Bounded CCA2-Secure Proxy Re-encryption from Lattices

Shingo Sato[1,2] and Junji Shikata[1,3]

[1] Institute of Advanced Sciences, Yokohama National University, Yokoaham, Japan
[2] Organization for the Promotion of Education, Yokohama National University, Yokoaham, Japan
[3] Graduate School of Environment and Information Sciences, Yokohama National University, Yokoaham, Japan
`{sato-shingo-zk, shikata-junji-rb}@ynu.ac.jp`

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 a generic construction of bounded $\mathsf{CCA2}$-secure PRE. Our generic construction starts 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 construction, we present a lattice-based PRE scheme with the required property. As a result, we can construct a bounded $\mathsf{CCA2}$-secure PRE scheme from lattices.

Keywords: Proxy re-encryption · Bounded CCA2 security · Post-quantum cryptography.

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. Additionally, PRE is classified into several types, as follows:

Unidirectional vs. Bidirectional. A PRE scheme is said to be *bidirectional* if $\mathsf{rk}_{A \to B}$ also allows to convert from ciphertexts under pk_B to pk_A. Meanwhile, *unidirectional* PRE just allows re-encryption from pk_A to pk_B, and not vice versa. Unidirectional PRE is more desirable because the proxies of bidirectional PRE are authorized in order to realize re-encryption functionalities.

Single-hop vs. Multi-hop. Suppose that ct_A is transformed to ct_B by using $\mathsf{rk}_{A \to B}$, and pk_C is another user's public key. A PRE scheme is said to be *single-hop* if a re-encryption key $\mathsf{rk}_{B \to C}$ cannot convert ct_B to another ciphertext ct_C under pk_C. In contrast, *multi-hop* PRE allows convert a ciphertext ct_B into a ciphertext ct_C under pk_C by using $\mathsf{rk}_{B \to C}$.

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 [29]. Accordingly, there are many PRE schemes such as Diffie-Hellman-based schemes (e.g., [5, 7, 11, 28]), 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 many researches on selected algorithms and candidates in the NIST (National Institute of Standards and Technology) PQC standardization process (e.g., [1,15,19,27,31]), due to advancement of quantum computers. As post-quantum PRE, several lattice-based PRE schemes have been proposed so far (e.g., [9, 16, 29, 34, 35]). 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 this security is one of the most desirable security notions and provides a wide range of applications, for PRE.

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, 29, 35]. 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 [29] 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 [34] presented a LWE-based construction of HRA-secure fine-grained PRE whose notion was introduced in [35].

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 bounded CCA2 security game 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 is sufficient (e.g., see [13, 20, 33]). Here, compact ciphertexts for bounded CCA2-secure PRE mean that ciphertext size does not depend on the parameters t_d, t_r linearly.

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, and 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]. The formalized security notion is a bounded variant of the standard CCA2 security (concretely, [11, Definition 2]) for single-hop unidirectional PRE.
- As a new property of PRE, we introduce re-encrytption key homomorphism in order to construct the objective bounded CCA2-secure PRE scheme. Accordingly, 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 lattices. This scheme is based on the underlying PKE scheme of CRYSTALS-Kyber (Kyber, for short) [6] which is a selected key encapsulation mechanism in the NIST PQC competition. 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 $\mathsf{CCA2}$-secure post-quantum PRE scheme with compact ciphertexts by applying our generic construction to our lattice-based PRE scheme. Furthermore, our lattice-based PRE is simple and practical because this scheme is constructed just by adding the re-encryption key generation and re-encryption algorithms to slightly modified Kyber's key generation, encryption, and decryption algorithms. Hence, the resulting bounded $\mathsf{CCA2}$-secure PRE scheme is also constructed simply.

1.3 Technical Overview

We explain an overview of technical aspects of our bounded $\mathsf{CCA2}$-secure PRE with compact ciphertexts.

Bounded $\mathsf{CCA2}$-secure PRE from CPA secure PRE. To construct bounded $\mathsf{CCA2}$-secure PRE with compact ciphertexts, we first consider a basic generic construction of bounded $\mathsf{CCA2}$-secure PRE $\Pi_{\mathsf{B\text{-}PRE}}$. This construction is based on the generic construction of bounded $\mathsf{CCA2}$-secure PKE [13], so that $\Pi_{\mathsf{B\text{-}PRE}}$ achieves the bounded $\mathsf{CCA2}$ security of PRE. Furthermore, $\Pi_{\mathsf{B\text{-}PRE}}$ 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 $\Pi_{\mathsf{B\text{-}PRE}}$ 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 family [4]. 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 $\Pi_{\mathsf{B\text{-}PRE}}$ 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 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

[4]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.

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 $\Pi_{\mathsf{B\text{-}PRE}}$ achieves the re-encryption functionality. This concrete construction and its security proof are given in Appendix A.

Bounded CCA2-secure PRE with Compact Ciphertexts. We explain an overview of our bounded CCA2-secure PRE $\Pi_{\mathsf{C\text{-}PRE}}$ with compact ciphertexts. This construction is based on the basic scheme $\Pi_{\mathsf{B\text{-}PRE}}$. 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 a property: secret-key to public-key homomorphism, which was formalized in [30].

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 $\Pi_{\mathsf{B\text{-}PRE}}$ 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 $\Pi_{\mathsf{B\text{-}PRE}}$, 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 related 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 way 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 $\sum_{i\in\{1,\ldots,v\}} \mathsf{rk}'_{(A,\alpha_i)\to(B,\beta_i)}$ is possible. Finally, we give a security proof for $\Pi_{\mathsf{C\text{-}PRE}}$ by assuming the RKH-CPA security of the underlying PRE.

Instantiation of Our Generic Construction. We construct a lattice-based PRE scheme to instantiate our generic construction. As mentioned beforehand, the key generation, encryption, and decryption algorithms of our scheme are constructed by slightly modifying those of Kyber PKE scheme. Although we need the introduced property re-encryption key homomorphism, all the existing post-quantum scheme cannot generate re-encryption keys satisfying such homomorphism. Hence, we present a new re-encryption key generation algorithm so that our lattice-based PRE scheme satisfies the objective homomorphism.

Additionally, we can use existing strongly unforgeable OTS schemes [25, 26] based on lattice assumptions, to instantiate our generic construction. As a result, we can obtain the objective bounded $\mathsf{CCA2}$-secure post-quantum PRE scheme.

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 $\boldsymbol{v} = (v_1, \ldots, v_n)^\top$. 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, $s \overset{\$}{\leftarrow} S$ means that an element $s \in S$ is chosen uniformly at random. For a positive integer η, $v \leftarrow \mathcal{U}_\eta$ denotes that each coefficient of $v \in R$ is drawn from a uniform distribution $\mathcal{U}_\eta$ over $[\eta]$. In the same way as this, $\boldsymbol{v} \leftarrow \mathcal{U}_\eta^k$ means that a k-dimensional vector $\boldsymbol{v} \in R^k$ is chosen from $\mathcal{U}_\eta^k$.

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* $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{ReKeyGen}, \mathsf{ReEnc})$*:*

- $\mathsf{Setup}(1^\lambda) \to \mathsf{pp}$*: 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.
- $\mathsf{Enc}(\mathsf{pk}, \mathsf{m}) \to \mathsf{ct}$*: The randomized algorithm* Enc *takes as input a pubic key* pk *and a message* $\mathsf{m} \in \mathcal{M}$*, and outputs a ciphertext* ct.

- $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) \to \mathsf{m}/\bot$: *The deterministic algorithm* Dec *takes as input a secret key* sk *and a ciphertext* ct*, and outputs a message* m *or the rejection symbol* $\bot$*.*
- $\mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j) \to \mathsf{rk}_{i\to j}$: *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* $\mathsf{Enc}, \mathsf{Dec}, \mathsf{ReKeyGen}, \mathsf{ReEnc}$*.*

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

Encryption Correctness. *For every* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$*, 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* $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KeyGen}(\mathsf{pp}), (\mathsf{pk}_j, \mathsf{sk}_j) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$*, 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 [8,11], we describe definitions of oracles in security games of PRE.

Definition 3. *An adversary against a PRE scheme* ($\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{ReKeyGen}, \mathsf{ReEnc}$) *is given access to the following oracles in a security game of PRE: Suppose that a public parameter* pp *and* n *key-pairs* $(\mathsf{pk}_1, \mathsf{sk}_1), \dots, (\mathsf{pk}_n, \mathsf{sk}_n)$ *are generated by* Setup *and* KeyGen *for* $n = \mathsf{poly}(\lambda)$*, respectively, where* λ *is a security parameter.*

- *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* $\bot$ *if* $i = i^* \wedge j \in \mathcal{L}_{\mathtt{Corrupt}}$ *or* $i = j$ *holds (where* i^* *is the user-index issued to the challenge query below); 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* $\bot$ *if* $i^* \in \mathcal{L}_{\mathtt{Corrupt}}$ *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* $\bot$ *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* (i, ct_i) *is a* derivative *of* (i^*, ct^*) *and* $j \in \mathcal{L}_{\mathtt{Corrupt}}$ *holds; 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 [11]:

Definition 4 (Derivatives for CCA2 security [11]). *Let* $\Pi_{\mathsf{PRE}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{ReKeyGen}, \mathsf{ReEnc})$ *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* Π_{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* Π_{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 chosen ciphertext attacks* (denoted as bounded CCA2 security) by following [11, 13]. For simplicity, we consider security notions under selective corruptions, rather than adaptive corruptions, throughout this paper.

Definition 5 (Bounded CCA2 security). *Let* t_d, t_r *be positive integers. For a security parameter, let* $n = \mathsf{poly}(\lambda)$ *be a positive integer. A single-hop unidirectional PRE scheme* $\Pi_{\mathsf{PRE}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{ReKeyGen}, \mathsf{ReEnc})$ *is* (t_d, t_r)-CCA2*-secure if for any PPT adversary* $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2)$ *against* Π_{PRE}*, its advantage*

$$\mathsf{Adv}^{(t_d,\,t_r)\text{-cca2}}_{\Pi_{\mathsf{PRE}},\mathcal{A},n}(\lambda) := \left|\Pr[\mathsf{Expt}^{(t_d,\,t_r)\text{-cca2}}_{\Pi_{\mathsf{PRE}},\mathcal{A},n}(\lambda) = 1] - 1/2\right|$$

is negligible in λ*, where the experiment* $\mathsf{Expt}^{(t_d,\,t_r)\text{-cca2}}_{\Pi_{\mathsf{PRE}},\mathcal{A},n}(\lambda)$ *is defined as follows:*

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

where $\mathcal{A}$ *is allowed to issue at most* t_d *queries to* $\mathtt{O.Dec}$ *and at most* t_r *queries to* $\mathtt{O.ReEnc}$*,* $\mathcal{L}_{\mathtt{Corrupt}}$ *is a subset of* $[n]$*, 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* $\Pi_{\mathsf{OTS}} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Vrfy})$ *is* strongly unforgeable *if for any PPT adversary against* Π_{OTS}*, its advantage* $\mathsf{Adv}^{\text{suf-ot}}_{\Pi_{\mathsf{OTS}},\mathcal{A}}(\lambda) = \Pr[\mathcal{A}\ wins]$ *is negligible in* λ*, where* $[\mathcal{A}\ wins]$ *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* $\mathcal{L}_{\mathsf{Sign}} \leftarrow \emptyset$*, 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* $\mathcal{L}_{\mathsf{Sign}} \neq \emptyset$*; otherwise it returns* $\sigma \leftarrow \mathsf{Sign}(\mathsf{sigk}, \mathsf{m})$ *and sets* $\mathcal{L}_{\mathsf{Sign}} \leftarrow \mathcal{L}_{\mathsf{Sign}} \cup \{(\mathsf{m}, \sigma)\}$*.*

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

2.3 Module-Learning with Errors Assumption

We describe the definition of the module-learning with errors (MLWE) assumption with uniform error distributions, 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 $\boldsymbol{a}_i \xleftarrow{\$} R_q^k$, $\boldsymbol{s} \xleftarrow{\$} \mathcal{U}_\eta^k$, and $e_i \xleftarrow{\$} \mathcal{U}_\eta$ are samples (uniformly) at random, and $b_i = \boldsymbol{a}_i^\top \boldsymbol{s} + e_i$.*

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

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

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 each column vector of a t-disjunct matrix $\boldsymbol{M}$ is some positive integer v.

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

3 Bounded CCA2-Secure PRE with Compact Ciphertexts

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

3.1 Re-encryption Key Homomorphism of PRE

In order to construct a bounded $\mathsf{CCA2}$-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 [30].

Definition 11 (Re-encryption key homomorphism). *Throughout this section, we consider a PRE scheme* $\Pi_{\mathsf{PRE}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{ReKeyGen}, \mathsf{ReEnc})$ *with the property that the secret keys, public keys and re-encryption keys are elements of groups $\mathcal{K}_{\mathsf{sk}}$, $\mathcal{K}_{\mathsf{pk}}$, and $\mathcal{K}_{\mathsf{rk}}$, respectively. For simplicity, we suppose that all of the group operations of $\mathcal{K}_{\mathsf{sk}}$, $\mathcal{K}_{\mathsf{pk}}$, and $\mathcal{K}_{\mathsf{rk}}$ are $+$. The PRE scheme* Π_{PRE} *is said to be* $\mathsf{re\text{-}encryption\ key\ homomorphic}$ *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 or deterministic 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* $\forall i \in [u], \forall j \in [u] : \mathsf{pk}_{A,i} \neq \mathsf{pk}_{B,j}$*), and outputs* u *re-encryption keys* $(\mathsf{rk}_{(A,i) \to (B,j)})_{i \in [u], j \in [u]}$*.*
– $\mathsf{ReKeyEval}((\mathsf{rk}_{(A,\alpha_i) \to (B,\beta_i)})_{i \in [u]}) \to \mathsf{rk}_{A \to B}$*: The deterministic or randomized algorithm* $\mathsf{ReKeyEval}$ *takes as input* v *re-encryption keys* $(\mathsf{rk}_{(A,\alpha_i) \to (B,\beta_i)})_{i \in [v]}$ *(where* $v \leq u$ *and* $\forall i \in [v] : \alpha_i \in [u] \wedge \beta_i \in [u]$*) 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* $\{(\mathsf{pk}_{i,j}, \mathsf{sk}_{i,j}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})\}_{i \in \{A,B\}, j \in [u]}$*, every* $(\mathsf{rk}_{(A,i) \to (B,j)})_{i \in [u], j \in [u]} \leftarrow \mathsf{HReKeyGen}((\mathsf{sk}_{A,i})_{i \in [u]}, (\mathsf{pk}_{B,i})_{i \in [u]})$*, every* $\mathsf{rk}_{A \to B} \leftarrow \mathsf{ReKeyEval}((\mathsf{rk}_{(A,\alpha_i) \to (B,\beta_i)})_{i \in [v]})$ *(for any* $\{(\alpha_i, \beta_i) \in [u] \times [u]\}_{i \in [v]}$*), and every* $\mathsf{m} \in \mathcal{M}$*, it holds that* $\mathsf{Dec}(\mathsf{sk}_B, \mathsf{ct}_B) = \mathsf{m}$ *with overwhelming probability, where* $\mathsf{ct}_B \leftarrow \mathsf{ReEnc}(\mathsf{rk}_{A \to B}, \mathsf{ct}_A)$*,* $\mathsf{ct}_A \leftarrow \mathsf{Enc}(\mathsf{pk}_A, \mathsf{m})$*,* $\mathsf{pk}_A = \mu(\mathsf{pk}_{A_1}, \dots, \mathsf{pk}_{A_u})$*, and* $\mathsf{sk}_B = \mu(\mathsf{sk}_{B_1}, \dots, \mathsf{sk}_{B_u})$*.*

Due to the additional property above, we need to introduce *security against chosen plaintext attacks with re-encryption key homomorphism* (denoted by RKH-CPA security) as a new security notion, since the two algorithms ReKeyGen and 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 `O.HReKeyGen`, instead of `O.ReKeyGen`. This formalization is Definition 12. Notice that, for simplicity, user-indices in Definition 12 are regarded as index-pairs $(i, j) \in [n] \times [u]$ rather than $i \in [n]$.

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

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

where $\mathcal{L}_{\mathtt{Corrupt}}$ *is a subset of* $[n] \times [u]$ *in the experiment above,* $(\mathsf{state}_0, \mathsf{state}_1)$ *is internal state information, and the oracle* `O.HReKeyGen` *is defined as follows:*

- *Homomorphic re-encryption key generation oracle* $\texttt{O.HReKeyGen}$ *given a homomorphic re-encryption key query* $((A,i)_{i\in[u]},(B,j)_{j\in[u]})$ *(where* $A,B\in[n]$*) returns* $\perp$ *if there exists some index* $(\hat{i},\hat{j})\in[u]\times[u]$ *such that* $((A,\hat{i})=(i^*,j^*)\wedge(B,\hat{j})\in\mathcal{L}_{\texttt{Corrupt}})$ *or* $(A,\hat{i})=(B,\hat{j})$ *(where* $\mathcal{L}_{\texttt{Honest}}=[n]\backslash\mathcal{L}_{\texttt{Corrupt}}$*); otherwise, it returns* $(\mathsf{rk}_{(A,i)\to(B,j)})_{i\in[u],j\in[u]}\leftarrow\mathsf{HReKeyGen}((\mathsf{sk}_{A,i})_{i\in[u]},(\mathsf{pk}_{B,j})_{j\in[u]})$.

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

The proposed scheme $\Pi_{\mathsf{C\text{-}PRE}}$ employs the following cryptographic primitives:

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

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

- $\mathsf{Setup}(1^\lambda)\to\mathsf{pp}$: Generate $\mathsf{pp}'\leftarrow\Pi'_{\mathsf{PRE}}.\mathsf{Setup}(\mathsf{pp})$. Let $\mathcal{M}=\mathcal{M}(\lambda)$ be the message space, which is the same as that space of Π'_{PRE}, let $\bar{n}=\bar{n}(\lambda),u=u(\lambda)$ be positive integers, and let $[\bar{n}]$ be the verification key space of Π_{OTS} [5]. 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 $(\mathsf{pk}'_i,\mathsf{sk}'_i)\leftarrow\Pi'_{\mathsf{PRE}}.\mathsf{KeyGen}(\mathsf{pp}')$ 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 $(\mathsf{vk},\mathsf{sigk})\leftarrow\Pi_{\mathsf{OTS}}.\mathsf{KeyGen}(1^\lambda)$.
  3. Compute $\{\tau_1,\dots,\tau_v\}\leftarrow\phi_{\boldsymbol{M}}(\mathsf{vk})$, where all $\tau_1,\dots,\tau_v\in[u]$ are distinct.
  4. Compute $\mathsf{pk}'_{\mathsf{vk}}\leftarrow\sum_{i\in[v]}\mathsf{pk}'_{\tau_i}$ and $\mathsf{ct}'_{\mathsf{vk}}\leftarrow\Pi'_{\mathsf{PRE}}.\mathsf{Enc}(\mathsf{pk}'_{\mathsf{vk}},\mathsf{m})$.
  5. Compute $\sigma\leftarrow\Pi_{\mathsf{OTS}}.\mathsf{Sign}(\mathsf{sigk},\mathsf{ct}'_{\mathsf{vk}})$.
  6. Output $\mathsf{ct}=(\mathsf{vk},\mathsf{ct}'_{\mathsf{vk}},\sigma)$.

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

- $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) \to \mathsf{m}/\bot$: Parse $\mathsf{sk} = (\mathsf{sk}'_i)_{i \in [u]}$ and $\mathsf{ct} = (\mathsf{vk}, \mathsf{ct}'_{\mathsf{vk}}, \sigma)$. Output $\bot$ if $\Pi_{\mathsf{OTS}}.\mathsf{Vrfy}(\mathsf{vk}, \mathsf{ct}'_{\mathsf{vk}}, \sigma) = \bot$ holds; otherwise, compute $\{\tau_1, \ldots, \tau_v\} \leftarrow \phi_{\boldsymbol{M}}(\mathsf{vk})$, $\mathsf{sk}'_{\mathsf{vk}} \leftarrow \sum_{i \in [v]} \mathsf{sk}'_{\tau_i}$, and output $\mathsf{m}' \leftarrow \Pi'_{\mathsf{PRE}}.\mathsf{Dec}(\mathsf{sk}'_{\mathsf{vk}}, \mathsf{ct}'_{\mathsf{vk}})$.
- $\mathsf{ReKeyGen}(\mathsf{sk}_A, \mathsf{pk}_B) \to \mathsf{rk}_{A \to B}$:
  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 $(\mathsf{rk}_{(A,i) \to (B,j)})_{i \in [u], j \in [u]} \leftarrow \Pi'_{\mathsf{PRE}}.\mathsf{HReKeyGen}((\mathsf{sk}'_{A,i})_{i \in [u]}, (\mathsf{pk}'_{B,j})_{j \in [u]})$.
  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 $\bot$ if $\Pi_{\mathsf{OTS}}.\mathsf{Vrfy}(\mathsf{vk}_A, \mathsf{ct}'_{\mathsf{vk}_A}, \sigma_A) = \bot$ holds.
  3. Generate $(\mathsf{vk}_B, \mathsf{sigk}_B) \leftarrow \Pi_{\mathsf{OTS}}.\mathsf{KeyGen}(1^\lambda)$.
  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 $\mathsf{rk}_{\mathsf{vk}_A \to \mathsf{vk}_B} \leftarrow \Pi'_{\mathsf{PRE}}.\mathsf{ReKeyEval}((\mathsf{rk}_{(A,\alpha_i) \to (B,\beta_i)})_{i \in [v]})$.
  6. Compute $\mathsf{ct}'_{\mathsf{vk}_B} \leftarrow \Pi'_{\mathsf{PRE}}.\mathsf{ReEnc}(\mathsf{rk}_{\mathsf{vk}_A \to \mathsf{vk}_B}, \mathsf{ct}'_{\mathsf{vk}_A})$.
  7. Compute $\sigma_B \leftarrow \Pi_{\mathsf{OTS}}.\mathsf{Sign}(\mathsf{sigk}_B, \mathsf{ct}'_{\mathsf{vk}_B})$.
  8. Output $\mathsf{ct}_B = (\mathsf{vk}_B, \mathsf{ct}'_{\mathsf{vk}_B}, \sigma_B)$.

Due to the correctness of $\Pi'_{\mathsf{PRE}}, \Pi_{\mathsf{OTS}}$ and the re-encryption key homomorphism of Π'_{PRE}, the correctness of $\Pi_{\mathsf{C\text{-}PRE}}$ holds. Proposition 1 shows this correctness, and we omit the proof of this proposition because this is proved clearly.

Proposition 1 (Correctness of $\Pi_{\mathsf{C\text{-}PRE}}$). *If the PRE scheme Π'_{PRE} is* correct *and* re-encryption key-homomorphic*, and the OTS scheme Π_{OTS} is* correct*, then the resulting PRE scheme $\Pi_{\mathsf{C\text{-}PRE}}$ is* correct.

3.3 Security Proof

The following theorem shows the bounded CCA2 security of $\Pi_{\mathsf{C\text{-}PRE}}$:

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

In particular, if there exists a PPT algorithm $\mathcal{A}$ against a (t,t)-CCA2-secure PRE scheme $\Pi_{\mathsf{C\text{-}PRE}}$, then there exists a PPT algorithm $\mathcal{B}$ against a u-RKH-CPA-secure PRE scheme Π'_{PRE} and a PPT algorithm $\mathcal{F}$ against strongly unforgeable *OTS scheme Π_{OTS}, such that*

$$\mathsf{Adv}^{(t,\,t)\text{-cca2}}_{\Pi_{\mathsf{C\text{-}PRE}},\mathcal{A},n}(\lambda) \leq n_h^3 u^2 \cdot \mathsf{Adv}^{u\text{-rkh-cpa}}_{\Pi'_{\mathsf{PRE}},\mathcal{B},n}(\lambda) + \mathsf{Adv}^{\text{suf-ot}}_{\Pi_{\mathsf{OTS}},\mathcal{F}}(\lambda).$$

Proof. Let $\mathcal{A}$ be a PPT adversary against the PRE scheme $\Pi_{\mathsf{C\text{-}PRE}}$. 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 outputs 1 in Game_i.
$\underline{\mathsf{Game}_0}$: This game is the same as the ordinary (t,t)-CCA2 game. Then, we have $\mathsf{Adv}^{(t,\,t)\text{-cca2}}_{\Pi_{\mathsf{C\text{-}PRE}},\mathcal{A},n}(\lambda) = |\Pr[W_0] - 1/2|$.

$\underline{\mathsf{Game}_1}$: 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 $\mathtt{O.ReEnc}$: At the beginning of the game, the experiment generates $(\mathsf{vk}^*, \mathsf{sigk}^*) \leftarrow \Pi_{\mathsf{OTS}}.\mathsf{KeyGen}(1^\lambda)$. 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 aborts if it holds that $\mathsf{vk}_i = \mathsf{vk}^*$, $\mathsf{ct}_i \neq \mathsf{ct}^*$, and $\Pi_{\mathsf{OTS}}.\mathsf{Vrfy}(\mathsf{vk}_i, \mathsf{ct}'_{\mathsf{vk}_i}, \sigma_i) = \top$ (this event is denoted as Bad); otherwise, it returns the result of $\mathtt{O.Dec}(i, \mathsf{ct}_i)$ (resp., $\mathtt{O.ReEnc}(i, j, \mathsf{ct}_i)$).

It is clear that Game_0 and Game_1 are identical unless Bad occurs. Thus, we construct a PPT algorithm $\mathcal{F}$ breaking the strongly unforgeable OTS scheme Π_{OTS} so that we bound the probability that Bad occurs.

On input a verification key vk^* of Π_{OTS}, $\mathcal{F}$ generates $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$ and $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$ for every $i \in [n]$ and gives $(\mathsf{pp}, \{\mathsf{pk}_i\}_{i\in[n]})$ to $\mathcal{A}$. Given $\mathcal{L}_{\mathtt{Corrupt}}$, $\mathcal{F}$ returns $\{\mathsf{sk}_i\}_{i\in\mathcal{L}_{\mathtt{Corrupt}}}$. 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 (resp. 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 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 $b \overset{\$}{\leftarrow} \{0,1\}$ 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)-CCA2 game since it has all key-pairs $(\mathsf{pk}_i, \mathsf{sk}_i)$. Hence, we have $\Pr[\mathsf{Bad}] \leq \mathsf{Adv}^{\text{suf-ot}}_{\Pi_{\mathsf{OTS}},\mathcal{F}}(\lambda)$, and it holds that $|\Pr[W_0] - \Pr[W_1]| \leq \mathsf{Adv}^{\text{suf-ot}}_{\Pi_{\mathsf{OTS}},\mathcal{F}}(\lambda)$.

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

- $\mathtt{O.ReKeyGen}(A, B)$: $\mathcal{B}$ returns $\perp$ if $A = i^* \wedge B \in \mathcal{L}_{\mathtt{Corrupt}}$ holds. If $\mathsf{T}_{\mathsf{rk}}[A,B] = \mathsf{rk}_{A\to B}$, $\mathcal{B}$ returns $\mathsf{rk}_{A\to B}$; 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 u-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 $\mathsf{derivative}$. 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 $\mathsf{vk}_A = \mathsf{vk}^*$, $\mathsf{ct}_A \neq \mathsf{ct}^*$, and $\Pi_{\mathsf{OTS}}.\mathsf{Vrfy}(\mathsf{vk}_A, \mathsf{ct}'_{\mathsf{vk}_A}, \sigma_A) = \top$.
  3. Return $\perp$ if $\Pi_{\mathsf{OTS}}.\mathsf{Vrfy}(\mathsf{vk}_A, \mathsf{ct}'_{\mathsf{vk}_A}, \sigma_A) = \perp$ 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 $\mathsf{m}' \leftarrow \Pi'_{\mathsf{PRE}}.\mathsf{Dec}(\mathsf{sk}_{\mathsf{vk}_A}, \mathsf{ct}'_{\mathsf{vk}_A})$.
- $\mathtt{O.ReEnc}(A, B, \mathsf{ct}_A)$: For $\mathsf{ct}_A = (\mathsf{vk}_A, \mathsf{ct}'_{\mathsf{vk}_A}, \sigma_A)$, $\mathcal{B}$ returns $\perp$ if (A, ct_A) is a $\mathsf{derivative}$ of (i^*, ct^*) and $B \in \mathcal{L}_{\mathtt{Corrupt}}$ holds. 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 $\mathsf{vk}_A = \mathsf{vk}^*$, $\mathsf{ct}_A \neq \mathsf{ct}^*$, and $\Pi_{\mathsf{OTS}}.\mathsf{Vrfy}(\mathsf{vk}_A, \mathsf{ct}'_A, \sigma_A) = \top$.
  3. Return $\perp$ if $\Pi_{\mathsf{OTS}}.\mathsf{Vrfy}(\mathsf{vk}_A, \mathsf{ct}'_{\mathsf{vk}_A}, \sigma_A) = \perp$ holds.
  4. If $\mathsf{T}_{\mathsf{rk}}[A, B] = \emptyset$, compute $(\mathsf{rk}_{(A,i)\to(B,j)})_{i\in[u],j\in[u]} \leftarrow \mathsf{HReKeyGen}((\mathsf{sk}_{A,i})_{i\in[u]}, (\mathsf{pk}_{B,j})_{j\in[u]})$ and set $\mathsf{T}_{\mathsf{rk}}[A, B] \leftarrow (\mathsf{rk}_{(A,i)\to(B,j)})_{i\in[u],j\in[u]}$. If $\mathsf{T}_{\mathsf{rk}}[A, B] = \mathsf{rk}_{A\to B}$, parse $\mathsf{rk}_{A\to B} = (\mathsf{rk}_{(A,i)\to(B,j)})_{i\in[u],j\in[u]}$.
  5. Generate $(\mathsf{vk}_B, \mathsf{sigk}_B) \leftarrow \Pi_{\mathsf{OTS}}.\mathsf{KeyGen}(1^\lambda)$.
  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. Compute $\mathsf{rk}_{\mathsf{vk}_A\to\mathsf{vk}_B} \leftarrow \sum_{i\in[v]} \mathsf{rk}_{(A,\alpha_i)\to(B,\beta_i)}$.
  8. Compute $\mathsf{ct}'_{\mathsf{vk}_B} \leftarrow \mathsf{ReEnc}(\mathsf{rk}_{\mathsf{vk}_A\to\mathsf{vk}_B}, \mathsf{ct}'_{\mathsf{vk}_A})$.
  9. Compute $\sigma_B \leftarrow \Pi_{\mathsf{OTS}}.\mathsf{Sign}(\mathsf{sigk}_B, \mathsf{ct}'_{\mathsf{vk}_B})$.
  10. Return $\mathsf{ct}_B = (\mathsf{vk}_B, \mathsf{ct}'_{\mathsf{vk}_B}, \sigma_B)$.
- $\mathtt{O.Challenge}_b(i', \mathsf{m}_0^*, \mathsf{m}_1^*)$:
  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 u-RKH-CPA game.
  3. Compute $\sigma^* \leftarrow \Pi_{\mathsf{OTS}}.\mathsf{Sign}(\mathsf{sigk}^*, \mathsf{ct}'^*_{\mathsf{vk}^*})$.
  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 $\Pr[\neg\mathsf{Abort}] \geq 1/(n_h^3 u^2)$ hold since Abort can occur in the simulation of the oracles $\mathtt{O.Dec}, \mathtt{O.ReEnc}, \mathtt{O.Challenge}_b$. Hence, we have

$$\begin{aligned}
\Pr[W_{\mathcal{B}}] &= \Pr[\mathsf{Abort}] \cdot \Pr[W_{\mathcal{B}} \mid \mathsf{Abort}] + \Pr[\neg\mathsf{Abort}] \cdot \Pr[W_{\mathcal{B}} \mid \neg\mathsf{Abort}] \\
&\geq \frac{1}{2}\left(1 - \frac{1}{n_h^3 u^2}\right) + \frac{1}{n_h^3 u^2} \cdot \Pr[W_{\mathcal{B}} \mid \neg\mathsf{Abort}] \\
&= \frac{1}{2} + \frac{1}{n_h^3 u^2}\left(\Pr[W_{\mathcal{B}} \mid \neg\mathsf{Abort}] - \frac{1}{2}\right).
\end{aligned}$$

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 $\varepsilon_{\mathcal{A}}/(n_h^3 u^2)$.

From the above discussion, we obtain

$$\mathsf{Adv}^{(t,\,t)\text{-cca2}}_{\Pi_{\mathsf{C\text{-}PRE}},\mathcal{A},n}(\lambda) \leq n_h^3 u^2 \cdot \mathsf{Adv}^{u\text{-kh-cpa}}_{\Pi'_{\mathsf{PRE}},\mathcal{B},n}(\lambda) + \mathsf{Adv}^{\text{suf-ot}}_{\Pi_{\mathsf{OTS}},\mathcal{F}}(\lambda),$$

and complete the proof. □

4 Re-Encryption Key Homomorphic PRE from Lattices

In order to instantiate our generic construction with compact ciphertexts, we give a lattice-based PRE scheme $\Pi_{\mathsf{L\text{-}PRE}}$ with re-encryption key homomorphism and prove that $\Pi_{\mathsf{L\text{-}PRE}}$ is RKH-CPA secure. The algorithms Setup, KeyGen, Enc, Dec of $\Pi_{\mathsf{L\text{-}PRE}}$ are similar to those of Kyber's underlying PKE scheme [6]. For simplicity, we consider a modified Kyber's algorithms, as follows: (i) Procedures with compression functions used in Kyber's PKE scheme are omitted in $\Pi_{\mathsf{L\text{-}PRE}}$; (ii) The distributions of secret keys and error vectors of $\Pi_{\mathsf{L\text{-}PRE}}$ are uniform distributions while Kyber's PKE scheme uses central binomial distributions which make easier to sample [24].

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 $\Pi_{\mathsf{L\text{-}PRE}}$, we employ the following functions:

– The bit-decomposition function BitDecomp given a vector $x \in \mathbb{Z}_q^N$ decomposes x into its bit representation.
– The powers-of-two function 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.

The PRE scheme $\Pi_{\mathsf{L\text{-}PRE}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{ReKeyGen}, \mathsf{ReEnc})$ with $(\mathsf{HReKeyGen}, \mathsf{ReKeyEval})$ is constructed 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 $q = q(\lambda)$, let $R := \mathbb{Z}[X]/(X^N + 1)$ and $R_q := \mathbb{Z}_q[X]/(X^N + 1)$, where $N = 2^{N'-1}$ such that $X^N + 1$ is the $2^{N'}$-th cyclotomic polynomial.
  - Let η, k be positive integers, and let $\ell := \lceil \log q \rceil$.
  - Sample $\boldsymbol{A} \xleftarrow{\$} R_q^{k \times k}$.

  Output $\mathsf{pp} = (\lambda, \mu, N, N', q, \eta, k, \ell, \boldsymbol{A})$.
– $\mathsf{KeyGen}(\mathsf{pp}) \to (\mathsf{pk}, \mathsf{sk})$: Parse $\mathsf{pp} = (\lambda, \mu, N, N', q, \ell, \eta, k, \boldsymbol{A})$. Sample $(\boldsymbol{s}, \hat{\boldsymbol{s}}) \leftarrow \mathcal{U}_\eta^k \times \mathcal{U}_\eta^k$ and $(\boldsymbol{e}, \hat{\boldsymbol{e}}) \leftarrow \mathcal{U}_\eta^k \times \mathcal{U}_\eta^k$. Compute $\boldsymbol{t} \leftarrow \boldsymbol{A}\boldsymbol{s} + \boldsymbol{e}$ and $\hat{\boldsymbol{t}} \leftarrow \boldsymbol{A}\hat{\boldsymbol{s}} + \hat{\boldsymbol{e}}$. Output $\mathsf{pk} = (\boldsymbol{t}, \hat{\boldsymbol{t}}) \in R_q^k \times R_q^k$ and $\mathsf{sk} = (\boldsymbol{s}, \hat{\boldsymbol{s}}) \in R^k \times R^k$.

- $\mathsf{Enc}(\mathsf{pk}, \mathsf{m}) \to \mathsf{ct}$: Parse $\mathsf{pk} = (\boldsymbol{t}, \hat{\boldsymbol{t}})$. Sample $(\boldsymbol{r}, \boldsymbol{e}_1, e_2) \leftarrow \mathcal{U}_\eta^k \times \mathcal{U}_\eta^k \times \mathcal{U}_\eta$, and compute $\boldsymbol{u} \leftarrow \boldsymbol{A}^\top \boldsymbol{r} + \boldsymbol{e}_1^\top$ and $v \leftarrow \boldsymbol{t}^\top \boldsymbol{r} + e_2 + \lceil \frac{q}{2} \rfloor \cdot \mathsf{m}$. Output $\mathsf{ct} = (\boldsymbol{u}, v) \in R_q^k \times R_q$.
- $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) \to \mathsf{m}$:
  1. Parse $\mathsf{sk} = (\boldsymbol{s}, \hat{\boldsymbol{s}})$ and $\mathsf{ct} = (\boldsymbol{u}, v)$.
  2. Set $\tilde{\boldsymbol{s}} \leftarrow \boldsymbol{s}$ if ct is generated by the algorithm Enc. Set $\tilde{\boldsymbol{s}} \leftarrow \hat{\boldsymbol{s}}$ if ct is generated by the algorithm ReEnc.
  3. Output $\mathsf{m} \leftarrow \lceil 2(v - \tilde{\boldsymbol{s}}^\top \boldsymbol{u})/q \rfloor \bmod 2$.
- $\mathsf{ReKeyGen}(\mathsf{sk}_A, \mathsf{pk}_B) \to \mathsf{rk}_{A \to B}$:
  1. Parse $\mathsf{sk}_A = (\boldsymbol{s}_A, \hat{\boldsymbol{s}}_A)$ and $\mathsf{pk}_B = (\boldsymbol{t}_B, \hat{\boldsymbol{t}}_B)$.
  2. Choose $\boldsymbol{R}_{A \to B,1}, \boldsymbol{R}_{A \to B,2} \leftarrow \mathcal{U}_\eta^{k \times k\ell}$ and $\boldsymbol{r}_{A \to B,3} \leftarrow \mathcal{U}_\eta^{k\ell}$.
  3. Compute $\boldsymbol{U}_{A \to B} \leftarrow \boldsymbol{A}^\top \boldsymbol{R}_{A \to B,1} + \boldsymbol{R}_{A \to B,2} \in R_q^{k \times k\ell}$.
  4. Compute $\boldsymbol{v}_{A \to B} \leftarrow \hat{\boldsymbol{t}}_B^\top \boldsymbol{R}_{A \to B,1} + \boldsymbol{r}_{A \to B,3}^\top - \mathsf{Powersof2}(\boldsymbol{s}_A^\top) \in R_q^{k\ell}$.
  5. 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 $\boldsymbol{u}_B \leftarrow \boldsymbol{U}_{A \to B} \cdot \mathsf{BitDecomp}(\boldsymbol{u}_A)$.
  3. Compute $v_B \leftarrow v_A + \boldsymbol{v}_{A \to B}^\top \cdot \mathsf{BitDecomp}(\boldsymbol{u}_A)$.
  4. 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. Choose $\boldsymbol{R}_{A \to B,1}, \boldsymbol{R}_{A \to B,2} \leftarrow \mathcal{U}_\eta^{k \times k\ell}$, and $\boldsymbol{r}_{(A,i) \to (B,j)} \leftarrow \mathcal{U}_\eta^{k\ell}$ for every $i \in [u]$ and $j \in [u]$.
  3. Compute $\boldsymbol{U}_{A \to B} \leftarrow \boldsymbol{A}^\top \boldsymbol{R}_{A \to B,1} + \boldsymbol{R}_{A \to B,2}$.
  4. Compute $\boldsymbol{v}_{(A,i) \to (B,j)} \leftarrow \hat{\boldsymbol{t}}_{B,j}^\top \boldsymbol{R}_{A \to B,1} + \boldsymbol{r}_{(A,i) \to (B,j),3} - \mathsf{Powersof2}(\boldsymbol{s}_{A,i}^\top)$ for every $i \in [u]$ and every $j \in [u]$.
  5. Output $(\mathsf{rk}_{(A,i) \to (B,j)})_{i \in [u], j \in [u]}$, where $\mathsf{rk}_{(A,i) \to (B,j)} = (\boldsymbol{U}_{A \to B}, \boldsymbol{v}_{(A,i) \to (B,j)})$.
- $\mathsf{ReKeyEval}((\mathsf{rk}_{(A,\alpha_i) \to (B,\beta_i)})_{i \in [v]}) \to \mathsf{rk}_{A \to B}$: Parse $\mathsf{rk}_{(A,\alpha_i) \to (B,\beta_i)} = (\boldsymbol{U}_{A \to B}, \boldsymbol{v}_{(A,\alpha_i) \to (B,\beta_i)})$ for every $i \in [v]$. Compute $\boldsymbol{v}_{A \to B} \leftarrow \sum_{i \in [v]} \boldsymbol{v}_{(A,\alpha_i) \to (B,\beta_i)} \in R_q^{k\ell}$. Output $\mathsf{rk}_{A \to B} = (\boldsymbol{U}_{A \to B}, \boldsymbol{v}_{A \to B})$.

The scheme $\Pi_{\mathsf{L\text{-}PRE}}$ is correct and re-encryption key homomorphic with overwhelming probability. The formal propositions and these proofs appear in Appendix B. We give informal propositions, as follows:

Proposition 2 (Correctness of $\Pi_{\mathsf{L\text{-}PRE}}$ (informal)). *The proposed PRE scheme $\Pi_{\mathsf{L\text{-}PRE}}$ is* correct *with overwhelming probability, under a suitable parameter setting by the algorithm* Setup*.*

Proposition 3 (Re-encryption key homomorphism of $\Pi_{\mathsf{L\text{-}PRE}}$ (informal)). *The proposed PRE scheme $\Pi_{\mathsf{L\text{-}PRE}}$ 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 $\Pi_{\mathsf{L\text{-}PRE}}$:

Theorem 2 **(RKH-CPA security of $\Pi_{\mathsf{L\text{-}PRE}}$).** *For a security parameter λ, suppose that $n = \mathsf{poly}(\lambda)$ is a positive integer and u is a positive integer which may be independent of λ. If the $\mathsf{MLWE}_{k+1,k,\eta}$ assumption holds, the proposed PRE scheme $\Pi_{\mathsf{L\text{-}PRE}}$ is u-RKH-CPA-secure.*

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

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

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 u-RKH-CPA security of the PRE scheme $\Pi_{\mathsf{L\text{-}PRE}}$. Let $n \cdot u$ $(= n_h + n_c)$ be the total number of users whose key-pairs are generated in the u-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 $\texttt{O.HReKeyGen}$ oracle. The challenge ciphertext under the public key of the user $(i^*, j^*) \in [n] \times [u]$ is denoted by $\mathsf{ct}^* = (\boldsymbol{u}^*, v^*)$. For simplicity, without loss of generality, the index $(i_\kappa, j_\kappa) \in \mathcal{L}_{\texttt{Honest}}$ of the κ-th honest user (where $i_\kappa \in [n], j_\kappa \in [u]$, and $\mathcal{L}_{\texttt{Honest}} = ([i] \times [j])_{i\in[n],j\in[u]} \backslash \mathcal{L}_{\texttt{Corrupt}}$) is denoted by $\kappa = (i_\kappa, j_\kappa)$.

In order to prove Theorem 2, we consider security games Game_0, $(\mathsf{Game}_1^{(\kappa)})_{\kappa\in[n_h]}$, $(\mathsf{Game}_2^{(\kappa)})_{\kappa\in[n_h]}$, $(\mathsf{Game}_3^{(\kappa)})_{\kappa\in[n_h]}$, 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. The games Game_0, $(\mathsf{Game}_1^{(\kappa)})_{\kappa\in[n_h]}$, $(\mathsf{Game}_2^{(\kappa)})_{\kappa\in[n_h]}$, $(\mathsf{Game}_3^{(\kappa)})_{\kappa\in[n_h]}$, Game_4 are defined as follows:

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

Let $\mathsf{Game}_1^{(0)}$ be the same game as Game_0. For each $\kappa \in [n_h]$, 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 $\hat{\boldsymbol{t}}_\kappa = \boldsymbol{A}\hat{\boldsymbol{s}}_\kappa + \hat{\boldsymbol{e}}_\kappa \in R_q^k$ is replaced by a uniformly random $\hat{\boldsymbol{t}}_\kappa \in R_q^k$, 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. Notice that $\mathcal{B}_1^{(\kappa)}$ is given one sample. Hence, we have $\left|\Pr[W_1^{(\kappa-1)}] - \Pr[W_1^{(\kappa)}]\right| \leq \mathsf{Adv}^{\text{mlwe}}_{k,k,\eta}(\mathcal{B}_1^{(\kappa)})$ for $\kappa \in [n_h]$.

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 $\kappa \in \mathcal{L}_{\texttt{Honest}}$.

$\underline{\mathsf{Game}_2^{(\kappa)}}$: 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 $\kappa=(A,\hat{i})$ and $(B,\hat{j})\in\mathcal{L}_{\texttt{Honest}}$ for some $\hat{i}\in[u]$ and every $\hat{j}\in[u]$, $\texttt{O.HReKeyGen}$ generates a uniformly random $\boldsymbol{U}_{A\to B}\in R_q^{k\times k\ell}$ and a uniformly random $\boldsymbol{v}_{(A,\hat{i})\to(B,j)}\in R_q^{k\ell}$ for every $j\in[u]$, instead of $\boldsymbol{U}_{A\to B}\leftarrow\boldsymbol{A}^\top\boldsymbol{R}_{A\to B,1}+\boldsymbol{R}_{A\to B,2}$ and $\boldsymbol{v}_{(A,\hat{i})\to(B,j)}\leftarrow\hat{\boldsymbol{t}}_{B,j}^\top\boldsymbol{R}_{A\to B,1}+\boldsymbol{r}_{(A,\hat{i})\to(B,j)}-\mathsf{Powersof2}(\boldsymbol{s}_{A,\hat{i}}^\top)$.

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

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 $\kappa\in\mathcal{L}_{\texttt{Honest}}$.

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

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 $\left|\Pr[W_3^{(\kappa-1)}]-\Pr[W_3^{(\kappa)}]\right|\leq\mathsf{Adv}_{k,k,\eta}^{\mathrm{mlwe}}(\mathcal{B}_3^{(\kappa)})$.

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

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 $\mathsf{MLWE}_{k+1,k,\eta}$ problem. Hence, we have $\left|\Pr[W_3^{(n_h)}]-\Pr[W_4]\right|\leq n_h\cdot\mathsf{Adv}_{k+1,k,\eta}^{\mathrm{mlwe}}(\mathcal{B}_4^{(i^*)})$. Furthermore, $\Pr[W_4]=1/2$ holds since the challenge ciphertext ct^* is independent of $b\in\{0,1\}$ in Game_4.

From the discussion above, we obtain

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

where $\mathcal{B}$ is a PPT algorithm against the $\mathsf{MLWE}_{k+1,k,\eta}$ assumption, such that $\mathsf{Adv}^{\text{mlwe}}_{k+1,k,\eta}(\mathcal{B}_i^{(\kappa)}) < \mathsf{Adv}^{\text{mlwe}}_{k+1,k,\eta}(\mathcal{B})$ for all $i \in [4]$ and $\kappa \in \mathcal{L}_{\texttt{Honest}}$. Therefore, this completes the proof. $\square$

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 lattice-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 lattice-based PRE and an existing lattice-based OTS scheme.

Although we just discussed single-hop PRE schemes, we can consider a multi-hop variant of our PRE schemes, $\Pi_{\mathsf{C\text{-}PRE}}, \Pi_{\mathsf{L\text{-}PRE}}$ as an extension of these schemes.

Acknowledgements. This work was supported by JST K Program Grant Number JPMJKP24U2, Japan. The authors would like to thank the anonymous referees for their helpful comments.

References

1. Almeida, J.B., Olmos, S.A., Barbosa, M., Barthe, G., Dupressoir, F., Grégoire, B., Laporte, V., Léchenet, J., Low, C., Oliveira, T., Pacheco, H., Quaresma, M., Schwabe, P., Strub, P.: Formally verifying kyber - episode V: machine-checked IND-CCA security and correctness of ML-KEM in easycrypt. In: CRYPTO (2). LNCS, vol. 14921, pp. 384–421. Springer (2024)
2. Ateniese, G., Benson, K., Hohenberger, S.: Key-private proxy re-encryption. In: CT-RSA. LNCS, vol. 5473, pp. 279–294. Springer (2009)
3. Ateniese, G., Fu, K., Green, M., Hohenberger, S.: Improved proxy re-encryption schemes with applications to secure distributed storage. In: NDSS. The Internet Society (2005)
4. Ateniese, G., Fu, K., Green, M., Hohenberger, S.: Improved proxy re-encryption schemes with applications to secure distributed storage. ACM Trans. Inf. Syst. Secur. **9**(1), 1–30 (2006)
5. Blaze, M., Bleumer, G., Strauss, M.: Divertible protocols and atomic proxy cryptography. In: EUROCRYPT. LNCS, vol. 1403, pp. 127–144. Springer (1998)
6. Bos, J.W., Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schanck, J.M., Schwabe, P., Seiler, G., Stehlé, D.: CRYSTALS - kyber: A cca-secure module-lattice-based KEM. In: EuroS&P. pp. 353–367. IEEE (2018)
7. Canard, S., Devigne, J., Laguillaumie, F.: Improving the security of an efficient unidirectional proxy re-encryption scheme. J. Internet Serv. Inf. Secur. **1**(2/3), 140–160 (2011)

8. Canetti, R., Hohenberger, S.: Chosen-ciphertext secure proxy re-encryption. In: CCS. pp. 185–194. ACM (2007)
9. Chandran, N., Chase, M., Liu, F., Nishimaki, R., Xagawa, K.: Re-encryption, functional re-encryption, and multi-hop re-encryption: A framework for achieving obfuscation-based security and instantiations from lattices. In: Public Key Cryptography. LNCS, vol. 8383, pp. 95–112. Springer (2014)
10. Chandran, N., Chase, M., Vaikuntanathan, V.: Functional re-encryption and collusion-resistant obfuscation. In: TCC. LNCS, vol. 7194, pp. 404–421. Springer (2012)
11. Chow, S.S.M., Weng, J., Yang, Y., Deng, R.H.: Efficient unidirectional proxy re-encryption. In: AFRICACRYPT. LNCS, vol. 6055, pp. 316–332. Springer (2010)
12. Cohen, A.: What about bob? the inadequacy of CPA security for proxy reencryption. In: Public Key Cryptography (2). LNCS, vol. 11443, pp. 287–316. Springer (2019)
13. Cramer, R., Hanaoka, G., Hofheinz, D., Imai, H., Kiltz, E., Pass, R., Shelat, A., Vaikuntanathan, V.: Bounded cca2-secure encryption. In: ASIACRYPT. LNCS, vol. 4833, pp. 502–518. Springer (2007)
14. Du, D.Z., Hwang, F.K.: Combinatorial Group Testing and Its Applications (2nd Edition), Series on Applied Mathematics, vol. 12. World Scientific (2000)
15. Duman, J., Hövelmanns, K., Kiltz, E., Lyubashevsky, V., Seiler, G.: Faster lattice-based kems via a generic fujisaki-okamoto transform using prefix hashing. In: CCS. pp. 2722–2737. ACM (2021)
16. Fan, X., Liu, F.: Proxy re-encryption and re-signatures from lattices. In: ACNS. LNCS, vol. 11464, pp. 363–382. Springer (2019)
17. Fuchsbauer, G., Kamath, C., Klein, K., Pietrzak, K.: Adaptively secure proxy re-encryption. In: Public Key Cryptography (2). LNCS, vol. 11443, pp. 317–346. Springer (2019)
18. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: STOC. pp. 169–178. ACM (2009)
19. Grubbs, P., Maram, V., Paterson, K.G.: Anonymous, robust post-quantum public key encryption. In: EUROCRYPT (3). LNCS, vol. 13277, pp. 402–432. Springer (2022)
20. Huguenin-Dumittan, L., Vaudenay, S.: On ind-qcca security in the ROM and its applications - CPA security is sufficient for TLS 1.3. In: EUROCRYPT (3). LNCS, vol. 13277, pp. 613–642. Springer (2022)
21. Ivan, A., Dodis, Y.: Proxy cryptography revisited. In: NDSS. The Internet Society (2003)
22. Liang, X., Weng, J., Yang, A., Yao, L., Jiang, Z., Wu, Z.: Attribute-based conditional proxy re-encryption in the standard model under LWE. In: ESORICS (2). LNCS, vol. 12973, pp. 147–168. Springer (2021)
23. Libert, B., Vergnaud, D.: Unidirectional chosen-ciphertext secure proxy re-encryption. In: Public Key Cryptography. LNCS, vol. 4939, pp. 360–379. Springer (2008)
24. Lyubashevsky, V.: Basic lattice cryptography: The concepts behind kyber (ML-KEM) and dilithium (ML-DSA). IACR Cryptol. ePrint Arch. p. 1287 (2024)
25. Lyubashevsky, V., Micciancio, D.: Asymptotically efficient lattice-based digital signatures. In: TCC. LNCS, vol. 4948, pp. 37–54. Springer (2008)
26. Lyubashevsky, V., Micciancio, D.: Asymptotically efficient lattice-based digital signatures. J. Cryptol. **31**(3), 774–797 (2018)
27. Maram, V., Xagawa, K.: Post-quantum anonymity of kyber. In: Public Key Cryptography (1). LNCS, vol. 13940, pp. 3–35. Springer (2023)

28. Miao, P., Patranabis, S., Watson, G.J.: Unidirectional updatable encryption and proxy re-encryption from DDH. In: Public Key Cryptography (2). LNCS, vol. 13941, pp. 368–398. Springer (2023)
29. Polyakov, Y., Rohloff, K., Sahu, G., Vaikuntanathan, V.: Fast proxy re-encryption for publish/subscribe systems. ACM Trans. Priv. Secur. **20**(4), 14:1–14:31 (2017)
30. Tessaro, S., Wilson, D.A.: Bounded-collusion identity-based encryption from semantically-secure public-key encryption: Generic constructions with short ciphertexts. In: Public Key Cryptography. LNCS, vol. 8383, pp. 257–274. Springer (2014)
31. Xagawa, K.: Anonymity of NIST PQC round 3 kems. In: EUROCRYPT (3). LNCS, vol. 13277, pp. 551–581. Springer (2022)
32. Yoneyama, K.: Compact authenticated key exchange from bounded cca-secure KEM. IEICE Trans. Fundam. Electron. Commun. Comput. Sci. **98-A**(1), 132–143 (2015)
33. Zhou, B., Jiang, H., Zhao, Y.: Cpa-secure kems are also sufficient for post-quantum TLS 1.3. In: ASIACRYPT (3). LNCS, vol. 15486, pp. 433–464. Springer (2024)
34. Zhou, Y., Liu, S., Han, S.: Multi-hop fine-grained proxy re-encryption. In: Public Key Cryptography (4). LNCS, vol. 14604, pp. 161–192. Springer (2024)
35. Zhou, Y., Liu, S., Han, S., Zhang, H.: Fine-grained proxy re-encryption: Definitions and constructions from LWE. In: ASIACRYPT (6). LNCS, vol. 14443, pp. 199–231. Springer (2023)

A Bounded CCA2 secure PRE from CPA secure PRE

In this section, we propose a generic construction of bounded CCA2-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 for presenting a generic construction of bounded CCA2-secure PRE.

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

```
$\underline{\mathsf{Expt}^{\mathrm{cpa}}_{\Pi_{\mathsf{PRE}},\mathcal{A},n}(\lambda)}$ :
    Generate $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$ and set $\mathsf{T}_{\mathsf{rk}} \leftarrow \emptyset$; ;
    $\forall i \in [n]$, generate $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$;
    $(\mathcal{L}_{\mathtt{Corrupt}}, \mathsf{state}_0) \leftarrow \mathcal{A}_0(\mathsf{pp}, \{\mathsf{pk}_i\}_{i\in[n]})$;
    $(i^*, \mathsf{m}_0^*, \mathsf{m}_1^*, \mathsf{state}_1) \leftarrow \mathcal{A}_1^{\mathtt{O.ReKeyGen}}(\mathsf{state}_0, \{\mathsf{sk}_i\}_{i\in\mathcal{L}_{\mathtt{Corrupt}}})$;
    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 $\mathcal{L}_{\mathtt{Corrupt}}$ is a subset of $[n]$, and $(\mathsf{state}_0, \mathsf{state}_1)$ 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 $(x_1, \ldots, x_v, z)$.

Definition 14 (AONT). *A PPT 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 a polynomial-time 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*

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

is negligible in λ*, where* O.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 CCA2-secure PRE scheme $\Pi_{\mathsf{B\text{-}PRE}}$ which is based on a generic construction [13] of bounded CCA2-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 CCA2-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 $\Pi'_{\mathsf{PRE}} = (\Pi'_{\mathsf{PRE}}.\mathsf{Setup}, \Pi'_{\mathsf{PRE}}.\mathsf{KeyGen}, \Pi'_{\mathsf{PRE}}.\mathsf{Enc}, \Pi'_{\mathsf{PRE}}.\mathsf{Dec}, \Pi'_{\mathsf{PRE}}.\mathsf{ReKeyGen}, \Pi'_{\mathsf{PRE}}.\mathsf{ReEnc})$ with the message space $\mathcal{M}' = \{0,1\}^{\bar{\mu}}$, where $\bar{\mu} = \bar{\mu}(\lambda)$ is a positive integer for a security parameter λ;
– a strongly unforgeable OTS scheme $\Pi_{\mathsf{OTS}} = (\Pi_{\mathsf{OTS}}.\mathsf{KeyGen}, \Pi_{\mathsf{OTS}}.\mathsf{Sign}, \Pi_{\mathsf{OTS}}.\mathsf{Vrfy})$;
– 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 $\Pi_{\mathsf{B\text{-}PRE}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{ReKeyGen}, \mathsf{ReEnc})$ is constructed as follows:

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

The correctness of $\Pi_{\mathsf{B\text{-}PRE}}$ follows that of the underlying primitives Π'_{PRE}, Π_{OTS}, and $(\mu, \bar{\mu}, v)$-AONT. Because this is proven in the straightforward way, we omit the proof.

A.3 Security Proof

Theorem 3 shows the bounded CCA2 security of the proposed scheme $\Pi_{\mathsf{B\text{-}PRE}}$.

Theorem 3 ((t,t)-CCA2 security of $\Pi_{\mathsf{B\text{-}PRE}}$). *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)-CCA2 game. If the PRE scheme Π'_{PRE} is CPA-secure, the OTS scheme Π_{OTS} is strongly unforgeable, and the algorithm Trans is $(\mu, \bar{\mu}, v)$-AONT, then the resulting PRE scheme $\Pi_{\mathsf{B\text{-}PRE}}$ is (t,t)-CCA2-secure.*

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

$$\mathsf{Adv}^{(t,t)\text{-cca2}}_{\Pi_{\mathsf{B\text{-}PRE}},\mathcal{A},n}(\lambda) \leq n_h^3 u^2 \cdot \mathsf{Adv}^{\text{cpa}}_{\mathcal{B}_1,\Pi'_{\mathsf{PRE}},nu}(\lambda) + \mathsf{Adv}^{\text{suf-ot}}_{\Pi_{\mathsf{OTS}},\mathcal{F}}(\lambda) + n_h^3 u^2 \cdot \mathsf{Adv}^{\text{ind}}_{\mathcal{B}_2,\mathsf{AONT}}(\lambda).$$

Proof. Let $\mathcal{A}$ be a PPT adversary against the PRE scheme $\Pi_{\mathsf{B\text{-}PRE}}$. Let $\mathsf{ct}^* = (\mathsf{vk}^*, (\mathsf{ct}'^*_i)_{i\in[v]}, z^*, \sigma^*)$ denote the challenge ciphertext under pk_{i^*}. To prove Theorem 3, we consider security games $\mathsf{Game}_0, \mathsf{Game}_1$.

<u>Game_0</u>: This game is the same as the (t,t)-$\mathsf{CCA2}$ security game.

<u>Game_1</u>: 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 $\Pi_{\mathsf{OTS}}.\mathsf{Vrfy}(\mathsf{vk}_i, (\mathsf{ct}'_{i,1}\| \cdots \|\mathsf{ct}'_{i,v}\|z_i), \sigma_i) = \top$ (this event is denoted by Bad). If so, the experiment aborts; otherwise, $\mathtt{O.Dec}$ computes $\mathsf{m}' \leftarrow \mathsf{Dec}(\mathsf{sk}_i, \mathsf{ct}_i)$ and returns $\mathsf{m}' \in \mathcal{M} \cup \{\perp\}$.

Game_0 and Game_1 are identical unless Bad occurs. In order to estimate the upper bound of the probability that Bad occurs, we construct a PPT algorithm $\mathcal{F}$ breaking the $\mathsf{strongly\ unforgeable}$ OTS scheme Π_{OTS}.

On input a verification key vk^* of Π_{OTS}, $\mathcal{F}$ gives $(\mathsf{pp}, \{\mathsf{pk}_i\}_{i\in[n]}, \{\mathsf{sk}_i\}_{i\in\mathcal{L}_{\mathsf{Corrupt}}})$ to $\mathcal{A}$ by generating $(\mathsf{pp}, \{(\mathsf{pk}_i, \mathsf{sk}_i)\}_{i\in[n]})$ by itself. By using the generated key-pairs, $\mathcal{F}$ simulates $\mathtt{O.ReKeyGen}$. 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 $\mathsf{strong\ unforgeability}$ game of Π_{OTS}, if it holds that $\mathsf{vk}_i = \mathsf{vk}^*$, $\mathsf{ct}_i \neq \mathsf{ct}^*$, and $\Pi_{\mathsf{OTS}}.\mathsf{Vrfy}(\mathsf{vk}_i, (\mathsf{ct}'_{i,1}\| \cdots \|\mathsf{ct}'_{i,v}\|z_i), \sigma_i)) = \top$ (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 $b \overset{\$}{\leftarrow} \{0,1\}$, 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 $\mathsf{strong\ unforgeability}$ 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 $\mathcal{F}$. It is clear that the output of $\mathcal{F}$ is a valid forgery in the $\mathsf{strong\ unforgeability}$ game if Bad occurs. Hence, the probability $\Pr[\mathsf{Bad}]$ is at most the advantage $\mathsf{Adv}^{\text{suf-ot}}_{\Pi_{\mathsf{OTS}},\mathcal{F}}(\lambda)$ of $\mathcal{F}$, and the two games are distinguishable with probability at most $\mathsf{Adv}^{\text{suf-ot}}_{\Pi_{\mathsf{OTS}},\mathcal{F}}(\lambda)$.

<u>**Simulation of Game_1.**</u> To bound the winning probability of $\mathcal{A}$ in Game_1, we consider the following experiment $\mathcal{B}$: At the beginning of the game, $\mathcal{B}$ generates $(\mathsf{pp}, \{(\mathsf{pk}_i, \mathsf{sk}_i)\}_{i\in[n]})$ in the same way as Game_1, and gives $(\mathsf{pp}, \{\mathsf{pk}_i\}_{i\in[n]})$ to $\mathcal{A}$. And then, $\mathcal{B}$ generates $(\mathsf{vk}^*, \mathsf{sigk}^*) \leftarrow \Pi_{\mathsf{OTS}}.\mathsf{KeyGen}(1^\lambda)$, chooses indices $i^* \overset{\$}{\leftarrow} [n_h]$, $j^* \overset{\$}{\leftarrow} \phi_{\boldsymbol{M}}(\mathsf{vk}^*)$, and simulates Game_1 except for the following procedure: $\mathcal{B}$ 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 $i^* \neq i'$. When $\mathcal{A}$ finally outputs $b' \in \{0,1\}$, $\mathcal{B}$ also outputs b'.

We estimate the probability that $\mathcal{B}$ outputs b' such that $b = b'$ (this event is denoted by $W_{\mathcal{B}}$). Let Abort be the event that $\mathcal{B}$ aborts in the above simulation of oracles. Notice that $\Pr[W_{\mathcal{B}} \mid \mathsf{Abort}] = 1/2$. Due to the t-disjunct property of $\boldsymbol{M}$, it holds that $\Pr[\neg\mathsf{Abort}] \geq 1/(n_h^3 u^2)$. Then, we have

$$\begin{aligned}\Pr[W_{\mathcal{B}}] &= \Pr[\mathsf{Abort}] \cdot \Pr[W_{\mathcal{B}} \mid \mathsf{Abort}] + \Pr[\neg\mathsf{Abort}] \cdot \Pr[W_{\mathcal{B}} \mid \neg\mathsf{Abort}] \\ &\geq \frac{1}{2}\left(1 - \frac{1}{n_h^3 u^2}\right) + \frac{1}{n_h^3 u^2} \cdot \Pr[W_{\mathcal{B}} \mid \neg\mathsf{Abort}] \\ &= \frac{1}{2} + \frac{1}{n_h^3 u^2}\left(\Pr[W_{\mathcal{B}} \mid \neg\mathsf{Abort}] - \frac{1}{2}\right).\end{aligned}$$

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 $\varepsilon_{\mathcal{A}}/(n_h^3 u^2)$. Here, let $\phi_{\boldsymbol{M}}(\mathsf{vk}^*) := \{\tau_1^*, \ldots, \tau_v^*\}$ (where $\tau_1^*, \ldots, \tau_v^* \in [u]$). 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 $0^{\bar{\mu}}$, 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 $\varepsilon_{\mathcal{B}} \leq |p^{(0)} - p^{(1)}| + |p^{(1)} - 1/2|$.

Reduction to CPA security. To bound $|p^{(0)} - p^{(1)}|$, we construct a PPT algorithm $\mathcal{B}_1$ against the CPA security of Π'_{PRE}, as follows: On input the public parameter $(\mathsf{pp}', \{\mathsf{pk}'_{i,j}\}_{(i,j)\in[n]\times[u]})$ in the CPA game, $\mathcal{B}_1$ generates pp by following the algorithm Setup and gives pp to $\mathcal{A}$. When $\mathcal{A}$ submits $\mathcal{L}_{\mathtt{Corrupt}}$, $\mathcal{B}_1$ generates $(\mathsf{vk}^*, \mathsf{sigk}^*) \leftarrow \Pi_{\mathsf{OTS}}.\mathsf{KeyGen}(1^\lambda)$, chooses $i^* \overset{\$}{\leftarrow} [n], j^* \overset{\$}{\leftarrow} \phi_{\boldsymbol{M}}(\mathsf{vk}^*)$, and obtains $\{\mathsf{sk}'_{i,j}\}_{(i,j)\in[n]\times[u]\setminus\{(i^*,j^*)\}}$ by issuing $\mathcal{L}'_{\mathtt{Corrupt}} = \{(i,j)\}\setminus\{(i^*,j^*)\}$ in the CPA game. Here, for simplicity, $(i,j) \in [n] \times [u]$ represents a user-index in the CPA game. Then $\mathcal{B}_1$ returns $\{\mathsf{sk}_i\}_{i\in\mathcal{L}_{\mathtt{Corrupt}}}$, where 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 $(A = i^* \wedge B \in \mathcal{L}_{\mathtt{Corrupt}})$ 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 a re-encryption query $((i^*,j^*),(B,j))$ in the CPA game, for every $j \in [u]$. For every $i \in [u]\setminus\{j^*\}$ and every $j \in [u]$, compute $\mathsf{rk}_{(i^*,i)\to(B,j)} \leftarrow \Pi'_{\mathsf{PRE}}.\mathsf{ReKeyGen}(\mathsf{sk}'_{i^*,i}, \mathsf{pk}'_{B,j})$.
  - (Case $A \neq i^*$): Compute $\mathsf{rk}_{(A,i)\to(B,j)} \leftarrow \Pi'_{\mathsf{PRE}}.\mathsf{ReKeyGen}(\mathsf{sk}'_{A,i}, \mathsf{pk}'_{B,j})$ for every $i \in [u]$ and every $j \in [u]$.

  $\mathcal{B}_1$ 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 $\mathsf{ct}_A = (\mathsf{vk}_A, (\mathsf{ct}'_{A,i})_{i\in[v]}, z, \sigma_A)$, $\mathcal{B}_1$ 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 $\Pi_{\mathsf{OTS}}.\mathsf{Vrfy}(\mathsf{vk}_A, (\mathsf{ct}'_{A,1}\|\cdots\|\mathsf{ct}'_{A,v}\|z), \sigma_A) = \top$.
  4. Compute $\mathsf{m}' \leftarrow \mathsf{Dec}(\mathsf{sk}_A, \mathsf{ct}_A)$ and return $\mathsf{m}' \in \mathcal{M} \cup \{\perp\}$.

– $\mathtt{O.ReEnc}(A, B, \mathsf{ct}_A)$: $\mathcal{B}_1$ 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 $B \in \mathcal{L}_{\mathtt{Corrupt}}$ 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. If $\mathsf{T}_{\mathsf{rk}}[A, B] = \emptyset$, compute $\mathsf{rk}_{A\to B} \leftarrow \mathsf{ReKeyGen}(\mathsf{sk}_A, \mathsf{pk}_B)$. If $\mathsf{T}_{\mathsf{rk}}[A, B] = \widehat{\mathsf{rk}}_{A\to B}$ $(\neq \emptyset)$, set $\mathsf{rk}_{A\to B} \leftarrow \widehat{\mathsf{rk}}_{A\to B}$.
4. Return $\mathsf{ct}_B \leftarrow \mathsf{ReEnc}(\mathsf{rk}_{A\to B}, \mathsf{ct}_A)$.

– $\texttt{O.Challenge}_b(i', \mathsf{m}_0^*, \mathsf{m}_1^*)$:
  1. Abort and output a random bit if $i^* \neq i'$.
  2. Compute $\{\tau_1^*, \ldots, \tau_v^*\} \leftarrow \phi_{\boldsymbol{M}}(\mathsf{vk}^*)$, $(x_1^*, \ldots, x_v^*, z^*) \leftarrow \mathsf{Trans}(\mathsf{m}_b^*)$.
  3. Obtain $\mathsf{ct}'^*_{j^*}$ by submitting a challenge query $(x^*_{j^*}, 0^{\bar{\mu}})$ to the CPA game.
  4. For every $j \in [v]\backslash\{j^*\}$, compute $\mathsf{ct}'^*_j \leftarrow \Pi'_{\mathsf{PRE}}.\mathsf{Enc}(\mathsf{pk}'_{\tau_j^*}, x_j^*)$.
  5. Compute $\sigma^* \leftarrow \Pi_{\mathsf{OTS}}.\mathsf{Sign}(\mathsf{sigk}^*, (\mathsf{ct}'^*_1 \| \cdots \| \mathsf{ct}'^*_v \| z^*))$.
  6. 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 $\texttt{O.Dec}$ and $\texttt{O.ReEnc}$. 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 $\left|p^{(0)} - p^{(1)}\right| \leq \mathsf{Adv}^{\mathrm{cpa}}_{\Pi'_{\mathsf{PRE}},\mathcal{B}_1,nu}(\lambda)$.

Reduction to AONT's Indistinguishability. To bound $|p^{(1)} - 1/2|$, we construct a PPT algorithm $\mathcal{B}_2$ against $(\mu, \bar{\mu}, v)$-AONT Trans. Suppose that $\mathcal{B}_2$ is given the oracle $\texttt{O.LR}$ in the $\mathsf{indistinguishability}$ game of AONT: $\mathcal{B}_2$ generates all n key-pairs $\{(\mathsf{pk}_i, \mathsf{sk}_i)\}_{i\in[n]}$ by itself and simulates the environment of $\mathcal{A}$ except for the following procedure of the oracle $\texttt{O.Challenge}_b(i', \mathsf{m}_0^*, \mathsf{m}_1^*)$:

1. Abort and output a random bit if $i^* = i'$.
2. Obtain $((x_i^*)_{i\in[v]\backslash\{j^*\}}, z^*)$ by issuing $(j^*, \mathsf{m}_0^*, \mathsf{m}_1^*)$ to $\texttt{O.LR}$.
3. Compute $\mathsf{ct}'^*_{j^*} \leftarrow \Pi'_{\mathsf{PRE}}.\mathsf{Enc}(\mathsf{pk}_{\tau^*_{j^*}}, 0^{\bar{\mu}})$ and $\mathsf{ct}'^*_i \leftarrow \Pi'_{\mathsf{PRE}}.\mathsf{Enc}(\mathsf{pk}_{\tau_i^*}, x_i^*)$ for every $i \in [v]\backslash\{j^*\}$, where $\{\tau_1^*, \ldots, \tau_v^*\} = \phi_{\boldsymbol{M}}(\mathsf{vk}^*)$.
4. Compute $\sigma^* \leftarrow \Pi_{\mathsf{OTS}}.\mathsf{Sign}(\mathsf{sigk}^*, (\mathsf{ct}'^*_1 \parallel \cdots \parallel \mathsf{ct}'^*_v \parallel z^*))$.
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 $\texttt{O.KeyGen}, \texttt{O.ReKeyGen}, \texttt{O.Dec}, \texttt{O.ReEnc}$ completely since it has the key-pairs of all users. $\texttt{O.Challenge}$ 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 $\mathsf{Adv}^{\mathrm{cpa}}_{\Pi'_{\mathsf{PRE}},\mathcal{B}_1}(\lambda) + \mathsf{Adv}^{\mathrm{ind}}_{\mathsf{Trans},\mathcal{B}_2}(\lambda) \geq \varepsilon_{\mathcal{A}}/(n_h^3 u^2)$.

From the discussion above, we obtain

$$\mathsf{Adv}^{(t,\,t)\text{-cca2}}_{\Pi_{\mathsf{PRE}},\mathcal{A},n}(\lambda) \leq n_h u^2 \cdot \mathsf{Adv}^{\mathrm{cpa}}_{\mathcal{B}_1,\Pi'_{\mathsf{PRE}},nu}(\lambda) + n_h^3 u^2 \cdot \mathsf{Adv}^{\mathrm{ind}}_{\mathsf{AONT},\mathcal{B}_2}(\lambda) + \mathsf{Adv}^{\mathrm{suf}}_{\Pi_{\mathsf{OTS}},\mathcal{F}}(\lambda).$$

and complete the proof. □

B Omitted Proofs for Our Lattice-based PRE Scheme

In this section, we give proofs for the $\mathsf{correctness}$ and $\mathsf{re\text{-}encryption\ key\ homomorphism}$ of our scheme $\Pi_{\mathsf{L\text{-}PRE}}$.

Proposition 4 (Correctness of $\Pi_{\mathsf{L\text{-}PRE}}$). *Let* $\mathsf{pp} = (\lambda, \mu, N, N', q, \eta, k, \ell, \boldsymbol{A})$ *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 the key-pairs of the users A and B, respectively, where for $i \in \{A, B\}$,* $\boldsymbol{t}_i = \boldsymbol{A}\boldsymbol{s}_i + \boldsymbol{e}_i$ *and* $\hat{\boldsymbol{t}}_i = \boldsymbol{A}\hat{\boldsymbol{s}}_i + \hat{\boldsymbol{e}}_i$*.*
– *Let* $\mathsf{ct}_A := (\boldsymbol{u}_A, v_A) \leftarrow \mathsf{Enc}(\mathsf{pk}_A, \mathsf{m})$ *for an arbitrary message* $\mathsf{m} \in \mathcal{M}$*.*

Denote $w := \boldsymbol{e}_A^\top \boldsymbol{r} + e_2 - \boldsymbol{s}_A^\top \boldsymbol{e}_1$ *and* $\hat{w} := w + \hat{\boldsymbol{e}}_B^\top \cdot \mathsf{BitDecomp}(\boldsymbol{u}_A) + \boldsymbol{r}_{A\to B,3}^\top \cdot \mathsf{BitDecomp}(\boldsymbol{u}_A) - \hat{\boldsymbol{s}}_B^\top \cdot \boldsymbol{R}_{A\to B,2} \cdot \mathsf{BitDecomp}(\boldsymbol{u}_A)$*, where* $\boldsymbol{r}, \boldsymbol{e}_1, e_2, \boldsymbol{R}_{A\to B,2}, \boldsymbol{r}_{A\to B,3}$ *are random values generated by running* $\mathsf{Enc}(\mathsf{pk}_A, \mathsf{m})$ *and* $\mathsf{ReKeyGen}(\mathsf{sk}_A, \mathsf{pk}_B)$*. Then, the proposed PRE scheme* $\Pi_{\mathsf{L\text{-}PRE}}$ *is* $\mathsf{correct}$ *with probability* $1 - \Pr[\|\hat{w}\|_\infty \geq q/4]$*.*

Proof. We consider an arbitrary message $\mathsf{m} \in \mathcal{M}$ throughout the proof of Proposition 4. First, we show the $\mathsf{encryption\text{-}correctness}$ of $\Pi_{\mathsf{L\text{-}PRE}}$. Then, the public value $\boldsymbol{t}_A$ is represented as $\boldsymbol{t}_A = \boldsymbol{A}\boldsymbol{s}_A + \boldsymbol{e}_A$. Additionally, the value $\boldsymbol{u}_A$ of the ciphertext $\mathsf{ct}_A = (\boldsymbol{u}_A, v_A)$ under pk_A is $\boldsymbol{u}_A = \boldsymbol{A}^\top \boldsymbol{r} + \boldsymbol{e}_1$, and the value v_A is $v_A = \boldsymbol{t}_A^\top \boldsymbol{r} + e_2 + \left\lceil \frac{q}{2} \right\rfloor \cdot \mathsf{m} = (\boldsymbol{A}\boldsymbol{s}_A + \boldsymbol{e}_A)^\top \boldsymbol{r} + e_2 + \left\lceil \frac{q}{2} \right\rfloor \cdot \mathsf{m}$. Then, we have

$$\begin{aligned} v_A - \boldsymbol{s}_A^\top \boldsymbol{u}_A &= (\boldsymbol{A}\boldsymbol{s}_A + \boldsymbol{e}_A)^\top \boldsymbol{r} + e_2 + \left\lceil \frac{q}{2} \right\rfloor \cdot \mathsf{m} - \boldsymbol{s}_A^\top (\boldsymbol{A}^\top \boldsymbol{r} + \boldsymbol{e}_1) \\ &= \left\lceil \frac{q}{2} \right\rfloor \cdot \mathsf{m} + \boldsymbol{e}_A^\top \boldsymbol{r} + e_2 - \boldsymbol{s}_A^\top \boldsymbol{e}_1. \end{aligned}$$

By setting $w = \boldsymbol{e}_A^\top \boldsymbol{r} + e_2 - \boldsymbol{s}_A^\top \boldsymbol{e}_1$, m is correctly recovered by the algorithm Dec, due to the assumption $||w||_\infty < q/4$. Hence, the proof of the $\mathsf{encryption\ correctness}$ is completed.

Next, we show the $\mathsf{re\text{-}encryption\ correctness}$ of $\Pi_{\mathsf{L\text{-}PRE}}$. 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 $\mathsf{ReKeyGen}(\mathsf{sk}_A, \mathsf{pk}_B)$ is

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

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:

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

Then, we have

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

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

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

Due to the assumption $||\hat{w}||_\infty < q/4$, m is recovered correctly. Hence, the proof of the re-encryption correctness is completed.

Therefore, we complete the proof of the correctness of $\Pi_{\mathsf{L\text{-}PRE}}$. $\square$

Proposition 5 (Re-encryption key homomorphism of $\Pi_{\mathsf{L\text{-}PRE}}$). *Suppose that u, v be positive integers such that $v \le u$. Let $\mathsf{pp} = (\lambda, \mu, N, N', q, \eta, k, \ell, \boldsymbol{A}) \leftarrow \mathsf{Setup}(1^\lambda)$, and let $\{\alpha_1, \dots, \alpha_v\} \in [u]^v$ and $\{\beta_1, \dots, \beta_v\} \in [u]^v$ be two sets of distinct user-indices. Then, users' key-pairs and a ciphertext are defined as follows:*

- *For every $i \in [u]$, 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 $\boldsymbol{t}_{A,i} = \boldsymbol{A}\boldsymbol{s}_{A,i} + \boldsymbol{e}_{A,i}$ and $\hat{\boldsymbol{t}}_{B,i} = \boldsymbol{A}\hat{\boldsymbol{s}}_{B,i} + \hat{\boldsymbol{e}}_{B,i}$;*
- *Let $\mathsf{ct}_A := (\boldsymbol{u}_A, v_A) \leftarrow \mathsf{Enc}(\boldsymbol{t}_A, \mathsf{m})$ for an arbitrary message $\mathsf{m} \in \mathcal{M}$, where $\boldsymbol{t}_A = \sum_{i\in[v]} \boldsymbol{t}_{A,\alpha_i}$, $\boldsymbol{u}_A = \boldsymbol{A}^\top \boldsymbol{r} + \boldsymbol{e}_1^\top$ and $v_A = \boldsymbol{t}_A^\top \boldsymbol{r} + e_2 + \lceil \frac{q}{2} \rfloor \cdot \mathsf{m}$.*

Denote $w_A := \boldsymbol{e}_A^\top \boldsymbol{r} + e_2 - \boldsymbol{s}_A^\top \boldsymbol{e}_1$ and

$$\begin{aligned}
w_B := w_A &+ \left(\sum_{i\in[v]} \boldsymbol{r}_{(A,\alpha_i)\to(B,\beta_i),3}^\top \right) \mathsf{BitDecomp}(\boldsymbol{u}_A) \\
&+ \left(\hat{\boldsymbol{e}}_B^\top \boldsymbol{R}_{A\to B,1} - \hat{\boldsymbol{s}}_B^\top \boldsymbol{R}_{A\to B,2}\right) \mathsf{BitDecomp}(\boldsymbol{u}_A),
\end{aligned}$$

where $\hat{\boldsymbol{s}}_B = \sum_{i\in[v]} \hat{\boldsymbol{s}}_{B,\beta_i}$, $\boldsymbol{R}_{A\to B,1}, \boldsymbol{R}_{A\to B,2}$, and $\boldsymbol{r}_{(A,\alpha_i)\to(B,\beta_i),3}$ for $i \in [v]$ are random values generated by running $\mathsf{HReKeyGen}((\mathsf{sk}_{A,i})_{i\in[u]}, (\mathsf{pk}_{B,i})_{j\in[u]})$.

Then, the proposed PRE scheme $\Pi_{\mathsf{L\text{-}PRE}}$ is re-encryption key homomorphic *with probability $1 - \Pr[\|w_B\|_\infty \ge q/4]$.*

Proof. We consider an arbitrary message $\mathsf{m} \in \mathcal{M}$ throughout the proof of Proposition 5. For every $i \in [u]$, the values of $\boldsymbol{t}_{A,i}$ and $\hat{\boldsymbol{t}}_{B,i}$ are $\boldsymbol{t}_{A,i} = \boldsymbol{A}\boldsymbol{s}_{A,i} + \boldsymbol{e}_{A,i}$ and $\hat{\boldsymbol{t}}_{B,i} = \boldsymbol{A}\hat{\boldsymbol{s}}_{B,i} + \hat{\boldsymbol{e}}_{B,i}$, respectively. Then, let $\boldsymbol{t}_A := \sum_{i\in[v]} \boldsymbol{t}_{A,\alpha_i} = \boldsymbol{A}\sum_{i\in[v]} \boldsymbol{s}_{A,\alpha_i} + \sum_{i\in[v]} \boldsymbol{e}_{A,\alpha_i} = \boldsymbol{A}\boldsymbol{s}_A + \boldsymbol{e}_A$; and $\hat{\boldsymbol{t}}_B := \sum_{i\in[v]} \hat{\boldsymbol{t}}_{B,\beta_i} = \boldsymbol{A}\sum_{i\in[v]} \hat{\boldsymbol{s}}_{B,\beta_i} + \sum_{i\in[v]} \hat{\boldsymbol{e}}_{B,\beta_i} =$

$\boldsymbol{A}\hat{\boldsymbol{s}}_B + \hat{\boldsymbol{e}}_B$, where $\boldsymbol{s}_A = \sum_{i\in[v]} \boldsymbol{s}_{A,\alpha_i}$, $\boldsymbol{e}_A = \sum_{i\in[v]} \boldsymbol{e}_{A,\alpha_i}$, $\hat{\boldsymbol{s}}_B = \sum_{i\in[v]} \hat{\boldsymbol{s}}_{B,\beta_i}$, and $\hat{\boldsymbol{e}}_B = \sum_{i\in[v]} \hat{\boldsymbol{e}}_{B,\beta_i}$.

Since $\mathsf{ct}_A = (\boldsymbol{u}_A, v_A)$ is an encryption of m under $\boldsymbol{t}_A$, we have $\boldsymbol{u}_A = \boldsymbol{A}^\top \boldsymbol{r} + \boldsymbol{e}_1$ and $v_A = \boldsymbol{t}_A^\top \boldsymbol{r} + e_2 + \lceil q/2 \rfloor \cdot \mathsf{m}$. 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 $i \in [u]$ and $j \in [u]$, the values of $(\boldsymbol{U}_{A\to B}, \boldsymbol{v}_{(A,i)\to(B,j)})$ are

$$\boldsymbol{U}_{A\to B} = \boldsymbol{A}^\top \boldsymbol{R}_{A\to B,1} + \boldsymbol{R}_{A\to B,2}; \text{ and}$$
$$\boldsymbol{v}_{(A,i)\to(B,j)} = \hat{\boldsymbol{t}}_{B,j}^\top \boldsymbol{R}_{A\to B,1} + \boldsymbol{r}_{(A,i)\to(B,j),3}^\top - \mathsf{Powersof2}(\boldsymbol{s}_{A,i}^\top).$$

Then, the value $\boldsymbol{v}_{A\to B}$ for $(\boldsymbol{U}_{A\to B}, \boldsymbol{u}_{A\to B}) \leftarrow \mathsf{ReKeyEval}((\mathsf{rk}_{(A,\alpha_i)\to(B,\beta_i)})_{i\in[v]})$ is $\boldsymbol{v}_{A\to B} := \sum_{i\in[v]} \boldsymbol{v}_{(A,\alpha_i)\to(B,\beta_i)} = \sum_{i\in[v]} \hat{\boldsymbol{t}}_{B,\beta_i}^\top \boldsymbol{R}_{A\to B,1} + \sum_{i\in[v]} \boldsymbol{r}_{(A,\alpha_i)\to(B,\beta_i),3}^\top - \sum_{i\in[v]} \mathsf{Powersof2}(\boldsymbol{s}_{A,\alpha_i}^\top)$. Since $\mathsf{ct}_B = (\boldsymbol{u}_B, v_B)$ is a re-encrypted ciphertext generated by using $(\boldsymbol{U}_{A\to B}, \boldsymbol{v}_{A\to B})$, we have

$$\boldsymbol{u}_B = (\boldsymbol{A}^\top \boldsymbol{R}_{A\to B,1} + \boldsymbol{R}_{A\to B,2}) \cdot \mathsf{BitDecomp}(\boldsymbol{u}_A); \text{ and}$$
$$\begin{aligned} v_B &= v_A + \left(\sum_{i\in[v]} \hat{\boldsymbol{t}}_{B,\beta_i}^\top \boldsymbol{R}_{A\to B,1} + \sum_{i\in[v]} \boldsymbol{r}_{(A,\alpha_i)\to(B,\beta_i),3}^\top - \sum_{i\in[v]} \mathsf{Powersof2}(\boldsymbol{s}_{A,\alpha_i}^\top) \right) \mathsf{BitDecomp}(\boldsymbol{u}_A) \\ &= v_A - \boldsymbol{s}_A^\top \boldsymbol{u}_A + \left((\boldsymbol{A}\hat{\boldsymbol{s}}_B + \hat{\boldsymbol{e}}_B)^\top \boldsymbol{R}_{A\to B,1} + \sum_{i\in[v]} \boldsymbol{r}_{(A,\alpha_i)\to(B,\beta_i),3}^\top \right) \mathsf{BitDecomp}(\boldsymbol{u}_A). \end{aligned}$$

Hence, it holds that

$$\begin{aligned} v_B - \hat{\boldsymbol{s}}_B^\top \boldsymbol{u}_B = v_A &- \boldsymbol{s}_A^\top \boldsymbol{u}_A \\ &+ \left((\boldsymbol{A}\hat{\boldsymbol{s}}_B + \hat{\boldsymbol{e}}_B)^\top \boldsymbol{R}_{A\to B,1} + \sum_{i\in[v]} \boldsymbol{r}_{(A,\alpha_i)\to(B,\beta_i),3}^\top \right) \mathsf{BitDecomp}(\boldsymbol{u}_A) \\ &- \hat{\boldsymbol{s}}_B^\top \left(\boldsymbol{A}^\top \boldsymbol{R}_{A\to B,1} \cdot \mathsf{BitDecomp}(\boldsymbol{u}_A) + \boldsymbol{R}_{A\to B,2} \cdot \mathsf{BitDecomp}(\boldsymbol{u}_A) \right) \\ &= v_A - \boldsymbol{s}_A^\top \boldsymbol{u}_A + \left(\sum_{i\in[v]} \boldsymbol{r}_{(A,\alpha_i)\to(B,\beta_i),3}^\top \right) \mathsf{BitDecomp}(\boldsymbol{u}_A) \\ &\quad + \left(\hat{\boldsymbol{e}}_B^\top \boldsymbol{R}_{A\to B,1} - \hat{\boldsymbol{s}}_B^\top \boldsymbol{R}_{A\to B,2} \right) \mathsf{BitDecomp}(\boldsymbol{u}_A). \end{aligned}$$

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

$$\begin{aligned} w_B := w_A &+ \left(\sum_{i\in[v]} \boldsymbol{r}_{(A,\alpha_i)\to(B,\beta_i),3}^\top \right) \mathsf{BitDecomp}(\boldsymbol{u}_A) \\ &+ \left(\hat{\boldsymbol{e}}_B^\top \boldsymbol{R}_{A\to B,1} - \hat{\boldsymbol{s}}_B^\top \boldsymbol{R}_{A\to B,2} \right) \mathsf{BitDecomp}(\boldsymbol{u}_A), \end{aligned}$$

where w_A is the error-term of $v_A - \boldsymbol{s}_A^\top \boldsymbol{u}_A$, i.e., $w_A = \boldsymbol{e}_A^\top \boldsymbol{r} + e_2 - \boldsymbol{s}_A^\top \boldsymbol{e}_1$.

Therefore, Dec correctly recovers m due to the assumption $\|w_B\|_\infty < q/4$, and this completes the proof. □