooses $i^* \xleftarrow{\$} [n], j^* \xleftarrow{\$} \phi_{\boldsymbol{M}}(\mathsf{vk}^*)$, and generates all key-pairs $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$ for all users $i \in [n]$. And then, $\mathcal{B}_2$ simulates the oracles $\mathtt{O.ReKeyGen}, \mathtt{O.Dec}, \mathtt{O.ReEnc}$ by using the generated key-pairs. Furthermore, $\mathcal{B}_2$ simulates the oracle $\mathtt{O.Challenge}_b(i', \mathsf{m}_0^*, \mathsf{m}_1^*)$ as follows:

1. Abort and output a random bit if $i^* = i'$.
2. Obtain $((x_i^*)_{i \in [v] \setminus \{j^*\}}, z^*)$ by issuing $(\mathsf{m}_b^*, 0^\mu)$ to the given oracle $\mathtt{O.LR}$.
3. Compute $\mathsf{ct}_i'^* \leftarrow \mathsf{PRE}'.\mathsf{Enc}(\mathsf{pk}_{\tau_i^*}, x_i^*)$ for every $i \in [v]$, where $\{\tau_1^*, \ldots, \tau_v^*\} = \phi_{\boldsymbol{M}}(\mathsf{vk}^*)$.
4. Compute $\sigma^* \leftarrow \mathsf{OTS.Sign}(\mathsf{sigk}^*, (\mathsf{ct}_1'^* \parallel \cdots \parallel \mathsf{ct}_v'^* \parallel z^*))$.
5. Return $\mathsf{ct}^* = (\mathsf{vk}^*, (\mathsf{ct}_i'^*)_{i \in [v]}, z^*, \sigma^*)$.

When $\mathcal{A}$ outputs the guessing bit $b' \in \{0,1\}$, $\mathcal{B}_2$ also outputs b'.

$\mathcal{B}_2$ simulates the oracles $\mathtt{O.KeyGen}, \mathtt{O.ReKeyGen}, \mathtt{O.Dec}, \mathtt{O.ReEnc}$ completely since it has the key-pairs of all users. The oracle $\mathtt{O.Challenge}$ is also simulated correctly since $\mathcal{B}_2$ can generate the challenge ciphertext without knowledge of $x_{j^*}^*$. Hence, the $\mathcal{B}_2$' advantage $\mathsf{Adv}^{\mathrm{ind}}_{\mathsf{Trans},\mathcal{B}_2}(\lambda)$ is at least $\left|p^{(1)} - 1/2\right|$. Therefore, we have $\mathsf{Adv}^{\mathrm{cpa}}_{\mathsf{PRE}',\mathcal{B}_1}(\lambda) + \mathsf{Adv}^{\mathrm{ind}}_{\mathsf{Trans},\mathcal{B}_2}(\lambda) \geq \varepsilon_{\mathcal{A}}/(n_h u)$. From the discussion above, we obtain

$$\mathsf{Adv}^{(t,t)\text{-cca2}}_{\mathsf{PRE},\mathcal{A}}(\lambda) \leq n_h u^2 \cdot \mathsf{Adv}^{\mathrm{cpa}}_{\mathcal{B}_1,\mathsf{PRE}'}(\lambda) + n_h u^2 \cdot \mathsf{Adv}^{\mathrm{ind}}_{\mathsf{AONT},\mathcal{B}_2}(\lambda) + \mathsf{Adv}^{\mathrm{suf}}_{\mathsf{OTS},\mathcal{F}}(\lambda).$$

and complete the proof. □

B Omitted Proofs for Our Kyber-based PRE Scheme

In this section, we give proofs of the correctness and re-encryption key homomorphism of our scheme K-PRE.

In order to show the correctness and re-encryption key homomorphic property of K-PRE, we employ the distribution ψ_d^k over R which is defined in [6]. Following [6], we describe the definition of ψ_d^k for positive integers d and k, as follows:

1. Choose $\boldsymbol{y} \leftarrow R^k$ uniformly at random.
2. Return $(\boldsymbol{y} - \mathsf{Decompress}_q(\mathsf{Compress}_q(\boldsymbol{y}, d), d)) \bmod^{\pm} q$.

B.1 Proof of Correctness

We show the correctness of K-PRE.

Proposition 5 (Correctness of K-PRE)**.** *Let* $\mathsf{pp} = (\lambda, \mu, N, N', q, \ell, \eta, k, d_t, d_u, d_v, \boldsymbol{A})$ *be a public parameter determined by running* $\mathsf{Setup}(1^\lambda)$ *and let* A, B *be distinct users. Then, the key-pairs of these users and a ciphertext under the user* A *'s public key are defined as follows:*

- *Let* $(\mathsf{pk}_A, \mathsf{sk}_A) = ((\boldsymbol{t}_A, \hat{\boldsymbol{t}}_A), (\boldsymbol{s}_A, \hat{\boldsymbol{s}}_A))$ *and* $(\mathsf{pk}_B, \mathsf{sk}_B) = ((\boldsymbol{t}_B, \hat{\boldsymbol{t}}_B), (\boldsymbol{s}_B, \hat{\boldsymbol{s}}_B))$ *be key-pairs of the users* A *and* B*, respectively, where* $\boldsymbol{t}_i = \mathsf{Compress}_q(\boldsymbol{A}\boldsymbol{s}_i + \boldsymbol{e}_i, d_t)$ *and* $\hat{\boldsymbol{t}}_i = \mathsf{Compress}_q(\boldsymbol{A}\hat{\boldsymbol{s}}_i + \hat{\boldsymbol{e}}_i, d_t)$ *for* $i \in \{A, B\}$*;*

- *Let* $\mathsf{ct}_A = (\boldsymbol{u}_A, v_A)$ *be a ciphertext generated by running* $\mathsf{Enc}(\mathsf{pk}_A, \mathsf{m})$ *for an arbitrary message* $\mathsf{m} \in \mathcal{M}$*, where* $\boldsymbol{t}_A = \mathsf{Decompress}(\boldsymbol{t}_A, d_t)$*,* $\boldsymbol{u} = \mathsf{Compress}_q(\boldsymbol{A}^\top \boldsymbol{r} + \boldsymbol{e}_{A,1}^\top, d_u)$*, and* $v = \mathsf{Compress}_q(\boldsymbol{t}_A^\top \boldsymbol{r} + e_{A,2} + \lceil \frac{q}{2} \rfloor \cdot \mathsf{m}, d_v)$*.*
- *Let* $\mathsf{ct}_B = (\boldsymbol{u}_B, v_B)$ *be a re-encrypted ciphertext running* $\mathsf{ReEnc}(\mathsf{rk}_{A\to B}, \mathsf{ct}_A)$*.*

Let $\boldsymbol{c}_{t,A}, \boldsymbol{c}_{t,B} \leftarrow \psi_{d_t}^k$*,* $\boldsymbol{c}_{u,A}, \boldsymbol{c}_{u,B} \leftarrow \psi_{d_u}^k$*,* $c_{v,A} \leftarrow \psi_{d_v}$*. Denote*

$$
\begin{aligned}
w &:= \boldsymbol{e}_A^\top \boldsymbol{r} + e_2 + c_{v,A} - \boldsymbol{s}_A^\top \boldsymbol{e}_{A,1} - \boldsymbol{s}_A^\top \boldsymbol{c}_{u,A};\\
\hat{w} &:= w + (e_{B,2} + (\hat{\boldsymbol{e}}_B^\top + \boldsymbol{c}_{t,B}^\top)\mathsf{BitDecomp}(\boldsymbol{u}_A)\\
&\qquad + \boldsymbol{r}_{A\to B,3}^\top \cdot \mathsf{BitDecomp}(\boldsymbol{u}_A) - \hat{\boldsymbol{s}}_B^\top \boldsymbol{R}_{A\to B,2} \cdot \mathsf{BitDecomp}(\boldsymbol{u}_A) - \hat{\boldsymbol{s}}_B^\top \boldsymbol{c}_{u,B};\\
\delta &:= \Pr\left[\|\hat{w}\|_\infty \geq q/4\right].
\end{aligned}
$$

Then, the proposed PRE scheme K-PRE *is* correct *with probability* $1 - \delta$*.*

Proof. We consider an arbitrary message $\mathsf{m} \in \mathcal{M}$ when showing the encryption correctness and re-encryption correctness of K-PRE. Recall that $\mathsf{pp} = (\lambda, \mu, N, N', q, \ell, \eta, k, d_t, d_u, d_v, \boldsymbol{A})$ is a public parameter determined by running $\mathsf{Setup}(1^\lambda)$ and let A, B be two distinct users. Then, these users' key-pairs and an encryption of m are defined as follows:

- $(\mathsf{pk}_A, \mathsf{sk}_A) = ((\boldsymbol{t}_A, \hat{\boldsymbol{t}}_A), (\boldsymbol{s}_A, \hat{\boldsymbol{s}}_A))$ and $(\mathsf{pk}_B, \mathsf{sk}_B) = ((\boldsymbol{t}_B, \hat{\boldsymbol{t}}_B), (\boldsymbol{s}_B, \hat{\boldsymbol{s}}_B))$ are key-pairs of the users A and B, respectively, where $\boldsymbol{t} = \mathsf{Compress}_q(\boldsymbol{A}\boldsymbol{s}_i + \boldsymbol{e}_i, d_t)$ and $\hat{\boldsymbol{t}} = \mathsf{Compress}_q(\boldsymbol{A}\hat{\boldsymbol{s}}_i + \hat{\boldsymbol{e}}_i, d_t)$ for $i \in \{A, B\}$; and
- $\mathsf{ct}_A = (\boldsymbol{u}_A, v_A)$ is a ciphertext generated by running $\mathsf{Enc}(\mathsf{pk}_A, \mathsf{m})$, where $\boldsymbol{u} = \mathsf{Compress}_q(\boldsymbol{A}^\top \boldsymbol{r} + \boldsymbol{e}_1^\top, d_u)$ and $v = \mathsf{Compress}_q(\boldsymbol{t}^\top \boldsymbol{r} + e_2 + \lceil \frac{q}{2} \rfloor \cdot \mathsf{m}, d_v)$.

First we show the encryption-correctness of K-PRE. Then, the public value $\boldsymbol{t}_A$ is represented as

$$
\begin{aligned}
\boldsymbol{t}_A &= \mathsf{Decompress}_q(\mathsf{Compress}_q(\boldsymbol{A}\boldsymbol{s}_A + \boldsymbol{e}_A, d_t), d_t)\\
&= \boldsymbol{A}\boldsymbol{s}_A + \boldsymbol{e}_A + \boldsymbol{c}_{t,A}
\end{aligned}
$$

for some value $\boldsymbol{c}_{t,A} \leftarrow \psi_{d_t}^k$.

Additionally, the value $\boldsymbol{u}_A$ of the ciphertext $\mathsf{ct}_A = (\boldsymbol{u}_A, v_A)$ under pk_A is

$$
\begin{aligned}
\boldsymbol{u}_A &= \mathsf{Decompress}_q(\mathsf{Compress}_q(\boldsymbol{A}_A^\top \boldsymbol{r}_A + \boldsymbol{e}_{A,1}, d_u), d_u)\\
&= \boldsymbol{A}^\top \boldsymbol{r} + \boldsymbol{e}_1 + \boldsymbol{c}_{u,A},
\end{aligned}
$$

for some $\boldsymbol{c}_{u,A} \leftarrow \psi_{d_u}^k$. And the value v_A is

$$
\begin{aligned}
v_A &= \mathsf{Decompress}_q(\mathsf{Compress}_q(\boldsymbol{t}_A^\top \boldsymbol{r} + e_2 + \left\lceil \frac{q}{2} \right\rfloor \cdot \mathsf{m}, d_v), d_v)\\
&= \boldsymbol{t}_A^\top \boldsymbol{r} + e_2 + c_{v,A} + \left\lceil \frac{q}{2} \right\rfloor \cdot \mathsf{m}\\
&= (\boldsymbol{A}\boldsymbol{s}_A + \boldsymbol{e}_A + \boldsymbol{c}_{t,A})^\top \boldsymbol{r} + e_2 + c_{v,A} + \left\lceil \frac{q}{2} \right\rfloor \cdot \mathsf{m}\\
&= (\boldsymbol{A}\boldsymbol{s}_A + \boldsymbol{e}_A)^\top \boldsymbol{r} + e_2 + c_{v,A} + \left\lceil \frac{q}{2} \right\rfloor \cdot \mathsf{m} + \boldsymbol{c}_{t,A}^\top \boldsymbol{r},
\end{aligned}
$$

for some $c_{v,A} \leftarrow \psi_{d_v}$.

Then, we have

$$\begin{aligned} v_A - \boldsymbol{s}_A^\top \boldsymbol{u}_A &= (\boldsymbol{A}\boldsymbol{s}_A + \boldsymbol{e}_A)^\top \boldsymbol{r}_A + e_{A,2} + c_{v,A} + \left\lceil \frac{q}{2} \right\rfloor \cdot \mathsf{m} + \boldsymbol{c}_{t,A}^\top \boldsymbol{r}_A \\ &\quad - \boldsymbol{s}_A^\top (\boldsymbol{A}^\top \boldsymbol{r}_A + \boldsymbol{e}_{A,1} + \boldsymbol{c}_{u,A}) \\ &= \left\lceil \frac{q}{2} \right\rfloor \cdot \mathsf{m} + \boldsymbol{e}_A^\top \boldsymbol{r}_A + e_{A,2} + c_{v,A} - \boldsymbol{s}_A^\top \boldsymbol{e}_{A,1} - \boldsymbol{s}_A^\top \boldsymbol{c}_{u,A}. \end{aligned}$$

Let $w := \boldsymbol{e}_A^\top \boldsymbol{r}_A + e_{A,2} + c_{v,A} - \boldsymbol{s}_A^\top \boldsymbol{e}_{A,1} - \boldsymbol{s}_A^\top \boldsymbol{c}_{u,A}$.

We define $\mathsf{m}' = \mathsf{Compress}_q(v_A - \boldsymbol{s}_A^\top \boldsymbol{u}_A, 1)$ and see that

$$\left\lceil \frac{q}{4} \right\rfloor \geq \left\| v_A - \boldsymbol{s}_A^\top \boldsymbol{u}_A - \left\lceil \frac{q}{2} \right\rfloor \cdot \mathsf{m}' \right\|_\infty = \left\| w + \left\lceil \frac{q}{2} \right\rfloor \cdot \mathsf{m} - \left\lceil \frac{q}{2} \right\rfloor \cdot \mathsf{m}' \right\|_\infty .$$

Due to the triangle inequality and the fact that $\|w\|_\infty < \lceil q/4 \rfloor$, it holds that

$$\left\| \left\lceil \frac{q}{2} \right\rfloor \cdot (\mathsf{m} - \mathsf{m}') \right\|_\infty < 2 \left\lceil \frac{q}{4} \right\rfloor$$

This implies $\mathsf{m} = \mathsf{m}'$, and the proof of the encryption correctness is completed.

Next, we show the re-encryption correctness of K-PRE. For simplicity, we also employ the above value of $(\boldsymbol{t}_A, \boldsymbol{u}_A, v_A)$. Then, a re-encryption key $\mathsf{rk}_{A\to B} = (\boldsymbol{U}_{A\to B}, \boldsymbol{v}_{A\to B})$ generated by running $\mathsf{ReKeyGen}(\mathsf{sk}_A, \mathsf{pk}_B)$ are

$$\begin{aligned} \boldsymbol{U}_{A\to B} &= \boldsymbol{A}^\top \boldsymbol{R}_{A\to B,1} + \boldsymbol{R}_{A\to B,2} \in R_q^{k\times kw}; \text{ and} \\ \boldsymbol{v}_{A\to B}^\top &= \hat{\boldsymbol{t}}_B^\top \boldsymbol{R}_{A\to B,1} + \boldsymbol{r}_{A\to B,3}^\top - \mathsf{Powersof2}(\boldsymbol{s}_A^\top). \end{aligned}$$

Additionally, a re-encrypted ciphertext $\mathsf{ct}_B = (\boldsymbol{u}_B, v_B)$ is generated by using the value of $(\boldsymbol{U}_{A\to B}, \boldsymbol{v}_{A\to B})$, as follows:

$$\begin{aligned} \boldsymbol{u}_B &= \left(\boldsymbol{A}_B^\top \boldsymbol{R}_{A\to B,1} + \boldsymbol{R}_{A\to B,2} \right) \cdot \mathsf{BitDecomp}(\boldsymbol{u}_A); \\ v_B &= v_A + (\hat{\boldsymbol{t}}_B^\top \boldsymbol{R}_{A\to B,1} + \boldsymbol{r}_{A\to B,3}^\top - \mathsf{Powersof2}(\boldsymbol{s}_A^\top)) \cdot \mathsf{BitDecomp}(\boldsymbol{u}_A). \end{aligned}$$

Moreover, the decompressed value of $(\boldsymbol{u}_B, v_B)$ is

$$\begin{aligned} \boldsymbol{u}_B &= \mathsf{Decompress}_q(\mathsf{Compress}_q(\boldsymbol{u}_B, d_u)) \\ &= \boldsymbol{A}^\top \boldsymbol{R}_{A\to B,1} \cdot \mathsf{BitDecomp}(\boldsymbol{u}_A) + \boldsymbol{R}_{A\to B,2} \cdot \mathsf{BitDecomp}(\boldsymbol{u}_A) + \boldsymbol{c}_{u,B}; \\ v_B &= \mathsf{Decompress}_q(\mathsf{Compress}_q(v_B, d_u)) \\ &= v_A + (\hat{\boldsymbol{t}}_B^\top \boldsymbol{R}_{A\to B,1} + \boldsymbol{r}_{A\to B,3}^\top - \mathsf{Powersof2}(\boldsymbol{s}_A^\top)) \cdot \mathsf{BitDecomp}(\boldsymbol{u}_A) + c_{v,B} \\ &= (v_A - \boldsymbol{s}_A^\top \boldsymbol{u}_A) + \hat{\boldsymbol{t}}_B^\top \boldsymbol{R}_{A\to B,1} \cdot \mathsf{BitDecomp}(\boldsymbol{u}_A) + \boldsymbol{r}_{A\to B,3}^\top \cdot \mathsf{BitDecomp}(\boldsymbol{u}_A) + c_{v,B}, \end{aligned}$$

for some $(\boldsymbol{c}_{u,B}, c_{v,B}) \in R^k \times R$. Additionally, the public value $\hat{\boldsymbol{t}}_B$ is

$$\begin{aligned} \hat{\boldsymbol{t}}_B &= \mathsf{Decompress}_q(\mathsf{Compress}_q(\boldsymbol{A}\hat{\boldsymbol{s}}_B + \hat{\boldsymbol{e}}_B, d_t), d_t) \\ &= \boldsymbol{A}\hat{\boldsymbol{s}}_B + \hat{\boldsymbol{e}}_B + \boldsymbol{c}_{t,B} \end{aligned}$$

for some $\boldsymbol{c}_{t,B} \in R^k$. Hence, we have

$$\begin{aligned}
&v_B - \hat{\boldsymbol{s}}_B^\top \boldsymbol{u}_B \\
&= (v_A - \boldsymbol{s}_A^\top \boldsymbol{u}_A) + \hat{\boldsymbol{t}}_B^\top \boldsymbol{R}_{A\to B,1} \cdot \mathsf{BitDecomp}(\boldsymbol{u}_A) + \boldsymbol{r}_{A\to B,3}^\top \cdot \mathsf{BitDecomp}(\boldsymbol{u}_A) + c_{v,B} \\
&\quad - \hat{\boldsymbol{s}}_B^\top (\boldsymbol{A}^\top \boldsymbol{R}_{A\to B,1} \cdot \mathsf{BitDecomp}(\boldsymbol{u}_A) + \boldsymbol{R}_{A\to B,2} \cdot \mathsf{BitDecomp}(\boldsymbol{u}_A) + \boldsymbol{c}_{u,B}) \\
&= \left(w + \left\lceil \frac{q_B}{2} \right\rfloor \mathsf{m}\right) + (\hat{\boldsymbol{t}}_B^\top \boldsymbol{R}_{A\to B,1} - (\boldsymbol{A}\hat{\boldsymbol{s}}_B)^\top \boldsymbol{R}_{A\to B,1}) \mathsf{BitDecomp}(\boldsymbol{u}_A) \\
&\quad + \boldsymbol{r}_{A\to B,3}^\top \cdot \mathsf{BitDecomp}(\boldsymbol{u}_A) - \hat{\boldsymbol{s}}_B^\top \boldsymbol{R}_{A\to B,2} \cdot \mathsf{BitDecomp}(\boldsymbol{u}_A) - \hat{\boldsymbol{s}}_B^\top \boldsymbol{c}_{u,B} \\
&= \left(w + \left\lceil \frac{q_B}{2} \right\rfloor \mathsf{m}\right) + (\hat{\boldsymbol{e}}_B^\top + \boldsymbol{c}_{t,B}^\top) \mathsf{BitDecomp}(\boldsymbol{u}_A) \\
&\quad + \boldsymbol{r}_{A\to B,3}^\top \cdot \mathsf{BitDecomp}(\boldsymbol{u}_A) - \hat{\boldsymbol{s}}_B^\top \boldsymbol{R}_{A\to B,2} \cdot \mathsf{BitDecomp}(\boldsymbol{u}_A) - \hat{\boldsymbol{s}}_B^\top \boldsymbol{c}_{u,B}.
\end{aligned}$$

The error-term $\hat{w}$ of $(v_B - \hat{\boldsymbol{s}}_B^\top \boldsymbol{u}_B)$ is defined as

$$\begin{aligned}
\hat{w} := w + (&e_{2,B} + (\hat{\boldsymbol{e}}_B^\top + \boldsymbol{c}_{t,B}^\top) \mathsf{BitDecomp}(\boldsymbol{u}_A) \\
&+ \boldsymbol{r}_{A\to B,3}^\top \cdot \mathsf{BitDecomp}(\boldsymbol{u}_A) - \hat{\boldsymbol{s}}_B^\top \boldsymbol{R}_{A\to B,2} \cdot \mathsf{BitDecomp}(\boldsymbol{u}_A) - \hat{\boldsymbol{s}}_B^\top \boldsymbol{c}_{u,B}.
\end{aligned}$$

In addition, let $\mathsf{m}' := \mathsf{Compress}_q(v_B - \hat{\boldsymbol{s}}_B^\top \boldsymbol{u}_B, 1)$. Hence, if $\|\hat{w}\|_\infty < \lceil q/4 \rfloor$, it holds that

$$\left\lceil \frac{q}{4} \right\rfloor \geq \left\| v_B - \boldsymbol{s}_B^\top \boldsymbol{u}_B - \left\lceil \frac{q}{2} \right\rfloor \cdot \mathsf{m}' \right\|_\infty = \left\| \hat{w} + \left\lceil \frac{q}{2} \right\rfloor \cdot \mathsf{m} - \left\lceil \frac{q}{2} \right\rfloor \cdot \mathsf{m}' \right\|_\infty .$$

Due to the triangle inequality and the fact $\|\hat{w}\|_\infty < \lceil q/4 \rfloor$, we obtain

$$\left\| \left\lceil \frac{q}{2} \right\rfloor \cdot (\mathsf{m} - \mathsf{m}') \right\|_\infty < 2 \cdot \left\lceil \frac{q}{4} \right\rfloor .$$

This indicates $\mathsf{m} = \mathsf{m}'$. Therefore, the reencryption-correctness is shown.

From the discussion above, we complete the proof of the correctness of the proposed PRE scheme K-PRE. □

B.2 Proof of Re-Encryption Key Homomorphism

We prove the re-encryption key homomorphism of K-PRE.

Proposition 6 (Re-encryption key homomorphism of K-PRE)**.** *Let* $\mathsf{pp} = (\lambda, \mu, N, N', q, \ell, \eta, k, d_t, d_u, d_v, \boldsymbol{A})$ *be a public parameter determined by running* $\mathsf{Setup}(1^\lambda)$*, and let* $A = \{a_1, \ldots, a_v\} \subseteq [u]$ *and* $B = \{b_1, \ldots, b_v\} \subseteq [u]$ *be two sets of distinct users. Then, the key-pairs of these users and a ciphertext are defined as follows:*

- *For each* $i \in [u]$*, let* $(\mathsf{pk}_{a_i}, \mathsf{sk}_{a_i}) = ((\boldsymbol{t}_{a_i}, \hat{\boldsymbol{t}}_{a_i}), (\boldsymbol{s}_{a_i}, \hat{\boldsymbol{s}}_{a_i}))$ *(resp.* $(\mathsf{pk}_{b_i}, \mathsf{sk}_{b_i}) = ((\boldsymbol{t}_{b_i}, \hat{\boldsymbol{t}}_{b_i}), (\boldsymbol{s}_{b_i}, \hat{\boldsymbol{s}}_{b_i}))$*) be the key-pair of the user* a_i *(resp. the user* b_i*), where* $\boldsymbol{t}_{a_i} = \mathsf{Compress}_q(\boldsymbol{A}\boldsymbol{s}_{a_i} + \boldsymbol{e}_{a_i}, d_t)$ *and* $\hat{\boldsymbol{t}}_{b_i} = \mathsf{Compress}_q(\boldsymbol{A}\hat{\boldsymbol{s}}_{b_i} + \hat{\boldsymbol{e}}_{b_i}, d_t)$*;*
- *Let* $\mathsf{ct}_A = (\boldsymbol{u}_A, v_A)$ *be a ciphertext generated by running* $\mathsf{Enc}(\mathsf{pk}_A, \mathsf{m})$ *for an arbitrary message* $\mathsf{m} \in \mathcal{M}$*, where* $\mathsf{pk}_A = \boldsymbol{t}_A = \sum_{i\in[u]} \boldsymbol{t}_{a_i}$*,* $\boldsymbol{u}_A = \mathsf{Compress}_q(\boldsymbol{A}^\top \boldsymbol{r} + \boldsymbol{e}_1^\top, d_u)$ *and* $v_A = \mathsf{Compress}_q(\boldsymbol{t}_A^\top \boldsymbol{r} + e_2 + \lceil \frac{q}{2} \rfloor \cdot \mathsf{m}, d_v)$*.*

Let $\boldsymbol{c}_{t,A}, \boldsymbol{c}_{t,B} \leftarrow \psi_{d_t}^k$*,* $\boldsymbol{c}_u \leftarrow \psi_{d_u}^k$*,* $c_v \leftarrow \psi_{d_v}$*. Let* $(\boldsymbol{U}_{A\to B}, \boldsymbol{v}_{a_i\to b_j})_{i\in[u],j\in[u]} \leftarrow \mathsf{HReKeyGen}((\mathsf{sk}_{a_i})_{i\in[u]}, (\mathsf{pk}_{b_j})_{j\in[u]})$ *and* $(\boldsymbol{U}_{A\to B}, \boldsymbol{v}_{A\to B}) \leftarrow \mathsf{ReKeyEval}((\mathsf{rk}_{a_i\to b_i})_{i\in[u]})$*.*

Denote

$$w := \left(\sum_{i\in[u]} \boldsymbol{r}_{a_i\to b_i}^\top\right) \mathsf{BitDecomp}(\boldsymbol{u}_A) + \left(\sum_{i\in[u]} \hat{\boldsymbol{e}}_{b_i} + \sum_{i\in[u]} \hat{\boldsymbol{c}}_{t,b_i}\right)^\top \boldsymbol{R}_{A\to B,1} + \hat{c}_{d_v}$$
$$- \hat{\boldsymbol{s}}_B^\top \boldsymbol{R}_{A\to B,2} \cdot \mathsf{BitDecomp}(\boldsymbol{u}_A) + \hat{\boldsymbol{s}}_B^\top \cdot \left(\sum_{i\in[u]} \hat{\boldsymbol{c}}_{t,b_i}\right); \text{ and}$$
$$\delta := \Pr\left[\|w\|_\infty \geq q/4\right]$$

for $\hat{c}_{d_u} \leftarrow \psi_{d_u}$ *and* $\hat{\boldsymbol{c}}_{t,b_i} \leftarrow \psi_{d_t}$ *($i \in [u]$), where let* $\hat{\boldsymbol{s}}_B := \sum_{i\in[u]} \hat{\boldsymbol{s}}_{b_i}$*, and for every* $i \in [u]$*,* $(\boldsymbol{R}_{A\to B,1}, \boldsymbol{R}_{A\to B,2}, \boldsymbol{r}_{a_i\to b_i})$ *is a tuple of values generated by running* $\mathsf{HReKeyGen}((\mathsf{sk}_i)_{i\in[u]}, (\mathsf{pk}_i)_{j\in[u]})$*.*

Then, the proposed PRE scheme K-PRE *satisfies* re-encryption key homomorphic *with probability* $1-\delta$*.*

Proof. We consider an arbitrary message $\mathsf{m} \in \mathcal{M}$ throughout the proof of Theorem 6. Recall that $\mathsf{pp} = (\lambda, \mu, N, N', q, \ell, \eta, k, d_t, d_u, d_v, \boldsymbol{A})$ is a public parameter determined by running $\mathsf{Setup}(1^\lambda)$, and let $A = \{a_1, \ldots, a_u\} \subseteq [n]$ and $B = \{b_1, \ldots, b_u\} \subseteq [n]$ be two sets of distinct users. For each $i \in [u]$, let $(\mathsf{pk}_{a_i}, \mathsf{sk}_{a_i}) = ((\boldsymbol{t}_{a_i}, \hat{\boldsymbol{t}}_{a_i}), (\boldsymbol{s}_{a_i}, \hat{\boldsymbol{s}}_{a_i}))$ (resp. $(\mathsf{pk}_{b_i}, \mathsf{sk}_{b_i}) = ((\boldsymbol{t}_{b_i}, \hat{\boldsymbol{t}}_{b_i}), (\boldsymbol{s}_{b_i}, \hat{\boldsymbol{s}}_{b_i}))$) be the key-pair of the user a_i (resp. the user b_i), where $\boldsymbol{t}_{a_i} = \mathsf{Compress}_q(\boldsymbol{A}\boldsymbol{s}_{a_i} + \boldsymbol{e}_{a_i}, d_t)$ and $\hat{\boldsymbol{t}}_{b_i} = \mathsf{Compress}_q(\boldsymbol{A}\hat{\boldsymbol{s}}_{b_i} + \hat{\boldsymbol{e}}_{b_i}, d_t)$.

For every $i \in [u]$, the values of $\boldsymbol{t}_{a_i}$ and $\hat{\boldsymbol{t}}_{b_i}$ is

$$\begin{aligned}
\boldsymbol{t}_{a_i} &= \mathsf{Decompress}_q(\mathsf{Compress}(\boldsymbol{A}\boldsymbol{s}_{a_i} + \boldsymbol{e}_{a_i}, d_t), d_t) \\
&= \boldsymbol{A}\boldsymbol{s}_{a_i} + \boldsymbol{e}_{a_i} + \boldsymbol{c}_{t,a_i}; \\
\hat{\boldsymbol{t}}_{b_i} &= \mathsf{Decompress}_q(\mathsf{Compress}(\boldsymbol{A}^\top\hat{\boldsymbol{s}}_{b_i} + \hat{\boldsymbol{e}}_{b_i}, d_t), d_t) \\
&= \boldsymbol{A}\hat{\boldsymbol{s}}_{b_i} + \hat{\boldsymbol{e}}_{b_i} + \hat{\boldsymbol{c}}_{t,b_i}
\end{aligned}$$

for some $(\boldsymbol{c}_{t,a_i}, \hat{\boldsymbol{c}}_{t,b_i}) \in R^k \times R^k$.

Then we define public keys $\mathsf{pk}_A, \mathsf{pk}_B$, as follows:

$$\mathsf{pk}_A := \sum_{i\in[u]} \boldsymbol{t}_{a_i} = \boldsymbol{A}\sum_{i\in[u]} \boldsymbol{s}_{a_i} + \sum_{i\in[u]} \boldsymbol{e}_{a_i} + \sum_{i\in[u]} \boldsymbol{c}_{t,a_i} = \boldsymbol{A}\boldsymbol{s}_A + \boldsymbol{e}_A + \boldsymbol{c}_{t,A};$$
$$\mathsf{pk}_B := \sum_{i\in[u]} \hat{\boldsymbol{t}}_{b_i} = \boldsymbol{A}\sum_{i\in[u]} \hat{\boldsymbol{s}}_{b_i} + \sum_{i\in[u]} \hat{\boldsymbol{e}}_{b_i} + \sum_{i\in[u]} \hat{\boldsymbol{c}}_{t,b_i} = \boldsymbol{A}\hat{\boldsymbol{s}}_B + \hat{\boldsymbol{e}}_B + \hat{\boldsymbol{c}}_{t,B},$$

where

- let $\boldsymbol{s}_A := \sum_{i\in[u]} \boldsymbol{s}_{a_i}$, $\boldsymbol{e}_A := \sum_{i\in[u]} \boldsymbol{e}_{a_i}$, and $\boldsymbol{c}_{t,A} := \sum_{i\in[u]} \boldsymbol{c}_{t,a_i}$; and
- let $\hat{\boldsymbol{s}}_B := \sum_{i\in[u]} \hat{\boldsymbol{s}}_{b_i}$, $\hat{\boldsymbol{e}}_B := \sum_{i\in[u]} \hat{\boldsymbol{e}}_{b_i}$, and $\hat{\boldsymbol{c}}_{t,B} := \sum_{i\in[u]} \hat{\boldsymbol{c}}_{t,b_i}$.

Let $\mathsf{ct}_A = (\boldsymbol{u}_A, v_A)$ be an encryption of m, under pk_A (i.e., $(\boldsymbol{u}_A, v_A) \leftarrow \mathsf{Enc}(\mathsf{pk}_A, \mathsf{m})$). The values of $\boldsymbol{u}_A$ and v_A are

$$\begin{aligned}
\boldsymbol{u}_A &= \mathsf{Decompress}_q(\mathsf{Compress}_q(\boldsymbol{A}^\top\boldsymbol{r} + \boldsymbol{e}_1, d_u), d_u) \\
&= \boldsymbol{A}^\top\boldsymbol{r} + \boldsymbol{e}_1 + \boldsymbol{c}_u; \text{ and} \\
v_A &= \mathsf{Decompress}_q(\mathsf{Compress}_q((\mathsf{pk}_A)^\top\boldsymbol{r} + e_2 + \lceil q/2\rfloor \cdot \mathsf{m}, d_v), d_v) \\
&= (\boldsymbol{A}\boldsymbol{s}_A + \boldsymbol{e}_A + \boldsymbol{c}_{t,A})^\top\boldsymbol{r} + e_2 + \lceil q/2\rfloor \cdot \mathsf{m} + c_v \\
&= (\boldsymbol{A}\boldsymbol{s}_A + \boldsymbol{e}_A)^\top\boldsymbol{r} + e_2 + \lceil q/2\rfloor \cdot \mathsf{m} + c_v + \boldsymbol{c}_{t,A}^\top\boldsymbol{r}
\end{aligned}$$

for some $(\boldsymbol{c}_u, c_v) \in R^k \times R$.

Let $(\mathsf{rk}_{a_i \to b_j})_{i\in[u],j\in[u]} \leftarrow \mathsf{HReKeyGen}((\mathsf{sk}_{a_i})_{i\in[u]}, (\mathsf{pk}_{b_j})_{j\in[u]})$. For every $i \in [u]$ and every $j \in [u]$, the value of the re-encryption key $\mathsf{rk}_{a_i\to b_j} = (\boldsymbol{U}_{A\to B}, \boldsymbol{v}_{a_i\to b_j})$ is

$$\begin{aligned}
\boldsymbol{U}_{A\to B} &= \boldsymbol{A}^\top \boldsymbol{R}_{A\to B,1} + \boldsymbol{R}_{A\to B,2};\\
\boldsymbol{v}_{a_i\to b_j} &= \hat{\boldsymbol{t}}_{b_j}^\top \boldsymbol{R}_{A\to B,1} + \boldsymbol{r}_{a_i\to b_j}^\top - \mathsf{Powersof2}(\boldsymbol{s}_{a_i}^\top).
\end{aligned}$$

Then, the homomorphicly evaluated value $\boldsymbol{v}_{A\to B}$ generated by running $(\boldsymbol{U}_{A\to B}, \boldsymbol{u}_{A\to B}) \leftarrow \mathsf{ReKeyEval}((\mathsf{rk}_{a_i\to b_i})_{i\in[u]})$ is

$$\boldsymbol{v}_{A\to B} := \sum_{i\in[u]} \boldsymbol{v}_{a_i\to b_i} = \sum_{i\in[u]} \hat{\boldsymbol{t}}_{b_i}^\top \boldsymbol{R}_{A\to B,1} + \sum_{i\in[u]} \boldsymbol{r}_{a_i\to b_i}^\top - \sum_{i\in[u]} \mathsf{Powersof2}(\boldsymbol{s}_{a_i}^\top).$$

Let $\mathsf{ct}_B = (\boldsymbol{u}_B, v_B)$ be a re-encrypted ciphertext generated by using the re-encryption key $(\boldsymbol{U}_{A\to B}, \boldsymbol{v}_{A\to B})$, and the value of $(\boldsymbol{u}_B, v_B)$ is

$$\begin{aligned}
\boldsymbol{u}_B &= (\boldsymbol{A}^\top \boldsymbol{R}_{A\to B,1} + \boldsymbol{R}_{A\to B,2}) \cdot \mathsf{BitDecomp}(\boldsymbol{u}_A);\\
v_B &= v_A + \left(\sum_{i\in[u]} \hat{\boldsymbol{t}}_{b_i}^\top \boldsymbol{R}_{A\to B,1} + \sum_{i\in[u]} \boldsymbol{r}_{a_i\to b_i}^\top - \sum_{i\in[u]} \mathsf{Powersof2}(\boldsymbol{s}_{a_i}^\top)\right) \mathsf{BitDecomp}(\boldsymbol{u}_A)\\
&= v_A - \sum_{i\in[u]} \boldsymbol{s}_{a_i}^\top \boldsymbol{u}_A + \left(\sum_{i\in[u]} \hat{\boldsymbol{t}}_{b_i}^\top \boldsymbol{R}_{A\to B,1} + \sum_{i\in[u]} \boldsymbol{r}_{a_i\to b_i}^\top\right) \mathsf{BitDecomp}(\boldsymbol{u}_A)\\
&= v_A - \boldsymbol{s}_A^\top \boldsymbol{u}_A + \left((\boldsymbol{A}\hat{\boldsymbol{s}}_B + \hat{\boldsymbol{e}}_B + \hat{\boldsymbol{c}}_{t,B})^\top \boldsymbol{R}_{A\to B,1} + \sum_{i\in[u]} \boldsymbol{r}_{a_i\to b_i}^\top\right) \mathsf{BitDecomp}(\boldsymbol{u}_A)\\
&= v_A - \boldsymbol{s}_A^\top \boldsymbol{u}_A\\
&\qquad + \hat{\boldsymbol{s}}_B^\top \boldsymbol{A}^\top \boldsymbol{R}_{A\to B,1} + \left(\sum_{i\in[u]} \boldsymbol{r}_{a_i\to b_i}^\top\right) \mathsf{BitDecomp}(\boldsymbol{u}_A) + (\hat{\boldsymbol{e}}_B + \hat{\boldsymbol{c}}_{t,B})^\top \boldsymbol{R}_{A\to B,1}.
\end{aligned}$$

Then, the decompressed values of $\boldsymbol{u}_B$ and v_B are

$$\begin{aligned}
\boldsymbol{u}_B &= \mathsf{Decompress}_q(\mathsf{Compress}_q(\boldsymbol{u}_A, d_u), d_u)\\
&= \boldsymbol{A}^\top \boldsymbol{R}_{A\to B,1} \cdot \mathsf{BitDecomp}(\boldsymbol{u}_A) + \boldsymbol{R}_{A\to B,2} \cdot \mathsf{BitDecomp}(\boldsymbol{u}_A) + \hat{\boldsymbol{c}}_{d_u};\\
v_B &= \mathsf{Decompress}_q(\mathsf{Compress}_q(v_A, d_v), d_v)\\
&= v_A - \boldsymbol{s}_A^\top \boldsymbol{u}_A\\
&\qquad + \hat{\boldsymbol{s}}_B^\top \boldsymbol{A}^\top \boldsymbol{R}_{A\to B,1} + \left(\sum_{i\in[u]} \boldsymbol{r}_{a_i\to b_i}^\top\right) \mathsf{BitDecomp}(\boldsymbol{u}_A)\\
&\qquad + (\hat{\boldsymbol{e}}_B + \hat{\boldsymbol{c}}_{t,B})^\top \boldsymbol{R}_{A\to B,1} + \hat{c}_{d_v}
\end{aligned}$$

for some $(\hat{\boldsymbol{c}}_{d_u}, \hat{c}_{d_v}) \in R^k \times R$. Hence, we have

$$\begin{aligned}
&v_A - \hat{\boldsymbol{s}}_B^\top \boldsymbol{u}_B \\
&= v_A - \boldsymbol{s}_A^\top \boldsymbol{u}_A \\
&\quad + \hat{\boldsymbol{s}}_B^\top \boldsymbol{A}^\top \boldsymbol{R}_{A\to B,1} + \left(\sum_{i\in[u]} \boldsymbol{r}_{a_i\to b_i}^\top\right) \mathsf{BitDecomp}(\boldsymbol{u}_A) + (\hat{\boldsymbol{e}}_B + \hat{\boldsymbol{c}}_{t,B})^\top \boldsymbol{R}_{A\to B,1} + \hat{c}_{d_v} \\
&\quad - \hat{\boldsymbol{s}}_B^\top \left(\boldsymbol{A}^\top \boldsymbol{R}_{A\to B,1} \cdot \mathsf{BitDecomp}(\boldsymbol{u}_A) + \boldsymbol{R}_{A\to B,2} \cdot \mathsf{BitDecomp}(\boldsymbol{u}_A) + \hat{\boldsymbol{c}}_{d_u}\right) \\
&= v_A - \boldsymbol{s}_A^\top \boldsymbol{u}_A + \left(\sum_{i\in[u]} \boldsymbol{r}_{a_i\to b_i}^\top\right) \mathsf{BitDecomp}(\boldsymbol{u}_A) + (\hat{\boldsymbol{e}}_B + \hat{\boldsymbol{c}}_{t,B})^\top \boldsymbol{R}_{A\to B,1} + \hat{c}_{d_v} \\
&\quad - \hat{\boldsymbol{s}}_B^\top \boldsymbol{R}_{A\to B,2} \cdot \mathsf{BitDecomp}(\boldsymbol{u}_A) + \hat{\boldsymbol{s}}_B^\top \hat{\boldsymbol{c}}_{d_u}.
\end{aligned}$$

The error-term w of $v_B - \hat{\boldsymbol{s}}_B^\top \boldsymbol{u}_B$ is defined as

$$\begin{aligned}
w := &\left(\sum_{i\in[u]} \boldsymbol{r}_{a_i\to b_i}^\top\right) \mathsf{BitDecomp}(\boldsymbol{u}_A) + (\hat{\boldsymbol{e}}_B + \hat{\boldsymbol{c}}_{t,B})^\top \boldsymbol{R}_{A\to B,1} + \hat{c}_{d_v} \\
&- \hat{\boldsymbol{s}}_B^\top \boldsymbol{R}_{A\to B,2} \cdot \mathsf{BitDecomp}(\boldsymbol{u}_A) + \hat{\boldsymbol{s}}_B^\top \hat{\boldsymbol{c}}_{d_u}.
\end{aligned}$$

Additionally, let $\mathsf{m}' := \mathsf{Compress}_q(v_A - \hat{\boldsymbol{s}}_B^\top \boldsymbol{u}_B, 1)$. Then it holds that

$$\left\lceil \frac{q}{4} \right\rfloor \geq \left\| v_B - \hat{\boldsymbol{s}}_B^\top \boldsymbol{u}_B - \left\lceil \frac{q}{2} \right\rfloor \cdot \mathsf{m}' \right\|_\infty = \left\| w + \left\lceil \frac{q}{2} \right\rfloor \cdot \mathsf{m} - \left\lceil \frac{q}{2} \right\rfloor \cdot \mathsf{m}' \right\|_\infty.$$

Due to the fact that $\|w\|_\infty < \lceil q/4 \rfloor$, it holds that

$$\left\| \left\lceil \frac{q}{2} \right\rfloor (\mathsf{m} - \mathsf{m}') \right\|_\infty < 2 \left\lceil \frac{q}{4} \right\rfloor,$$

and this indicates $\mathsf{m} = \mathsf{m}'$. The proof is completed. □