Various Proxy Re-Encryption Schemes from Lattices

Xiong Fan * Feng-Hao Liu †

Abstract

Proxy re-encryption (PRE) was introduced by Blaze, Bleumer and Strauss [Eurocrypt '98]. Basically, PRE allows a semi-trusted proxy to transform a ciphertext encrypted under one key into an encryption of the same plaintext under another key, without revealing the underlying plaintext. Since then, many interesting applications have been explored, and many constructions in various settings have been proposed. In 2007, Cannetti and Honhenberger [CCS '07] defined a stronger notion – CCA-security and construct a bi-directional PRE scheme. Later on, several work considered CCA-secure PRE based on bilinear group assumptions. Very recently, Kirshanova [PKC '14] proposed the first single-hop CCA1-secure PRE scheme based on learning with errors (LWE) assumption.

In this work, we first point out a subtle but serious mistake in the security proof of the work by Kirshanova. This reopens the direction of lattice-based CCA1-secure constructions, even in the single-hop setting. Then we propose a new LWE-based single-hop CCA1-secure PRE scheme. Finally, we extend the construction to support multi-hop re-encryptions for different levels of security under different settings.

*Cornell University, xfan@cs.cornell.edu.

†Florida Atlantic University, fenghao.liu@fau.edu.

1 Introduction

Proxy re-encryption (PRE) allows a (semi-trusted) proxy to transform an encryption of m under Alice's public key into another encryption of the same message under Bob's public key. The proxy, however, cannot learn the underlying message m, and thus both parties' privacy can be maintained. This primitive (and its variants) have various applications ranging from encrypted email forwarding [BBS98], securing distributed file systems [AFGH05], to digital rights management (DRM) systems [Smi05]. In addition application-driven purposes, various works have shown connections between re-encryption (and its variants) with other cryptographic primitives, such as program obfuscation [HRsV07, CCV12, CCL$^+$14] and fully-homomorphic encryption [CLTV, ABF$^+$13]. Thus studies along this line are both important and interesting for theory and practice.

The concept of proxy re-encryption (PRE) was introduced by Blaze, Bleumer, and Strauss [BBS98], who also gave the first construction of a CPA (i.e. chosen-plaintext attacks) secure bi-directional multi-hop[1] PRE scheme under the Decisional Diffie-Hellman assumption. It was explicitly left as an interesting open problem [BBS98] to construct a uni-directional PRE, which apparently provides more find-grained security. Later Ateniese, Fu, Green and Hohenberger [AFGH05] constructed the first CPA secure *uni-directional* scheme based on bilinear maps, yet their construction can only support a single-hop re-encryption. Hohenberger et al. [HRsV07] and Chase et al. [CCV12] used an obfuscation-based approach and constructed CPA secure uni-directional single-hop PRE scheme (and its variants). Recently, Chase et al. [CCL$^+$14], using the obfuscation-based approach, constructed the first CPA secure uni-directional multi-hop PRE scheme based on lattices assumptions.

As argued that CPA security can be insufficient for some useful scenarios, Canetti and Hohenberger [CH07] considered a natural stronger security notion — chosen-ciphertext attacks (CCA) security where the adversary has access to a decryption oracle. Intuitively, this security notion guarantees that the underlying message of the challenge ciphertext remains hidden even if the adversary can somehow obtain decryptions of other ciphertexts. They give a meaningful security formulation of CCA secure PRE, and then constructed the first CCA-secure bidirectional multi-hop PRE scheme. Later, Shao et al. [SCL10] constructed a CCA-secure uni-directional single-hop PRE, and Chow et al. [CWYD10] proposed another CCA-secure uni-directional scheme in random oracle model. Libert and Vergnaud [LV08] improved the result by constructing a CCA uni-directional single-hop PRE without random oracles, and this remains the state of the art of the current construction (for the setting of uni-directional CCA PRE under the definition of [CH07]). We note that it is unclear how to extend security of the previous obfuscation-approach [HRsV07, CCV12, CCL$^+$14] (that are only CPA-secure) to the CCA setting. One particular technical challenge is that the re-encryption key output by the simulator might be distinguishable given the CCA decryption oracle, and thus the previous security analyses cannot go through.

For CPA security, our understanding is quite well — we know how to construct PRE schemes that are uni-directional and multi-hop in the standard model. However, for CCA security, our understanding is much limited in the following sense. Obviously, there is no known scheme that achieves both uni-directional and multi-hop at the same time. Moreover, all currently known constructions [BBS98, AFGH05, CH07, LV08, SCL10, CWYD10] are based on Diffie-Hellman-typed assumptions, either in bilinear groups or not. Yoshinori Aono et al. [ABPW13] showed how to construct CCA-secure PRE scheme in the random oracle model. Very recently, Kirshanova [Kir14] proposed a single-hop construction based on lattices, and argued that it is CCA1 secure[2]. However, after a careful examination of her security proof, we found a subtle mistake in the security

[1] Basically a bi-directional scheme allows the proxy, given a re-encryption key $\mathsf{rk}_{A,B}$, to transform $\mathsf{Enc}(\mathsf{pk}_A, m)$ to $\mathsf{Enc}(\mathsf{pk}_B, m)$ and vice versa, while a unidirectional scheme has a more fine-grained access control: by giving a unidirectional re-encryption key $\mathsf{rk}_{A\to B}$, the proxy can only transform ciphertexts in that direction.

On the other hand, a single-hop scheme only supports re-encryption once, i.e. a re-encrypted ciphertext cannot be further re-encrypted; a multi-hop scheme supports re-encryption over re-encryption multiple times.

[2] CCA1 security is weaker in the sense that the attacker does not have the decryption oracle after receiving the challenge ciphertext.

proof, and thus how to construct a lattice-based PRE that achieves CCA1-security, (even for the single-hop case) remains open.

In this paper, we revisit the study of lattice-based PRE constructions. In particular, we make contributions in the following three folds:

- First as discussed above, we point out a subtle mistake in the security proof of the work [Kir14]. We also argue that it cannot be easily fixed.
- Second, we propose a basic construction of PRE scheme that is single-hop CCA*-secure[3] under the learning with errors (LWE) assumption.
- Third, we prove that the basic construction, with a slight modification, can be extended to the multi-hop setting and achieve CCA* security for tree-based networks. On the other hand, the basic construction, with another slight modification, can achieve multi-hop for any acyclic-graph-based network, yet fallbacks to the CPA security.

To summarize, we give a unified framework to construct PRE with three modes that achieves different levels of security and functionalities: (1) single-hop and CCA* security; (2) multi-hop for tree-based networks and CCA* security; (3) multi-hop for arbitrary acyclic networks and CPA security.

1.1 Technique Highlights

In the following, we highlight our technical ideas for the three contributions as described above.

Part I: the subtle mistake in the work [Kir14]. For clarification of exposition, we first present the main idea of the construction [Kir14]. Then we will point out where the subtlety is and explain why the problem cannot be easily fixed.

Basically, the encryption mechanism can be regarded as an extension of CCA-secure public key encryption scheme of the work [MP12]. For concreteness, we consider two users: User one has public key $\mathsf{pk}_1 = (\mathbf{A}_0, \mathbf{A}_1, \mathbf{A}_2, \mathbf{H})$, and User two has public key $\mathsf{pk}_2 = (\mathbf{A}'_0, \mathbf{A}'_1, \mathbf{A}'_2, \mathbf{H}')$, where each public key consists of four matrices. The secret key of User one consists of matrices $\mathbf{R}_1, \mathbf{R}_2$ satisfying $\mathbf{A}_1 = -\mathbf{A}_0\mathbf{R}_1, \mathbf{A}_2 = -\mathbf{A}_0\mathbf{R}_2$, and it is similar for the case of User two. We note that the readers here do not need to worry about the dimensions, but just keep in mind the structure: each user has four matrices in the public key, and the secret key consists of two "short" matrices $\mathbf{R}_1, \mathbf{R}_2$. To encrypt under pk_1, we consider an encryption matrix $\mathbf{A}_u = [\mathbf{A}_0|\mathbf{A}_1 + \mathbf{HG}|\mathbf{A}_2 + \mathbf{H}_u\mathbf{G}]$, where $\mathbf{H}_u$ is a random invertible matrix (as a tag to the ciphertext), then encrypt messages using the dual-Regev style encryption [GPV08], i.e. $\mathsf{ct} = \boldsymbol{s}^T\mathbf{A}_u + \boldsymbol{e} + \mathsf{encode}(m)$. Similarly, we can encrypt under pk_2 with the same structure.

To generate a re-encryption key from User 1 to User 2, the work [Kir14] considers a short matrix $\mathbf{X}$ (defined as below) that satisfies the following relation:

$$[\mathbf{A}_0|\mathbf{A}_1 + \mathbf{HG}|\mathbf{A}_2 + \mathbf{H}_u\mathbf{G}] \begin{bmatrix} \mathbf{X}_{00} & \mathbf{X}_{01} & \mathbf{X}_{02} \\ \mathbf{X}_{10} & \mathbf{X}_{11} & \mathbf{X}_{12} \\ 0 & 0 & \mathbf{I} \end{bmatrix} = [\mathbf{A}'_0|\mathbf{A}'_1 + \mathbf{H}'\mathbf{G}|\mathbf{A}'_2 + \mathbf{H}_u\mathbf{G}].$$

In particular, for the last column of the re-encryption key matrix, it holds that

$$[\mathbf{A}_0|\mathbf{A}_1 + \mathbf{HG}] \begin{bmatrix} \mathbf{X}_{02} \\ \mathbf{X}_{12} \end{bmatrix} = \mathbf{A}'_2 - \mathbf{A}_2. \tag{1}$$

[3] We define a meaningful notion CCA* that lies between CCA1 and CCA2. See Remark 2.3 for further discussions about the notion

It is not hard to see that $\mathsf{ct} \cdot \mathbf{X} = \boldsymbol{s}^T \cdot \mathbf{A}'_u + \tilde{\boldsymbol{e}} + \mathsf{encode}(m)$, a ciphertext of m under pk_2, so the correctness property is guaranteed.

To prove security, the work [Kir14] uses a standard reduction argument based on the LWE assumption: suppose there exists an adversary that can break the PRE scheme, then there exists a reduction, with oracle access to the adversary, who can break the underlying LWE assumption. For this type of proofs, typically the reduction needs to embed the hard instance (LWE instance for this case), then simulates a scheme (PRE) to the adversary, and finally the reduction can use the adversary to break the underlying hardness assumption. It is **crucially important** that the simulated scheme cannot be distinguished by the adversary; otherwise, the adversary can always output $\perp$ if he detects the scheme is different from the real scheme, and such adversary is useless to the reduction. The security proof in the work [Kir14] missed this point. At a high level, her reduction simulated a PRE scheme that *can* be distinguishable by the adversary easily, so the whole argument breaks down. Below we further elaborate on the details.

For simplicity we consider a simple case where there are only two honest users, Users one and two and the adversary only gets one re-encryption key from User one to User two. The challenge ciphertext comes from an encryption of User one, i.e. pk_1. For such case, the reduction of the work [Kir14] pre-selects a tag matrix $\mathbf{H}_{u^*}$ (for the challenge ciphertext), matrices $\mathbf{R}_1^*, \mathbf{R}_2^*$, and then embeds an LWE instance $\mathbf{A}^*$ in the encryption matrix: $\mathbf{A}_u^* = [\mathbf{A}^* | -\mathbf{A}^*\mathbf{R}_1^* | -\mathbf{A}^*\mathbf{R}_2^* + (\mathbf{H}_u - \mathbf{H}_{u^*})\mathbf{G}]$. In this case, the reduction sets $\mathsf{pk}_1 = (\mathbf{A}_0, \mathbf{A}_1, \mathbf{A}_2, \mathbf{H})$ to be $(\mathbf{A}^*, -\mathbf{A}^*\mathbf{R}_1^* - \mathbf{H}^*\mathbf{G}, -\mathbf{A}^*\mathbf{R}_2^* - \mathbf{H}_{u^*}\mathbf{G}, \mathbf{H}^*)$ for some random invertible $\mathbf{H}^*$.

To generate re-encryption key from the challenge user to other users, (in this case from User one to User two), the reduction first pre-samples small matrices $\mathbf{X}_{00}, \mathbf{X}_{01}, \mathbf{R}'_1, \mathbf{R}'_2$, and a random invertible matrix $\mathbf{H}'$. Then it computes:

$$\mathbf{A}'_0 = [\mathbf{A}^* | -\mathbf{A}^*\mathbf{R}_1^*] \begin{bmatrix} \mathbf{X}_{00} \\ \mathbf{X}_{10} \end{bmatrix}$$

$$\mathbf{A}'_1 = [\mathbf{A}^* | -\mathbf{A}^*\mathbf{R}_1^*] \begin{bmatrix} \mathbf{X}_{00} \\ \mathbf{X}_{10} \end{bmatrix} \cdot \mathbf{R}'_1, \qquad \mathbf{A}'_2 = [\mathbf{A}^* | -\mathbf{A}^*\mathbf{R}_1^*] \begin{bmatrix} \mathbf{X}_{00} \\ \mathbf{X}_{10} \end{bmatrix} \cdot \mathbf{R}'_2$$

Then the reduction sets

$$\mathsf{pk}_2 = (\mathbf{A}'_0, \mathbf{A}'_1, \mathbf{A}'_2, \mathbf{H}'), \quad \mathsf{rk}_{1\to 2} = \begin{bmatrix} \begin{pmatrix} \mathbf{X}_{00} \\ \mathbf{X}_{10} \end{pmatrix} & \begin{pmatrix} \mathbf{X}_{00} \\ \mathbf{X}_{10} \end{pmatrix} \mathbf{R}'_1 & \begin{pmatrix} \mathbf{X}_{00} \\ \mathbf{X}_{10} \end{pmatrix} \mathbf{R}'_2 \\ 0 & 0 & \mathbf{I} \end{bmatrix}$$

generated as above. Then obviously the matrices $\mathbf{A}'_1, \mathbf{A}'_2$ can be expressed as $\mathbf{A}'_1 = \mathbf{A}'_0\mathbf{R}'_1, \mathbf{A}'_2 = \mathbf{A}'_0\mathbf{R}'_2$, where $\mathbf{R}'_1, \mathbf{R}'_2$ are small matrices and still act as secret key for User two. Therefore, the reduction can still use the same algorithm in the real scheme to answer decryption queries for User two.

However, if $\mathbf{A}'_2$ is generated in this way, then it is easy to check and compare with Equation (1):

$$[\mathbf{A}_0 | \mathbf{A}_1 + \mathbf{H}\mathbf{G}] \begin{bmatrix} \mathbf{X}_{02} \\ \mathbf{X}_{12} \end{bmatrix} = [\mathbf{A}^* | -\mathbf{A}^*\mathbf{R}_1^*] \begin{bmatrix} \mathbf{X}_{00} \\ \mathbf{X}_{10} \end{bmatrix} \cdot \mathbf{R}'_2 \neq \mathbf{A}'_2 - \mathbf{A}_2. \tag{2}$$

This means adversary, given the simulated $\mathsf{pk}_1, \mathsf{pk}_2, \mathsf{rk}_{1\to 2}$, adversary can easily tell whether they are from the real scheme or the simulated scheme. Thus, the security proof in this way [Kir14] is not correct.

A straightforward fix would be to set $\mathbf{A}'_2 = [\mathbf{A}^* | -\mathbf{A}^*\mathbf{R}_1^*] \begin{bmatrix} \mathbf{X}_{00} \\ \mathbf{X}_{10} \end{bmatrix} \cdot \mathbf{R}'_2 + \mathbf{A}_2 = \mathbf{A}'_0 \cdot \mathbf{R}'_2 + \mathbf{A}_2$ so that Equations (1) and (2) match. But in this way it is not clear how to express $\mathbf{A}_2$ as $\mathbf{A}'_0\mathbf{R}$ for some small matrix $\mathbf{R}$, because it is not clear how to express $\mathbf{A}_2$ as $\mathbf{A}'_0\tilde{\mathbf{R}}$ for some small $\tilde{\mathbf{R}}$. Note that $\mathbf{R}$ serves as the secret key of pk_2 to simulate decryption queries. Consequently, it is not clear how the reduction can answer decryption queries as the previous approach. It seems that this construction/proof is facing a dilemma: either the reduction can answer the decryption queries but the re-encryption key can be distinguished, or the reduction can generate an indistinguishable re-encryption key but cannot answer the decryption queries.

Part II: our new construction for single-hop PRE. To overcome the dilemma, we consider a new matrix structure: the setup algorithm outputs a public matrix $\mathbf{A}$, and each user extends the previous matrix structure to be $\mathbf{A}_u = [\mathbf{A}|\mathbf{A}_1 + \mathbf{H}\mathbf{G}|\mathbf{A}_2 + \mathbf{H}_u\mathbf{G}]$, where $\mathbf{A}_1 = -\mathbf{A}\mathbf{R}_1$, $\mathbf{A}_2 = -\mathbf{A}\mathbf{R}_2$ and the matrices $\mathbf{R}_1, \mathbf{R}_2$ are the corresponding secret key. The shared matrix $\mathbf{A}$ offers a significant advantage for the simulation: the reduction can embed the LWE instance $\mathbf{A}^*$ as the public shared matrix, and then sets

$$\mathbf{A}'_2 = [\mathbf{A}^*| - \mathbf{A}^*\mathbf{R}_1^*] \begin{bmatrix} \mathbf{X}_{00} \\ \mathbf{X}_{10} \end{bmatrix} \cdot \mathbf{R}'_2 - \mathbf{A}^*\mathbf{R}_2^*.$$

This allows the reduction to express $\mathbf{A}'_2$ as $\mathbf{A}^*\mathbf{R}$ for some small and known matrix $\mathbf{R}$. Then the reduction can use this to simulate the decryption queries, while the Equation (1) will match for the real scheme and the simulated scheme. We present the detailed scheme and analysis in Section 3.

Part III: extension to multi-hop PRE. We further observe that the matrix structure in our construction can be extended to the multi-hop case with slight modification. Interestingly, our scheme itself can support general network structures (for functionalities), yet our security proof (for CCA security), however, requires the structure of tree-structured networks (i.e. the adversary can only query re-encryption keys that form a tree among the users). If the adversary's queries form a general graph, then security of our scheme becomes unclear: we are not able to prove security under the current techniques, but there is no known attack, either. We leave it as an interesting open problem to determine whether our construction is secure under general network structures.

A technical reason for this phenomenon comes from the order of sampling for the simulation. We give a simple example for illustration: let there be three parties in the network, Users one, two, and three. It is easy for the reduction to simulate in the following order pk_1, $\mathsf{rk}_{1\to2}$, pk_2, $\mathsf{rk}_{2\to3}$, and then pk_3 *without* knowing a trapdoor of the LWE instance $\mathbf{A}^*$. The reduction, however, would get stuck if he needs to further generate $\mathsf{rk}_{1\to3}$, which should be consistent with the already sampled pk_1 and pk_3. We recall that the reduction is able to check whether $\mathsf{rk}_{1\to3}$ is consistent with pk_1 and pk_3 in both the real scheme and the simulated scheme (as Equation (1)). Thus, the reduction must simulate such consistency as the real scheme. Even though there are techniques from the Ring-LWE [LPR10, GGH13] that allows sampling in the *reverse* order of pk_3, $\mathsf{rk}_{2\to3}$, pk_2, $\mathsf{rk}_{1\to2}$, pk_1, it does not help to solve the problem because the reduction still does not know how to generate $\mathsf{rk}_{1\to3}$ after pk_1 and pk_3 are sampled, without a trapdoor of $\mathbf{A}^*$.

On the other hand, if the adversary does not have access to the decryption oracle, then we can slightly modify the construction to achieve multi-hop PRE for arbitrary acyclic graphs. We use an idea from the work [CCL+14] to blur the relation between public keys and re-encryption keys. That is, the adversary can *no longer* check whether whether $\mathsf{rk}_{1\to3}$ is consistent with pk_1 and pk_3! This gives an extra power to the reduction that it can simulate re-encryption keys without knowing the trapdoors at all. We note that this technique does not extend to the CCA setting because the simulation strategy can be detected by the CCA decryption oracle. Thus, we construct a PRE scheme that falls back to the CPA security but achieves multi-hop for arbitrary acyclic graphs.

1.2 Related Works

In the following Table 1 we compare our contributions with previous work as described before.

In addition to the schemes proposed in the above mentioned work, recently Nuñez et al. [NAL15] proposed a nice framework capturing more fine-grained CCA-security of PRE, corresponding to the adversary's ability in the security experiment. Our multi-hop CCA*-secure PRE construction described in Appendix A can be categorized as $\mathsf{CCA}_{1,2}$ model in their paper regarding a special structure (trees). Also as Nuñez et al. pointed out, Kirshanova's scheme [Kir14] suffered from another attack by querying the re-encryption oracle with

Work	Direction	Hop	Assumption	Security	ROM
Blaze et al. [BBS98]	bi-directional	multi	DDH	PRE-CPA	no
Ateniese et al. [AFGH05]	uni-directional	single	eDBDH	PRE-CPA	no
Canetti el al. [CH07]	bi-directional	multi	DBDH	PRE-CCA	no
Libert et al. [LV08]	uni-directional	single	3-wDBDHI	PRE-RCCA	no
Shao et al. [SCL10]	uni-directional	single	3-QDBDH	PRE-CCA	no
Chow et al. [CWYD10]	uni-directional	single	CDH	PRE-CCA	yes
Hohenberger et al. [HRsV07]	uni-directional	single	sDDH & DLIN	PRE-CPA	no
Chandran et al. [CCV12]	uni-directional	single	SXDH	PRE-CPA	no
Yoshinori Aono et al. [ABPW13]	uni-directional	single	LWE	PRE-CCA	yes
Chandran et al. [CCL$^+$14]	uni-directional	multi	LWE	PRE-CPA	no
This paper	uni-directional	single	LWE	PRE-CCA*	no
	uni-directional	multi	LWE	PRE-CCA*	no
	uni-directional	multi	LWE	PRE-CPA	no

Table 1: Comparison of our work with previous work. Here ROM stands for random oracle, DH means Diffie-Helleman. DDH means decisional DH, CDH means computational DH, eDBDH means exponential decisional bilinear DH, 3-wDBDHI means 3-party weak decisional bilinear DH inversion, 3-QDBDH means 3-Quotient Decision bilinear DH, SXDH means strong external DH, sDDH means strong decisional DH, DLIN mean decisional linear, and LWE stands for learning with errors assumption.

some "maliciously prepared" ciphertexts. We note that even though Kirshanova's scheme might circumvent the attack by posing (slight) restrictions on the security model, the security proof of Kirshanova's scheme is still flawed as we described above.

2 Preliminaries

Notations. Let PPT denote probabilistic polynomial time. We use bold uppercase letters to denote matrices, and bold lowercase letters for vectors. We let λ be the security parameter, $[n]$ denote the set $\{1, ..., n\}$, and $|\boldsymbol{t}|$ denote the number of bits in a string or vector $\boldsymbol{t}$. We denote the i-th bit value of a string $\boldsymbol{s}$ by $\boldsymbol{s}[i]$. We use $[\cdot|\cdot]$ to denote the concatenation of vectors or matrices, and use l_∞ norm for the norms of all vectors and matrices used in our paper.

2.1 Proxy Re-Encryption Scheme

In this section, we describe the definition of uni-directional PRE, which can be regarded as a natural extension of bi-directional PRE defined in [CH07]. We present the syntax and security definition introduced in [CH07], with some modification to suit the unidirectional setting. The PRE scheme consists a tuple of PPT algorithms $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{ReKeyGen}, \mathsf{ReEnc})$, which can be defined as follows:

$\mathsf{Setup}(1^\lambda)$: On input the security parameter 1^λ, the setup algorithm sets some parameters for the PRE system and outputs a public parameter pp.

$\mathsf{KeyGen}(\mathsf{pp})$: On input the public parameter pp, the key generation algorithm outputs a public key pk and a secret key sk for one user in the system. Without loss of generality, pp is known to all parties and can be included in pk and sk.

$\mathsf{Enc}(\mathsf{pk}, \mu)$: On input a public key pk and a message μ, the encryption algorithm outputs a ciphertext ct for message μ.

$\mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$: On input a secret key sk and a ciphertext ct, the decryption algorithm outputs a message μ or an error symbol $\perp$.

$\mathsf{ReKeyGen}(\mathsf{pk}, \mathsf{sk}, \widetilde{\mathsf{pk}})$: On input a public/secret pair $(\mathsf{pk}, \mathsf{sk})$ from one user, and public key $\widetilde{\mathsf{pk}}$ from another, the re-encryption key generation algorithm outputs a re-encryption key as $\mathsf{rk}_{\mathsf{pk}\to\widetilde{\mathsf{pk}}}$.

$\mathsf{ReEnc}(\mathsf{rk}_{\mathsf{pk}\to\widetilde{\mathsf{pk}}}, \mathsf{ct})$: On input a re-encryption key $\mathsf{rk}_{\mathsf{pk}\to\widetilde{\mathsf{pk}}}$ and a ciphetext ct under the key pk, the re-encryption algorithm outputs a ciphertext ct' under public key $\widetilde{\mathsf{pk}}$.

Definition 2.1 (Single/multi-hop PRE)**.** *We say a PRE scheme is $H(\lambda)$-level multi-hop if a proxy can further re-encrypt already re-encryted ciphertexts up to $H(\lambda)$ times. A PRE scheme is single-hop if $H(\lambda) = 1$.*

Correctness. For correctness, we consider two cases for the PRE scheme: one for "fresh" ciphertexts generated by encryption algorithm, and the other for re-encryption ciphertexts generated by the re-encryption algorithm. We say that an $H(\lambda)$-level multi-hop PRE scheme is correct if the following holds.

1. For all pp output by $\mathsf{Setup}(1^\lambda)$, $(\mathsf{pk}, \mathsf{sk})$ output by $\mathsf{KeyGen}(\mathsf{pp})$, and all message μ, it holds that $\mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(\mathsf{pk}, \mu)) = \mu$ with overwhelming probability.
2. For any integers $n \in [H(\lambda)]$, any sequence of secret/public key pairs $\{(\mathsf{pk}_i, \mathsf{sk}_i)\}_{i\in[n]}$ output $\mathsf{KeyGen}(\mathsf{pp})$, any re-encryption keys $\mathsf{rk}_{i\to i+1} \leftarrow \mathsf{ReKeyGen}(\mathsf{pk}_i, \mathsf{sk}_i, \mathsf{pk}_{i+1})$, any message μ, and $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}_1, \mu)$, it holds that
$$\mathsf{Dec}(\mathsf{sk}_n, \mathsf{ReEnc}(\mathsf{rk}_{n-1\to n}, ..., \mathsf{ReEnc}(\mathsf{rk}_{1\to 2}, \mathsf{ct})...)) = \mu.$$

Security definition. Here we present the chosen-ciphertext security of uni-directional PRE appeared in [Kir14], which can be regarded as a natural extension of CCA-PRE notion introduced in [CH07] who considered the bi-directional setting. Let $\mathcal{A}$ denote any PPT adversary, and Π be a PRE scheme. We define the notion of CCA-secure PRE in the uni-directional setting using the following experiment $\mathbf{Expt}_{\mathcal{A}}^{\mathsf{CCA\text{-}PRE}}(1^\lambda)$, which describes the interaction between several oracles and an adversary $\mathcal{A}$. Recall that as we discussed before, we include public parameters pp in each user's public key pk and secret key sk, so we will omit them in the description for simplicity.

Definition 2.2 (Uni-directional CCA-PRE)**.** *The experiment consists of an execution of $\mathcal{A}$ with the following oracles with detail as follows:*

- *The challenger runs the key generation algorithm $\mathsf{KeyGen}(\mathsf{pp})$ and sends the challenge public key pk^* to adversary $\mathcal{A}$.*
- *Proceeding adaptively, the adversary $\mathcal{A}$ has access to the following oracles:*

  Uncorrupted key generation: *Obtain a new key pair $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$. Send pk_i back to adversary $\mathcal{A}$ and add index i to the honest user set $\mathcal{H}$.*

  Corrupted key generation: *Obtain a new key pair $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$. Send the key pair $(\mathsf{pk}_i, \mathsf{sk}_i)$ back to adversary $\mathcal{A}$ and add index i to the corrupted user set $\mathcal{C}$.*

  Re-encryption key generation $\mathcal{O}_{\mathsf{ReKeyGen}}$: *On input an index pair (i, j) sent by the adversary, we consider the following cases:*

  - *If $(i \in \mathcal{H}, j \in \mathcal{C})$ or $(i \in \mathcal{C}, j \in \mathcal{H})$ or $\{i, j\} \nsubseteq \mathcal{C} \cup \mathcal{H}$, the oracle returns $\perp$;*
  - *else if the pair (i, j) is queried for the first time, the oracle returns a re-encryption key $\mathsf{rk}_{i\to j} \leftarrow \mathsf{ReKeyGen}(\mathsf{pk}_i, \mathsf{sk}_i, \mathsf{pk}_j)$;*
  - *else (the pair (i, j) has been queried before), the oracle returns the re-encryption key $\mathsf{rk}_{i\to j}$.*

To avoid trivial attacks, we require that either both (i, j) are in $\mathcal{H}$, or alternatively both are corrupted, i.e. we do not allow re-encryption key generation queries between a corrupted and a uncorrupted parties.

Re-encryption oracle $\mathcal{O}_{\mathsf{ReEnc}}$: *On input (i, j, ct), if $j \in \mathcal{C}$ or $\{i, j\} \nsubseteq \mathcal{C} \cup \mathcal{H}$, then return $\perp$. Otherwise, the oracle returns a re-encrypted ciphertext $\mathsf{ct}' \leftarrow \mathsf{ReEnc}(\mathsf{rk}_{i\to j}, \mathsf{ct})$.*

Decryption oracle $\mathcal{O}_{\mathsf{Dec}}$: *On input (i, ct), if $i \notin \mathcal{C} \cup \mathcal{H}$ or ct is not a valid ciphertext, then return a special symbol $\perp$. Otherwise, return $\mathsf{Dec}(\mathsf{sk}_i, \mathsf{ct})$.*

Challenge oracle: *This oracle can be queried only once. On input (i^*, μ_0, μ_1), the oracle chooses a bit $b \in \{0, 1\}$ and returns $\mathsf{ct}^* \leftarrow \mathsf{Enc}(\mathsf{pk}_{i^*}, \mu_b)$ as the challenge ciphertext.*

Decision oracle: *This oracle can be queried only once. On input b' from adversary $\mathcal{A}$, the oracle outputs 1 if $b' = b$, and 0 otherwise.*

The advantage of an adversary in the above experiment $\mathbf{Expt}_{\mathcal{A}}^{\mathsf{CCA\text{-}PRE}}(1^\lambda)$ is defined as $|\mathbf{Pr}[b' = b] - \frac{1}{2}|$. A uni-directional PRE scheme is CCA-PRE secure if all PPT *adversaries have at most a negligible advantage in the above experiment.*

Remark 2.3. *We described a meaningful relaxation of the CCA-PRE security, which we name CCA*-PRE security. The relaxation lies in the decryption oracle queried after challenge oracle: for tag-based schemes (i.e. whose ciphertexts consist of tags), we restrict the that the adversary cannot query the decryption oracle with ciphertexts using the same tag as the challenge ciphertext. That is, the decryption oracle (queried after the challenge ciphertext) always outputs a special symbol $\perp$ upon queries with the challenge ciphertext's tag.*

It is obvious to see that CCA security is stronger than CCA1 security (where the adversary cannot access the decryption oracle after the challenge ciphertext), and is slightly weaker than CCA(2) security. This relaxation is meaningful and can be nearly the best we can achieve if we further require the property of* unlinkability *for re-encrypted ciphertexts. That is, if we want the re-encrypted algorithm to produce statistically indistinguishable ciphertexts, i.e. the re-encrypted ciphertexts are almost identically distributed as fresh ones, then arguably it is not possible to achieve CCA2 security, because the decryption oracle cannot distinguish a re-encryption of challenge ciphertext from a fresh ciphertext, so an adversary can easily break the security game by querying the decryption oracle with a re-encrypted ciphertext of the challenge ciphertext. For tag-based schemes, if we design a scheme such that the tag remains the same for re-encrypted ciphertexts, then we can make sure that the challenge ciphertext will not be decrypted directly to the adversary. The CCA* security guarantees the challenge ciphertext remains hidden, even if the adversary can obtain decryptions of ciphertexts with other tags.*

The above definition is general that the adversary can query re-encryption keys with no restrictions. In some cases such as [CCL+14], they can prove security only if the structure of the re-encryption keys queried by the adversary forms an acyclic graph. Therefore, we consider several meaningful relaxations of the security model to capture this concept. We consider a relaxation – CCA-PRE security with respect to a specified family of graphs (e.g. acyclic graphs). In this model, the security game is essentially the same as Definition 2.2, but the adversary's queries are restricted so that the structure of the re-encryption keys must form a graph in the family of graphs. Another relaxation is CPA-PRE security with respect to a family of graphs. This model has the same spirit, except the adversary does not have access to the decryption oracle. The construction by Chandran et al. [CCL+14] satisfies this model.

Definition 2.4 (Graph-based uni-directional CCA-PRE)**.** *Let $\mathcal{G}$ be a family of graphs, and $\mathcal{A}$ be an adversary. The experiment $\mathbf{Expt}_{\mathcal{A},\mathcal{G}}^{\mathsf{graph\text{-}CCA\text{-}PRE}}(1^\lambda)$ is the same as that in Definition 2.2, except the re-encryption keys obtained by the adversary must form a graph $G \in \mathcal{G}$. A uni-directional PRE scheme is CCA-PRE secure with respect to the family $\mathcal{G}$ if all* PPT *adversaries have at most a negligible advantage in the above experiment.*

We define graph-based unidirectional CPA-PRE in a similar way as below.

Definition 2.5 (Graph-based uni-directional CPA-PRE)**.** *Let $\mathcal{G}$ be a family of graphs, and the experiment $\mathbf{Expt}^{\mathsf{graph\text{-}CPA\text{-}PRE}}_{\mathcal{A},\mathcal{G}}(1^\lambda)$ be analogous to experiment $\mathbf{Expt}^{\mathsf{graph\text{-}CCA\text{-}PRE}}_{\mathcal{A},\mathcal{G}}(1^\lambda)$ except that the adversary does not have access to decryption oracle $\mathcal{O}_{\mathsf{Dec}}$. We say a PRE scheme is CPA secure with respect to the family $\mathcal{G}$ if all* PPT *adversaries have at most a negligible advantage in the above experiment.*

Remark 2.6. *If $\mathcal{G}$ is the family of trees, then we call the scheme a tree-based unidirectional unidirectional CCA*/CPA-PRE; if that is the family of acyclic graph, then we call the scheme an acyclic graph-based unidirectional unidirectional CCA*/CPA-PRE. In particular, our construction in Section A is a tree-based unidirectional CCA* scheme, and the construction in Section B is an acyclic graph-based CPA-PRE.*

2.2 Lattice Background

A full-rank m-dimensional integer lattice $\Lambda \subset \mathbb{Z}^m$ is a discrete additive subgroup whose linear span is $\mathbb{R}^m$. Let Λ be a discrete subset of $\mathbb{Z}^m$. For any vector $\boldsymbol{c} \in \mathbb{R}^m$, and any positive parameter $\sigma \in \mathbb{R}$, let $\rho_{\sigma,\boldsymbol{c}}(\boldsymbol{x}) = \exp(-\pi||\boldsymbol{x}-\boldsymbol{c}||^2/\sigma^2)$ be the Gaussian function on $\mathbb{R}^m$ with center $\boldsymbol{c}$ and parameter σ. Next, we set $\rho_{\sigma,\boldsymbol{c}}(\Lambda) = \sum_{\boldsymbol{x}\in\Lambda}\rho_{\sigma,\boldsymbol{c}}(\boldsymbol{x})$ be the discrete integral of $\rho_{\sigma,\boldsymbol{x}}$ over Λ and $\mathcal{D}_{\Lambda,\sigma,\boldsymbol{c}}(\boldsymbol{y}) := \frac{\rho_{\sigma,\boldsymbol{c}}(\boldsymbol{y})}{\rho_{\sigma,\boldsymbol{c}}(\Lambda)}$. Let S^m denote the set of vectors in $\mathbb{R}^{m+1}$ whose length is 1. Then the norm of a matrix $\mathbf{R} \in \mathbb{R}^{m\times m}$ is defined to be $\mathsf{sup}_{\boldsymbol{x}\in S^m}||\mathbf{R}\boldsymbol{x}||$. We have the following lemma, which bounds the norm for some specified distributions.

Lemma 2.7 ([ABB10])**.** *Regarding the norm defined above, we have the following bounds:*

- *Let $\mathbf{R} \in \{-1,1\}^{m\times m}$ be chosen at random, then $\mathbf{Pr}[||\mathbf{R}|| > 12\sqrt{2m}] < e^{-2m}$.*
- *Let $\mathbf{R}$ be sampled from $\mathcal{D}_{\mathbb{Z}^{m\times m},\sigma}$, then we have $\mathbf{Pr}[||\mathbf{R}|| > \sigma\sqrt{m}] < e^{-2m}$.*

Randomness extraction. We will use the following lemma to argue the indistinghishability of two different distributions, which is a generalization of the leftover hash lemma proposed by Dodis et al. [DRS04].

Lemma 2.8 ([ABB10])**.** *Suppose that $m > (n+1)\log q + w(\log n)$. Let $\mathbf{R} \in \{-1,1\}^{m\times k}$ be chosen uniformly at random for some polynomial $k = k(n)$. Let $\mathbf{A}, \mathbf{B}$ be matrix chosen randomly from $\mathbb{Z}_q^{n\times m}, \mathbb{Z}_q^{n\times k}$ respectively. Then, for all vectors $\boldsymbol{w} \in \mathbb{Z}^m$, the two following distributions are statistically close:*

$$(\mathbf{A}, \mathbf{AR}, \mathbf{R}^T\boldsymbol{w}) \approx (\mathbf{A}, \mathbf{B}, \mathbf{R}^T\boldsymbol{w})$$

Learning With Errors. The LWE problem was introduced by Regev [Reg05], who showed that solving it *on the average* is as hard as (quantumly) solving several standard lattice problems *in the worst case*.

Definition 2.9 (LWE)**.** *For an integer $q = q(n) \geq 2$, and an error distribution $\chi = \chi(n)$ over $\mathbb{Z}_q$, the learning with errors problem $\mathsf{LWE}_{n,m,q,\chi}$ is to distinguish between the following pairs of distributions:*

$$\{\mathbf{A}, \mathbf{A}\boldsymbol{s}+\boldsymbol{x}\} \text{ and } \{\mathbf{A}, \boldsymbol{u}\}$$

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

G-trapdoors and sampling algorithms. We briefly describe the main results in [MP12]: the definition of $\mathbf{G}$-trapdoor and the algorithms $\mathsf{Invert}^{\mathcal{O}}$ and $\mathsf{Sample}^{\mathcal{O}}$. Roughly speaking, a $\mathbf{G}$-trapdoor is a transformation, represented by a matrix $\mathbf{R}$ from a public matrix $\mathbf{A}$ to a special matrix $\mathbf{G}$. The formal definition is as follows:

Definition 2.10 ([MP12]). *Let $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ and $\mathbf{G} \in \mathbb{Z}_q^{n\times w}$ be matrices with $m \geq w \geq n$. A $\mathbf{G}$-trapdoor for $\mathbf{A}$ is a matrix $\mathbf{R} \in \mathbb{Z}^{m-w} \times w$ such that $\mathbf{A}\left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right] = \mathbf{HG}$ for some invertible matrix $\mathbf{H} \in \mathbb{Z}_q^{n\times n}$. We refer to $\mathbf{H}$ as the tag or label of the trapdoor. The quality of the trapdoor is measured by its largest singular value $s_1(\mathbf{R})$.*

In order to embed matrix $\mathbf{G}$ into a uniformly looking matrix $\mathbf{A}$ together with a transformation $\mathbf{R}$, we should start with a uniform matrix $\mathbf{A}_0$ and a matrix $\mathbf{R}$, and construct $\mathbf{A} = [\mathbf{A}_0| - \mathbf{A}_0\mathbf{R} + \mathbf{HG}]$. For an appropriate chosen dimensions $(\mathbf{A}, \mathbf{AR})$ is negligible from uniformly random distribution by the Lattice-based Leftover Hash Lemma.

Lemma 2.11 ([MP12]). *There is an efficient algorithm $\mathsf{Sample}^{\mathcal{O}}(\mathbf{R}, \mathbf{A}', \mathbf{H}, \boldsymbol{u}, s)$, where $\mathbf{R}$ is a $\mathbf{G}$-trapdoor for matrix $\mathbf{A}$ with invertible tag $\mathbf{H}$, a vector $\boldsymbol{u} \in \mathbb{Z}^n$ and an oracle $\mathcal{O}$ for Gaussian sampling over a desired coset $\Lambda_q^{\boldsymbol{v}}(\mathbf{G})$. It will output a vector drawn from a distribution within negligible statistical distance of $\mathcal{D}_{\Lambda^{\boldsymbol{u}}(\mathbf{A}),s}$, where $\mathbf{A} = [\mathbf{A}'| - \mathbf{A}'\mathbf{R} + \mathbf{HG}]$.*

In the following, we provide two extensions of the LWE inversion algorithms proposed by Micciancio and Peikert [MP12], which would be used in the security proof and scheme respectively.

- $\mathsf{Invert}^{\mathcal{O}}(\mathbf{R}_1, \mathbf{R}_2, \mathbf{A}, \boldsymbol{b})$: On input a vector $\boldsymbol{b} = \boldsymbol{s}^t\mathbf{A} + \boldsymbol{e}^t$, a matrix $\mathbf{A} = [\mathbf{A}_0| - \mathbf{A}_0\mathbf{R}_1 + \mathbf{H}_1\mathbf{G}| - \mathbf{A}_0\mathbf{R}_2 + \mathbf{H}_2\mathbf{G}]$ and its corresponding $\mathbf{G}$-trapdoor $\mathbf{R}_1, \mathbf{R}_2$ with invertible tag $\mathbf{H}_1, \mathbf{H}_2$, the algorithm first computes $\boldsymbol{b}' = \boldsymbol{b}^t\left[\begin{smallmatrix}\mathbf{R}_1+\mathbf{R}_2\\ \mathbf{I}\\ \mathbf{I}\end{smallmatrix}\right]$, and then run the oracle $\mathcal{O}(\boldsymbol{b}')$ to get $(\boldsymbol{s}', \boldsymbol{e}')$. The algorithm outputs $\boldsymbol{s} = (\mathbf{H}_1 + \mathbf{H}_2)^{-t}\boldsymbol{s}'$ and $\boldsymbol{e} = \boldsymbol{b} - \mathbf{A}^t$.
- $\mathsf{Invert}'^{\mathcal{O}}(\mathbf{R}_1, \mathbf{R}_2, \mathbf{A}, \boldsymbol{b})$: On input a vector $\boldsymbol{b} = \boldsymbol{s}^t\mathbf{A}+\boldsymbol{e}^t$, a matrix $\mathbf{A} = [\mathbf{A}_0|-\mathbf{A}_0\mathbf{R}_1|-\mathbf{A}_0\mathbf{R}_2+\mathbf{H}_2\mathbf{G}]$ and its corresponding $\mathbf{G}$-trapdoor $\mathbf{R}_1, \mathbf{R}_2$ with invertible tag $\mathbf{H}_1, \mathbf{H}_2$, the algorithm first computes $\boldsymbol{b}' = \boldsymbol{b}^t\left[\begin{smallmatrix}\mathbf{R}_1+\mathbf{R}_2\\ \mathbf{I}\\ \mathbf{I}\end{smallmatrix}\right]$, and then run the oracle $\mathcal{O}(\boldsymbol{b}')$ to get $(\boldsymbol{s}', \boldsymbol{e}')$. The algorithm outputs $\boldsymbol{s} = \mathbf{H}_2^{-t}\boldsymbol{s}'$ and $\boldsymbol{e} = \boldsymbol{b} - \mathbf{A}^t$.

3 Single-hop CCA*-Secure PRE Construction

In this section, we present our first construction of single-hop PRE. We achieve the CCA*-PRE security as defined in Definition 2.2.

As the work of Micciancio and Peikert [MP12], our scheme uses a special collection of elements defined over ring $\mathcal{R} = \mathbb{Z}_q[x]/(f(x))$, where $f(x) = x^n + f_{n-1}x^{n-1} + \cdots + f_0$ is a irreducible modulo every p dividing q. Since $\mathcal{R}$ is a free $\mathbb{Z}_q$-module of rank n, thus elements of $\mathcal{R}$ can be represented as vectors in $\mathbb{Z}_q$ relative to standard basis of monomials $1, x, ..., x^{n-1}$. Multiplication by any fixed element of $\mathcal{R}$ then acts as a linear transformation on $\mathbb{Z}_q^n$ according to the rule

$$x \cdot (a_0, ..., a_{n-1})^t = (0, a_0, ..., a_{n-2})^t - a_{n-1}(f_0, f_1, ..., f_{n-1})^t$$

and so can be represented by an matrix in $\mathbb{Z}_q^{n\times n}$ relative to the standard basis. In other words, there is an injective ring homomorphism $h : \mathcal{R} \to \mathbb{Z}_q^{n\times n}$ that maps any $a \in \mathcal{R}$ to matrix $\mathbf{H} = h(a)$ representing multiplication by a. As introduced in [MP12], we need a very large set $\mathcal{U} = \{u_1, ..., u_l\}$ with the "unit differences" property: for any $i \neq j$, the difference $u_i - u_j \in \mathcal{R}^*$, and hence $h(u_i - u_j) = h(u_i) - h(u_j) \in \mathbb{Z}_q^{n\times n}$ is invertible.

The PRE system has message space $\{0,1\}^{nk}$, which we map bijectively to the cosets of $\Lambda/2\Lambda$ for $\Lambda = \Lambda(\mathbf{G}^t)$ via some encoding function encode that is efficient to evaluate and invert. In particular, letting $\mathbf{S} \in \mathbb{Z}^{nk\times nk}$ be any basis of Λ, we can map $\boldsymbol{\mu} \in \{0,1\}^{nk}$ to $\mathsf{encode}(\boldsymbol{\mu}) = \mathbf{S}\boldsymbol{\mu} \in \mathbb{Z}^{nk}$. The PRE scheme $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{ReKeyGen}, \mathsf{ReEnc})$ can be described as follows:

- $\mathsf{Setup}(1^\lambda, 1^N)$: On input the security parameter λ and the number N of users in the PRE system, the global setup algorithm set the lattice parameter $n = n(\lambda, N), m = m(\lambda, N), q = q(\lambda, N)$ and Gaussian parameter $s = s(\lambda, N)$. Then it randomly selects a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, and outputs the public parameter $\mathsf{pp} = (\mathbf{A}, n, m, q, s)$.

- $\mathsf{KeyGen}(\mathsf{pp})$: On input the public parameter pp, the key generation algorithm for i-th user chooses random matrices $\mathbf{R}_{i1}, \mathbf{R}_{i2} \leftarrow \mathcal{D}$, letting $\mathbf{A}_{i1} = \mathbf{A}\mathbf{R}_{i1} \bmod q$ and $\mathbf{A}_{i2} = \mathbf{A}\mathbf{R}_{i2} \bmod q$. The public key is $\mathsf{pk}_i = \mathbf{A}_i = [\mathbf{A}| - \mathbf{A}_{i1}| - \mathbf{A}_{i2}]$, and the secret key is $\mathsf{sk}_i = [\mathbf{R}_{i1}|\mathbf{R}_{i2}]$.

- $\mathsf{Enc}(\mathsf{pk}_i, \boldsymbol{\mu}, \ell)$: On input a public key pk_i, a message $\boldsymbol{\mu}$ and a level $\ell \in \{1, 2\}$, the encryption algorithm chooses non-zero $u \leftarrow \mathcal{U}$ and let the message/level-dependent matrix

$$\mathbf{A}_{i,u,l} = [\mathbf{A}| - \mathbf{A}_{i1} + h(\ell)\mathbf{G}| - \mathbf{A}_{i2} + h(u)\mathbf{G}]$$

  Choose $\boldsymbol{s} \leftarrow \mathbb{Z}_q^n, \boldsymbol{e}_0, \boldsymbol{e}_1 \leftarrow \mathcal{D}_{\mathbb{Z},s}^m$ and $\boldsymbol{e}_2 \leftarrow \mathcal{D}_{\mathbb{Z},s}^{nk}$. Let

$$\boldsymbol{b}^t = (\boldsymbol{b}_0, \boldsymbol{b}_1, \boldsymbol{b}_2) = 2(\boldsymbol{s}^t \mathbf{A}_{i,u,\ell} \bmod q) + \boldsymbol{e}^t + (0, 0, \mathsf{encode}(\boldsymbol{\mu})^t) \bmod 2q$$

  where $\boldsymbol{e} = (\boldsymbol{e}_0, \boldsymbol{e}_1, \boldsymbol{e}_3)$. Output the ciphertext $\mathsf{ct} = (u, \boldsymbol{b}, \ell)$.

- $\mathsf{Dec}(\mathsf{sk}_i, \mathsf{ct})$: On input a secret key sk_i and ciphertext $\mathsf{ct} = (u, \boldsymbol{b}, \ell)$, the decryption algorithm

  1. If ct does not parse or $u = 0$, output $\perp$. Otherwise, reconstruct the message/level-dependent matrix $\mathbf{A}_{i,u,\ell}$

$$\mathbf{A}_{i,u,l} = [\mathbf{A}| - \mathbf{A}_{i1} + h(\ell)\mathbf{G}| - \mathbf{A}_{i2} + h(u)\mathbf{G}]$$

     Call $\mathsf{Invert}^{\mathcal{O}}([\mathbf{R}_{i1}|\mathbf{R}_{i2}], \mathbf{A}_u, \boldsymbol{b} \bmod q)$ to get values $\boldsymbol{z} \in \mathbb{Z}_q^n$ and $\boldsymbol{e} = (\boldsymbol{e}_0, \boldsymbol{e}_1, \boldsymbol{e}_2)$ for which $\boldsymbol{b}^t = \boldsymbol{z}^t + \boldsymbol{e}^t \bmod q$. If the algorithm Invert fail for any reason, output $\perp$.
  2. Check the length of the obtained error vectors.
  3. Let $\boldsymbol{v} = \boldsymbol{b} - \boldsymbol{e}$, and parse $\boldsymbol{v} = (\boldsymbol{v}_0, \boldsymbol{v}_1, \boldsymbol{v}_2)$. If $\boldsymbol{v}_0 \notin 2\Lambda(\mathbf{A}^t)$, output $\perp$. Finally, output

$$\mathsf{encode}^{-1}(\boldsymbol{v}^t \begin{bmatrix} \mathbf{R}_{i1} & \mathbf{R}_{i2} \\ \mathbf{I} & 0 \\ 0 & \mathbf{I} \end{bmatrix} \bmod 2q) \in \{0,1\}^{nk}$$

     if it exists, otherwise output $\perp$.

- $\mathsf{ReKeyGen}(\mathsf{pk}_i, \mathsf{sk}_i, \mathsf{pk}_j)$: On input a public/secret key pair $(\mathsf{pk}_i, \mathsf{sk}_i)$ from the i-th user and public key pk_j from j-th user, the re-encryption key generation algorithm do:

  1. Parse i-th public/secret key pair and j-th public key as follows:

$$\mathsf{pk}_i = [\mathbf{A}| - \mathbf{A}_{i1}| - \mathbf{A}_{i2}], \quad \mathsf{sk}_i = [\mathbf{R}_{i1}|\mathbf{R}_{i2}], \quad \mathsf{pk}_j = [\mathbf{A}| - \mathbf{A}_{j1}| - \mathbf{A}_{j2}]$$

  2. Use extended sampling algorithm $\mathsf{Sample}^{\mathcal{O}}$ to sample $\mathbf{X}_{01}, \mathbf{X}_{02}, \mathbf{X}_{11}, \mathbf{X}_{12} \in \mathbb{Z}^{nk \times nk}$ such that

$$[\mathbf{A}| - \mathbf{A}_{i1} + h(1)\mathbf{G}] \begin{bmatrix} \mathbf{X}_{01} \\ \mathbf{X}_{11} \end{bmatrix} = -\mathbf{A}_{j1} + h(2)\mathbf{G}, \ [\mathbf{A}| - \mathbf{A}_{i1} + h(1)\mathbf{G}] \begin{bmatrix} \mathbf{X}_{02} \\ \mathbf{X}_{12} \end{bmatrix} = -\mathbf{A}_{j2} + \mathbf{A}_{i2}$$

     Therefore, it holds that

$$[\mathbf{A}| - \mathbf{A}_{i1} + h(1)\mathbf{G}| - \mathbf{A}_{i2} + \mathbf{B}] \begin{bmatrix} \mathbf{I} & \mathbf{X}_{01} & \mathbf{X}_{02} \\ 0 & \mathbf{X}_{11} & \mathbf{X}_{12} \\ 0 & 0 & \mathbf{I} \end{bmatrix} = [\mathbf{A}| - \mathbf{A}_{j1} + h(2)\mathbf{G}| - \mathbf{A}_{j2} + \mathbf{B}]$$

     for any matrix $\mathbf{B} \in \mathbb{Z}^{n \times nk}$.

3. Output the re-encryption key

$$\mathsf{rk}_{i\to j} = \{\mathbf{X}_{01}, \mathbf{X}_{02}, \mathbf{X}_{11}, \mathbf{X}_{12}\}$$

- $\mathsf{ReEnc}(\mathsf{rk}_{i\to j}, \mathsf{ct})$: On input the re-encryption key $\mathsf{rk}_{i\to j}$ from i-th user to j-th user, and a ciphertext $\mathsf{ct} = (u, \boldsymbol{b}, \ell)$, the re-encryption algorithm output a special symbol $\perp$ if $\ell = 2$. Otherwise, it computes

$$\boldsymbol{b}^t \cdot \mathsf{rk}_{i\to j} = \boldsymbol{s}^t[\mathbf{A}| - \mathbf{A}_{j1} + h(1)\mathbf{G}| - \mathbf{A}_{j2} + h(u)\mathbf{G}] + \boldsymbol{e}'^t + (0, 0, \mathsf{encode}(\mu)^t)$$

where $\boldsymbol{e}' = (\boldsymbol{e}'_0, \boldsymbol{e}'_1, \boldsymbol{e}'_2)$, and

$$\boldsymbol{e}'_0 = \boldsymbol{e}_0, \qquad \boldsymbol{e}'_1 = \boldsymbol{e}_0\mathbf{X}_{01} + \boldsymbol{e}_1\mathbf{X}_{11}, \qquad \boldsymbol{e}'_2 = \boldsymbol{e}_0\mathbf{X}_{02} + \boldsymbol{e}_1\mathbf{X}_{12} + \boldsymbol{e}_2$$

Then, it outputs $\mathsf{ct}' = (u, \boldsymbol{b}', 2)$.

Correctness. We show that our construction, with appropriate parameter setting specified in Section 3.2, satisfies the correctness condition defined above.

Lemma 3.1 (Correctness)**.** *Let* $(\mathbf{A}_i, (\mathbf{R}_{i1}, \mathbf{R}_{i2}))$ *and* $(\mathbf{A}_j, (\mathbf{R}_{j1}, \mathbf{R}_{j2}))$ *be the public/secret key pair for* i, j*-th user respectively in the PRE system. Let* $\mathsf{ct} = (u, \boldsymbol{b}, 1)$ *be the ciphertext of plaintext* $\boldsymbol{\mu}$ *for* i*-th user, and* $\mathsf{ct}' = \mathsf{ReEnc}(\mathsf{rk}_{i\to j}, \mathsf{ct})$ *be the re-encrypted ciphertext for* j*-th user. Then as we require above, it holds* $\mu \leftarrow \mathsf{Dec}(\mathsf{ct}', \mathsf{sk}_j)$.

Proof. Parse ciphertext ct as $\mathsf{ct} = (u, \boldsymbol{b}, 1)$. Per correctness, we have

$$\boldsymbol{b}^t = 2(\boldsymbol{s}^t\mathbf{A}_{i,u,1} \bmod q) + \boldsymbol{e}^t + (0, 0, \mathsf{encode}(\boldsymbol{\mu})^t) \bmod 2q$$

where $\mathbf{A}_{i,u,1} = [\mathbf{A}| - \mathbf{A}_{i1} + h(1)\mathbf{G}| - \mathbf{A}_{i2} + h(u)\mathbf{G}]$. The re-encryption process can be re-phrased as follows:

$$\begin{aligned}
&\boldsymbol{s}^t[\mathbf{A}| - \mathbf{A}_{i1} + h(1)\mathbf{G}| - \mathbf{A}_{i2} + h(u)\mathbf{G}] \begin{bmatrix} \mathbf{I} & \mathbf{X}_{01} & \mathbf{X}_{02} \\ 0 & \mathbf{X}_{11} & \mathbf{X}_{12} \\ 0 & 0 & \mathbf{I} \end{bmatrix} + \ \mathsf{noise}\ + (0, 0, \mathsf{encode}(\boldsymbol{\mu})^t) \\
&= \boldsymbol{s}^t[\mathbf{A}| - \mathbf{A}_{j1} + h(2)\mathbf{G}| - \mathbf{A}_{j2} + h(u)\mathbf{G}] + \ \mathsf{noise}\ + (0, 0, \mathsf{encode}(\boldsymbol{\mu})^t) \\
&= \boldsymbol{s}^t\mathbf{A}_{j,u,2} + \ \mathsf{noise}\ + (0, 0, \mathsf{encode}(\boldsymbol{\mu})^t)
\end{aligned}$$

where $\mathbf{A}_{j,u,2} = [\mathbf{A}| - \mathbf{A}_{j1} + h(2)\mathbf{G}| - \mathbf{A}_{j2} + h(u)\mathbf{G}]$ and the noise terms is defined in algorithm ReEnc. It is obvious that the re-encrypted ciphertext can be decrypted using j-th secret key, thus we omit the detail for decryption here. □

3.1 Security Proof

We follow the intuition explained in the introduction part to prove security as the following theorem.

Theorem 3.2. *Assuming the hardness of* $\mathsf{LWE}_{q,\alpha'}$ *(*$\alpha' = \alpha/3 \geq 2\sqrt{n}/q$*), the proxy re-encryption scheme is selectively CCA*-secure as defined in Definition 2.2.*

Proof. First, using the same technique in [MP12], we can transform the samples from LWE distribution to what we will need below. Given access to an LWE distribution $A_{\boldsymbol{s},\alpha'}$ over $\mathbb{Z}_q \times \mathbb{T}$ (where $\mathbb{T} = \mathbb{R}/\mathbb{Z}$), we can transform its samples $(\boldsymbol{a}, b = \langle \boldsymbol{s}, \boldsymbol{a}\rangle/q + e \bmod 1)$ to have the form $(\boldsymbol{a}, 2(\langle \boldsymbol{s}, \boldsymbol{\alpha}\rangle \bmod q) + e' \bmod 2q)$ for $e' \leftarrow D_{\mathbb{Z},\alpha q}$, by mapping b to $2qb + D_{\mathbb{Z}-2qb,s} \bmod 2q$ where $s^2 = (\alpha q)^2 - (2\alpha' q)^2 \geq 4n \geq \eta_\epsilon(\mathbb{Z})^2$. The transformation maps the uniform distribution over $\mathbb{Z}_q^n \times \mathbb{T}$ to the uniform distribution over $\mathbb{Z}_q^n \times \mathbb{Z}_{2q}$. Once the LWE samples are of the desired form, we construct column-wise matrix $\mathbf{A}^*$ from these samples $\boldsymbol{a}$ and a vector $\boldsymbol{b}^*$ from the corresponding b. Without loss of generality, we randomly choose one uncorrputed user as the challenge user, which will result in a polynomial loss in the security proof. We proceed via a sequence of hybrid games:

Hybrid H_0 : The game H_0 is exactly the CCA* attack with the real system described above.

Hybrid H_1 : In game H_1, we change the way to generate uncorrupted pk, challenge ciphertext ct^* and re-encryption keys rk, that are hard to distinguish from the counterparts in game H_0. We set the public parameter matrix $\mathbf{A} = \mathbf{A}^*$, where $\mathbf{A}^*$ is from LWE instance $(\mathbf{A}^*, \boldsymbol{b}^*)$, and select a random element $u^* \in \mathcal{U}$.

- **Uncorrupted key generation oracle**: To obtain a public key for uncorrupted user of the challenge user $i^* \in \mathcal{C}$, the oracle chooses random matrices $\mathbf{R}_{i^*1}, \mathbf{R}_{i^*2}$ from $\{-1, 1\}^{m\times m}$, then output the public key to be
$$\mathsf{pk}_{i^*} = [\mathbf{A}^*| - \mathbf{A}^*\mathbf{R}_{i^*1} - h(1)\mathbf{G}| - \mathbf{A}^*\mathbf{R}_{i^*2} - h(u^*)\mathbf{G}]$$
For uncorrupted query $i \in \mathcal{C}$ other than challenge user i^*, the oracle firstly chooses and stores matrices $\mathbf{X}_{i,01}, \mathbf{X}_{i,11}, \mathbf{X}_{i,02}, \mathbf{X}_{i,12}$ from $\mathcal{D}_{\mathbb{Z},s}^{nk\times nk}$, and set
$$\mathbf{A}_{i1} = [\mathbf{A}| - \mathbf{A}_{i^*1}] \begin{bmatrix}\mathbf{X}_{i,01}\\ \mathbf{X}_{i,11}\end{bmatrix} \qquad \mathbf{A}_{i2} = [\mathbf{A}| - \mathbf{A}_{i^*1}] \begin{bmatrix}\mathbf{X}_{i,02}\\ \mathbf{X}_{i,12}\end{bmatrix}$$
where $\mathbf{A}_{i^*1} = \mathbf{A}\mathbf{R}_{i^*1}, \mathbf{A}_{i^*2} = \mathbf{A}\mathbf{R}_{i^*2}$. Then the oracle outputs the public key for i-th query as
$$\mathsf{pk}_i = [\mathbf{A}| - \mathbf{A}_{i1} - h(2)\mathbf{G}| - \mathbf{A}_{i2} + \mathbf{A}_{i^*2}]$$
Also since the oracle does not reveal any secret key of uncorrupted users, the output of $\{\mathsf{pk}_i\}_{i\in\mathcal{C}}$ other than the challenge ciphertext does not reveal the choice of u^* as well.
- **Corrupted key generation oracle**: To obtain the pair of secret and public keys for user $i \in \mathcal{H}$, the oracle chooses random matrices $\mathbf{R}_{i1}, \mathbf{R}_{i2} \leftarrow \mathcal{D}$, letting $\mathbf{A}_{i1} = \mathbf{A}\mathbf{R}_{i1} \bmod q$ and $\mathbf{A}_{i2} = \mathbf{A}\mathbf{R}_{i2} \bmod q$, and sends back $\mathsf{sk}_i = [\mathbf{R}_{i1}|\mathbf{R}_{i2}]$.
- **Re-encryption key generation oracle**: First check the constrains defined in Definition 2.2, i.e. if $(i \in \mathcal{H}, j \in \mathcal{C})$ or $(i \in \mathcal{C}, j \in \mathcal{H})$ or $\{i, j\} \nsubseteq \mathcal{C} \cup \mathcal{H}$, the oracle returns $\perp$; Otherwise, since there exist some differences in the key generation process between the first query and the rest queries, we divide the process into two cases as follows:
  1. The generation of re-encryption key from challenge user i^* to other i-th user from level 1 to level 2: The oracle can use the pre-sampled matrices $\{\mathbf{X}\}$ in the generation of pk_i to re-construct the re-encryption key $\mathsf{rk}_{i^*\to i}$ as:
  $$\mathsf{rk}_{i^*\to i} = \begin{bmatrix}\mathbf{I} & \mathbf{X}_{i,01} & \mathbf{X}_{i,02}\\ 0 & \mathbf{X}_{i,11} & \mathbf{X}_{i,12}\\ 0 & 0 & \mathbf{I}\end{bmatrix}$$
  where matrices $\mathbf{X}_{i,01}, \mathbf{X}_{i,11}, \mathbf{X}_{i,02}, \mathbf{X}_{i,12}$ are pre-sampled previously in the key generation oracle. Each entry of the resulting re-encryption key is an inner product of a discrete Gaussian vector and a vector consisting of $\{0, 1\}$, so the simulated re-encryption keys from challenge user to other uncorrupted users have the same distribution as a re-encryption key in the scheme.
  2. The generation of re-encryption key from i-th user to j-th user ($i \neq i^*$) from level 1 to level 2: We first reconstruct the level-dependent matrix $\mathbf{A}_i$ for i-th users as:
  $$\mathbf{A}_i = [\mathbf{A}| - \mathbf{A}_{i1} + (h(1) - h(2))\mathbf{G}] = [\mathbf{A}^*| - \mathbf{A}^*\mathbf{R}_{i1}^* + (h(1) - h(2))\mathbf{G}]$$
  where $\mathbf{R}_{i1}^* = \mathbf{X}_{i,01} - \mathbf{R}_{i^*1}\mathbf{X}_{i,11}$. We can still compute the re-encryption key matrix using Sample algorithm in the same way as in H_0, since matrix $(h(1) - h(2))$ is non-zero in encryption matrix $\mathbf{A}_i$. The generation of re-encryption keys in this case still uses algorithm $\mathsf{Sample}^{\mathcal{O}}$, thus the distribution of simulated re-encryption keys is statistically close to that in the real scheme.

- **Decryption oracle**: On decryption query (i, ct) from adversary $\mathcal{A}$, the oracle first parses ciphertext $\mathsf{ct} = (u, \boldsymbol{b}, \ell)$, and outputs $\perp$ if $u = 0$. If the decryption queries are made after the challenge oracle query, then oracle outputs $\perp$ if $u = u^*$. Then oracle divides the decryption process into following four cases:

  1. If $i = i^*$ and $\ell = 1$, oracle reconstructs the message/level-dependent matrix $\mathbf{A}_{i^*,u,1}$ as

  $$\mathbf{A}_{i^*,u,1} = [\mathbf{A}^*| - \mathbf{A}^*\mathbf{R}_{i^*1}| - \mathbf{A}^*\mathbf{R}_{i^*2} + (h(u) - h(u^*))\mathbf{G}]$$

  Call $\mathsf{Invert}'^{\mathcal{O}}([\mathbf{R}_{i^*1}|\mathbf{R}_{i^*2}], \mathbf{A}_{i^*,u}, \boldsymbol{b} \bmod q)$ to get some $\boldsymbol{z} \in \mathbb{Z}^n$ and $\boldsymbol{e}$. Then oracle performs step 3 exactly as in Dec, except using $[\mathbf{R}_{i^*1}|\mathbf{R}_{i^*2}]$ to decode message.

  2. If $i = i^*$ and $\ell = 2$, oracle reconstructs the message/level-dependent matrix $\mathbf{A}_{i^*,u,2}$ as

  $$\mathbf{A}_{i^*,u,1} = [\mathbf{A}^*| - \mathbf{A}^*\mathbf{R}_{i^*1} + (h(2) - h(1))\mathbf{G}| - \mathbf{A}^*\mathbf{R}_{i^*2} + (h(u) - h(u^*))\mathbf{G}]$$

  Call $\mathsf{Invert}^{\mathcal{O}}([\mathbf{R}_{i^*1}|\mathbf{R}_{i^*2}], \mathbf{A}_{i^*,u}, \boldsymbol{b} \bmod q)$ to get some $\boldsymbol{z} \in \mathbb{Z}^n$ and $\boldsymbol{e}$. Then oracle performs step 3 exactly as in Dec, except using $[\mathbf{R}_{i^*1}|\mathbf{R}_{i^*2}]$ to decode message.

  3. If $i \neq i^*$ and $\ell = 1$, oracle reconstructs the message/level-dependent matrix $\mathbf{A}_{i,u,1}$ as

  $$\begin{aligned}\mathbf{A}_{i,u,1} &= [\mathbf{A}^*| - \mathbf{A}^*\mathbf{R}^*_{i1} + (h(1) - h(2))\mathbf{G}| - \mathbf{A}^*\mathbf{R}^*_{i2} - \mathbf{A}_{i^*3} + (h(u) - h(u^*))\mathbf{G}] \\ &= [\mathbf{A}^*| - \mathbf{A}^*\mathbf{R}^*_{i1} + (h(1) - h(2))\mathbf{G}| - \mathbf{A}^*(\mathbf{R}^*_{i2} + \mathbf{R}_{i^*2}) + (h(u) - h(u^*))\mathbf{G}]\end{aligned}$$

  where $\mathbf{R}^*_{i2} = (\mathbf{X}_{i,02} - \mathbf{R}_{i^*1}\mathbf{X}_{i,12})$. Call algorithm $\mathsf{Invert}^{\mathcal{O}}([\mathbf{R}^*_{i1}|\mathbf{R}^*_{i2} + \mathbf{R}_{i^*2}], \mathbf{A}_{i,u}, \boldsymbol{b} \bmod q)$ to get some $\boldsymbol{z} \in \mathbb{Z}^n$ and $\boldsymbol{e}$. Then oracle performs step 3 exactly as in Dec, except using $[\mathbf{R}^*_{i1}|\mathbf{R}^*_{i2} + \mathbf{R}_{i^*2}]$ to decode message.

  4. If $i \neq i^*$ and $\ell = 2$, oracle reconstructs the message/level-dependent matrix $\mathbf{A}_{i,u,2}$ as

  $$\begin{aligned}\mathbf{A}_{i,u,2} &= [\mathbf{A}^*| - \mathbf{A}^*\mathbf{R}^*_{i1}| - \mathbf{A}^*\mathbf{R}^*_{i2} - \mathbf{A}_{i^*3} + (h(u) - h(u^*))\mathbf{G}] \\ &= [\mathbf{A}^*| - \mathbf{A}^*\mathbf{R}^*_{i1}| - \mathbf{A}^*(\mathbf{R}^*_{i2} + \mathbf{R}_{i^*2}) + h(u - u^*)\mathbf{G}]\end{aligned}$$

  where $\mathbf{R}^*_{i2} = (\mathbf{X}_{i,02} - \mathbf{R}_{i^*1}\mathbf{X}_{i,12})$. Call algorithm $\mathsf{Invert}'^{\mathcal{O}}([\mathbf{R}^*_{i1}|\mathbf{R}^*_{i2} + \mathbf{R}_{i^*2}], \mathbf{A}_{i,u}, \boldsymbol{b} \bmod q)$ to get some $\boldsymbol{z} \in \mathbb{Z}^n$ and $\boldsymbol{e}$. Then oracle performs step 3 exactly as in Dec, except using $[\mathbf{R}^*_{i1}|\mathbf{R}^*_{i2} + \mathbf{R}_{i^*2}]$ to decode message.

  In summarize, the decryption oracle can answer any decryption queries for uncorrupted users as long as $u \neq u^*$, which is ensure with overwhelming probability because u^* is statistically hidden, and by the "unit difference" property on set $\mathcal{U}$ we have $h(u) - h(u^*) = h(u - u^*)$ is invertible, as require by calling $\mathsf{Invert}^{\mathcal{O}}$ algorithm.

- **Challenge oracle**: The oracle produces challenge ciphertext $(u, \boldsymbol{b}, 1)$ on a message $\boldsymbol{\mu}^* \in \{0,1\}^{nk}$ as follows. Let $u = u^*$, then the message/level-dependent matrix is $\mathbf{A}_{i^*,u^*,1} = [\mathbf{A}^*| - \mathbf{A}^*\mathbf{R}_{i^*1}| - \mathbf{A}^*\mathbf{R}_{i^*2}]$. Then oracle sets the first nk coordinates of challenge ciphertext $\boldsymbol{b}$ to be $\boldsymbol{b}_0 = \boldsymbol{b}^*$, where $(\mathbf{A}^*, \boldsymbol{b}^*)$ is the LWE instance. The last $2nk$ coordinates can be set as

  $$\boldsymbol{b}_1 = \boldsymbol{b}_0^t\mathbf{R}_{i^*1} + \boldsymbol{e}_1 \bmod 2q, \qquad \boldsymbol{b}_2 = \boldsymbol{b}_0^t\mathbf{R}_{i^*2} + \boldsymbol{e}_2 + \mathsf{encode}(\boldsymbol{\mu}^*) \bmod 2q$$

  where $\boldsymbol{e}_1, \boldsymbol{e}_2 \in \mathcal{D}^{nk}_{\mathbb{Z},s}$. Then oracle output the challenge ciphertext $\mathsf{ct} = (\mu^*, \boldsymbol{b})$.

Hybrid H_2 : In game H_2, we are given a LWE instance $(\mathbf{A}^*, \widetilde{\boldsymbol{b}^*})$, where $\widetilde{\boldsymbol{b}^*} \in \mathbb{Z}^{nk}_{2q}$ is uniformly random. We only change how the first nk coordinate of challenge ciphertext is created, by letting it be $\widetilde{\boldsymbol{b}^*}$. We construct the pubic key for each user, answer re-encryption key generation and decryption query, and construct the last $3nk$ coordinates of challenge ciphertext in the exactly way as in H_1.

Claim 3.3. *Hybrids* H_0 *and* H_1 *are statistically close.*

Proof. The adversary obtains all the public keys and re-encryption keys as he wants in hybrid H_0 and H_1, where in H_1, challenge public key is

$$\mathsf{pk}_{i^*} = [\mathbf{A}^* | - \mathbf{A}^*\mathbf{R}_{i^*1} - h(1)\mathbf{G} | - \mathbf{A}^*\mathbf{R}_{i^*2} - h(u^*)\mathbf{G}]$$

Note the challenge public key pk_{i^*} is still $\mathsf{negl}(\lambda)$-uniform for any choice of u^*, thus is identically distributed as in H_0. Therefore also conditioned on any fixed choice of pk_{i^*}, the value of u^* is statistically hidden from the adversary. For uncurrupted users other than the challenge user, we have

$$\mathbf{A}_{i1} = [\mathbf{A} | - \mathbf{A}_{i^*1}] \begin{bmatrix} \mathbf{X}_{i,01} \\ \mathbf{X}_{i,11} \end{bmatrix}, \qquad \mathbf{A}_{i2} = [\mathbf{A} | - \mathbf{A}_{i^*1}] \begin{bmatrix} \mathbf{X}_{i,02} \\ \mathbf{X}_{i,12} \end{bmatrix}$$

Since matrices $\mathbf{X}_{i,01}, \mathbf{X}_{i,11}, \mathbf{X}_{i,02}, \mathbf{X}_{i,12}$ are sampled from discrete Gaussian distribution $\mathcal{D}_{\mathbb{Z},s}^{nk \times nk}$, the public keys are identically distributed as in H_0. For re-encryption keys, because they are all sampled from discrete Gaussian distribution in hybrid $\mathsf{H}_0, \mathsf{H}_1$, they are identically distributed as well. Therefore, hybrid H_0 and H_1 are identically distributed.

We now show that the distribution of $(u^*, \boldsymbol{b})$ is within $\mathsf{negl}(\lambda)$ statistical distance of that in game H_0 from the adversary's view. Clearly, u^* and $\boldsymbol{b}_0$ have the same distribution as in H_0, because u^* is $\mathsf{negl}(\lambda)$-uniform given the public keys of uncorrupted users, and $\boldsymbol{b}_0 = 2(\boldsymbol{s}^t\mathbf{A}^* \bmod q) + \tilde{\boldsymbol{e}_0^t}$ is from the LWE instance. Since the noise item in $\boldsymbol{b}_1$ is $\tilde{\boldsymbol{e}_0^t}\mathbf{R}_{i^*1} + \boldsymbol{e}_1$, by Corollary 3.10 in [Reg05], vector $\boldsymbol{b}_1$ are within $\mathsf{negl}(\lambda)$-statistical distance from discrete Gaussian distribution $\mathcal{D}_{\mathbb{Z}^n k,s}$. The same argument also applies to $\boldsymbol{b}_2$. □

Claim 3.4. *Assuming the hardness of LWE assumption, then hybrid* H_1 *and* H_2 *are computationally indistinguishable.*

Proof. By a straightforward reduction. □

Claim 3.5. *In hybrid* H_2*, the probability of adversary wining the game is negligible in the security parameter.*

Proof. This claim can be proved by Leftover Hash Lemma 2.8,

$$(\mathbf{A}^*, \widetilde{\boldsymbol{b}^*}, -\mathbf{A}^*\mathbf{R}_{i^*1}, -\mathbf{A}^*\mathbf{R}_{i^*2}, \widetilde{\boldsymbol{b}^*}\mathbf{R}_{i^*1}, \widetilde{\boldsymbol{b}^*}\mathbf{R}_{i^*2})$$

is $\mathsf{negl}(\lambda)$-uniform when matrices $\mathbf{R}_{i^*1}, \mathbf{R}_{i^*2}, \mathbf{R}_{i^*3}$ are chosen as in H_2. Therefore, the challenge ciphertext has the same distribution (up to $\mathsf{negl}(\lambda)$ statistical distance) for any encrypted message, and so the adversarys advantage is negligible. □

Combining hybrids $\mathsf{H}_0, \mathsf{H}_1, \mathsf{H}_2$ and the above claims, we complete the proof. □

3.2 Parameter Selection

In this section, we set the lattice parameters used in our construction. $\mathbf{G} \in \mathbb{Z}_q^{n \times nk}$ is a gadget matrix for $q = \mathsf{poly}(n), n = \mathsf{poly}(\lambda)$ and $k = O(\log q) = O(\log n)$. For matrix $\mathbf{A} \in \mathbf{Z}_q^{n \times m}$ in the public parameters and secret keys $\mathbf{R} \leftarrow \mathcal{D}$, we set $m = O(nk)$ and $\mathcal{D} = \mathcal{D}_{\mathbf{Z}, w(\sqrt{\log n})}^{m \times nk}$ respectively. We set the deviation s for discrete Gaussian distribution used in security proof to be $s = w(\sqrt{\log n})$. For the error rate α in the LWE assumption, we set sufficiently large $1/\alpha = O(nk) \cdot w(\sqrt{\log n})$.

References

[ABB10] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient lattice (H)IBE in the standard model. In Gilbert [Gil10], pages 553–572.

[ABF+13] Joël Alwen, Manuel Barbosa, Pooya Farshim, Rosario Gennaro, S Dov Gordon, Stefano Tessaro, and David A Wilson. On the relationship between functional encryption, obfuscation, and fully homomorphic encryption. In *Cryptography and Coding*, pages 65–84. Springer, 2013.

[ABPW13] Yoshinori Aono, Xavier Boyen, Le Trieu Phong, and Lihua Wang. Key-private proxy re-encryption under LWE. In Goutam Paul and Serge Vaudenay, editors, *INDOCRYPT 2013*, volume 8250 of *LNCS*, pages 1–18. Springer, December 2013.

[AFGH05] Giuseppe Ateniese, Kevin Fu, Matthew Green, and Susan Hohenberger. Improved proxy re-encryption schemes with applications to secure distributed storage. In *NDSS 2005*. The Internet Society, February 2005.

[BBS98] Matt Blaze, Gerrit Bleumer, and Martin Strauss. Divertible protocols and atomic proxy cryptography. In Kaisa Nyberg, editor, *EUROCRYPT'98*, volume 1403 of *LNCS*, pages 127–144. Springer, May / June 1998.

[CCL+14] Nishanth Chandran, Melissa Chase, Feng-Hao Liu, Ryo Nishimaki, and Keita Xagawa. Re-encryption, functional re-encryption, and multi-hop re-encryption: A framework for achieving obfuscation-based security and instantiations from lattices. In Krawczyk [Kra14], pages 95–112.

[CCV12] Nishanth Chandran, Melissa Chase, and Vinod Vaikuntanathan. Functional re-encryption and collusion-resistant obfuscation. In Ronald Cramer, editor, *TCC 2012*, volume 7194 of *LNCS*, pages 404–421. Springer, March 2012.

[CH07] Ran Canetti and Susan Hohenberger. Chosen-ciphertext secure proxy re-encryption. In Peng Ning, Sabrina De Capitani di Vimercati, and Paul F. Syverson, editors, *ACM CCS 07*, pages 185–194. ACM Press, October 2007.

[CLTV] Ran Canetti, Huijia Lin, Stefano Tessaro, and Vinod Vaikuntanathan. Obfuscation of probabilistic circuits and applications. pages 468–497.

[CWYD10] Sherman SM Chow, Jian Weng, Yanjiang Yang, and Robert H Deng. Efficient unidirectional proxy re-encryption. In *Progress in Cryptology–AFRICACRYPT 2010*, pages 316–332. Springer, 2010.

[DRS04] Yevgeniy Dodis, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 523–540. Springer, May 2004.

[GGH13] Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 1–17. Springer, May 2013.

[Gil10] Henri Gilbert, editor. *EUROCRYPT 2010*, volume 6110 of *LNCS*. Springer, May 2010.

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Richard E. Ladner and Cynthia Dwork, editors, *40th ACM STOC*, pages 197–206. ACM Press, May 2008.

[HRsV07] Susan Hohenberger, Guy N. Rothblum, abhi shelat, and Vinod Vaikuntanathan. Securely obfuscating re-encryption. In Salil P. Vadhan, editor, *TCC 2007*, volume 4392 of *LNCS*, pages 233–252. Springer, February 2007.

[Kir14] Elena Kirshanova. Proxy re-encryption from lattices. In Krawczyk [Kra14], pages 77–94.

[Kra14] Hugo Krawczyk, editor. *PKC 2014*, volume 8383 of *LNCS*. Springer, March 2014.

[LL15] Kim Laine and Kristin Lauter. Key recovery for LWE in polynomial time. Cryptology ePrint Archive, Report 2015/176, 2015. `http://eprint.iacr.org/2015/176`.

[LPR10] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In Gilbert [Gil10], pages 1–23.

[LV08] Benoît Libert and Damien Vergnaud. Unidirectional chosen-ciphertext secure proxy re-encryption. In Ronald Cramer, editor, *PKC 2008*, volume 4939 of *LNCS*, pages 360–379. Springer, March 2008.

[MP12] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 700–718. Springer, April 2012.

[NAL15] David Nunez, Isaac Agudo, and Javier Lopez. A parametric family of attack models for proxy re-encryption. In *Computer Security Foundations Symposium (CSF), 2015 IEEE 28th*, pages 290–301. IEEE, 2015.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th ACM STOC*, pages 84–93. ACM Press, May 2005.

[SCL10] Jun Shao, Zhenfu Cao, and Peng Liu. CCA-Secure PRE scheme without random oracles. Cryptology ePrint Archive, Report 2010/112, 2010. `http://eprint.iacr.org/2010/112`.

[Smi05] Tony Smith. *DVD Jon: Buy DRM-less Tracks from Apple iTunes*, 2005. `http://www.theregister.co.uk/2005/03/18/itunes_pymusique/`.

A Multi-hop CCA*-Secure PRE Construction

In this section, we present our construction for multi-hop CCA* secure PRE scheme as defined in Definition 2.4. The construction can be viewed as a natural variant of the single-hop scheme presented in Section 3. The maximum times of re-encryption applied to the ciphertext depends on the types of LWE assumptions. If we rely on sub-exponential LWE assumption, then we can apply poly-log times of re-encryption. If we build our scheme based on standard LWE assumption, then we can apply re-encryption constant times. We elaborate this tradeoff in the parameter setting in Section A.2.

We construct the large set $\mathcal{U}$ and encode messages $\mu \in \{0,1\}^{nk}$ using the same technique as the single-hop case as Section 3. Here instead of encoding the level ℓ in the encryption matrix (recall that in the previous scheme $\ell \in \{1,2\}$), in the multi-hop scheme, each user chooses a random public element $v_i \in \mathcal{U}$. The encryption algorithm will choose a random element $u \in \mathcal{U}$ (as a tag) for each encryption, and embeds the matrices $h(v_i), h(u)$ into the encryption matrix as: $\mathbf{A}_{i,u} = [\mathbf{A}|-\mathbf{A}_{i1}+h(v_i)\mathbf{G}|-\mathbf{A}_{i2}+h(u)\mathbf{G}]$. We describe the scheme in detail as follows:

- $\mathsf{Setup}(1^\lambda, H)$: On input the security parameter λ and the maximum allowed re-encryption hop H, the setup algorithm set the lattice parameter $n = n(\lambda, H), m = m(\lambda, H), q = q(\lambda, H)$ and Gaussian parameter $s = s(\lambda, H)$. Then it randomly selects a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, and outputs the public parameter $\mathsf{pp} = (\mathbf{A}, n, m, q, s)$.

- $\mathsf{KeyGen}(\mathsf{pp})$: On input the public parameter pp, the key generation algorithm for i-th user chooses a random element $v_i \in \mathcal{U}$, and random matrices $\mathbf{R}_{i1}, \mathbf{R}_{i2} \leftarrow \mathcal{D}$, letting $\mathbf{A}_{i1} = \mathbf{A}\mathbf{R}_{i1} \bmod q$ and $\mathbf{A}_{i2} = \mathbf{A}\mathbf{R}_{i2} \bmod q$. The public key is $\mathsf{pk}_i = ([\mathbf{A}| - \mathbf{A}_{i1}| - \mathbf{A}_{i2}], h(v_i))$, and the secret key is $\mathsf{sk}_i = [\mathbf{R}_{i1}|\mathbf{R}_{i2}]$.

- $\mathsf{Enc}(\mathsf{pk}_i, \boldsymbol{\mu})$: On input a public key pk_i, a message $\boldsymbol{\mu}$, the encryption algorithm chooses non-zero $u \leftarrow \mathcal{U}$ and let the message/user-dependent matrix

  $$\mathbf{A}_{i,u} = [\mathbf{A}| - \mathbf{A}_{i1} + h(v_i)\mathbf{G}| - \mathbf{A}_{i2} + h(u)\mathbf{G}]$$

  The rest of encryption algorithm is the same as encryption algorithm in single-hop CCA-PRE scheme described in Section 3.

- $\mathsf{Dec}(\mathsf{sk}_i, \mathsf{ct})$: On input a secret key sk_i and ciphertext $\mathsf{ct} = (u, \boldsymbol{b})$, the decryption algorithm

  1. If ct does not parse or $u = 0$, output $\perp$. Otherwise, reconstruct the message/user-dependent matrix $\mathbf{A}_{i,u}$

     $$\mathbf{A}_{i,u} = [\mathbf{A}| - \mathbf{A}_{i1} + h(v_i)\mathbf{G}| - \mathbf{A}_{i2} + h(u)\mathbf{G}]$$

     The rest of decryption algorithm is the same as decryption algorithm in single-hop CCA-PRE scheme described in Section 3.

- $\mathsf{ReKeyGen}(\mathsf{pk}_i, \mathsf{sk}_i, \mathsf{pk}_j)$: On input a public/secret key pair $(\mathsf{pk}_i, \mathsf{sk}_i)$ from the i-th user and public key pk_j from j-th user, the re-encryption key generation algorithm do:

  1. First, it parses i-th public/secret key pair and j-th public key as follows:

     $$\mathsf{pk}_i = ([\mathbf{A}| - \mathbf{A}_{i1}| - \mathbf{A}_{i2}], h(v_i)), \quad \mathsf{sk}_i = [\mathbf{R}_{i1}|\mathbf{R}_{i2}], \quad \mathsf{pk}_j = ([\mathbf{A}| - \mathbf{A}_{j1}| - \mathbf{A}_{j2}], h(v_j))$$

  2. Use extended sampling algorithm $\mathsf{Sample}^{\mathcal{O}}$ to sample $\mathbf{X}_{01}, \mathbf{X}_{02}, \mathbf{X}_{11}, \mathbf{X}_{12} \in \mathbb{Z}^{nk \times nk}$ such that

     $$[\mathbf{A}| - \mathbf{A}_{i1} + h(v_i)\mathbf{G}] \begin{bmatrix} \mathbf{X}_{01} \\ \mathbf{X}_{11} \end{bmatrix} = -\mathbf{A}_{j1} + h(v_j)\mathbf{G},$$

     $$[\mathbf{A}| - \mathbf{A}_{i1} + h(v_i)\mathbf{G}] \begin{bmatrix} \mathbf{X}_{02} \\ \mathbf{X}_{12} \end{bmatrix} = -\mathbf{A}_{j2} + \mathbf{A}_{i2}$$

Therefore, it holds that

$$[\mathbf{A}| - \mathbf{A}_{i1} + h(v_i)\mathbf{G}| - \mathbf{A}_{i2} + \mathbf{B}] \begin{bmatrix} \mathbf{I} & \mathbf{X}_{01} & \mathbf{X}_{02} \\ 0 & \mathbf{X}_{11} & \mathbf{X}_{12} \\ 0 & 0 & \mathbf{I} \end{bmatrix} = [\mathbf{A}| - \mathbf{A}_{j1} + h(v_j)\mathbf{G}| - \mathbf{A}_{j2} + \mathbf{B}]$$

for any matrix $\mathbf{B} \in \mathbb{Z}^{n \times nk}$.

3. Output the re-encryption key

$$\mathsf{rk}_{i \to j} = \{\mathbf{X}_{01}, \mathbf{X}_{02}, \mathbf{X}_{11}, \mathbf{X}_{12}\}$$

- $\mathsf{ReEnc}(\mathsf{rk}_{i \to j}, \mathsf{ct})$: The re-encryption algorithm for tree-based PRE is the same as the counterpart in single-hop CCA-PRE secure scheme in 3.

It is obviously to see that with appropriate parameter setting as specified in Section A.2, the correctness of the construction can be implied by the correctness of previous scheme as proof for Lemma 3.1.

A.1 Security Proof

In this section, we prove a weaker version of Definition 2.4 – a selective CCA*-PRE security, where the adversary first commits to a tree structure before the experiment proceeds. Given the tree $\mathcal{T}$ beforehand, the reduction can simulate the public keys and re-encryption keys in a specified order. On the other hand, if the structure is unknown, then it is unclear how the reduction can simulate all the rk's in an arbitrary order of queries. This limitation comes from current techniques in security proof, yet there are no known attacks to the construction, to our knowledge. It is an interesting open problem to determine whether the construction achieves the full security notion as defined in Definition 2.4.

Next, we briefly describe the intuition behind our proof. For example, suppose User two is a child of User one in tree $\mathcal{T}$, then reduction first simulates pk_1 and $\mathsf{rk}_{1 \leftarrow 2}$, and then multiply them to get pk_2. The reduction can still provide decryption oracle because the tag of message in encryption matrix $\mathbf{A}_{i,u} = [\mathbf{A}| - \mathbf{A}_{i1} + h(v_i)\mathbf{G}| - \mathbf{A}_{i2} + h(u)\mathbf{G}]$ would not vanish. The limitation in this approach is that the reduction has to know the tree structure beforehand, and it cannot generalize to acyclic graph since it is not clear how to coordinate the public key from two different paths of simulation.

Theorem A.1. *Assuming the hardness of* $\mathsf{LWE}_{q,\alpha'}$ *where* $\alpha' = \alpha/3 \geq 2\sqrt{n}/q$*, the proxy re-encryption scheme is a tree-based CCA*-secure as defined in Definition 2.4 for any tree structure* $\mathcal{T}$*.*

Proof. First, we obtain the desired LWE form $(\mathbf{A}^*, \boldsymbol{b}^*)$ by using the same technique described in proof of Theorem 3.2. We proceed via a sequence of games.

Hybrid H_0 : The game H_0 is exactly the CCA* attack with the real system described above.

Hybrid H_1 : In game H_1, we change how the public keys and re-encryption key for users in the tree $\mathcal{T}$ and challenge ciphertext $\mathsf{ct}^* = (u^*, \boldsymbol{b}^*)$ are constructed, and the way that decryption queries are answered, but in a way that only introduces $\mathsf{negl}(\lambda)$ statistical distance with game H_0. Suppose adversary $\mathcal{A}$ commits the tree structure $\mathcal{T} = (E, V, r)$ at the beginning of the game. We let the matrix in public parameter pp to be $\mathbf{A} = \mathbf{A}^*$, and choose a random element $u^* \in \mathcal{U}$. The changes can be described in detail as follows:

- **Uncorrupted public and re-encryption key generation:** To generate a public key for the root user $r \in V$, the oracle chooses a random element $v_r \in \mathcal{U}$ and random matrices $\mathbf{R}_{r1}, \mathbf{R}_{r2}$ from $\{-1, 1\}^{m \times m}$, then output the public key to be

$$\mathsf{pk}_r = ([\mathbf{A}| - \mathbf{A}\mathbf{R}_{r1} - h(v_r)\mathbf{G}| - \mathbf{A}\mathbf{R}_{r2} - h(u^*)\mathbf{G}], h(v_r))$$

For generation of other vertices in the tree $\mathcal{T}$, the oracle computes the public key by induction. For descendant i of the root vertex r, the oracle firstly chooses and stores random element $v_i \in \mathcal{U}$ and matrices $\mathbf{X}_{i,01}, \mathbf{X}_{i,11}, \mathbf{X}_{i,02}, \mathbf{X}_{i,12}$ from $\mathcal{D}_{\mathbb{Z},s}^{nk\times nk}$, and set

$$\mathbf{A}_{i1} = [\mathbf{A}| - \mathbf{A}_{r1}] \begin{bmatrix} \mathbf{X}_{i,01} \\ \mathbf{X}_{i,11} \end{bmatrix}, \qquad \mathbf{A}_{i2} = [\mathbf{A}| - \mathbf{A}_{r1}] \begin{bmatrix} \mathbf{X}_{i,02} \\ \mathbf{X}_{i,12} \end{bmatrix}$$

where $\mathbf{A}_{r1} = -\mathbf{A}\mathbf{R}_{r1}, \mathbf{A}_{r2} = -\mathbf{A}\mathbf{R}_{r2}$. Then the oracle outputs the public key for vertex i as

$$\mathsf{pk}_i = ([\mathbf{A}| - \mathbf{A}_{i1} - h(v_i)\mathbf{G}| - \mathbf{A}_{i2} + \mathbf{A}_{r2} - h(u^*)\mathbf{G}], h(v_i))$$

For descendant k of vertex j, where j is a descendant of vertex i, the oracle firstly chooses and stores random element $v_k \in \mathcal{U}$, matrices $\mathbf{X}_{k,01}, \mathbf{X}_{k,11}, \mathbf{X}_{k,02}, \mathbf{X}_{k,12}$ from $\mathcal{D}_{\mathbb{Z},s}^{nk\times nk}$, and set

$$\mathbf{A}_{k1} = [\mathbf{A}|\mathbf{A}_{j1}] \begin{bmatrix} \mathbf{X}_{k,01} \\ \mathbf{X}_{k,11} \end{bmatrix} \qquad \mathbf{A}_{k2} = [\mathbf{A}|\mathbf{A}_{j1}] \begin{bmatrix} \mathbf{X}_{k,02} \\ \mathbf{X}_{k,12} \end{bmatrix}$$

where $\mathbf{A}_{j1} = -\mathbf{A}(\mathbf{X}_{j,01} - \mathbf{X}_{i,01}\mathbf{X}_{j,11} + \cdots + \mathbf{R}_{r1}\cdots\mathbf{X}_{i,01}\mathbf{X}_{j,11}), \mathbf{A}_{r2} = -\mathbf{A}(\mathbf{X}_{j,02} - \mathbf{X}_{i,02}\mathbf{X}_{j,12} + \cdots + \mathbf{R}_{r2}\cdots\mathbf{X}_{i,02}\mathbf{X}_{j,12})$. Then the oracle outputs the public key for vertex k as

$$\mathsf{pk}_k = ([\mathbf{A}| - \mathbf{A}_{k1} - h(v_k)\mathbf{G}| - \mathbf{A}_{k2} + \mathbf{A}_{j2} - h(u^*)\mathbf{G}], h(v_k))$$

For re-encryption key from vertex i to vertex j, if $(i,j) \in E$, then output re-encryption key as the stored matrices $\{\mathbf{X}_{j,01}, \mathbf{X}_{j,11}, \mathbf{X}_{j,02}, \mathbf{X}_{j,12}\}$ sampled in the generation of public key for vertex j.

- **Decryption oracle:** On decryption query (i, ct) from adversary $\mathcal{A}$, the oracle first parse ciphertext $\mathsf{ct} = (u, \boldsymbol{b})$, and output $\perp$ if $u = 0$. If the decryption queries are made after the challenge oracle query, then oracle outputs $\perp$ if $u = u^*$. Then oracle divides the decryption process into following two cases:
  1. If $i = r$, the oracle first re-constructs the message/user-dependent matrix as

  $$\begin{aligned} \mathbf{A}_{r,u} &= [\mathbf{A}| - \mathbf{A}_{r1} - h(v_r)\mathbf{G} + h(v_r)\mathbf{G}|\mathbf{A}_{r2} - h(u^*)\mathbf{G} + h(u)\mathbf{G}] \\ &= [\mathbf{A}| - \mathbf{A}_{r1}|\mathbf{A}_{r2} + h(u-u^*)\mathbf{G}] \end{aligned}$$

  where $\mathbf{A}_{r1} = \mathbf{A}\mathbf{R}_{r1}, \mathbf{A}_{r2} = \mathbf{A}\mathbf{R}_{r2}$. Call $\mathsf{Invert}'^{\mathcal{O}}([\mathbf{R}_{r1}|\mathbf{R}_{r2}], \mathbf{A}_{r,u}, \boldsymbol{b} \bmod q)$ to get some $\boldsymbol{z} \in \mathbb{Z}^n$ and $\boldsymbol{e}$. Then oracle performs step 3 exactly as in Dec, except using $[\mathbf{R}_{r1}|\mathbf{R}_{r2}]$ to decode message.
  2. If $i \neq r$, and suppose vertex i is the descendant of vertex j, the oracle first re-constructs the message/user-dependent matrix as

  $$\begin{aligned} \mathbf{A}_{i,u} &= [\mathbf{A}| - \mathbf{A}_{i1} - h(v_i)\mathbf{G} + h(v_i)\mathbf{G}| - \mathbf{A}_{i2} + \mathbf{A}_{j2} - h(u^*)\mathbf{G} + h(u)\mathbf{G}] \\ &= [\mathbf{A}| - \mathbf{A}_{i1}| - \mathbf{A}_{i2} + \mathbf{A}_{j2} + h(u-u^*)\mathbf{G}] \end{aligned}$$

  where $\mathbf{A}_{i1} = \mathbf{A}\mathbf{T}_{i1} = \mathbf{A}(\mathbf{X}_{i,01} - \mathbf{X}_{j,01}\mathbf{X}_{i,11} + \cdots + \mathbf{R}_{r1}\cdots\mathbf{X}_{j,01}\mathbf{X}_{i,11})$, and $\mathbf{A}_{i2} - \mathbf{A}_{j2} = \mathbf{A}\mathbf{T}_{i2}$ for small matrix $\mathbf{T}_i$ computed from the expression of matrices $\mathbf{A}_{i2}, \mathbf{A}_{j2}$. Then Call $\mathsf{Invert}'^{\mathcal{O}}([\mathbf{T}_{i1}|\mathbf{T}_{i2}], \mathbf{A}_{r,u}, \boldsymbol{b} \bmod q)$ to get some $\boldsymbol{z} \in \mathbb{Z}^n$ and $\boldsymbol{e}$. Then oracle performs step 3 exactly as in Dec, except using $[\mathbf{T}_{i1}|\mathbf{T}_{i2}]$ to decode message.
- **Challenge oracle:** The oracle produces challenge ciphertext $(u, \boldsymbol{b})$ on a challenge query $(i^*, \boldsymbol{\mu})$ as follows. Let $u = u^*$, then the message/user-dependent matrix is $\mathbf{A}_{i^*,u^*} = [\mathbf{A}^*| - \mathbf{A}^*\mathbf{T}_{i^*1}| - \mathbf{A}^*\mathbf{T}_{i^*2}]$. Then oracle sets the first nk coordinates of challenge ciphertext $\boldsymbol{b}$ to be $\boldsymbol{b}_0 = \boldsymbol{b}^*$, where $(\mathbf{A}^*, \boldsymbol{b}^*)$ is the LWE instance. The last $2nk$ coordinates can be set as

$$\boldsymbol{b}_1 = \boldsymbol{b}_0^t\mathbf{T}_{i^*1} + \boldsymbol{e}_1 \bmod 2q, \qquad \boldsymbol{b}_2 = \boldsymbol{b}_0^t\mathbf{T}_{i^*2} + \boldsymbol{e}_2 + \mathsf{encode}(\boldsymbol{\mu}^*) \bmod 2q$$

where $\boldsymbol{e}_1, \boldsymbol{e}_2 \in \mathcal{D}_{\mathbb{Z},s}^{nk}$. Then oracle output the challenge ciphertext $\mathsf{ct} = (\mu^*, \boldsymbol{b})$. We now show that the distribution of $(u^*, \boldsymbol{b})$ is within $\mathsf{negl}(\lambda)$ statistical distance of that in game H_0 from the adversary's view. Clearly, u^* and $\boldsymbol{b}_0$ have the same distribution as in H_0, because u^* is $\mathsf{negl}(\lambda)$-uniform given the public keys of uncorrupted users, and $\boldsymbol{b}_0 = 2(\boldsymbol{s}^t \mathbf{A}^* \bmod q) + \tilde{\boldsymbol{e}_0^t}$ is from the LWE instance. Since the noise item in $\boldsymbol{b}_1$ is $\tilde{\boldsymbol{e}_0^t}\mathbf{R}_{i^*1} + \boldsymbol{e}_1$, by Corollary 3.10 in [Reg05], vector $\boldsymbol{b}_1$ are within $\mathsf{negl}(\lambda)$-statistical distance from discrete Gaussian distribution $\mathcal{D}_{\mathbb{Z}^n k,s}$. The same argument also applies to $\boldsymbol{b}_2$.

Hybrid H_3 : The same as the hybrid H_2 in the proof of Theorem 3.2.

Claim A.2. *Hybrids* H_0 *and* H_1 *are statistically close.*

Claim A.3. *Assuming the hardness of LWE assumption, then hybrid* H_1 *and* H_2 *are computationally indistinguishable.*

Claim A.4. *In hybrid* H_2*, the probability of adversary wining the game is negligible in the security parameter.*

The proof of the above claims is analogous to tht proofs in Theorem 3.2, thus we omit the detail here. Combining hybrids $\mathsf{H}_0, \mathsf{H}_1, \mathsf{H}_2$ and the above claims, we complete the proof. □

A.2 Parameter Selection

$\mathbf{G} \in \mathbb{Z}_q^{n \times nk}$ is a gadget matrix for $k = O(\log q) = O(\log n)$. For matrix $\mathbf{A} \in \mathbf{Z}_q^{n \times m}$ in the public parameters and secret keys $\mathbf{R} \leftarrow \mathcal{D}$, we set $m = O(nk)$ and $\mathcal{D} = \mathcal{D}_{\mathbf{Z}, w(\sqrt{\log n})}^{m \times nk}$ respectively. We set the deviation s for discrete Gaussian distribution used in security proof to be $s = w(\sqrt{\log n})$. After each re-encryption the noise will grow a $\sqrt{m}s$ factor. For the error rate α in the LWE assumption, we set sufficiently large $1/\alpha = O(nk) \cdot w(\sqrt{\log n})$. We use H to denote the maximum depth of the tree structure used in our construction, and we have $H = O(n/\log n)$. In order to achieve fixed poly-log depth H, so q has to set to be greater then $m^{H/2}$. Thus we have to rely sub-exponential LWE to set $q = 2^{n^\epsilon}, 0 < \epsilon < 1$, then $n = 2^{\lambda^\epsilon}, 0 < \epsilon < 1$. We notice that in [LL15], Laine and Lauter proposed an attack for LWE assumption when q is exponentially large (i.e. 2^{cn}), but in our setting, $q = 2^{n^\epsilon}$ is sub-exponentially in terms of the security parameter, so our assumption still holds. If we want to rely on standard LWE assumption, then we can achieve constant depth by setting $q = \mathsf{poly}(n), n = \mathsf{poly}(\lambda)$.

B Multi-Hop CPA-Secure PRE Construction

Our construction, with slight modifications, can be extended to support multi-hop with respect to general acyclic graphs, but the scheme falls back to the CPA-security. This extension is inspired by the re-encryption key generation idea introduced in [CCL$^+$14].

The Setup algorithm is the same as that of the single-hop scheme in Section 3, while the KeyGen is slightly different. Here we can use a simpler encryption matrix of the form $\mathbf{A}_i = [\mathbf{A}| - \mathbf{A}_{i1} + h(v_i)\mathbf{G}]$ (recall that v_i is a random element that User i owns), and the encryption algorithm does not need to sample a random element $u \in \mathcal{U}$ (as a tag). Therefore, the key generation algorithm only needs to generate $\mathbf{A}_{i1} = \mathbf{A} \cdot \mathbf{R}_{i1}$, and sets $\mathbf{A}_{i1}$ and $\mathbf{R}_{i1}$ as the public/secret keys (for User i). These two algorithms can be viewed as a simplified version of the previous scheme, so we omit the details. In the following, we present the encryption, and the re-encryption procedure.

$\mathsf{Enc}(\mathsf{pk}_i, \boldsymbol{\mu})$: On input a public key pk_i, a message $\boldsymbol{\mu}$, the encryption algorithm sets the user-dependent matrix

$$\mathbf{A}_i = [\mathbf{A}| - \mathbf{A}_{i1} + h(v_i)\mathbf{G}]$$

Choose $\boldsymbol{s} \leftarrow \mathbb{Z}_q^n, \boldsymbol{e}_0 \leftarrow \mathcal{D}_{\mathbb{Z},s}^m$ and $\boldsymbol{e}_1 \leftarrow \mathcal{D}_{\mathbb{Z},s}^{nk}$. Let

$$\boldsymbol{b}^t = (\boldsymbol{b}_0, \boldsymbol{b}_1) = 2(\boldsymbol{s}^t \mathbf{A}_i \bmod q) + \boldsymbol{e}^t + (0, \mathsf{encode}(\boldsymbol{\mu})^t) \bmod 2q$$

where $\boldsymbol{e} = (\boldsymbol{e}_0, \boldsymbol{e}_1)$. Output the ciphertext $\mathsf{ct} = \boldsymbol{b}$.

$\mathsf{ReKeyGen}(\mathsf{pk}_i, \mathsf{sk}_i, \mathsf{pk}_j)$: On input a public/secret key pair $(\mathsf{pk}_i, \mathsf{sk}_i)$ and a public key pk_j, the re-encryption key generation algorithm dose:

1. First, parse i-th public/secret key pair and j-th public key as follows:

$$\mathsf{pk}_i = ([\mathbf{A}| - \mathbf{A}_{i1}], h(v_i)), \qquad \mathsf{sk}_i = \mathbf{R}_{i1}, \qquad \mathsf{pk}_j = ([\mathbf{A}| - \mathbf{A}_{j1}], h(v_j))$$

   Then sample random matrix $\mathbf{U} \in \mathbb{Z}_q^{n \times m}$ and short noise matrices $\mathbf{V} \in \mathcal{D}_{\mathbb{Z},s}^{n \times n}, \mathbf{E}_0,$ $\mathbf{E}_1, \in \mathcal{D}_{\mathbb{Z},s}^{n \times m}$.

2. Use extended sampling algorithm $\mathsf{Sample}^{\mathcal{O}}$ to sample

$$\mathbf{X}_{00}, \mathbf{X}_{10}, \mathbf{X}_{01}, \mathbf{X}_{11} \in \mathbb{Z}^{nk \times nk}$$

   such that

$$[\mathbf{A}| - \mathbf{A}_{i1} + h(v_i)\mathbf{G}] \begin{bmatrix} \mathbf{X}_{00} \\ \mathbf{X}_{10} \end{bmatrix} = \mathbf{V}\mathbf{A} + \mathbf{E}_0$$

$$[\mathbf{A}| - \mathbf{A}_{i1} + h(v_i)\mathbf{G}] \begin{bmatrix} \mathbf{X}_{01} \\ \mathbf{X}_{11} \end{bmatrix} = \mathbf{V}(-\mathbf{A}_{j1} + h(v_j)\mathbf{G}) + \mathbf{E}_1$$

   Therefore, it holds that

$$\begin{aligned} &[\mathbf{A}| - \mathbf{A}_{i1} + h(v_i)\mathbf{G}] \begin{bmatrix} \mathbf{X}_{00} & \mathbf{X}_{01} \\ \mathbf{X}_{10} & \mathbf{X}_{11} \end{bmatrix} \\ &= [\mathbf{V}\mathbf{A} + \mathbf{E}_1 | \mathbf{V}(-\mathbf{A}_{j1} + h(v_j)\mathbf{G}) + \mathbf{E}_2] \end{aligned}$$

3. Output the re-encryption key

$$\mathsf{rk}_{i \to j} = \{\mathbf{X}_{00}, \mathbf{X}_{01}, \mathbf{X}_{10}, \mathbf{X}_{11}\}$$

Correctness. We provide a sketch proof below to show that the correctness still holds regarding the modification described above. The parameter selection for our CPA-secure PRE construction is the same as that of multi-hop CPA-secure scheme described in Section A.2, thus we omit the details here.

Lemma B.1 (Correctness)**.** *The correctness of the modified multi-hop PRE holds regarding polynomial proxy re-encryptions with appropriate parameters.*

Proof. For simplicity, we only prove the case that the ciphertext is re-encrypted from freshly generated ciphertext. Assume a ciphertext $\mathsf{ct} = \boldsymbol{b}$ is an encryption of message μ under user i's public key and then re-encrypted by the proxy to user j, thus first we have

$$\boldsymbol{b}^t = 2(\boldsymbol{s}^t \mathbf{A}_i \bmod q) + \boldsymbol{e}^t + (0, \mathsf{encode}(\boldsymbol{\mu})^t) \bmod 2q$$

where $\mathbf{A}_i = [\mathbf{A}| - \mathbf{A}_{i1} + h(v_i)\mathbf{G}| - \mathbf{A}_{i2} + \mathbf{G}]$. The re-encryption process from user i to j can be re-phased as follows:

$$\begin{aligned} &\boldsymbol{s}^t [\mathbf{A}| - \mathbf{A}_{i1} + h(v_i)\mathbf{G}] \begin{bmatrix} \mathbf{X}_{00} & \mathbf{X}_{01} \\ \mathbf{X}_{10} & \mathbf{X}_{11} \end{bmatrix} + \ \mathsf{noise} \ + (0, \mathsf{encode}(\boldsymbol{\mu})^t) \\ &= \boldsymbol{s}^t [\mathbf{V}\mathbf{A} + \mathbf{E}_0 | \mathbf{V}(-\mathbf{A}_{j1} + h(v_j)\mathbf{G}) + \mathbf{E}_1] + \ \mathsf{noise} \ + (0, \mathsf{encode}(\boldsymbol{\mu})^t) \\ &= \boldsymbol{s}^t [\mathbf{V}\mathbf{A}_j + \mathbf{E}_0 | \mathbf{E}_1] + \ \mathsf{noise} \ + (0, \mathsf{encode}(\boldsymbol{\mu})^t) \end{aligned}$$

The noise terms is defined analogously as in previous algorithm ReEnc, thus it is obviously that the re-encrypted ciphertext can be decrypted using j-th secret key. □

We prove the security of the above construction in the CPA-PRE model, where adversary dose not have access to decryption oracle in security experiment. The security proof can be obtained from the previous proofs with some minor modifications, thus we only present a proof sketch below.

Theorem B.2 (CPA-secure PRE)**.** *Assuming the hardness of* $\mathsf{LWE}_{q,\alpha'}$ *where* $\alpha' = \alpha/3 \geq 2\sqrt{n}/q$*, the scheme is a graph-based CPA-secure as defined in Definition 2.5 for any general acyclic graph.*

Proof. First, we obtain the desired LWE form $(\mathbf{A}^*, \boldsymbol{b}^*)$ by using the same technique described in proof of Theorem 3.2. We proceed via a sequence of games.

Hybrid H_0 : The game H_0 is exactly the CPA-PRE attack with the real system described above.

Hybrid H_1 : In game H_1, we change the way to generate uncorrupted pk, challenge ciphertext ct^* and re-encryption keys rk, that are hard to distinguish from the counterparts in game H_0. We set the public parameter matrix $\mathbf{A} = \mathbf{A}^*$, where $\mathbf{A}^*$ is from LWE instance $(\mathbf{A}^*, \boldsymbol{b}^*)$.

- **Uncorrupted key generation oracle**: To obtain a public key for uncorrupted user of the user i, the oracle chooses random matrices $\mathbf{R}_{i1}$ from $\{-1, 1\}^{m\times m}$ and a random element $v_i \in \mathcal{U}$, then output public key to be
$$\mathsf{pk}_{i^*} = ([\mathbf{A}| - \mathbf{A}\mathbf{R}_{i1} - h(v_i)\mathbf{G}], h(v_i))$$
- **Re-encryption key generation oracle**: Since there is no decryption oracle in CPA security model, adversary cannot check the correctness of re-encrypted ciphertext. Moreover, because the re-encrypted user-dependent matrix
$$[\mathbf{V}\mathbf{A} + \mathbf{E}_0|\mathbf{V}(-\mathbf{A}_{j1} + h(v_j)\mathbf{G}) + \mathbf{E}_1]$$
where matrices $\mathbf{V}, \mathbf{E}_0, \mathbf{E}_1$ are sampled from discrete Gaussian distribution, the distribution remains computationally indistinguishable from random distribution by the LWE assumption. Therefore, we can sample some matrices $\mathbf{X}_{00}, \mathbf{X}_{10}, \mathbf{X}_{01}$,
$\mathbf{X}_{11} \leftarrow \mathcal{D}_{\mathbb{Z},s}^{nk\times nk}$, which are distributed identically as in hybrid H_0, and output them as re-encryption key from user i to j.
- **Challenge oracle**: The challenge ciphertext is generated using the same technique described as in Theorem 3.2, with some minor modifications to fit the encryption algorithm defined above, thus we omit the details here.

Hybrid H_3 : The same as the hybrid H_2 in the proof of Theorem 3.2.

Claim B.3. *Hybrids* H_0 *and* H_1 *are statistically close.*

Claim B.4. *Assuming the hardness of LWE assumption, then hybrid* H_1 *and* H_2 *are computationally indistinguishable.*

Claim B.5. *In hybrid* H_2*, the probability of adversary wining the game is negligible in the security parameter.*

The proof of the above claims is analogous to the proofs in Theorem 3.2, thus we omit the detail here. Combining hybrids $\mathsf{H}_0, \mathsf{H}_1, \mathsf{H}_2$ and the above claims, we complete the proof. □