Bounded CCA Secure Proxy Re-encryption Based on Kyber

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

[1] Institute of Advanced Sciences, Yokohama National University, Yokohama, Japan
[2] Graduate School of Environment and Information Sciences, Yokohama National University, Yokohama, Japan
sato-shingo-zk@ynu.ac.jp,shikata-junji-rb@ynu.ac.jp

Abstract. Proxy re-encryption (PRE) allows semi-honest party (called proxy) to convert a ciphertext under a public key into a ciphertext under another public key. Due to this functionality, there are various applications such as encrypted email forwarding, key escrow, and securing distributed file systems. Meanwhile, post-quantum cryptography (PQC) is one of the most important research areas because development of quantum computers has been advanced recently. In particular, there are many researches on public key encryption (PKE) algorithms selected/submitted in the NIST (National Institute of Standards and Technology) PQC standardization. However, there is no post-quantum PRE scheme secure against adaptive chosen ciphertext attacks (denoted by CCA security) while many (post-quantum) PRE schemes have been proposed so far. In this paper, we propose a bounded CCA secure PRE scheme based on CRYSTALS-Kyber which is a selected algorithm in the NIST PQC competition. To this end, we present generic constructions of bounded CCA secure PRE. Our generic constructions start from PRE secure against chosen plaintext attacks (denoted by CPA security). In order to instantiate our generic constructions, we present a CPA secure PRE scheme based on CRYSTALS-Kyber.

1 Introduction

1.1 Background

The notion of proxy re-encryption (PRE) was introduced in [4], 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 an encryption of the same message under another public key. That is, a party Alice with a public-secret key-pairs $(\mathsf{pk}_A, \mathsf{sk}_A)$ can generate a re-encryption key $\mathsf{rk}_{A,B}$ converting a ciphertext under pk_A into a ciphertext under the public key pk_B of another party Bob and give $\mathsf{rk}_{A,B}$ to a proxy. Then, this proxy can transform ciphertexts under pk_A into ciphertexts under pk_B, without knowing secret information such as messages and secret keys. Security of PRE ensures confidentiality of messages even though the adversary has several re-encryption keys. Due to the functionality of PRE, there are various applications such as encrypted email forwarding [4], key escrow [17], securing distributed file systems [2], and more. In particular, in the case where a ciphertext under a legacy parameter may not be able to ensure a concrete security (such as 128-bit security) in the future, because of advancement of cryptanalysis against cryptosystems, it is desirable to convert such a ciphertext into a ciphertext under a new parameter so that this one can guarantee the objective security in the future. Hence, we focus on constructing such a PRE scheme.

Post-quantum cryptography. Post-quantum cryptography (PQC) is one of the most active research areas. Development of cryptosystems resistant to attacks using quantum computers has been required due to recent advancement of quantum computers. In particular, PKE algorithms selected/submitted in the NIST (national institute of standards and technology) PQC competition will be widespread. Hence, there are a lot of works related to these NIST PQC candidates (e.g., [**?**, 13, 16, 18, 22]). In particular, it is reasonable to focus on CRYSTALS-Kyber (Kyber, for short) which is a key encapsulation mechanism (KEM) scheme selected in the NIST PQC competition, and we provide Kyber-based cryptosystems with advanced functionalities.

Related Work. Blaze, Bleumer, and Strauss introduced the notion of PRE and proposed a PRE scheme based on the DDH assumption [4]. This scheme is bidirectional, multi-hop, and secure against chosen plaintext attacks (denoted by CPA security). Ateniese, Fu, Green, and Hohenberger presented the first unidirectional PRE scheme with bilinear maps, and this scheme supports only single-hop re-encryption [3]. Canetti and Hohenberger gave the formalization of security against chosenc ciphertext attacks (denoted by CCA security) for PRE and presented a bidirectional multi-hop PRE scheme with CCA security under the random oracle model [7]. Phong et al. proposed a lattice-based unidirectional PRE scheme with security against (adaptive) chosen ciphertext attacks (denoted by CCA security or CCA2 security) [19]. Polyakov et al. presented practical unidirectional PRE schemes [20]. In [8], Cohen introduced the notion of *security against honest re-encryption attacks* (denoted by HRA security) and showed that one of the PRE schemes of [20] is insecure in the HRA security model. Davidon et al. modified the HRA insecure PRE scheme so that this modified scheme satisfies both HRA security and post-compromize security [10]. Fan and Liu gave tag-based PRE schemes based on the LWE assumption and these achieve security against non-adaptive chosen ciphertext attacks (denoted by CCA1 security) [14].

1.2 Contribution

Our goal is to propose a post-quantum PRE scheme with a variant of CCA security. To this end, we propose generic constructions of bounded CCA secure PRE. One is a generic construction starting from any CPA secure PRE scheme and strongly unforgeable one-time signature scheme. The other one is constructed from any CPA secure PRE with an additional property, and its ciphertexts are more compact, compared to the first one. Moreover, we present a PRE scheme converting Kyber's ciphertexts so that we can instantiate our second construction. Details on our contribution are as follows:

- We formalize a notion of bounded CCA security of PRE so that we present generic constructions of single-hop unidirectional PRE with such security. In the security model of bounded CCA security, the adversary is allowed to issue limited numbers of queries to the decryption or reencryption oracles in its security game. Although this security notion is weaker than the existing CCA security notion [7], bounded CCA security is practical, and we can construct bounded CCA secure PRE by using CPA secure PRE. In particular, there is no post-quantum PRE scheme with CCA security introduced in [7]. Hence, it is important to consider our formalized security notion and give a PRE scheme with such security.
- We propose two generic construction of single-hop unidirectional PRE. the first one is constructed from any single-hop unidirectional PRE scheme with CPA security and any strongly unforgeable one-time signature scheme. This scheme is based on the CHK transformation [5] and the bounded-collusion identity-based encryption scheme [11] constructed from any CPA secure PKE. However, it is not straightforward to construct such a scheme so that this one can ensure re-encryption.
- The building blocks of our second scheme are the same as those of the first one except that the underlying PRE is required to be $\mathsf{key\ homomorphism}$. This required property for PRE is introduced in this paper and similar to the $\mathsf{key\ homomorphism}$ of PKE [15, 21]. By employing such a CPA secure PRE, the ciphertext-size of the second scheme is smaller than that of the first one.
- In order to give a post-quantum instantiation of our generic constructions, we preset a Kyber-based single-hop unidirectional PRE scheme with CPA security and $\mathsf{key\ homomorphism}$. This scheme can convert Kyber ciphertexts into other ciphertexts, and we can apply our second generic construction with bounded CCA security to this CPA secure scheme. We have chosen Kyber since this will be used widely as a selected algorithm in the NIST PQC competition. Hence, giving this instantiation is meaningful.

2 Preliminaries

Throughout this paper, we use the following notation: For a positive integer n, let $[n] := \{1, \dots, n\}$. For n values $x_1, \dots, x_n$ 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 indexes 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. For a probabilistic algorithm $\mathcal{A}$, $y \leftarrow \mathcal{A}(x; r)$ means that $\mathcal{A}$ on input x outputs y by using randomness r.

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

Furthermore, we describe definitions of cryptographic primitives and computational assumptions.

2.1 Proxy Re-encryption

In this section, we describe the syntax, security definitions, and several properties of (single-hop) unidirectional proxy re-encryption (PRE).

Following [1], 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. *Here, both* pk *and* sk *implicitly include the public parameter* pp.
- $\mathsf{Enc}(\mathsf{pk}, \mathsf{m}) \to \mathsf{ct}$*: The randomized algorithm* Enc *takes as input a pubic key* pk *and a message* $\mathsf{m} \in \mathcal{M}$*, and outputs a ciphertext* ct.
- $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) \to \mathsf{m}/\perp$*: The deterministic algorithm* Dec *takes as input a secret key* sk *and a ciphertext* ct*, and outputs a message* m *or the rejection symbol* $\perp$.
- $\mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j) \to \mathsf{rk}_{i\to j}$*: The randomized or deterministic algorithm takes as input a secrete key* sk_i *and a public key* pk_j*, and outputs a re-encryption key* $\mathsf{rk}_{i\to j}$.
- $\mathsf{ReEnc}(\mathsf{rk}_{i\to j}, \mathsf{ct}_i) \to \mathsf{ct}_j$*: The randomized 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* $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{ReKeyGen}, \mathsf{ReEnc})$ *is said to be* correct *if for every* $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$ *and every* $\mathsf{m} \leftarrow \mathcal{M}$*, the following holds:*

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

Re-encryption Correctness. *For every* $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KeyGen}(\mathsf{pp}), (\mathsf{pk}_j, \mathsf{sk}_j) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$*, 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* $\mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{pk}_i, \mathsf{m})$*.*

In order to describe security definitions of PRE, we describe derivatives of single-hop unidirectional PRE ciphertexts by following [7]:

Definition 3 (Derivatives of single-hop PRE ciphertexts [7]). *Suppose that the challenge ciphertext* ct^* *and the corresponding index* i^* *are defined in a security game of PRE.* Derivative *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)*.*

Following [7], we describe definitions of oracles in security games of PRE, as follows:

Definition 4. *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:*

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

Following [1], we describe the definitions of *security against chosen plaintext attacks* (denoted by CPA security).

Definition 5 (CPA security). *A 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* $\mathsf{Adv}^{\mathrm{cpa}}_{\Pi_{\mathsf{pre}},\mathcal{A}}(\lambda) := \left|\Pr[\mathsf{Expt}^{\mathrm{cpa}}_{\Pi_{\mathsf{pre}},\mathcal{A}}(\lambda) = 1] - 1/2\right|$ *is negligible in* λ*, where the experiment* $\mathsf{Expt}^{\mathrm{cpa}}_{\Pi_{\mathsf{pre}},\mathcal{A}}(\lambda)$ *is defined as follows:*

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

where state_0 *and* state_1 *are state information.*

As a new security notion of PRE, we formalize a bounded variant of security against chosen ciphertext attacks (denoted by bounded CCA security) by following [7, 9].

Definition 6 (Bounded CCA security). *Let* t_d, t_r *be positive integers. A PRE scheme* $\Pi_{\mathsf{pre}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{ReKeyGen}, \mathsf{ReEnc})$ *is* (t_d, t_r)*-*CCA *secure if for any PPT adversary* $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2)$ *against* Π_{pre}*, its advantage* $\mathsf{Adv}^{\mathrm{cca}}_{\Pi_{\mathsf{pre}},\mathcal{A}}(\lambda) := \left|\Pr[\mathsf{Expt}^{\mathrm{cca}}_{\Pi_{\mathsf{pre}},\mathcal{A}}(\lambda) = 1] - 1/2\right|$ *is negligible in* λ*, where the experiment* $\mathsf{Expt}^{\mathrm{cca}}_{\Pi_{\mathsf{pre}},\mathcal{A}}(\lambda)$ *is defined as follows:*

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

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

Special Properties. As a new property of PRE, we formalize *re-encryption key homomorphism.* This property is inspired by the *secret-to-public key homomorphism* defined in [15, 21].

Definition 7 (Re-encryption key homomorphism). *Let* $\Pi_{\mathsf{pre}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{ReKeyGen}, \mathsf{ReEnc})$ *be a PRE scheme with the secret key space* $\mathcal{K}_{\mathsf{sk}} = \mathcal{K}_{\mathsf{sk}}(\lambda)$*, the public key space* $\mathcal{K}_{\mathsf{pk}} = \mathcal{K}_{\mathsf{pk}}(\lambda)$*, and the re-encryption key space* $\mathcal{K}_{\mathsf{rk}} = \mathcal{K}_{\mathsf{rk}}(\lambda)$ *for a security parameter* λ *and a public parameter* $\mathsf{pp} \leftarrow \mathsf{Setup}(1^{\lambda})$*. The PRE scheme* Π_{pre} *is said to be* re-encryption key homomorphic *if there exist the following map* $\mu : \mathcal{K}_{\mathsf{sk}} \to \mathcal{K}_{\mathsf{pk}}$ *and polynomial-time algorithms* $(\mathsf{HReKeyGen}, \mathsf{ReKeyEval})$ *with positive integers* $u = \mathsf{poly}(\lambda), v = \mathsf{poly}(\lambda)$ *(where* $u \geq v$*):*

- *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* $\mu(\mathsf{sk} + \mathsf{sk}') = \mu(\mathsf{sk}) \cdot \mu(\mathsf{sk}')$*;*
- $\mathsf{HReKeyGen}((\mathsf{sk}_{A,i})_{i\in[u]}, (\mathsf{pk}_{B,i})_{i\in[u]}) \to (\mathsf{rk}^{(i\to j)}_{A\to B})_{i\in[u],j\in[u]}$*: The randomized algorithm* $\mathsf{HReKeyGen}$ *takes as input u secret keys* $(\mathsf{sk}_{A,i})_{i\in[u]}$ *and u public keys* $(\mathsf{pk}_{B,i})_{i\in[u]}$*, and outputs u re-encryption keys* $(\mathsf{rk}^{(i\to j)}_{A\to B})_{i\in[u],j\in[u]}$*.*
- $\mathsf{ReKeyEval}((\mathsf{rk}^{(a_i\to b_i)}_{A\to B})_{i\in[v]}) \to \mathsf{rk}_{A\to B}$*: The deterministic or randomized algorithm* $\mathsf{ReKeyEval}$ *takes as input v re-encryption keys* $(\mathsf{rk}^{(a_i\to b_i)}_{A\to B})_{i\in[v]}$ *(for all distinct* $a_1, \ldots, a_v \in [u]$ *and all distinct* $b_1, \ldots, b_v \in [u]$*) and outputs a new re-encryption key* $\mathsf{rk}_{A\to B}$*.*
- *For every* $\mathsf{pp} \leftarrow \mathsf{Setup}(\lambda)$*, every* $\{(\mathsf{pk}_{A,i}, \mathsf{sk}_{A,i}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})\}_{i\in[u]}$*, every* $\{(\mathsf{pk}_{B,i}, \mathsf{sk}_{B,i}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})\}_{i\in[u]}$*, every* $(\mathsf{rk}^{(i\to j)}_{A\to B})_{i\in[u],j\in[u]} \leftarrow \mathsf{HReKeyGen}((\mathsf{sk}_{A,i})_{i\in[u]}, (\mathsf{pk}_{B,i})_{i\in[u]})$*, every* $\mathsf{rk}_{A\to B} \leftarrow \mathsf{ReKeyEval}((\mathsf{rk}^{(a_i\to b_i)}_{A\to B})_{i\in[v]})$ *(for all* $a_1, \ldots, a_v, b_1, \ldots, b_v \in [u]$*), and every* $\mathsf{m} \in \mathcal{M}$*, it holds that* $\mathsf{Dec}(\mathsf{sk}_B, \mathsf{ct}_B) = \mathsf{m}$ *with overwhelming probability, where* $\mathsf{pk}_A = \mu(\mathsf{pk}_{A,a_1}, \ldots, \mathsf{pk}_{A,a_v})$*,* $\mathsf{sk}_B = \mu(\mathsf{sk}_{B,b_1}, \ldots, \mathsf{sk}_{B,b_v})$*,* $\mathsf{ct}_A \leftarrow \mathsf{Enc}(\mathsf{pk}_A, \mathsf{m})$*, and* $\mathsf{ct}_B \leftarrow \mathsf{ReEnc}(\mathsf{rk}_{A\to B}, \mathsf{ct}_A)$*.*

As a new security notion of PRE, we introduce *security against chosen plaintext attacks with key homomorphism* (denoted by KH-CPA security) and formalize this security notion, as follows:

Definition 8 (KH-CPA security). *A PRE scheme* $\Pi_{\mathsf{pre}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{ReKeyGen}, \mathsf{ReEnc}, \mathsf{HReKeyGen})$ *with* key homomorphism *is* KH-CPA *secure if for any PPT adversary* $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2)$ *against* Π_{pre}*, its advantage* $\mathsf{Adv}^{\text{kh-cpa}}_{\Pi_{\mathsf{pre}},\mathcal{A}}(\lambda) := \left|\Pr[\mathsf{Expt}^{\text{kh-cpa}}_{\Pi_{\mathsf{pre}},\mathcal{A}}(\lambda) = 1] - 1/2\right|$ *is negligible in* λ*, where the experiment* $\mathsf{Expt}^{\text{kh-cpa}}_{\Pi_{\mathsf{pre}},\mathcal{A}}(\lambda)$ *is defined as follows:*

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

where

- *the key-generation oracle* $\mathtt{O}.\widehat{\mathtt{KeyGen}}$ *given a key-generation query* $(n, \mathcal{U}_{\mathtt{Corrupt}})$ *computes* $(\mathsf{pk}_{i,j}, \mathsf{sk}_{i,j}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$ *for every* $(i,j) \in [n] \times [u]$ *and returns* $(\{\mathsf{pk}_{i,j}\}_{i\in[n], j\in[u]}, \{\mathsf{sk}_{i,j}\}_{i\in\mathcal{U}_{\mathtt{Corrupt}}, j\in[u]})$*; and*
- *the homomorphic re-encryption key generation oracle* $\mathtt{O.HReKeyGen}$ *given a homomorphic re-encryption key query* $(A, B) \in [n] \times [n]$ *returns* $\perp$ *if* $A \in \mathcal{U}_{\mathtt{Honest}} \wedge B \in \mathcal{U}_{\mathtt{Corrupt}}$ *holds, and returns* $(\mathsf{rk}^{(i\to j)}_{A\to B})_{i\in[u], j\in[u]} \leftarrow \mathsf{HReKeyGen}((\mathsf{sk}_{A,i})_{i\in[u]}, (\mathsf{pk}_{B,i})_{i\in[u]})$ *otherwise.*

2.2 One-Time Signatures

We describe the syntax and a security definition of one-time signatures (OTSs).

Definition 9 (One-time signatures). *For a security parameter* λ*, let* $\mathcal{M} = \mathcal{M}(\lambda)$ *be a message space. A one-time signature (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.
- $\mathsf{Sign}(\mathsf{sigk}, \mathsf{m}) \to \mathsf{sig}$*: 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* sig.
- $\mathsf{Vrfy}(\mathsf{vk}, \mathsf{m}, \mathsf{sig}) \to \top/\perp$*: The deterministic algorithm* Vrfy *takes as input a verification key* vk*, a message* $\mathsf{m} \in \mathcal{M}$*, and a signature* sig*, and it outputs* $\top$ *(accept) or* $\perp$ *(reject).*

An OTS scheme is required to satisfy correctness, as follows:

Definition 10 (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* $\mathsf{Vrfy}(\mathsf{vk}, \mathsf{m}, \mathsf{sig}) = \top$ *with overwhelming probability, where* $\mathsf{sig} \leftarrow \mathsf{Sign}(\mathsf{sigk}, \mathsf{m})$.

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

Definition 11 (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* $\mathsf{Adv}^{\text{suf-ot}}_{\Pi_{\mathsf{ots}},\mathcal{A}}(\lambda) = \Pr[\mathcal{A} \text{ wins}]$ *is negligible in* λ*, where* $[\mathcal{A} \text{ wins}]$ *is the event that* $\mathcal{A}$ *wins in the following security game between a challenger and* $\mathcal{A}$*:*

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

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

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

2.3 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 $(x_1, \ldots, x_v, z)$.

We describe the definition of AONTs, as follows:

Definition 12 (AONT). *An efficient randomized algorithm* Trans *is* $(\mu, \bar{\mu}, v)$-AONT *if the following conditions hold:*

1. *Given* $X \in \{0,1\}^\mu$, Trans *outputs* $v+1$ *blocks* $(x_1, \ldots, x_v, z) \in (\{0,1\}^{\bar{\mu}})^{v+1}$, *where for* $i \in [v]$, x_i *is a secret share, and* z *is a public share.*
2. *There exists an efficient inverse function* $\mathsf{Inverse}$ *which, on input* $(x_1, \ldots, x_v, z) \in (\{0,1\}^{\bar{\mu}})^{v+1}$, *outputs* $X \in \{0,1\}^\mu$.
3. *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* $\mathtt{O.LR}$ *is the* $\mathsf{left\text{-}or\text{-}right}$ *oracle which, on input* $(j, X_0, X_1) \in [v] \times (\{0,1\}^\mu)^2$, *returns* $(x_1, \ldots, x_{j-1}, x_{j+1}, \ldots, x_v, z)$.

2.4 Module-Learning with Errors (Module-LWE)

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

Definition 13 (Module-LWE). *For a security parameter* λ, *let* $n = n(\lambda), k = k(\lambda), \eta = \eta(\lambda)$ *denote positive integers. The module-LWE problem is to distinguish between uniform samples* $(\boldsymbol{a}_i, b_i) \in R_q^k \times R_q$ *from* m *samples* $(\boldsymbol{a}_i, b_i) \in R_q^k \times R_q$ *for* $i \in [m]$, *where* $\boldsymbol{a}_i \xleftarrow{\$} R_q^k$, $\boldsymbol{s} \xleftarrow{\$} \beta_\eta^k$, *and* $e_i \xleftarrow{\$} \beta_\eta$ *are samples (uniformly) at random, and* $b_i = \boldsymbol{a}_i^T \boldsymbol{s} + e_i$.

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

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

is negligible in λ.

2.5 Disjunct Matrices

We describe the definition of disjunct matrices, as follows:

Definition 14 (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* $\forall j' \in \{s_1, \ldots, s_t\}$, $m_{q,j'} = 0$.

The number of tests required in combinatorial non-adaptive group testing using t-disjunct matrices is bounded by $u = O(t^2 \log n)$ (e.g., see [12]).

3 Bounded CCA secure PRE from CPA secure PRE

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

3.1 Generic Construction

Our construction is based on a generic construction [21] of bounded CCA secure PKE. Additionally, we utilize the re-encryption functionality and CPA security of the underlying PRE in order to achieve both re-encryption functionality and bounded CCA security.

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 $\mathcal{M}_{\Pi'_{\mathsf{pre}}} = \{0,1\}^{\bar{\mu}}$, where $\bar{\mu} = \bar{\mu}(\lambda)$ is a positive integer for a security parameter λ;
- An 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 $\mathsf{Inverse}$, where $\mu = \mu(\lambda)$ and $v = v(\lambda)$ are positive integers for a security parameter λ.

The proposed PRE scheme $\Pi_{\mathsf{pre}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{ReKeyGen}, \mathsf{ReEnc})$ 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}[3].
  - 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 $(x_1, \ldots, x_v, z) \leftarrow \mathsf{Trans}(\mathsf{m})$.
  4. Compute $\{\sigma_1, \ldots, \sigma_v\} \leftarrow \phi_{\boldsymbol{M}}(\mathsf{vk})$, where all $\sigma_1, \ldots, \sigma_v \in [u]$ are distinct.
  5. Compute $\mathsf{ct}'_i \leftarrow \Pi'_{\mathsf{pre}}.\mathsf{Enc}(\mathsf{pk}_{\sigma_i}, x_i)$ for every $i \in [v]$.
  6. Compute $\mathsf{sig} \leftarrow \Pi_{\mathsf{ots}}.\mathsf{Sign}(\mathsf{sigk}, (\mathsf{ct}'_1 \parallel \cdots \parallel \mathsf{ct}'_v \parallel z))$.

[3]By using a collision resistant hash function, we can compress the size of the verification keys of Π_{ots} into the space $[\bar{n}]$, so that we can employ a group testing-based methodology without using exponential-sized matrices.

7. Output $\mathsf{ct} = (\mathsf{vk}, (\mathsf{ct}'_i)_{i\in[v]}, z, \mathsf{sig})$.

– $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) \to \mathsf{m}/\bot$:
  1. Parse $\mathsf{sk} = (\mathsf{sk}'_i)_{i\in[u]}$ and $\mathsf{ct} = (\mathsf{vk}, (\mathsf{ct}'_i)_{i\in[v]}, z, \mathsf{sig})$.
  2. Output $\bot$ if $\Pi_{\mathsf{ots}}.\mathsf{Vrfy}(\mathsf{vk}, (\mathsf{ct}'_1 \parallel \cdots \parallel \mathsf{ct}'_v \parallel z), \mathsf{sig}) = \bot$.
  3. Compute $\{\sigma_1, \ldots, \sigma_v\} \leftarrow \phi_{\boldsymbol{M}}(\mathsf{vk})$.
  4. Compute $x'_i \leftarrow \Pi'_{\mathsf{pre}}.\mathsf{Dec}(\mathsf{sk}'_{\sigma_i}, \mathsf{ct}'_i)$ for every $i \in [v]$.
  5. Output $\mathsf{m}' \leftarrow \mathsf{Inverse}(x'_1, \ldots, x'_v, z)$ if $x'_i \neq \bot$ holds for every $i \in [v]$; otherwise, output $\bot$.

– $\mathsf{ReKeyGen}(\mathsf{sk}_A, \mathsf{pk}_B) \to \mathsf{rk}_{A\to B}$:
  1. Parse $\mathsf{sk}_A = (\mathsf{sk}'_{A,i})_{i\in[u]}$ and $\mathsf{pk}_B = (\mathsf{pk}'_{B,i})_{i\in[u]}$.
  2. For every $i \in [u]$ and $j \in [u]$, compute $\mathsf{rk}^{(i\to j)}_{A\to B} \leftarrow \Pi'_{\mathsf{pre}}.\mathsf{ReKeyGen}(\mathsf{sk}'_{A,i}, \mathsf{pk}'_{B,j})$.
  3. Output $\mathsf{rk}_{A\to B} = (\mathsf{rk}^{(i\to j)}_{A\to B})_{i\in[u],j\in[u]}$.

– $\mathsf{ReEnc}(\mathsf{rk}_{A\to B}, \mathsf{ct}_A) \to \mathsf{ct}_B$:
  1. Parse $\mathsf{rk}_{A\to B} = (\mathsf{rk}^{(i\to j)}_{A\to B})_{i\in[u],j\in[u]}$ and $\mathsf{ct}_A = (\mathsf{vk}_A, (\mathsf{ct}'_{A,i})_{i\in[v]}, z, \mathsf{sig}_A)$.
  2. Output $\bot$ if $\Pi_{\mathsf{ots}}.\mathsf{Vrfy}(\mathsf{vk}_A, (\mathsf{ct}'_{A,i})_{i\in[v]}, \mathsf{sig}_A) = \bot$.
  3. Generate $(\mathsf{vk}_B, \mathsf{sigk}_B) \leftarrow \Pi_{\mathsf{ots}}.\mathsf{KeyGen}(1^\lambda)$.
  4. Compute $\{\sigma^{(A)}_1, \ldots, \sigma^{(A)}_v\} \leftarrow \phi_{\boldsymbol{M}}(\mathsf{vk}_A)$ and $\{\sigma^{(B)}_1, \ldots, \sigma^{(B)}_v\} \leftarrow \phi_{\boldsymbol{M}}(\mathsf{vk}_B)$.
  5. For $i \in [v]$, compute $\mathsf{ct}'_{B,i} \leftarrow \Pi'_{\mathsf{pre}}.\mathsf{ReEnc}(\mathsf{rk}^{(\sigma^{(A)}_i \to \sigma^{(B)}_i)}_{A\to B}, \mathsf{ct}'_{A,i})$.
  6. Compute $\mathsf{sig}_B \leftarrow \Pi_{\mathsf{ots}}.\mathsf{Sign}(\mathsf{sigk}_B, (\mathsf{ct}'_{B,1} \parallel \cdots \parallel \mathsf{ct}'_{B,v} \parallel z))$.
  7. Output $\mathsf{ct}_B = (\mathsf{vk}_B, (\mathsf{ct}'_{B,i})_{i\in[v]}, z, \mathsf{sig}_B)$.

Proposition 1 (Correctness of Π_{pre}). *If the PRE scheme Π'_{pre} and the OTS scheme Π_{ots} are* correct*, then the resulting PRE scheme Π_{pre} is also* correct.

It is clear that Π_{pre} is correct due to the correctness of Π'_{pre} and Π_{ots}, and a property of AONT Trans. Thus, we omit the proof of the correctness of Π_{pre}.

3.2 Security Proof

The following theorem shows the bounded CCA security (i.e., (t,t)-CCA security) of the proposed scheme Π_{pre}:

Theorem 1 (Security of Π_{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)-*CPA *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 Π_{pre} is (t,t)-*CCA *secure.*

*Concretely, if there exists a PPT algorithm $\mathcal{A}$ against the (t,t)-*CCA *secure PRE Π_{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*

$$\mathsf{Adv}^{\mathrm{cca}}_{\Pi_{\mathsf{pre}},\mathcal{A}}(\lambda) \leq n_h u \cdot \mathsf{Adv}^{\mathrm{cpa}}_{\mathcal{B}_1,\Pi'_{\mathsf{pre}}}(\lambda) + \mathsf{Adv}^{\mathrm{suf}}_{\Pi_{\mathsf{ots}},\mathcal{F}}(\lambda) + n_h u \cdot \mathsf{Adv}^{\mathrm{ind}}_{\mathcal{B}_2,\mathsf{AONT}}(\lambda).$$

Proof. Let $\mathcal{A}$ be a PPT adversary against the PRE scheme Π_{pre}. Let $i^* \in [n]$ be the user which $\mathcal{A}$ submits to the challenge oracle $\mathtt{O.Challenge}$ and let $\mathsf{ct}^* = (\mathsf{vk}^*, (\mathsf{ct}'^*_k)_{k\in[v]}, z^*, \mathsf{sig}^*)$ denote the challenge ciphertext under pk_{i^*}. Queries issued to the decryption oracle $\mathtt{O.Dec}$ and the re-encryption oracle $\mathtt{O.ReEnc}$ are denoted by decryption queries and re-encryption queries, respectively.

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

<u>Game_0</u>: The same game as the (t,t)-CCA security game. Then, we have $\mathsf{Adv}^{\mathrm{cca}}_{\Pi_{\mathsf{pre}},\mathcal{A}}(\lambda) = |\Pr[W_0] - 1/2|$.

<u>Game_1</u>: The same game as Game_0 except for the following procedure of the decryption oracle $\mathtt{O.Dec}$ and the re-encryption oracle $\mathtt{O.ReEnc}$: For a decryption or re-encryption query on $\mathsf{ct}_i = (\mathsf{vk}_i, (\mathsf{ct}'_{i,j})_{j\in[v]}, z_i, \mathsf{sig}_i)$, 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 $\Pi_{\mathsf{ots}}.\mathsf{Vrfy}(\mathsf{vk}_i, (\mathsf{ct}'_{i,1}\| \cdots \|\mathsf{ct}'_{i,v}\|z_i), \mathsf{sig}_i) = \top$. If so, the experiment aborts; otherwise, $\mathtt{O.Dec}$ computes $\mathsf{m}' \leftarrow \mathsf{Dec}(\mathsf{sk}_i, \mathsf{ct}_i)$ and returns $\mathsf{m}' \in \mathcal{M} \cup \{\perp\}$.

Let Bad be the event that $\mathcal{A}$ issues a decryption or re-encryption query on ct_i such that $\mathsf{vk}_i = \mathsf{vk}^*$, $\mathsf{ct}_i \neq \mathsf{ct}^*$, and $\Pi_{\mathsf{ots}}.\mathsf{Vrfy}(\mathsf{vk}_i, (\mathsf{ct}'_{i,1}\| \cdots \|\mathsf{ct}'_{i,v}\|z_i), \mathsf{sig}_i)) = \top$. Then, Game_0 and Game_1 are identical unless Bad occurs. Hence, we bound the probability that Bad occurs. In order to estimate this upper bound, we construct a PPT algorithm $\mathcal{F}$ breaking the strongly unforgeable OTS scheme Π_{ots}. On input a verification key vk^* of Π_{ots}, $\mathcal{F}$ generates $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$ and gives pp to $\mathcal{A}$. When $\mathcal{A}$ submits $(n, \mathcal{U}_{\mathsf{Corrupt}})$, $\mathcal{F}$ generates $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$ for every $i \in [n]$ and returns $(\{\mathsf{pk}_i\}_{i\in[n]}, \{\mathsf{sk}_i\}_{i\in\mathcal{U}_{\mathsf{Corrupt}}})$. By using these generated key-pairs, $\mathcal{F}$ simulates the oracle $\mathtt{O.ReKeyGen}$. Additionally, the oracle $\mathtt{O.Dec}$ (resp., $\mathtt{O.ReEnc}$) is simulated as follows: For a decryption query (i, ct_i) (resp., (i, j, ct_i)) (where $\mathsf{ct}_i = (\mathsf{vk}_i, (\mathsf{ct}'_{i,j})_{j\in[v]}, z_i, \mathsf{sig}_i)$), $\mathcal{F}$ aborts and outputs a forgery $((\mathsf{ct}'_{i,1}\| \cdots \|\mathsf{ct}'_{i,v}\|z_i), \mathsf{sig}_i)$ in the strong unforgeability game of Π_{ots}, if it holds that $\mathsf{vk}_i = \mathsf{vk}^*$, $\mathsf{ct}_i \neq \mathsf{ct}^*$, and $\Pi_{\mathsf{ots}}.\mathsf{Vrfy}(\mathsf{vk}_i, (\mathsf{ct}'_{i,1}\| \cdots \|\mathsf{ct}'_{i,v}\|z_i), \mathsf{sig}_i)) = \top$ (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 $(i^*, \mathsf{m}^*_0, \mathsf{m}^*_1)$, $\mathcal{F}$ chooses $b \xleftarrow{\$} \{0,1\}$ computes $((\mathsf{ct}'^*_i)_{i\in[v]}, z^*)$ by following the procedure of $\mathsf{Enc}(\mathsf{pk}_{i^*}, \mathsf{m}^*_b)$. Then, this algorithm issues $(\mathsf{ct}'^*_1\| \cdots \|\mathsf{ct}'^*_v\|z^*)$ to the signing oracle in the strong unforgeability game and obtains sig^*. And then, $\mathcal{F}$ returns the challenge ciphertext $\mathsf{ct}^* = (\mathsf{vk}^*, (\mathsf{ct}'^*_i)_{i\in[v]}, z^*, \mathsf{sig}^*)$. Finally, when $\mathcal{A}$ outputs $b' \in \{0,1\}$ and Bad has not occurred, $\mathcal{F}$ halts and outputs 0.

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

In order to bound the winning probability of $\mathcal{A}$ in Game_1, we consider the following experiment $\mathcal{B}$: At the beginning of the (t,t)-CCA game, $\mathcal{B}$ gives $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$ to $\mathcal{A}$ and simulates $(\{\mathsf{pk}_i\}_{i\in[n]}, \{\mathsf{sk}_i\}_{i\in\mathcal{U}_{\mathsf{Corrupt}}}) \leftarrow \mathtt{O.KeyGen}(n, \mathcal{U}_{\mathsf{Corrupt}})$.

And then, $\mathcal{B}$ generates $(\mathsf{vk}^*, \mathsf{sigk}^*) \leftarrow \Pi_{\mathsf{ots}}.\mathsf{KeyGen}(1^\lambda)$, chooses indices $i^* \xleftarrow{\$} [n_h]$, $j^* \xleftarrow{\$} \phi_{\boldsymbol{M}}(\mathsf{vk}^*)$ (where n_h is the number of honest users), and simulates the environment of $\mathcal{A}$ except for the following: The experiment aborts and outputs a random bit if $\mathcal{A}$ issues

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

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

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

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

The $\mathcal{A}$'s advantage $\varepsilon_{\mathcal{A}}$ in Game_1 is equivalent to $|\Pr[W_{\mathcal{B}} \mid \neg\mathsf{Abort}] - 1/2|$. Hence, the $\mathcal{B}$'s advantage $\varepsilon_{\mathcal{B}} = |\Pr[W_{\mathcal{B}}] - 1/2|$ is at least $\varepsilon_{\mathcal{A}}/(n_h u)$. Here, let $\phi_{\boldsymbol{M}}(\mathsf{vk}^*) := \{\sigma_1^*, \ldots, \sigma_v^*\}$ and $\sigma_{k^*}^* := j^*$ (where $\sigma_1^*, \ldots, \sigma_v^* \in [u]$ and $k^* \in [v]$). In order to bound $\varepsilon_{\mathcal{B}}$, we change the environment of $\mathcal{A}$. In this modified environment, the j^*-th share $x_{j^*}^*$ generated by Trans is replaced with the all-zero string $0^{|x_{j^*}^*|}$, when producing the challenge ciphertext. The probability $\Pr[W_{\mathcal{B}}]$ is defined as $p^{(0)}$, and the probability that $W_{\mathcal{B}}$ occurs in the modified environment is defined as $p^{(1)}$. Then we have $\varepsilon_{\mathcal{B}} \leq |p^{(0)} - p^{(1)}| + |p^{(1)} - 1/2|$.

In order to bound $|p^{(0)} - p^{(1)}|$, we construct a PPT algorithm $\mathcal{B}_1$ against the CPA security of Π'_{pre}, as follows: On input the public parameter pp' in the CPA game, $\mathcal{B}_1$ generates pp by following the algorithm Setup and gives pp to $\mathcal{A}$. When $\mathcal{A}$ submits the key generation query $(n, \mathcal{U}_{\mathtt{Corrupt}})$, $\mathcal{B}_1$ generates $(\mathsf{vk}^*, \mathsf{sigk}^*) \leftarrow \Pi_{\mathsf{ots}}.\mathsf{KeyGen}(1^\lambda)$, chooses $i^* \xleftarrow{\$} [n], j^* \xleftarrow{\$} \phi_{\boldsymbol{M}}(\mathsf{vk}^*)$, and obtains $(\{\mathsf{pk}'_{i,j}\}_{(i,j)\in[n]\times[u]}, \{\mathsf{sk}'_{i,j}\}_{(i,j)\in[n]\times[u]\setminus\{(i^*,j^*)\}})$ by querying the key generation oracle in the CPA game. Here, for simplicity, we suppose that $(i,j) \in [n]\times[u]$ represents a user in the CPA game and let $\mathcal{U}_{\mathtt{Honest}} = [n]\setminus\mathcal{U}_{\mathtt{Corrupt}}$ denote the set of honest users in the (t,t)-CCA game. Then $\mathcal{B}_1$ returns $(\{\mathsf{pk}_i\}_{i\in[n]}, \{\mathsf{sk}_i\}_{i\in\mathcal{U}_{\mathtt{Corrupt}}})$, where let $\mathsf{pk}_i := (\mathsf{pk}'_{i,j})_{j\in[u]}$ for every $i \in [n]$, let $\mathsf{sk}_i := (\mathsf{sk}'_{i,j})_{j\in[u]}$ for every $i \in [n]\setminus\{i^*\}$, and let $\mathsf{sk}_{i^*} := (\mathsf{sk}'_{i^*,j})_{j\in[u]\setminus\{j^*\}}$. Furthermore, this algorithm simulates the oracles $\mathtt{O.ReKeyGen}, \mathtt{O.ReEnc}, \mathtt{O.Dec}, \mathtt{O.Challenge}_b$, as follows:

- $\mathtt{O.ReKeyGen}(A, B)$: If $A \in \mathcal{U}_{\mathtt{Honest}} \wedge B \in \mathcal{U}_{\mathtt{Corrupt}}$ holds, $\mathcal{B}_1$ returns $\perp$; otherwise it does the following:

- Case ($A = i^*$): Obtain $\mathsf{rk}_{i^* \to B}^{(j^* \to j)}$ by issuing $((i^*, j^*), (B, j))$ to the re-encryption key generation oracle in the CPA game, for every $j \in [u]$. For every $i \in [u] \backslash \{j^*\}$ and every $j \in [u]$, compute $\mathsf{rk}_{A \to B}^{(i \to j)} \leftarrow \Pi'_{\mathsf{pre}}.\mathsf{ReKeyGen}(\mathsf{sk}'_{i^*,i}, \mathsf{pk}'_{B,j})$.
- Case ($A \neq i^*$): Compute $\mathsf{rk}_{A \to B}^{(i \to j)} \leftarrow \Pi'_{\mathsf{pre}}.\mathsf{ReKeyGen}(\mathsf{sk}'_{A,i}, \mathsf{pk}'_{B,j})$ for every $i \in [u]$ and every $j \in [u]$.

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

– $\mathtt{O.Dec}(A, \mathsf{ct}_A)$. $\mathcal{B}_1$ parses $\mathsf{ct}_A = (\mathsf{vk}_A, (\mathsf{ct}'_{A,i})_{i \in [v]}, z, \mathsf{sig}_A)$ and does the following:
1. Return $\perp$ if (A, ct_A) is a $\mathsf{derivative}$ of (i^*, ct^*).
2. Abort and output a random bit if $A = i^* \wedge j^* \in \phi_{\boldsymbol{M}}(\mathsf{vk}_A)$ holds.
3. Returns $\perp$ if it holds that $\mathsf{vk}_A = \mathsf{vk}_{i^*}$, $\mathsf{ct}_A \neq \mathsf{ct}^*$ and $\Pi_{\mathsf{ots}}.\mathsf{Vrfy}(\mathsf{vk}_A, (\mathsf{ct}'_{A,1} \| \cdots \| \mathsf{ct}'_{A,v} \| z), \mathsf{sig}_A) = \top$.
4. Return $\perp$ if $\Pi_{\mathsf{ots}}.\mathsf{Vrfy}(\mathsf{vk}_A, (\mathsf{ct}'_{A,1} \| \cdots \| \mathsf{ct}'_{A,v} \| z), \mathsf{sig}_A) = \perp$ holds.
5. Compute $x'_i \leftarrow \Pi'_{\mathsf{pre}}.\mathsf{Dec}(\mathsf{sk}'_{A,i}, \mathsf{ct}'_{A,i})$ for every $i \in [v]$,.
6. Return $\mathsf{m}' \leftarrow \mathsf{Inverse}(x'_1, \dots, x'_v, z)$ if $x'_i \neq \perp$ holds for every $i \in [v]$; otherwise return $\perp$.

– $\mathtt{O.ReEnc}(A, B, \mathsf{ct}_A)$. $\mathcal{B}_1$ parses $\mathsf{ct}_A = (\mathsf{vk}_A, (\mathsf{ct}'_{A,i})_{i \in [v]}, z, \mathsf{sig}_A)$ and does the following:
1. Return $\perp$ if $j \in \mathcal{U}_{\mathtt{Corrupt}}$ holds and (A, ct_A) is a $\mathsf{derivative}$ of (i^*, ct^*).
2. Abort and output a random bit if $A = i^* \wedge j^* \in \phi_{\boldsymbol{M}}(\mathsf{vk}_A)$ holds.
3. Returns $\perp$ if it holds that $\mathsf{vk}_A = \mathsf{vk}_{i^*}$, $\mathsf{ct}_A \neq \mathsf{ct}^*$ and $\Pi_{\mathsf{ots}}.\mathsf{Vrfy}(\mathsf{vk}_A, (\mathsf{ct}'_{A,1} \| \cdots \| \mathsf{ct}'_{A,v} \| z), \mathsf{sig}_A) = \top$.
4. Return $\perp$ if $\Pi_{\mathsf{ots}}.\mathsf{Vrfy}(\mathsf{vk}_A, (\mathsf{ct}'_{A,1} \| \cdots \| \mathsf{ct}'_{A,v} \| z), \mathsf{sig}_A) = \perp$ holds.
5. Generate $(\mathsf{vk}_B, \mathsf{sigk}_B) \leftarrow \Pi_{\mathsf{ots}}.\mathsf{KeyGen}(1^\lambda)$.
6. For every $i \in [u]$ and $j \in [u]$, compute $\mathsf{rk}_{A \to B}^{(i \to j)} \leftarrow \Pi'_{\mathsf{pre}}.\mathsf{ReKeyGen}(\mathsf{sk}'_{A,i}, \mathsf{pk}'_{B,j})$.
7. Let $\{\sigma_1^{(A)}, \dots, \sigma_v^{(A)}\} \leftarrow \phi_{\boldsymbol{M}}(\mathsf{vk}_A)$ and $\{\sigma_1^{(B)}, \dots, \sigma_v^{(B)}\} \leftarrow \phi_{\boldsymbol{M}}(\mathsf{vk}_B)$.
8. Compute $\mathsf{ct}'_{B,i} \leftarrow \Pi'_{\mathsf{pre}}.\mathsf{ReEnc}(\mathsf{rk}_{A \to B}^{(\sigma_i^{(A)} \to \sigma_i^{(B)})}, \mathsf{ct}'_{A,i})$ for every $i \in [v]$.
9. Compute $\mathsf{sig}_B \leftarrow \Pi_{\mathsf{ots}}.\mathsf{Sign}(\mathsf{sigk}_B, (\mathsf{ct}'_{B,1} \| \cdots \| \mathsf{ct}'_{B,v} \| z))$.
10. Return $\mathsf{ct}_B = (\mathsf{vk}_B, (\mathsf{ct}'_{B,i})_{i \in [v]}, z, \mathsf{sig}_B)$.

– $\mathtt{O.Challenge}_b(i', \mathsf{m}_0^*, \mathsf{m}_1^*)$. $\mathcal{B}_1$ does the following:
1. Abort and output a random bit if $i^* \neq i'$.
2. Let $\{\sigma_1^*, \dots, \sigma_v^*\} \leftarrow \phi_{\boldsymbol{M}}(\mathsf{vk}^*)$.
3. Compute $(x_1^*, \dots, x_v^*, z^*) \leftarrow \mathsf{Trans}(\mathsf{m}_b^*)$.
4. Obtain $\mathsf{ct}'^*_{j^*}$ by submitting $(x^*_{j^*}, 0^{\bar{\mu}})$ to the CPA game.
5. For every $j \in [v] \backslash \{j^*\}$, then compute $\mathsf{ct}'^*_j \leftarrow \Pi'_{\mathsf{pre}}.\mathsf{Enc}(\mathsf{pk}'_{\sigma_j^*}, x_j^*)$.
6. Compute $\mathsf{sig}^* \leftarrow \Pi_{\mathsf{ots}}.\mathsf{Sign}(\mathsf{sigk}^*, (\mathsf{ct}'^*_1 \| \cdots \| \mathsf{ct}'^*_v \| z^*))$.
7. Return $\mathsf{ct}^* = (\mathsf{vk}^*, (\mathsf{ct}'^*_i)_{i \in [v]}, z^*, \mathsf{sig}^*)$.

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

In order to bound the winning probability in the modified environment (i.e., $|p^{(1)} - 1/2|$), we construct a PPT algorithm $\mathcal{B}_2$ against $(\mu, \bar{\mu}, v)$-AONT Trans. By using $\mathcal{A}$, we construct $\mathcal{B}_2$ given the oracle O.LR in the game of Trans: At the beginning of the (t, t)-CCA game, $\mathcal{B}_2$ gives $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$ to $\mathcal{A}$. When $\mathcal{A}$ issues $(n, \mathcal{U}_{\mathtt{Corrupt}})$, $\mathcal{B}_2$ generates $(\mathsf{vk}^*, \mathsf{sigk}^*) \leftarrow \Pi_{\mathsf{ots}}.\mathsf{KeyGen}(1^\lambda)$, chooses $i^* \xleftarrow{\$} [n], j^* \xleftarrow{\$} \phi_{\boldsymbol{M}}(\mathsf{vk}^*)$, and generates all key-pairs $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$ for all users $i \in [n]$. And then, $\mathcal{B}_2$ simulates the oracles O.ReKeyGen, O.ReEnc, O.Dec by using the generated key-pairs. Furthermore, $\mathcal{B}_2$ simulates $\mathtt{O.Challenge}_b(i', \mathsf{m}^*_0, \mathsf{m}^*_1)$ as follows:

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

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

$\mathcal{B}_2$ simulates the oracles O.KeyGen, O.ReKeyGen, O.ReEnc, O.Dec completely since it has the key-pairs of all users. The oracle 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 $\mathsf{Adv}^{\mathrm{cpa}}_{\Pi'_{\mathsf{pre}},\mathcal{B}_1}(\lambda) + \mathsf{Adv}^{\mathrm{ind}}_{\mathsf{Trans},\mathcal{B}_2}(\lambda) \geq \varepsilon_{\mathcal{A}}/(n_h u)$. From the discussion above, we obtain

$$\mathsf{Adv}^{\mathrm{cca}}_{\Pi_{\mathsf{pre}},\mathcal{A}}(\lambda) \leq n_h u \cdot \mathsf{Adv}^{\mathrm{cpa}}_{\mathcal{B}_1,\Pi'_{\mathsf{pre}}}(\lambda) + n_h u \cdot \mathsf{Adv}^{\mathrm{ind}}_{\mathsf{AONT},\mathcal{B}_2}(\lambda) + \mathsf{Adv}^{\mathrm{suf}}_{\Pi_{\mathsf{ots}},\mathcal{F}}(\lambda).$$

and complete the proof.

4 Bounded CCA secure PRE with Compact Ciphertexts

In this section, we present a generic construction with compact ciphertexts, which starts from any CPA secure PRE with key homomorphism and any strongly unforgeable OTS.

4.1 Construction

We provide a bounded CCA secure PRE scheme with compact ciphertexts. This scheme is based on our generic construction in Section 3.1 and constructed from any CPA secure PRE with key-homomorphism (Definition 7).

In the proposed scheme, we employ the following (cryptographic) primitives:

– 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 key homomorphism (i.e., there exist PPT algorithms $\Pi'_{\mathsf{pre}}.\mathsf{HReKeyGen}, \Pi'_{\mathsf{pre}}.\mathsf{ReKeyEval}$);

– A one-time signature scheme $\Pi_{\mathsf{ots}} = (\Pi_{\mathsf{ots}}.\mathsf{KeyGen}, \Pi_{\mathsf{ots}}.\mathsf{Sign}, \Pi_{\mathsf{ots}}.\mathsf{Vrfy})$.

The proposed PRE scheme $\Pi^{\mathsf{kh}}_{\mathsf{pre}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{ReKeyGen}, \mathsf{ReEnc})$ 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 $[n]$ 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.

  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 $\{\sigma_1, \ldots, \sigma_v\} \leftarrow \phi_{\boldsymbol{M}}(\mathsf{vk})$, where all $\sigma_1, \ldots, \sigma_v \in [u]$ are distinct.
  4. Compute $\mathsf{pk}_{\mathsf{vk}} \leftarrow \prod_{i\in[v]} \mathsf{pk}'_{\sigma_i}$.
  5. Compute $\mathsf{ct}' \leftarrow \Pi'_{\mathsf{pre}}.\mathsf{Enc}(\mathsf{pk}_{\mathsf{vk}}, \mathsf{m})$.
  6. Compute $\mathsf{sig} \leftarrow \Pi_{\mathsf{ots}}.\mathsf{Sign}(\mathsf{sigk}, \mathsf{ct}')$.
  7. Output $\mathsf{ct} = (\mathsf{vk}, \mathsf{ct}', \mathsf{sig})$.
- $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) \to \mathsf{m}/\perp$:
  1. Parse $\mathsf{sk} = (\mathsf{sk}'_i)_{i\in[n]}$ and $\mathsf{ct} = (\mathsf{vk}, \mathsf{ct}', \mathsf{sig})$.
  2. Output $\perp$ if $\Pi_{\mathsf{ots}}.\mathsf{Vrfy}(\mathsf{vk}, \mathsf{ct}', \mathsf{sig}) = \perp$.
  3. Compute $\mathsf{sk}_{\mathsf{vk}} \leftarrow \prod_{i\in[v]} \mathsf{sk}'_{\sigma_i}$, where $\{\sigma_1, \ldots, \sigma_v\} \leftarrow \phi_{\boldsymbol{M}}(\mathsf{vk})$.
  4. Output $\mathsf{m}' \leftarrow \Pi'_{\mathsf{pre}}.\mathsf{Dec}(\mathsf{sk}_{\mathsf{vk}}, \mathsf{ct}')$.
- $\mathsf{ReKeyGen}(\mathsf{sk}_A, \mathsf{pk}_B) \to \mathsf{rk}_{A\to B}$:
  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 $(\mathsf{rk}^{(i\to j)}_{A\to B})_{i\in[u],j\in[u]} \leftarrow \Pi'_{\mathsf{pre}}.\mathsf{HReKeyGen}((\mathsf{sk}'_{A,i})_{i\in[u]}, (\mathsf{pk}'_{B,j})_{j\in[u]})$.
  3. Output $\mathsf{rk}_{A\to B} = (\mathsf{rk}^{(i\to j)}_{A\to B})_{i\in[u],j\in[u]}$.
- $\mathsf{ReEnc}(\mathsf{rk}_{A\to B}, \mathsf{ct}_A) \to \mathsf{ct}_B$:
  1. Parse $\mathsf{rk} = (\mathsf{rk}^{(i\to j)}_{A\to B})_{i\in[u],j\in[u]}$ and $\mathsf{ct}_A = (\mathsf{vk}_A, \mathsf{ct}'_A, \mathsf{sig}_A)$.
  2. Output $\perp$ if $\Pi_{\mathsf{ots}}.\mathsf{Vrfy}(\mathsf{vk}_A, \mathsf{ct}'_A, \mathsf{sig}_A) = \top$ holds.
  3. Generate $(\mathsf{vk}_B, \mathsf{sigk}_B) \leftarrow \Pi_{\mathsf{ots}}.\mathsf{KeyGen}(1^\lambda)$.
  4. Compute $\{\sigma^{(A)}_1, \ldots, \sigma^{(A)}_v\} \leftarrow \phi_{\boldsymbol{M}}(\mathsf{vk}_A)$ and $\{\sigma^{(B)}_1, \ldots, \sigma^{(B)}_v\} \leftarrow \phi_{\boldsymbol{M}}(\mathsf{vk}_B)$.
  5. Compute $\mathsf{rk}_{\mathsf{vk}_A\to\mathsf{vk}_B} \leftarrow \mathsf{ReKeyEval}((\mathsf{rk}^{(\sigma^{(A)}_i\to\sigma^{(B)}_i)}_{A\to B})_{i\in[v]})$.
  6. Compute $\mathsf{ct}'_B \leftarrow \Pi'_{\mathsf{pre}}.\mathsf{ReEnc}(\mathsf{rk}_{\mathsf{vk}_A\to\mathsf{vk}_B}, \mathsf{ct}'_A)$.
  7. Compute $\mathsf{sig}_B \leftarrow \Pi_{\mathsf{ots}}.\mathsf{Sign}(\mathsf{sigk}_B, \mathsf{ct}'_B)$.
  8. Output $\mathsf{ct}_B = (\mathsf{vk}_B, \mathsf{ct}'_B, \mathsf{sig}_B)$.

Due to the correctness of $\Pi'_{\mathsf{pre}}, \Pi_{\mathsf{ots}}$ and the key homomorphism of Π'_{pre}, the correctness of $\Pi^{\mathsf{kh}}_{\mathsf{pre}}$ holds, as follows:

Proposition 2 (Correctness of Π_{pre}). *If the PRE scheme Π'_{pre} is* correct *and* key homomorphism*, and the OTS scheme Π_{ots} is* strongly unforgeable*, then the resulting scheme $\Pi^{\mathsf{kh}}_{\mathsf{pre}}$ is* correct.

4.2 Security Proof

The following theorem shows the bounded CCA security of $\Pi_{\mathsf{pre}}^{\mathsf{kh}}$:

Theorem 2 (Security of Π_{pre}). *Suppose that the matrix $\boldsymbol{M} \in \{0,1\}^{u \times n}$ is a t-disjunct matrix and n_h is a number of honest users in (t,t)-CCA game. If the PRE scheme Π_{pre} is $\mathsf{KH\text{-}CPA}$ secure, and the OTS scheme Π_{ots} is* strongly unforgeable*, then the resulting PRE scheme $\Pi_{\mathsf{pre}}^{\mathsf{kh}}$ is (t,t)-CCA secure.*

Concretely, if there exists a PPT algorithm $\mathcal{A}$ against a (t,t)-CCA secure PRE scheme $\Pi_{\mathsf{pre}}^{\mathsf{kh}}$, then there exists a PPT algorithm $\mathcal{B}$ against a $\mathsf{KH\text{-}CPA}$ secure PRE scheme Π'_{pre} and a PPT algorithm $\mathcal{F}$ against strongly unforgeable *OTS scheme Π_{ots}, such that*

$$\mathsf{Adv}^{\mathrm{cca}}_{\Pi_{\mathsf{pre}}^{\mathsf{kh}},\mathcal{A}}(\lambda) \leq n_h u \cdot \mathsf{Adv}^{\text{kh-cpa}}_{\Pi'_{\mathsf{pre}},\mathcal{B}}(\lambda) + \mathsf{Adv}^{\mathrm{suf}}_{\Pi_{\mathsf{ots}},\mathcal{F}}(\lambda).$$

Proof. Let $\mathcal{A}$ be a PPT adversary against the PRE scheme $\Pi_{\mathsf{pre}}^{\mathsf{kh}}$. For a user $i^* \in [n]$ submitted to the challenge oracle $\mathtt{O.Challenge}$, let $\mathsf{ct}^* = (\mathsf{vk}^*, \mathsf{ct}'^*, \mathsf{sig}^*)$ denote the challenge ciphertext under pk_{i^*}.

In order to prove Theorem 2, we consider security games $\mathsf{Game}_0, \mathsf{Game}_1$. For $i \in \{0,1\}$, let W_i be the event that the experiment in Game_i outputs 1.
$\underline{\mathsf{Game}_0}$: The same game as (t,t)-CCA security game. Then, we have $\mathsf{Adv}^{\mathrm{cca}}_{\Pi_{\mathsf{pre}},\mathcal{A}}(\lambda) = |\Pr[W_0] - 1/2|$.

$\underline{\mathsf{Game}_1}$: The same game 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 $\mathsf{ct}_i = (\mathsf{vk}_i, \mathsf{ct}'_i, \mathsf{sig}_i)$), the experiment checks whether it holds that $\mathsf{vk}_i = \mathsf{vk}^*$, $\mathsf{ct}_i \neq \mathsf{ct}^*$, and $\Pi_{\mathsf{ots}}.\mathsf{Vrfy}(\mathsf{vk}_i, \mathsf{ct}'_i, \mathsf{sig}_i) = \top$. If so, this experiment aborts; otherwise, it returns the result of $\mathtt{O.Dec}(i, \mathsf{ct}_i)$ (resp., $\mathtt{O.ReEnc}(i, j, \mathsf{ct}_i)$).

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

Additionally, when $\mathcal{A}$ submits $(i^*, \mathsf{m}_0^*, \mathsf{m}_1^*)$, $\mathcal{F}$ chooses $b \xleftarrow{\$} \{0,1\}$ and computes ct'^* by following the procedure of $\mathsf{Enc}(\mathsf{pk}_{i^*}, \mathsf{m}_b^*)$. Then, this algorithm issues

ct'^* to the signing oracle $\mathtt{O.Sign}$ in the strong unforgeability game and obtains sig^*. $\mathcal{F}$ returns the challenge ciphertext $\mathsf{ct}^*_{i^*} = (\mathsf{vk}^*, \mathsf{ct}'^*, \mathsf{sig}^*)$. Finally, when $\mathcal{A}$ outputs $b' \in \{0,1\}$ and Bad has not occurred, $\mathcal{F}$ halts and aborts.

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

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

- $\mathtt{O.ReKeyGen}(A,B)$: $\mathcal{B}$ checks whether $A \in \mathcal{U}_{\mathtt{Honest}} \wedge B \in \mathcal{U}_{\mathtt{Corrupt}}$ holds. If so, this algorithms returns $\perp$; otherwise, it obtains $(\mathsf{rk}^{(i\to j)}_{A\to B})_{i\in[u],j\in[u]}$ by issuing (A,B) to the homomorphic re-encryption key generation oracle in the KH-CPA game, and returns $\mathsf{rk}_{A\to B} = (\mathsf{rk}^{(i\to j)}_{A\to B})_{i\in[u],j\in[u]}$.
- $\mathtt{O.ReEnc}(A,B,\mathsf{ct}_A)$: For $\mathsf{ct}_A = (\mathsf{vk}_A, \mathsf{ct}'_A, \mathsf{sig}_A)$, $\mathcal{B}$ returns $\perp$ if $B \in \mathcal{U}_{\mathtt{Corrupt}}$ holds and ct_A is its derivative of the challenge ciphertext. Otherwise, this algorithm does the following:
  1. If (A,B) is not queried to $\mathtt{O.ReKeyGen}$, compute $(\mathsf{rk}^{(i\to j)}_{A\to B})_{i\in[u],j\in[u]} \leftarrow \mathsf{HReKeyGen}((\mathsf{sk}_{A,i})_{i\in[u]}, (\mathsf{pk}_{B,j})_{j\in[u]})$.
  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 $\mathsf{vk}_A = \mathsf{vk}^*$, $\mathsf{ct}_A = \mathsf{ct}^*$, and $\Pi_{\mathsf{ots}}.\mathsf{Vrfy}(\mathsf{vk}_A, \mathsf{ct}'_A, \mathsf{sig}_A) = \top$.
  4. Return $\perp$ if $\Pi_{\mathsf{ots}}.\mathsf{Vrfy}(\mathsf{vk}_A, \mathsf{ct}'_A, \mathsf{sig}_A) = \perp$.
  5. Generate $(\mathsf{vk}_B, \mathsf{sigk}_B) \leftarrow \Pi_{\mathsf{ots}}.\mathsf{KeyGen}(1^\lambda)$.
  6. Compute $\{\sigma_1, \ldots, \sigma_v\} \leftarrow \phi_{\boldsymbol{M}}(\mathsf{vk}_B)$.
  7. If $A = i^*\ j^* \in \{\sigma_1, \ldots, \sigma_v\}$, then abort and output a random bit. Otherwise, compute $\mathsf{rk}_{\mathsf{vk}_A\to\mathsf{vk}_B} \leftarrow \sum_{i\in[v]} \mathsf{rk}^{(\sigma^{(A)}_i\to\sigma^{(B)}_i)}_{A\to B}$.
  8. Compute $\mathsf{ct}'_B \leftarrow \mathsf{ReEnc}(\mathsf{rk}_{\mathsf{vk}_A\to\mathsf{vk}_B}, \mathsf{ct}'_A)$.
  9. Compute $\mathsf{sig}_B \leftarrow \Pi_{\mathsf{ots}}.\mathsf{Sign}(\mathsf{sigk}_B, \mathsf{ct}'_B)$.
  10. Return $\mathsf{ct}_B = (\mathsf{vk}_B, \mathsf{ct}'_B, \mathsf{sig}_B)$.
- $\mathtt{O.Dec}(A,\mathsf{ct}_A)$: For $\mathsf{ct}_A = (\mathsf{vk}_A, \mathsf{ct}'_A, \mathsf{sig}_A)$, $\mathcal{B}$ returns $\perp$ if the challenge ciphertext is defined and ct_A is its derivative. Otherwise, this algorithm does the following:
  1. Abort and output a random bit if $A = i^* \wedge j^* \in \phi_{\boldsymbol{M}}(\mathsf{vk}_A)$ holds.

   2. Return $\perp$ if it holds that $\mathsf{vk}_A = \mathsf{vk}^*$, $\mathsf{ct}_A = \mathsf{ct}^*$, and $\Pi_{\mathsf{ots}}.\mathsf{Vrfy}(\mathsf{vk}_A, \mathsf{ct}'_A, \mathsf{sig}_A) = \top$.
   3. If $\Pi_{\mathsf{ots}}.\mathsf{Vrfy}(\mathsf{vk}_A, \mathsf{ct}'_A, \mathsf{sig}_A) = \perp$, return $\perp$.
   4. Compute $\mathsf{sk}_{\mathsf{vk}} \leftarrow \sum_{i\in[v]} \mathsf{sk}'_{\sigma_i}$, where $\{\sigma_1, \ldots, \sigma_v\} = \phi_{\boldsymbol{M}_A}(\mathsf{vk}_A)$.
   5. Return $\mathsf{m}' \leftarrow \Pi'_{\mathsf{pre}}.\mathsf{Dec}(\mathsf{sk}_{\mathsf{vk}}, \mathsf{ct}'_A)$.
- $\mathtt{O.Challenge}(i', \mathsf{m}_0^*, \mathsf{m}_1^*)$: $\mathcal{B}$ does the following:
   1. Abort and output a random bit if $i^* \neq i'$ holds.
   2. Obtain the ciphertext ct'^* by issuing $((i^*, j^*), \mathsf{m}_0^*, \mathsf{m}_1^*)$ to the challenge oracle in the KH-CPA game.
   3. Compute $\mathsf{sig}^* \leftarrow \Pi_{\mathsf{ots}}.\mathsf{Sign}(\mathsf{sigk}_{i^*}, \mathsf{ct}'^*)$.
   4. Return $\mathsf{ct}^*_{i^*} = (\mathsf{vk}^*, \mathsf{ct}'^*, \mathsf{sig}^*)$.

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. Additionally, let $W_{\mathcal{B}}$ denote the event that $\mathcal{B}$ outputs a bit $b' \in \{0,1\}$ such that $b = b'$. Then, $\Pr[W_{\mathcal{B}} \mid \mathsf{Abort}] = 1/2$ and $\Pr[\neg\mathsf{Abort}] \geq 1/(n_h u)$ hold. Hence, we have

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

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

From the above discussion, we obtain

$$\mathsf{Adv}^{\mathrm{cca}}_{\Pi_{\mathsf{pre}},\mathcal{A}}(\lambda) \leq n_h u \cdot \mathsf{Adv}^{\mathrm{kh\text{-}cpa}}_{\Pi'_{\mathsf{pre}},\mathcal{B}}(\lambda) + \mathsf{Adv}^{\mathrm{suf}}_{\Pi_{\mathsf{ots}},\mathcal{F}}(\lambda),$$

and complete the proof. □

5 Kyber-based PRE with Key Homomorphism

In order to instantiate our generic construction with compact ciphertexts, we give a Kyber-based PRE scheme $\Pi^{\mathsf{Kyber}}_{\mathsf{pre}}$ with key homomorphism and prove that $\Pi^{\mathsf{Kyber}}_{\mathsf{pre}}$ is KH-CPA secure. Concretely, the algorithms Setup, KeyGen, Enc, Dec of $\Pi^{\mathsf{Kyber}}_{\mathsf{pre}}$ are the same as those of Kyber [6], and then we add the algorithms ReKeyGen, ReEnc, HReKeyGen, ReKeyEval in order to guarantee the re-encryption functionality of PRE.

To describe this PRE scheme, we employ the following functions:

- The compression functions used in Kyber [6]:

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

We describe the PRE scheme $\Pi_{\mathsf{pre}}^{\mathsf{Kyber}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{ReKeyGen}, \mathsf{ReEnc})$ with $(\mathsf{HReKeyGen}, \mathsf{ReKeyEval})$, as follows:

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

  Output $\mathsf{pp} = (\lambda, \mu, N, N', q, \ell, \eta, k, d_t, d_u, d_v, \boldsymbol{A})$.
- $\mathsf{KeyGen}(\mathsf{pp}) \to (\mathsf{pk}, \mathsf{sk})$:
  1. Sample $(\boldsymbol{s}, \hat{\boldsymbol{s}}) \leftarrow \beta_\eta^k \times \beta_\eta^k$ and $(\boldsymbol{e}, \hat{\boldsymbol{e}}) \leftarrow \beta_\eta^k \times \beta_\eta^k$.
  2. Compute $\boldsymbol{t} \leftarrow \mathsf{Compress}_q(\boldsymbol{A}\boldsymbol{s} + \boldsymbol{e}, d_t)$ and $\hat{\boldsymbol{t}} \leftarrow \mathsf{Compress}_q(\boldsymbol{A}\hat{\boldsymbol{s}} + \hat{\boldsymbol{e}}, d_t)$.
  3. Output $\mathsf{pk} = (\boldsymbol{t}, \hat{\boldsymbol{t}})$ and $\mathsf{sk} = (\boldsymbol{s}, \hat{\boldsymbol{s}})$.
- $\mathsf{Enc}(\mathsf{pk}, \mathsf{m}) \to \mathsf{ct}$:
  1. Parse $\mathsf{pk} = (\boldsymbol{t}, \hat{\boldsymbol{t}})$.
  2. Compute $\boldsymbol{t} \leftarrow \mathsf{Decompress}(\boldsymbol{t}, d_t)$.
  3. Sample $(\boldsymbol{r}, \boldsymbol{e}_1, e_2) \leftarrow \beta_\eta^k \times \beta_\eta^k \times \beta_\eta$.
  4. Compute $\boldsymbol{u} \leftarrow \mathsf{Compress}_q(\boldsymbol{A}^\top \boldsymbol{r} + \boldsymbol{e}_1^\top, d_u)$.
  5. Compute $v \leftarrow \mathsf{Compress}_q(\boldsymbol{t}^\top \boldsymbol{r} + e_2 + \lceil \frac{q}{2} \rfloor \cdot \mathsf{m}, d_v)$.
  6. Output $\mathsf{ct} = (\boldsymbol{u}, v)$.
- $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) \to \mathsf{m}$:
  1. Parse $\mathsf{sk} = (\boldsymbol{s}, \hat{\boldsymbol{s}})$ and $\mathsf{ct} = (\boldsymbol{u}, v)$.
  2. Compute $\boldsymbol{u} \leftarrow \mathsf{Decompress}_q(\boldsymbol{u}, d_u)$ and $v \leftarrow \mathsf{Decompress}_q(v, d_v)$.
  3. Output $\mathsf{m} \leftarrow \mathsf{Compress}_q(v - \boldsymbol{s}^\top \boldsymbol{u}, 1)$.
- $\mathsf{ReKeyGen}(\mathsf{sk}_A, \mathsf{pk}_B) \to \mathsf{rk}_{A \to B}$:
  1. Parse $\mathsf{sk}_A = (\boldsymbol{s}_A, \hat{\boldsymbol{s}}_A)$ and $\mathsf{pk}_B = (\boldsymbol{t}_B, \hat{\boldsymbol{t}}_B)$.
  2. Compute $\hat{\boldsymbol{t}}_B \leftarrow \mathsf{Decompress}_q(\hat{\boldsymbol{t}}_B, d_t)$.
  3. Choose $\boldsymbol{R}_{A \to B,1}, \boldsymbol{R}_{A \to B,2} \leftarrow \beta_\eta^{k \times k\ell}$ and $\boldsymbol{r}_{A \to B,3} \leftarrow \beta_\eta^k$.

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

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

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

– $\mathsf{ReKeyEval}((\mathsf{rk}_{A\to B}^{(a_i\to b_i)})_{i\in[v]}) \to \mathsf{rk}_{A\to B}$:
  1. Parse $\mathsf{rk}_{A_i\to B_i} = (\boldsymbol{U}_{A\to B}, \boldsymbol{v}_{A_i\to B_i})$ for every $i \in [u]$.
  2. Compute $\boldsymbol{v}_{A\to B} \leftarrow \sum_{i\in[v]} \boldsymbol{v}_{A\to B}^{(a_i\to b_i)} \in R_q^k$.
  3. Output $\mathsf{rk}_{A\to B} = (\boldsymbol{U}_{A\to B}, \boldsymbol{v}_{A\to B})$.

Propositions 3 and 4 show the correctness and key-homomorphism of $\Pi_{\mathsf{pre}}^{\mathsf{Kyber}}$, respectively. Here, the proof of Proposition 3 is appeared in Appendix A.1.

Proposition 3 (Correctness). *Let* $\mathsf{pp} = (\lambda, \mu, N, N', q, \ell, \eta, k, d_t, d_u, d_v, \boldsymbol{A})$ *be a public parameter determined by running* $\mathsf{Setup}(1^\lambda)$ *and let* A, B *be distinct users. Then, the key-pairs of these users and a ciphertext under the A 's public key are defined as follows:*

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

Let $\boldsymbol{c}_{t,A}, \boldsymbol{c}_{t,B} \leftarrow \psi_{d_t}^k$*,* $\boldsymbol{c}_u \leftarrow \psi_{d_u}^k$*,* $c_v \leftarrow \psi_{d_v}$ *be distributed according to the following distribution* ψ_d^k *over R:*

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

Denote

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

Then, $\Pi_{\mathsf{pre}}^{\mathsf{Kyber}}$ *is* correct *with probability* δ.

Proposition 4. *Let* $\mathsf{pp} = (\lambda, \mu, N, N', q, \ell, \eta, k, d_t, d_u, d_v, \boldsymbol{A})$ *be a public parameter determined by running* $\mathsf{Setup}(1^\lambda)$*, and let* $A = \{a_1, \dots, a_v\} \subseteq [u]$ *and* $B = \{b_1, \dots, b_v\} \subseteq [u]$ *be two sets of distinct users. Then, the key-pairs of these users and a ciphertext are defined as follows:*

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

Let $\boldsymbol{c}_{t,A}, \boldsymbol{c}_{t,B} \leftarrow \psi_{d_t}^k$*,* $\boldsymbol{c}_u \leftarrow \psi_{d_u}^k$*,* $c_v \leftarrow \psi_{d_v}$ *be distributed according to the following distribution* ψ_d^k *over* R*:*

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

Let $(\boldsymbol{U}_{A\to B}, \boldsymbol{v}_{A\to B}^{(i\to j)})_{i\in[u], j\in[u]} \leftarrow \mathsf{HReKeyGen}((\mathsf{sk}_{A,i})_{i\in[u]}, (\mathsf{pk}_{B,i})_{j\in[u]})$ *and* $(\boldsymbol{U}_{A\to B}, \boldsymbol{v}_{A\to B}) \leftarrow \mathsf{ReKeyEval}((\mathsf{rk}_{A\to B}^{(a_i\to b_i)})_{i\in[v]})$*.*

Denote

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

for $\hat{c}_{d_u} \leftarrow \psi_{d_u}$ *and* $\hat{\boldsymbol{c}}_{t,b_i} \leftarrow \psi_{d_t}$ $(i \in [v])$*, where let* $\hat{\boldsymbol{s}}_B := \sum_{i\in[v]} \hat{\boldsymbol{s}}_{b_i}$*, let* $\hat{\boldsymbol{c}}_{t,B} := \sum_{i\in[v]} \hat{\boldsymbol{c}}_{t,b_i}$ *and for every* $i \in [v]$*,* $\boldsymbol{r}_{A\to B,3}^{(a_i\to b_i)}$ *is generated when running* $\mathsf{HReKeyGen}((\mathsf{sk}_i)_{i\in[u]}, (\mathsf{pk}_i)_{j\in[u]})$*.*

Then, the PRE scheme $\Pi_{\mathsf{pre}}^{\mathsf{Kyber}}$ *satisfies* re-encryption key homomorphic *with probability* δ*.*

Proof. We consider an arbitrary message $\mathsf{m} \in \mathcal{M}$ throughout the proof of Theorem 4. Recall that $\mathsf{pp} = (\lambda, \mu, N, N', q, \ell, \eta, k, d_t, d_u, d_v, \boldsymbol{A})$ is a public parameter determined by running $\mathsf{Setup}(1^\lambda)$, and let $A = \{a_1, \ldots, a_v\} \subseteq [u]$ and $B = \{b_1, \ldots, b_v\} \subseteq [u]$ be two sets of distinct users. For each $i \in [v]$, let $(\mathsf{pk}_{A,a_i}, \mathsf{sk}_{A,a_i}) = ((\boldsymbol{t}_{A,a_i}, \hat{\boldsymbol{t}}_{A,a_i}), (\boldsymbol{s}_{A,a_i}, \hat{\boldsymbol{s}}_{A,a_i}))$ (resp. $(\mathsf{pk}_{B,b_i}, \mathsf{sk}_{B,b_i}) = ((\boldsymbol{t}_{B,b_i}, \hat{\boldsymbol{t}}_{B,b_i}), (\boldsymbol{s}_{B,b_i}, \hat{\boldsymbol{s}}_{B,b_i})))$ be the key-pair of the user (A, a_i) (resp. the user (B, b_i)), where $\boldsymbol{t}_{A,a_i} = \mathsf{Compress}_q(\boldsymbol{A}\boldsymbol{s}_{A,a_i} + \boldsymbol{e}_i, d_t)$ and $\hat{\boldsymbol{t}}_{B,b_i} = \mathsf{Compress}_q(\boldsymbol{A}\hat{\boldsymbol{s}}_{B,b_i} + \hat{\boldsymbol{e}}_{B,b_i}, d_t)$.

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

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

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

Let $A := \{a_i\}_{i\in[v]}$ and $B := \{b_i\}_{i\in[v]}$. Then we define public keys $\mathsf{pk}_A, \mathsf{pk}_B$, as follows:

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

where

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

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

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

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

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

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

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

$$\boldsymbol{v}_{A\to B} := \sum_{i\in[v]} \boldsymbol{v}_{A\to B}^{(a_i\to b_i)} = \sum_{i\in[v]} \hat{\boldsymbol{t}}_{B,b_i}^\top \boldsymbol{R}_{A\to B,1} + \sum_{i\in[v]} \boldsymbol{r}_{A\to B,3}^{(a_i\to b_i)\top} - \sum_{i\in[v]} \mathsf{Powersof2}(\boldsymbol{s}_{A,a_i}^\top).$$

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

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

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

$$\begin{aligned}
\boldsymbol{u}_B &= \mathsf{Decompress}_q(\mathsf{Compress}_q(\boldsymbol{u}_A, d_u), d_u)\\
&= \boldsymbol{A}^\top \boldsymbol{R}_{A\to B,1} \cdot \mathsf{BitDecomp}(\boldsymbol{u}_A) + \boldsymbol{R}_{A\to B,2} \cdot \mathsf{BitDecomp}(\boldsymbol{u}_A) + \hat{\boldsymbol{c}}_{d_u};\\
v_B &= \mathsf{Decompress}_q(\mathsf{Compress}_q(v_A, d_v), d_v)\\
&= v_A - \boldsymbol{s}_A^\top \boldsymbol{u}_A\\
&\quad + \hat{\boldsymbol{s}}_B^\top \boldsymbol{A}^\top \boldsymbol{R}_{A\to B,1} + \left(\sum_{i\in[v]} \boldsymbol{r}_{A\to B,3}^{(a_i\to b_i)\top}\right) \mathsf{BitDecomp}(\boldsymbol{u}_A) + (\hat{\boldsymbol{e}}_B + \hat{\boldsymbol{c}}_{t,B})^\top \boldsymbol{R}_{A\to B,1} + \hat{c}_{d_v}
\end{aligned}$$

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

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

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

$$\begin{aligned}
w := &\left(\sum_{i\in[v]} \boldsymbol{r}_{A\to B,3}^{(a_i\to b_i)\top}\right) \mathsf{BitDecomp}(\boldsymbol{u}_A) + (\hat{\boldsymbol{e}}_B + \hat{\boldsymbol{c}}_{t,B})^\top \boldsymbol{R}_{A\to B,1} + \hat{c}_{d_v} \\
&- \hat{\boldsymbol{s}}_B^\top \boldsymbol{R}_{A\to B,2} \cdot \mathsf{BitDecomp}(\boldsymbol{u}_A) + \hat{\boldsymbol{s}}_B^\top \hat{\boldsymbol{c}}_{d_u}.
\end{aligned}$$

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

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

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

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

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

Theorem 3. *If the* $\mathsf{MLWE}_{k+1,k\ell,\eta}$ *assumption holds, the proposed PRE scheme* $\Pi_{\mathsf{pre}}^{\mathsf{Kyber}}$ *is* $\mathsf{KH\text{-}CPA}$ *secure.*

Concretely, suppose that n *is the total number of users,* n_h *is the number of honest users,* q_{rk} *is the maximum number of queries issued to the re-encryption key generation oracle, and* u *is the number of key-pairs given to a user. If there exists a PPT adversary* $\mathcal{A}$ *against the* $\mathsf{KH\text{-}CPA}$ *security of* $\Pi_{\mathsf{pre}}^{\mathsf{Kyber}}$*, then there exists a PPT algorithm* $\mathcal{B}$ *against the* $\mathsf{MLWE}_{k+1,k\ell,\eta}$ *problem, such that*

$$\mathsf{Adv}_{\Pi_{\mathsf{pre}}^{\mathsf{Kyber}},\mathcal{A}}^{\text{kh-cpa}}(\lambda) \leq n_h(q_{rk}ku + 2u + 1) \cdot \mathsf{Adv}_{k+1,k\ell,\eta}^{\text{mlwe}}(\mathcal{B}).$$

The proof of Theorem 3 is appeared in Appendix A.2.

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

References

1. G. Ateniese, K. Benson, and S. Hohenberger. Key-private proxy re-encryption. In *CT-RSA*, volume 5473 of *Lecture Notes in Computer Science*, pages 279–294. Springer, 2009.
2. G. Ateniese, K. Fu, M. Green, and S. Hohenberger. Improved proxy re-encryption schemes with applications to secure distributed storage. In *NDSS*. The Internet Society, 2005.
3. G. Ateniese, K. Fu, M. Green, and S. Hohenberger. Improved proxy re-encryption schemes with applications to secure distributed storage. *ACM Trans. Inf. Syst. Secur.*, 9(1):1–30, 2006.
4. M. Blaze, G. Bleumer, and M. Strauss. Divertible protocols and atomic proxy cryptography. In *EUROCRYPT*, volume 1403 of *LNCS*, pages 127–144. Springer, 1998.
5. D. Boneh, R. Canetti, S. Halevi, and J. Katz. Chosen-ciphertext security from identity-based encryption. *SIAM J. Comput.*, 36(5):1301–1328, 2007.
6. J. W. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanck, P. Schwabe, and D. Stehlé. CRYSTALS - kyber: a cca-secure module-lattice-based KEM. *IACR Cryptol. ePrint Arch.*, page 634, 2017.
7. R. Canetti and S. Hohenberger. Chosen-ciphertext secure proxy re-encryption. In *CCS*, pages 185–194. ACM, 2007.
8. A. Cohen. What about bob? the inadequacy of CPA security for proxy reencryption. In *Public Key Cryptography (2)*, volume 11443 of *LNCS*, pages 287–316. Springer, 2019.
9. R. Cramer, G. Hanaoka, D. Hofheinz, H. Imai, E. Kiltz, R. Pass, A. Shelat, and V. Vaikuntanathan. Bounded cca2-secure encryption. In *ASIACRYPT*, volume 4833 of *Lecture Notes in Computer Science*, pages 502–518. Springer, 2007.
10. A. Davidson, A. Deo, E. Lee, and K. Martin. Strong post-compromise secure proxy re-encryption. In *ACISP*, volume 11547 of *LNCS*, pages 58–77. Springer, 2019.
11. Y. Dodis, J. Katz, S. Xu, and M. Yung. Key-insulated public key cryptosystems. In *EUROCRYPT*, volume 2332 of *Lecture Notes in Computer Science*, pages 65–82. Springer, 2002.
12. D.-Z. Du and F. K. Hwang. *Combinatorial Group Testing and Its Applications (2nd Edition)*, volume 12 of *Series on Applied Mathematics*. World Scientific, 2000.
13. J. Duman, K. Hövelmanns, E. Kiltz, V. Lyubashevsky, and G. Seiler. Faster lattice-based kems via a generic fujisaki-okamoto transform using prefix hashing. In *CCS*, pages 2722–2737. ACM, 2021.
14. X. Fan and F. Liu. Proxy re-encryption and re-signatures from lattices. In *ACNS*, volume 11464 of *LNCS*, pages 363–382. Springer, 2019.
15. S. Goldwasser, A. B. Lewko, and D. A. Wilson. Bounded-collusion IBE from key homomorphism. In *TCC*, volume 7194 of *Lecture Notes in Computer Science*, pages 564–581. Springer, 2012.
16. P. Grubbs, V. Maram, and K. G. Paterson. Anonymous, robust post-quantum public key encryption. In *EUROCRYPT (3)*, volume 13277 of *LNCS*, pages 402–432. Springer, 2022.
17. A. Ivan and Y. Dodis. Proxy cryptography revisited. In *NDSS*. The Internet Society, 2003.
18. V. Maram and K. Xagawa. Post-quantum anonymity of kyber. In *Public Key Cryptography (1)*, volume 13940 of *LNCS*, pages 3–35. Springer, 2023.

19. L. T. Phong, L. Wang, Y. Aono, M. H. Nguyen, and X. Boyen. Proxy re-encryption schemes with key privacy from LWE. *IACR Cryptol. ePrint Arch.*, page 327, 2016.
20. Y. Polyakov, K. Rohloff, G. Sahu, and V. Vaikuntanathan. Fast proxy re-encryption for publish/subscribe systems. *ACM Trans. Priv. Secur.*, 20(4):14:1–14:31, 2017.
21. S. Tessaro and D. A. Wilson. Bounded-collusion identity-based encryption from semantically-secure public-key encryption: Generic constructions with short ciphertexts. In *Public Key Cryptography*, volume 8383 of *LNCS*, pages 257–274. Springer, 2014.
22. K. Xagawa. Anonymity of NIST PQC round 3 kems. In *EUROCRYPT (3)*, volume 13277 of *Lecture Notes in Computer Science*, pages 551–581. Springer, 2022.

A Omitted Proofs

A.1 Proof of Proposition 3

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

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

First we show the encryption-correctness of $\Pi_{\mathsf{pre}}^{\mathsf{Kyber}}$. Then, the public value $\boldsymbol{t}_A$ is represented as

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

for some value $\boldsymbol{c}_{t,A} \in R^k$.

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

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

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

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

for some $c_{v,A} \in R$.

Then, we have

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

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

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

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

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

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

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

Next, we show the $\mathsf{re\text{-}encryption\text{-}correctness}$ of $\Pi_{\mathsf{pre}}^{\mathsf{Kyber}}$. For convenience, we also employ the above value of $(\boldsymbol{t}_A, \boldsymbol{u}_A, v_A)$. 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 represented as follows:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

From the discussion above, we complete the proof of the correctness of the proposed PRE scheme $\Pi_{\mathsf{pre}}^{\mathsf{Kyber}}$. □

A.2 Proof of Theorem 3

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

$\underline{\mathsf{Game}_0}$: The original KH-CPA security game. Then, we have $\mathsf{Adv}_{\Pi_{\mathsf{pre}},\mathcal{A}}^{\text{kh-cpa}}(\lambda) = |\Pr[W_0] - 1/2|$.

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

$\underline{\mathsf{Game}_1^{(\kappa)}}$: The same game as $\mathsf{Game}_1^{(\kappa-1)}$ except that, for every $i \in [u]$, $\hat{\boldsymbol{t}}_{\kappa,i} = \mathsf{Compress}_q(\boldsymbol{A}\hat{\boldsymbol{s}}_{\kappa,i}+\hat{\boldsymbol{e}}_{\kappa,i}, d_t)$ is replaced by a uniformly random $\hat{\boldsymbol{t}}_{\kappa,i} = \mathsf{Compress}_q(\hat{\boldsymbol{b}}_{\kappa,i}, d_t)$ (where $\hat{\boldsymbol{b}}_{\kappa,i} \in R_q^k$ is chosen uniformly at random), when generating the public key $\mathsf{pk}_{\kappa,i} = (\boldsymbol{t}_{\kappa,i}, \hat{\boldsymbol{t}}_{\kappa,i})$ 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 $\hat{\boldsymbol{s}}_{\kappa,i}$ is not necessary to simulate the environments of $\mathcal{A}$ in both $\mathsf{Game}_1^{(\kappa-1)}$ and $\mathsf{Game}_1^{(\kappa)}$. By using $\mathcal{A}$'s output, $\mathcal{B}_1^{(\kappa)}$ can distinguish between a $\mathsf{MLWE}_{k,k,\eta}$ sample and a uniformly random one, in the straightforward way. Hence, we have $\left|\Pr[W_1^{(\kappa-1)}] - \Pr[W_1^{(\kappa)}]\right| \leq u{\cdot}\mathsf{Adv}_{k,k,\eta}^{\mathrm{mlwe}}(\mathcal{B}_1^{(\kappa)})$ for every $\kappa \in [n_h]$, due to the union bound on $i \in [u]$.

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

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

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

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

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

There exists a PPT algorithm $\mathcal{B}_3^{(\kappa)}$ against the $\mathsf{MLWE}_{k,k,\eta}$ problem. Since $\boldsymbol{s}_{\kappa,i}$ 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 $\left|\Pr[W_3^{(\kappa-1)}] - \Pr[W_3^{(\kappa)}]\right| \leq \mathsf{Adv}_{k,k,\eta}^{\mathrm{mlwe}}(\mathcal{B}_3^{(\kappa)})$, due to the union bound on $i \in [u]$.

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

Since the secret key $(\boldsymbol{s}_{\kappa,i}, \hat{\boldsymbol{s}}_{\kappa,i})$ for every $\kappa \in [n_h]$ and $i \in [u]$ is not used in both $\mathsf{Game}_3^{(n_h)}$ and Game_4, it is possible to simulate the environments of $\mathcal{A}$ in these games and construct a PPT algorithm $\mathcal{B}_4^{(i^*)}$ against the $\mathsf{MLWE}_{k,k,\eta}$ problem. Hence, we have $\left|\Pr[W_3^{(n_h)}] - \Pr[W_4]\right| \leq n_h \cdot \mathsf{Adv}_{k+1,k,\eta}^{\mathrm{mlwe}}(\mathcal{B}_4^{(i^*)})$. Furthermore, $\Pr[W_4] = 1/2$ holds since $\mathcal{A}$'s view is independent of $b \in \{0,1\}$ in Game_4.

From the discussion above, we obtain

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

where $\mathcal{B}$ is a PPT algorithm against the $\mathsf{MLWE}_{k+1,k\ell,\eta}$ problem, such that it holds that $\mathsf{Adv}_{k+1,k\ell,\eta}^{\mathrm{mlwe}}(\mathcal{B}_i^{(L)}) < \mathsf{Adv}_{k+1,k\ell,\eta}^{\mathrm{mlwe}}(\mathcal{B})$ for all $i \in [4]$ and all $L \in \mathcal{U}_{\texttt{Honest}}$. This completes the proof. □