






Fine-Grained Proxy Re-Encryption: Definitions & Constructions from LWE

Yunxiao Zhou^{1,2}, Shengli Liu^{2,3}  , Shuai Han^{1,2}  , and Haibin Zhang⁴ 

¹ School of Cyber Science and Engineering,
Shanghai Jiao Tong University, Shanghai 200240, China
`{cloudzhou,dalen17}@sjtu.edu.cn`

² State Key Laboratory of Cryptology, P.O. Box 5159, Beijing 100878, China

³ Department of Computer Science and Engineering,
Shanghai Jiao Tong University, Shanghai 200240, China
`slliu@sjtu.edu.cn`

⁴ Beijing Institute of Technology, Beijing 100081, China
`bchainzhang@aliyun.com`

Abstract. Proxy re-encryption (PRE) allows a proxy with a re-encryption key to translate a ciphertext intended for Alice (delegator) to another ciphertext intended for Bob (delegatee) without revealing the underlying message. However, with PRE, Bob can obtain the whole message from the re-encrypted ciphertext, and Alice cannot take flexible control of the extent of the message transmitted to Bob.

In this paper, we propose a new variant of PRE, called Fine-Grained PRE (FPRE), to support fine-grained re-encryptions. An FPRE is associated with a function family \mathcal{F} , and each re-encryption key $rk_{A \rightarrow B}^f$ is associated with a function $f \in \mathcal{F}$. With FPRE, Alice now can authorize re-encryption power to proxy by issuing $rk_{A \rightarrow B}^f$ to it, with f chosen by herself. Then the proxy can translate ciphertext encrypting m to Bob's ciphertext encrypting $f(m)$ with such a fine-grained re-encryption key, and Bob only obtains a function of message m . In this way, Alice can take flexible control of the message spread by specifying functions.

For FPRE, we formally define its syntax and formalize security notions including CPA security, ciphertext pseudo-randomness, unidirectionality, non-transitivity, collusion-safety under adaptive corruptions in the multi-user setting. Moreover, we propose a new security notion named *ciphertext unlinkability*, which blurs the link between a ciphertext and its re-encrypted ciphertext to hide the proxy connections between users. We establish the relations between those security notions.

As for constructions, we propose two FPRE schemes, one for bounded linear functions and the other for deletion functions, based on the learning-with-errors (LWE) assumption. Our FPRE schemes achieve all the aforementioned desirable securities under adaptive corruptions in the standard model. As far as we know, our schemes provide the *first* solution to PRE with security under adaptive corruptions in the standard model.

1 Introduction

A proxy re-encryption (PRE) scheme is a public-key encryption (PKE) scheme augmented with two functionalities. One is the generation of re-encryption key $rk_{i \rightarrow j}$ for user i and user j . The other is ciphertext re-encryption which translates a ciphertext $ct^{(i)}$ encrypting message m under user i 's public key $pk^{(i)}$ to a ciphertext $ct^{(j)}$ encrypting the same message m under user j 's public key $pk^{(j)}$. With PRE, user i (delegator) can authorize re-encryption power to a proxy by issuing a re-encryption key $rk_{i \rightarrow j}$. Then the proxy can use $rk_{i \rightarrow j}$ to accomplish the ciphertext translation from $ct^{(i)}$ to $ct^{(j)}$, which further enables user j (delegatee) to recover the message. Beyond the traditional semantic security of PKE, PRE also requires that the knowledge of $rk_{i \rightarrow j}$ does not help any proxy to gain (in a computational sense) any information on the message encrypted in ciphertexts $ct^{(i)}$ and $ct^{(j)}$.

PRE has found lots of applications since introduced by Blaze et al. [4]. For example, a patient i may issue a re-encryption key $rk_{i \rightarrow j}$ to a hospital. When he receives his own medical testing report $ct^{(i)}$ encrypted under his public key $pk^{(i)}$ and would like to see a doctor for diagnosis, he can forward $ct^{(i)}$ to the hospital. Then the hospital converts $ct^{(i)}$ to $ct^{(j)}$ under doctor j 's public key, and the doctor can use his/her own secret key $sk^{(j)}$ to decrypt $ct^{(j)}$ to recover the patient's original medical testing report, which helps him for disease diagnosis.

- **Unidirectional vs. Bidirectional.** A PRE scheme is *unidirectional* if $rk_{i \rightarrow j}$ only allows re-encryption from $pk^{(i)}$ to $pk^{(j)}$ but *not vice versa*. In contrast, a *bidirectional* PRE scheme allows bidirectional ciphertext translations between $pk^{(i)}$ and $pk^{(j)}$ with a single $rk_{i \leftrightarrow j}$. Compared to unidirectional PRE, the proxies in a bidirectional PRE scheme are authorized more re-encryption power, and this is not welcomed especially when the other direction is not permitted by user j . Therefore, unidirectional PRE is preferable to its bidirectional counterpart. Moreover, as shown by [6], a unidirectional PRE implies a bidirectional one.

- **Single-hop vs. Multi-hop.** A PRE scheme is *single-hop* if a re-encrypted ciphertext cannot be further re-encrypted by any re-encryption key. In contrast, with a *multi-hop* PRE, a ciphertext $ct^{(i)}$ is translated to $ct^{(j)}$ by $rk_{i \rightarrow j}$, and $ct^{(j)}$ can be further translated to $ct^{(k)}$ by $rk_{j \rightarrow k}$. With the multi-hop property, a malicious proxy may lead to lost of control of authorization. For example, a proxy with $rk_{i \rightarrow j}, rk_{j \rightarrow k}$ can easily obtains re-encryption power from i to k which may be undesirable for user i .

- **Non-Interactive vs. Interactive.** A PRE scheme is *non-interactive* if the generation of re-encryption key $rk_{i \rightarrow j}$ does not need user j 's secret key $sk^{(j)}$. In contrast, an *interactive* PRE needs $sk^{(j)}$, and hence user j must be on-line and involved in the generation of $rk_{i \rightarrow j}$. Clearly the non-interactive property is preferable to the interactive one.

In this paper, we focus on *unidirectional*, *single-hop* and *non-interactive* PRE.

Security of PRE. PRE is usually deployed in multi-user settings, where the adversary is able to corrupt some users by obtaining their secret keys, and it is

also able to obtain re-encryption keys between some users. The main security notion for a PRE scheme is indistinguishability under chosen-plaintext attacks (CPA) or chosen-ciphertext attacks (CCA) for some challenge ciphertext ct^* under some target user i^* , against probabilistic polynomial-time (PPT) adversaries who corrupt users and obtain re-encryption keys of its own choices. Of course, the knowledge of corrupted secret keys and re-encryption keys should not lead to trivial decryption of ct^* . According to the way that the adversary corrupts the users, there are two types of security notions.

- **Security under selective corruptions.** At the beginning of the security game, the adversary submits a set of users that it wants to corrupt, and the challenger returns all the secret keys of users in the corruption set.
- **Security under adaptive corruptions.** Throughout the security game, the adversary issues corruption queries adaptively.

Obviously, CPA/CCA security under adaptive corruptions is stronger than that under selective corruptions. In [12], Fuchsbauer et al. proposed a security reduction from selective corruptions to adaptive corruptions for PRE, but it suffers from a super-polynomial security loss $n^{O(\log n)}$ with n the number of users. To the best of our knowledge, all existing PRE schemes with adaptive corruptions are based on the Random Oracle (RO) model, and there is no PRE scheme achieving even CPA security under adaptive corruptions in the standard model.

Similarly, the rest of security notions including CPR, UNID, NTR, CUL, CS can be defined either under selective corruptions or under adaptive corruptions.

- **Ciphertext Pseudo-Randomness (CPR).** CPR is similar to but stronger than CPA security. It requires that the challenge ciphertext is computationally indistinguishable to an element randomly chosen from the ciphertext space.
- **Unidirectionality (UNID).** Roughly speaking, unidirectionality requires that it is hard for adversaries to compute $rk_{i \rightarrow j}$ with the knowledge of $rk_{j \rightarrow i}$.
- **Non-Transitivity (NTR).** For a single-hop unidirectional PRE scheme, non-transitivity requires that it is hard for adversaries to compute $rk_{i \rightarrow k}$ even with the knowledge of $rk_{i \rightarrow j}$ and $rk_{j \rightarrow k}$. It is easy to see that NTR, as well as UNID, captures the precise authorization of re-encryption power.
- **Ciphertext Unlinkability (CUL).** For a single-hop unidirectional PRE scheme, if $ct^{(j)}$ is encrypted from $ct^{(i)}$ with re-encryption key $rk_{i \rightarrow j}$, then $ct^{(j)}$ is linked to $ct^{(i)}$. Ciphertext unlinkability requires that the linked ciphertext pair $(ct^{(i)}, ct^{(j)})$ is computationally indistinguishable from independently generated ciphertexts $ct^{(i)}$ and $ct^{(j)}$. The indistinguishability should also be considered in the corruption scenario in the multi-user settings. As far as we know, there is no formal security definition for ciphertext-unlinkability yet.
- **Collusion-Safety (CS).** For a single-hop unidirectional PRE scheme, collusion-safety requires that it is hard for adversaries to compute secret key $sk^{(i)}$ even with the knowledge of $rk_{i \rightarrow j}$ and $sk^{(j)}$. This notion is also called *master secret security* in [3, 9, 16, 21, 22, 23]. Note that secret key $sk^{(i)}$ may not be unique

w.r.t. $pk^{(i)}$. Therefore, in this paper, we consider a stronger notion of Collusion-Safety (CS), which requires the CPA security of the ciphertext under public key $pk^{(i)}$ against adversaries who obtains $rk_{i \rightarrow j}, sk^{(j)}$ and can also corrupt users and issue re-encryption queries. Clearly, CS implies the master secret security.

Related Works. Let us recall existing works on single-hop unidirectional PRE. In [23], Shao et al. designed the first CCA-secure unidirectional single-hop PRE scheme from the DDH assumption under adaptive corruptions in the RO model. However, Chow et al. [9] showed an attack on the scheme of [23], and presented a fixed PRE scheme, which achieves CCA-security but only under selective corruptions in the RO model. Moreover, Selvi et al. [21] pointed out a weakness in the proof in [9] and presented a PRE scheme achieving CCA security under selective corruptions, also in the RO model. Later, Canard et al. [5] proposed a CCA-secure PRE scheme under adaptive corruptions again in the RO model.

As for standard model, Ateniese et al. [3] designed the first unidirectional single-hop PRE scheme from the DBDH assumption, achieving weak-CPA security. Later, Libert et al. [16] designed the first CCA-secure scheme from 3-QDBDH assumption. In [14], Kirshanova proposed the first lattice-based weak CCA1-secure scheme. However, Fan et al. [11] pointed out a mistake of the proof in [14] and presented a new lattice-based scheme, achieving tag-based CCA (tbCCA) security, with a security level between weak-CCA1 in [14] and CCA security in [16]. Unfortunately, these schemes only achieve selective security in the standard model.

There are also a variety of PRE with extended functionalities. Shao [22] proposed the notion of anonymous identity-based PRE and presented a scheme achieving CCA security under adaptive corruptions in the RO model. Chandran et al. [7, 8] generalized PRE to functional re-encryption scheme and constructed functional PRE schemes from obfuscations, which are secure under selective corruption. In their schemes, the policy function F defines the access policy. Only the policy is satisfied, can a user decrypt the re-encrypted ciphertext to recover the original message successfully. Similarly, Weng et al. [25] and Liang et al. [15] proposed attribute-based conditional PRE schemes to achieve attribute-based policy control, and their schemes are CPA secure under selective corruptions. Recently, Miao et al. [17] proposed a unidirectional multi-hop updatable PRE from DDH with security under selective corruptions in the standard model.

The related works on PRE shows that there is no PRE or its variant schemes achieving CPA or CCA security under adaptive corruptions in the standard model.¹ It is natural to ask:

Q1: Can we construct a PRE scheme meeting the CPA security under adaptive corruptions in the standard model, possibly also achieving ciphertext unlinkability, unidirectionality, non-transitivity and collusion-safety?

¹ In fact, to the best of our knowledge, there is no PRE with security under adaptive corruptions in the standard model, no matter single-hop or multi-hop, unidirectional or bidirectional, interactive or non-interactive PREs.

Fine-Grained PRE. Up to now, the re-encryption of $ct^{(i)}$ under $pk^{(i)}$ only generates $ct^{(j)}$ under $pk^{(j)}$ encrypting the same message as $ct^{(i)}$. This is an *all-or-nothing* style of re-encryption.

Let us take the patient-hospital-doctor example again. With PRE, either a doctor sees all the medical testing data, say (m_1, m_2, m_3) , in the reports or nothing at all. In fact, the patient may only want to reveal a part of the data, like $(m_1, *, m_3)$, to the doctor and hide some sensitive data m_2 from the doctor. Even more generally, the patient may only allow revealing a function of his data, say $f(m_1, m_2, m_3)$, to another party. If we resort to PRE, then the PRE must be able to do a fine-grained re-encryption authorization to proxies. More precisely, re-encryption key $rk_{i \rightarrow j}^f$ for user i and user j must be associated with a function f . With $rk_{i \rightarrow j}^f$, the proxy is authorized with re-encryption power, but limited by function f . Accordingly, the proxy is only able to translate a ciphertext $ct^{(i)}$ encrypting message m under user i 's public key $pk^{(i)}$ to a ciphertext $ct^{(j)}$ encrypting $f(m)$ under $pk^{(j)}$ of user j . Such a fine-grained single-hop unidirectional PRE is able to accurately control message spread to other parties by means of functions $\{f\}$. Up to now, there is no work on this topic. Thus, question Q1 is now upgraded to another interesting question:

Q2: *Can we construct a Fine-Grained PRE scheme achieving the CPA security under adaptive corruptions in the standard model, possibly also achieving ciphertext unlinkability, unidirectionality, non-transitivity and collusion-safety?*

Our Contributions. In this work, we answer the above question in the affirmative. Our contributions are three-fold.

- *Formal Definitions for Fine-Grained PRE and Its Securities.* We present the formal definitions for the new concept Fine-Grained PRE (FPRE), which generalizes PRE by enabling fine-grained re-encryption power.

Moreover, we present the formal definitions for a set of security notions for FPRE, including CPA security, ciphertext pseudorandomness (CPR), unidirectionality (UNID), non-transitivity (NTR) and collusion-safety (CS). We also propose a new security notion named ciphertext unlinkability (CUL), which blurs the link between a ciphertext and its re-encrypted ciphertext. All the security notions are formalized in a multi-user setting where the adversary is able to corrupt users and obtain re-encryption keys *adaptively*.

We establish the relations between these security notions: CPA implies both UNID and NTR, CUL implies CPA, and CPR (trivially) implies CPA. See Fig. 1 for an overview.

- *Construction of FPRE for Bounded Linear Functions from LWE.* We propose a unidirectional, single-hop, non-interactive FPRE scheme $\text{FPRE}_{\text{LWE}}^{\text{lin}}$, and the fine-grained function family consists of bounded linear functions \mathcal{F}_{lin} (with coefficients of bounded norm). Our $\text{FPRE}_{\text{LWE}}^{\text{lin}}$ achieves CPA, UNID, NTR, CS and CUL security *under adaptive corruptions in the standard model*, based on the learning-with-errors (LWE) assumption. The LWE assumption makes our scheme quantum-safe. In addition, our scheme is *key-optimal* (in

the sense of [3]), where the delegatee’s secret storage remains constant regardless of the number of delegations it accepts.

When setting the linear function to be the identity function, we immediately get a single-hop unidirectional (traditional) PRE scheme, which contributes as the *first* PRE scheme with security under adaptive corruptions in the standard model.

- *Construction of FPRE for Deletion Functions from LWE.* As a by-product, our $\text{FPRE}_{\text{LWE}}^{\text{lin}}$ for bounded linear functions can be easily adapted to a scheme $\text{FPRE}_{\text{LWE}}^{\text{del}}$ for deletion functions \mathcal{F}_{del} , which can be applied in various realistic scenarios.

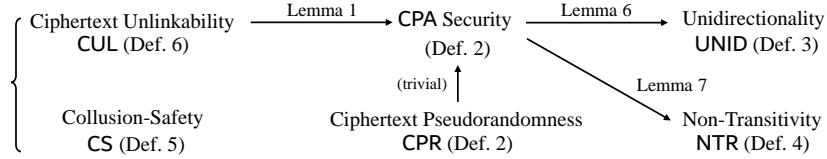


Fig. 1. Security notions of FPRE under adaptive corruptions and their relations.

We refer to Table 1 for a comparison of our scheme with known single-hop unidirectional PRE schemes.²

Technical Overview of Our LWE-based FPRE Scheme. Below we give a high-level overview of our FPRE scheme $\text{FPRE}_{\text{LWE}}^{\text{lin}}$ for the bounded linear function family based on LWE, and in particular, explain how we realize fine-grained re-encryptions. For simplicity, we do not specify the dimensions of matrices/vectors.

We start with the dual Regev PKE scheme [20] encrypting a multiple-bit message $\mathbf{m} \in \{0, 1\}^\ell$. The ciphertext under user i ’s public key $\mathbf{A}^{(i)} = \begin{pmatrix} \overline{\mathbf{A}}^{(i)} \\ \underline{\mathbf{A}}^{(i)} \end{pmatrix}$ is

$$ct^{(i)} = \mathbf{A}^{(i)} \mathbf{s} + \mathbf{e} + \begin{pmatrix} \mathbf{0} \\ \lfloor q/2 \rfloor \mathbf{m} \end{pmatrix} = \begin{pmatrix} \overline{\mathbf{A}}^{(i)} \mathbf{s} + \mathbf{e}_1 \\ \underline{\mathbf{A}}^{(i)} \mathbf{s} + \mathbf{e}_2 + \lfloor q/2 \rfloor \cdot \mathbf{m} \end{pmatrix}, \quad (1)$$

where $\mathbf{e} = \begin{pmatrix} \mathbf{e}_1 \\ \mathbf{e}_2 \end{pmatrix}$ is an error vector.

² We explain the security notions in Table 1: “weak-CPA/CCA1” does not allow the adversary to issue any re-encryption key query from an honest user to a corrupted user, while “CPA/CCA” allows such queries (except for trivial attacks). “tbCCA” refers to tag-based CCA and was first introduced in [11], with a security level between weak-CCA1 and CCA. “HRA” refers to security against honest re-encryption attacks), proposed in [10], and it does not allow such re-encryption key query, but provides re-encryption oracle to answer re-encryptions of honestly generated ciphertexts for corrupted users. On the one hand, HRA does not allow the adversary to obtain any re-encryption key from the honest user to the corrupted user, which is weaker than our CPA; on the other hand, the adversary in the HRA model can obtain re-encryptions of the honestly-generated ciphertexts from the challenge user to the corrupted user, which is not allowed in our CPA model.

Table 1. Comparison of single-hop unidirectional PRE schemes. The column **Standard Model?** asks whether the security is proved in the standard model. The column **Adaptive Corruptions?** asks whether all the security notions support adaptive corruptions. The column **Security** shows the type of security that the scheme achieves, where “HRA” refers to security against honest re-encryption attack [10] and “tbCCA” refers to tag-based CCA [11]. The column **UNID** shows whether the scheme has unidirectionality. The column **CUL** shows whether the scheme has ciphertext unlinkability. The column **NTR** shows whether the scheme has non-transitivity. The column **CS** shows whether the scheme is collusion-safe. The column **Assumption** shows the assumptions that the security of the scheme is based on, where DBDH refers to the Decision Bilinear Diffie-Hellman assumption, DCDH refers to the Divisible CDH assumption and 3-QDBDH refers to the 3-Quotient DBDH assumption. The column **Post Quantum?** asks whether the scheme is based on a post quantum assumption like LWE. The column **Fine-Grained?** asks whether the scheme supports fine-grained re-encryptions. “–” means that no proof is provided.

PRE Scheme	Standard Model?	Adaptive Corruptions?	Security	UNID	NTR	CUL	CS	Assumption	Post Quantum?	Fine-Grained?
SC09 [23]	×	✓	CCA	✓	✓	–	✓	DDH	×	×
CWYD10 [9]	×	×	CCA	✓	✓	–	✓	CDH	×	×
CDL11 [5]	×	✓	CCA	✓	✓	–	✓	CDH	×	×
Shao12 [22]	×	✓	CCA	✓	✓	–	✓	DBDH	×	×
SPR17 [21]	×	×	CCA	✓	✓	–	✓	DCDH	×	×
AFGH05 [3]	✓	×	weak-CPA	✓	–	–	✓	DBDH	×	×
LV08 [16]	✓	×	CCA	✓	✓	–	✓	3-QDBDH	×	×
Kirshanova14 [14]	✓	×	weak-CCA1	✓	–	–	–	LWE	✓	×
FL19 [11]	✓	×	tbCCA	✓	✓	–	–	LWE	✓	×
SDDR21 [24]	✓	×	HRA	✓	✓	–	–	LWE	✓	×
LWYYJW21 [15]	✓	×	CPA	✓	–	–	–	LWE	✓	×
This work	✓	✓	CPA	✓	✓	✓	✓	LWE	✓	✓

(a) **The generation of re-encryption key.** Let \mathbf{I} be the identity matrix.

Multiplying a small-norm matrix $\left(\mathbf{R} \mid \begin{smallmatrix} \mathbf{0} \\ \mathbf{I} \end{smallmatrix}\right)$ to $ct^{(i)}$ yields

$$\left(\mathbf{R} \mid \begin{smallmatrix} \mathbf{0} \\ \mathbf{I} \end{smallmatrix}\right) \cdot ct^{(i)} = \underbrace{\left(\mathbf{R}\overline{\mathbf{A}}^{(i)} + \begin{smallmatrix} \mathbf{0} \\ \mathbf{I} \end{smallmatrix}\mathbf{A}^{(i)}\right)}_{(*)} \cdot \mathbf{s} + \underbrace{\mathbf{R}\mathbf{e}_1 + \begin{smallmatrix} \mathbf{0} \\ \mathbf{I} \cdot \mathbf{e}_2 \end{smallmatrix}}_{\mathbf{e}'}} + \begin{smallmatrix} \mathbf{0} \\ \lfloor q/2 \rfloor \cdot \mathbf{I} \cdot \mathbf{m} \end{smallmatrix}.$$

With the help of the trapdoor $\mathbf{T}^{(i)}$ of $\overline{\mathbf{A}}^{(i)}$, a small-norm \mathbf{R} can be found with the pre-image sampling algorithm [13] so that $\mathbf{R}\overline{\mathbf{A}}^{(i)} = \mathbf{A}^{(j)} - \begin{smallmatrix} \mathbf{0} \\ \mathbf{I} \end{smallmatrix}\mathbf{A}^{(i)}$. Consequently, $\mathbf{A}^{(j)} = (*)$ and $ct^{(j)} := \left(\mathbf{R} \mid \begin{smallmatrix} \mathbf{0} \\ \mathbf{I} \end{smallmatrix}\right) \cdot ct^{(i)} = \mathbf{A}^{(j)} \cdot \mathbf{s} + \mathbf{e}' + \begin{smallmatrix} \mathbf{0} \\ \lfloor q/2 \rfloor \cdot \mathbf{m} \end{smallmatrix}$, which can be decrypted to recover message \mathbf{m} by user j . Taking \mathbf{R} as the re-generation key $rk_{i \rightarrow j}$, then we can translate $ct^{(i)}$ to $ct^{(j)}$ successfully.

(b) **The CPA security of $ct^{(i)}$.** There is a dilemma to prove the CPA security of $ct^{(i)}$: applying the LWE assumption to $ct^{(i)}$ requires $\mathbf{A}^{(i)}$ be a uniformly distributed matrix with trapdoor unknown; but the generation of $rk_{i \rightarrow j} = \mathbf{R}$ needs a trapdoor $\mathbf{T}^{(i)}$. Moreover, we can hardly change the generation of \mathbf{R} to

avoid using $\mathbf{T}^{(i)}$ since the relation between public keys $pk^{(i)} = \mathbf{A}^{(i)}, pk^{(j)} = \mathbf{A}^{(j)}$ and the re-generation key \mathbf{R} can be easily checked by $\mathbf{R}\overline{\mathbf{A}}^{(i)} = \mathbf{A}^{(j)} - \begin{pmatrix} \mathbf{0} \\ \mathbf{I} \end{pmatrix} \mathbf{A}^{(i)}$.

To solve the problem, we change the way of generating $rk_{i \rightarrow j} = \mathbf{R}$: now \mathbf{R} is sampled with the pre-image sampling algorithm so that $\mathbf{R}\overline{\mathbf{A}}^{(i)} = \mathbf{A}^{(j)} \cdot \mathbf{S} + \mathbf{E} - \begin{pmatrix} \mathbf{0} \\ \mathbf{I} \end{pmatrix} \mathbf{A}^{(i)}$, instead of $\mathbf{R}\overline{\mathbf{A}}^{(i)} = \mathbf{A}^{(j)} - \begin{pmatrix} \mathbf{0} \\ \mathbf{I} \end{pmatrix} \mathbf{A}^{(i)}$, where \mathbf{S} and \mathbf{E} are small-norm matrices following discrete Gaussian distribution. Thus, $\mathbf{A}^{(j)} \cdot \mathbf{S} + \mathbf{E} = (*)$ and

$$ct^{(j)} := \left(\mathbf{R} \mid \begin{matrix} \mathbf{0} \\ \mathbf{I} \end{matrix} \right) \cdot ct^{(i)} = \mathbf{A}^{(j)} \cdot \mathbf{s} + \underbrace{\mathbf{E} \cdot \mathbf{s} + \mathbf{e}'}_{\mathbf{e}''} + \begin{pmatrix} \mathbf{0} \\ \lfloor q/2 \rfloor \cdot \mathbf{I} \cdot \mathbf{m} \end{pmatrix}. \quad (2)$$

When \mathbf{s} is a small-norm vector, \mathbf{e}'' is small enough so that $ct^{(j)}$ can also be successfully decrypted by user j 's secret key.

When targeting on the CPA security of $ct^{(i)}$, the adversary must not corrupt j if it has already obtained $rk_{i \rightarrow j} = \mathbf{R}$ (to avoid trivial attacks). According to the LWE assumption, $\mathbf{A}^{(j)} \cdot \mathbf{S} + \mathbf{E}$ is computationally indistinguishable to a uniform distribution, and so is $\mathbf{R}\overline{\mathbf{A}}^{(i)} \left(= \mathbf{A}^{(j)} \cdot \mathbf{S} + \mathbf{E} - \begin{pmatrix} \mathbf{0} \\ \mathbf{I} \end{pmatrix} \mathbf{A}^{(i)} \right)$. Then we can modify the generation of $rk_{i \rightarrow j} = \mathbf{R}$ by sampling from a discrete Gaussian distribution independently. According to [13], such a \mathbf{R} makes $\mathbf{R}\overline{\mathbf{A}}^{(i)}$ uniformly distributed. Therefore, the adversary hardly realizes this modification of \mathbf{R} . Now the generation of \mathbf{R} is free of trapdoor $\mathbf{T}^{(i)}$, and the LWE assumption assures the pseudo-randomness of $ct^{(i)}$.

Up to now, we obtain a multi-hop PRE since $ct^{(j)}$ can be similarly re-encrypted to $ct^{(k)}$ with $rk_{j \rightarrow k}$ generated in a similar way.

(c) Achieving single-hop with ciphertexts of two levels. We resort to two level ciphertexts to achieve single-hop for PRE, inspired by [3, 16]. The first-level ciphertext $ct_1^{(i)}$ is defined as in (1). The second-level ciphertext $ct_2^{(i)} = \mathbf{A}'^{(i)} \mathbf{s} + \mathbf{e} + \begin{pmatrix} \mathbf{0} \\ \lfloor q/2 \rfloor \mathbf{m} \end{pmatrix}$ is also a dual Regev ciphertext. Then user i 's public key is $pk^{(i)} = (\mathbf{A}^{(i)}, \mathbf{A}'^{(i)})$ and secret key is $sk^{(i)} = (\mathbf{T}^{(i)}, \mathbf{K}^{(i)})$ with $\mathbf{A}'^{(i)} = \mathbf{K}^{(i)} \overline{\mathbf{A}}'^{(i)}$. Now the re-encryption key $rk_{i \rightarrow j}$ translates the first-level ciphertext $ct_1^{(i)}$ under i to the second-level ciphertext $ct_2^{(j)}$ under j . If user j has no trapdoor for $\overline{\mathbf{A}}'^{(j)}$, then he cannot generate re-encryption key $rk_{j \rightarrow k}$ for further hops. This can be accomplished by putting $\overline{\mathbf{A}}' (= \overline{\mathbf{A}}'^{(j)})$ in public parameter and shared by all users. In this way, we obtain a single-hop PRE.

(d) Achieving fine-grained re-encryption for deletion functions and bounded linear functions. According to (2), the re-encrypted ciphertext $ct_2^{(j)}$ is in fact an encryption of $\mathbf{I} \cdot \mathbf{m}$. If \mathbf{I} is replaced by a binary matrix \mathbf{M} (of bounded norm), then $rk_{i \rightarrow j} := \left(\mathbf{R} \mid \begin{matrix} \mathbf{0} \\ \mathbf{M} \end{matrix} \right)$ and $ct_2^{(j)}$ becomes an encryption of $\mathbf{M} \cdot \mathbf{m}$, which is a bounded linear function of \mathbf{m} . For example, setting $\mathbf{M} := \mathbf{I}'$ with \mathbf{I}' is almost equal to \mathbf{I} except that its v -th row is a zero-row, then the v -th entry in $\mathbf{I} \cdot \mathbf{m}$ is erased to 0. This highlights the idea of designing fine-grained PRE with deletion functions (see Sect. 5 for details.) To support larger message space and more

linear functions, we set $q = p^2$ and replace $\lfloor q/2 \rfloor$ with p in ciphertexts. In this way, the message space and (small-norm) \mathbf{M} can be defined over \mathbb{Z}_p .

(e) Achieving adaptive security for fine-grained PRE. The CPA security of fine-grained PRE focuses on the first-level ciphertext $ct_1^{(i)}$ for the target user i , even if the adversary obtains re-encryption keys and corrupts users adaptively. We allow the adversary obtains $rk_{i' \rightarrow j'}$ and corrupts j' to obtain $sk^{(j')}$, as long as $i' \neq i$ ($i' = i$ will lead to a trivial attack). Therefore, our CPA security is actually stronger than the CPA notion defined in [3, 10, 12], where $rk_{i' \rightarrow j'}$ is invisible to the adversary when i' is uncorrupted and j' is corrupted.

To prove adaptive CPA security, we first use the guessing strategy for the target user i , and this only leads to a security loss of n , the number of users. Note that if j is corrupted, then the adversary cannot see $rk_{i \rightarrow j}$ (to avoid trivial attacks). But the adversary is able to obtain $rk_{i \rightarrow j}$ for all uncorrupted j . For those $rk_{i \rightarrow j}$, the generation of \mathbf{R} can be indistinguishably changed to independently sampling \mathbf{R} with some discrete Gaussian distribution, thanks to the LWE assumption which makes $\mathbf{R}\mathbf{A}^{(i)} (= \mathbf{A}^{(j)} \cdot \mathbf{S} + \mathbf{E} - \binom{\mathbf{0}}{\mathbf{I}}\mathbf{A}^{(i)})$ pseudo-random. As a result, all $rk_{i \rightarrow j}$ w.r.t. uncorrupted j are independent of the challenge ciphertext $ct_1^{(i)}$. Meanwhile, the generations of $rk_{i' \rightarrow j'}$ and $sk^{(i')}$ with $i' \neq i$ are only related to $\mathbf{A}^{(i')}$, and hence are also independent of $ct_1^{(i)}$. Then we can use the LWE assumption to prove the pseudo-randomness of the challenge ciphertext $ct_1^{(i)}$, and CPA security follows.

Moreover, the sampling of \mathbf{R} retains enough entropy, and a uniform $ct_1^{(i)}$ can act as an extractor on \mathbf{R} so that $ct_2^{(j)} := \left(\mathbf{R} \mid \begin{smallmatrix} \mathbf{0} \\ \mathbf{I} \end{smallmatrix} \right) \cdot ct_1^{(i)}$ is statistically close to a uniform distribution. Hence it is hard for an adversary to realize the link between $ct_1^{(i)}$ and its re-encrypted ciphertext $ct_2^{(j)}$, thus CUL security is achieved.

It is easy to check that CPA security implies unidirectionality (UNID) and non-transitivity (NTR). In the CPA security, adversary is able to see $rk_{i' \rightarrow i}$ and a corrupted secret key $sk^{(i')}$. If UNID does not hold, then the adversary can recover $rk_{i \rightarrow i'}$ from $rk_{i' \rightarrow i}$ and obtain the ability to decrypt the challenge ciphertext of i from $rk_{i \rightarrow i'}$ and $sk^{(i')}$, thus breaking the CPA security. Similarly, in the CPA security, adversary is able to see $rk_{i \rightarrow j}$, $rk_{j \rightarrow k}$ and a corrupted secret key $sk^{(k)}$. If NTR does not hold, the adversary can recover $rk_{i \rightarrow k}$ from $rk_{i \rightarrow j}$ and $rk_{j \rightarrow k}$. Then it obtains the ability to decrypt the challenge ciphertext of i from $rk_{i \rightarrow k}$ and $sk^{(k)}$, thus breaking the CPA security.

2 Preliminaries

Notations. Let $\lambda \in \mathbb{N}$ denote the security parameter throughout the paper, and all algorithms, distributions, functions and adversaries take 1^λ as an implicit input. If x is defined by y or the value of y is assigned to x , we write $x := y$. For $i, j \in \mathbb{N}$ with $i < j$, define $[i, j] := \{i, i+1, \dots, j\}$ and $[j] := \{1, 2, \dots, j\}$. For a set \mathcal{X} , denote by $x \leftarrow_s \mathcal{X}$ the procedure of sampling x from \mathcal{X} uniformly at random. If \mathcal{D} is distribution, $x \leftarrow_s \mathcal{D}$ means that x is sampled according to \mathcal{D} .

All our algorithms are probabilistic unless stated otherwise. We use $y \leftarrow_s \mathcal{A}(x)$ to define the random variable y obtained by executing algorithm \mathcal{A} on input x . If \mathcal{A} is deterministic we write $y \leftarrow \mathcal{A}(x)$. “PPT” abbreviates probabilistic polynomial-time. Denote by negl some negligible function. By $\Pr_i[\cdot]$ we denote the probability of a particular event occurring in game \mathbf{G}_i .

For random variables X and Y , the min-entropy of X is defined as $\mathbf{H}_\infty(X) := -\log(\max_x \Pr[X = x])$, and the statistical distance between X and Y is defined as $\Delta(X, Y) := \frac{1}{2} \cdot \sum_x |\Pr[X = x] - \Pr[Y = x]|$. If $\Delta(X, Y) = \text{negl}(\lambda)$, we say that X and Y are statistically indistinguishable (close), and denote it by $X \approx_s Y$.

Let $n, m, m', q \in \mathbb{N}$, and let $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$, $\mathbf{v} \in \mathbb{Z}_q^n$, $\mathbf{B} \in \mathbb{Z}_q^{m' \times n}$. Define the lattice $\Lambda(\mathbf{A}) := \{\mathbf{A}\mathbf{x} \mid \mathbf{x} \in \mathbb{Z}^n\}$, the q -ary lattice $\Lambda_q(\mathbf{A}) := \{\mathbf{A}\mathbf{x} \mid \mathbf{x} \in \mathbb{Z}_q^n\} + q\mathbb{Z}^m$, its “orthogonal” lattice $\Lambda_q^\perp(\mathbf{A}) := \{\mathbf{x} \in \mathbb{Z}^m \mid \mathbf{x}^\top \mathbf{A} = \mathbf{0} \pmod{q}\}$, and the “shifted” lattice $\Lambda_q^\mathbf{v}(\mathbf{A}) := \{\mathbf{r} \in \mathbb{Z}^m \mid \mathbf{r}^\top \mathbf{A} = \mathbf{v}^\top \pmod{q}\}$, which can be further extended to $\Lambda_q^\mathbf{B}(\mathbf{A}) := \{\mathbf{R} \in \mathbb{Z}^{m' \times m} \mid \mathbf{R}\mathbf{A} = \mathbf{B} \pmod{q}\}$. Let $\|\mathbf{v}\|$ (resp., $\|\mathbf{v}\|_\infty$) denote its ℓ_2 (resp., infinity) norm. For a matrix \mathbf{A} , we define $\|\mathbf{A}\|$ (resp., $\|\mathbf{A}\|_\infty$) as the largest ℓ_2 (resp., infinity) norm of \mathbf{A} ’s rows. A distribution χ is B -bounded if its support is limited to $[-B, B]$. Let \mathbb{Z}_q be the ring of integers modulo q , and its elements are represented by the integers in $(-q/2, q/2]$.

Due to space limitations, we present lattice backgrounds in Appendix A.1, where we recall the definition of discrete Gaussian distribution, LWE assumption, and the TrapGen, Invert, SamplePre algorithms introduced in [1, 13, 19].

3 Fine-Grained PRE

In this section, we propose a new primitive called *Fine-Grained PRE* (FPRE), by extending the concept of proxy re-encryption (PRE) to fine-grained settings. (For completeness, we recall the formal definition of PRE in Appendix A.2.) Moreover, we formalize a set of security notions for FPRE, including CPA security, ciphertext pseudorandomness (CPR), unidirectionality (UNID), non-transitivity (NTR), collusion safety (CS), and ciphertext unlinkability (CUL), all of which are under *adaptive corruptions*. We refer to Fig. 1 in the introduction for an overview of the relations between these security notions.

Single-hop PRE can prevent the proxies from spreading ciphertexts without permission, and it is usually achieved by ciphertexts of two levels. Only the first level ciphertexts can be re-encrypted (to second level ciphertexts) and the second level ciphertexts can not be re-encrypted anymore. Accordingly, we define FPRE with two encryption and two decryption algorithms.

Now we present the syntax of fine-grained PRE.

Definition 1 (Fine-Grained PRE). *Let \mathcal{F} be a family of functions from \mathcal{M} to \mathcal{M} , where \mathcal{M} is a message space. A fine-grained proxy re-encryption (FPRE) scheme for function family \mathcal{F} is defined with a tuple of PPT algorithms $\text{FPRE} = (\text{Setup}, \text{KGen}, \text{FReKGen}, \text{Enc}_1, \text{Enc}_2, \text{FReEnc}, \text{Dec}_1, \text{Dec}_2)$.*

- $\text{pp} \leftarrow_s \text{Setup}$: *The setup algorithm outputs a public parameter pp , which serves as an implicit input of other algorithms.*

- $(pk, sk) \leftarrow_s \text{KGen}(\text{pp})$: Taking pp as input, the key generation algorithm outputs a pair of public key and secret key (pk, sk) .
- $rk_{i \rightarrow j}^f \leftarrow_s \text{FReKGen}(pk^{(i)}, sk^{(i)}, pk^{(j)}, f)$: Taking as input a public-secret key pair $(pk^{(i)}, sk^{(i)})$, another public key $pk^{(j)}$ and a function $f \in \mathcal{F}$, the fine-grained re-encryption key generation algorithm outputs a fine-grained re-encryption key $rk_{i \rightarrow j}^f$ that allows re-encrypting ciphertexts intended to i into ciphertexts encrypted for j .
- $ct_1 \leftarrow_s \text{Enc}_1(pk, m)$: Taking as input a public key pk and a message $m \in \mathcal{M}$, this algorithm outputs a first-level ciphertext ct_1 that can be further re-encrypted into a second-level ciphertext.
- $ct_2 \leftarrow_s \text{Enc}_2(pk, m)$: Taking as input a public key pk and a message $m \in \mathcal{M}$, this algorithm outputs a second-level ciphertext ct_2 that cannot be re-encrypted anymore.
- $ct_2^{(j)} \leftarrow_s \text{FReEnc}(rk_{i \rightarrow j}^f, ct_1^{(i)})$: Taking as input a re-encryption key $rk_{i \rightarrow j}^f$ and a first-level ciphertext intended for i , the fine-grained re-encryption algorithm outputs a second-level ciphertext re-encrypted for user j .
- $m \leftarrow \text{Dec}_1(sk, ct_1)$: Taking as input a secret key sk and a first-level ciphertext ct_1 , the deterministic decryption algorithm outputs a message m .
- $m \leftarrow \text{Dec}_2(sk, ct_2)$: Taking as input a secret key sk and a second-level ciphertext ct_2 , the deterministic decryption algorithm outputs a message m .

Correctness. For all $m \in \mathcal{M}$, $\text{pp} \leftarrow_s \text{Setup}$, $(pk, sk) \leftarrow_s \text{KGen}(\text{pp})$, $ct_1 \leftarrow_s \text{Enc}_1(pk, m)$ and $ct_2 \leftarrow_s \text{Enc}_2(pk, m)$, it holds that $\text{Dec}_1(sk, ct_1) = m = \text{Dec}_2(sk, ct_2)$.

Fine-Grained One-Hop Correctness. For all $m \in \mathcal{M}$, $f \in \mathcal{F}$, $\text{pp} \leftarrow_s \text{Setup}$, $(pk^{(i)}, sk^{(i)}) \leftarrow_s \text{KGen}(\text{pp})$, $(pk^{(j)}, sk^{(j)}) \leftarrow_s \text{KGen}(\text{pp})$, $rk_{i \rightarrow j}^f \leftarrow_s \text{FReKGen}(pk^{(i)}, sk^{(i)}, pk^{(j)}, f)$, $ct_1^{(i)} \leftarrow_s \text{Enc}_1(pk^{(i)}, m)$ and $ct_2^{(j)} \leftarrow_s \text{FReEnc}(rk_{i \rightarrow j}^f, ct_1^{(i)})$, it holds

$$\text{Dec}_2(sk^{(j)}, ct_2^{(j)}) = f(m).$$

The notion of FPRE generalizes PRE by enabling fine-grained re-encryption, which is captured by the two fine-grained algorithms FReKGen and FReEnc. With a fine-grained re-encryption key $rk_{i \rightarrow j}^f$ generated by FReKGen, one can convert a first-level ciphertext $ct_1^{(i)}$ encrypting message m for user i into a second-level ciphertext $ct_2^{(j)}$ encrypting a function $f(m)$ of m for another user j by invoking FReEnc, where $f \in \mathcal{F}$. If the function family \mathcal{F} consists of only the identity function, i.e., $f(m) = m$, then we recover the (traditional) PRE.

Next, we formalize the indistinguishability of ciphertexts under chosen-plaintext attacks (CPA) and Ciphertext Pseudorandomness (CPR) for FPRE.

Definition 2 (CPA Security & Ciphertext Pseudorandomness for FPRE).

An FPRE scheme FPRE is CPA secure, if for any PPT adversary \mathcal{A} and any polynomial n , it holds that $\text{Adv}_{\text{FPRE}, \mathcal{A}, n}^{\text{CPA}}(\lambda) := \left| \Pr[\text{Exp}_{\text{FPRE}, \mathcal{A}, n}^{\text{CPA}} \Rightarrow 1] - \frac{1}{2} \right| \leq \text{negl}(\lambda)$, where the experiment $\text{Exp}_{\text{FPRE}, \mathcal{A}, n}^{\text{CPA}}$ is defined in Fig. 2.

An FPRE scheme FPRE has ciphertext pseudorandomness (CPR), if for any PPT \mathcal{A} and any polynomial n , it holds that $\text{Adv}_{\text{FPRE}, \mathcal{A}, n}^{\text{CPR}}(\lambda) := \left| \Pr[\text{Exp}_{\text{FPRE}, \mathcal{A}, n}^{\text{CPR}} \Rightarrow 1] - \frac{1}{2} \right| \leq \text{negl}(\lambda)$, where $\text{Exp}_{\text{FPRE}, \mathcal{A}, n}^{\text{CPR}}$ is also defined in Fig. 2.

$\text{Exp}_{\text{FPRE}, \mathcal{A}, n}^{\text{CPA}}$ / $\text{Exp}_{\text{FPRE}, \mathcal{A}, n}^{\text{CPR}}$:	
$\text{pp} \leftarrow \text{Setup}$. For $i \in [n]$: $(pk^{(i)}, sk^{(i)}) \leftarrow \text{KGen}(\text{pp})$ $\mathcal{Q}_{rk} := \emptyset$ //record re-encryption key queries $\mathcal{Q}_c := \emptyset$ //record corruption queries $i^* := \perp$ //record challenge user	
$(i^*, m_0, m_1, st) \leftarrow \mathcal{A}^{\mathcal{O}_{\text{REKEY}}(\cdot, \cdot, \cdot), \mathcal{O}_{\text{COR}}(\cdot)}(\text{pp}, \{pk^{(i)}\}_{i \in [n]})$	
$(i^*, m, st) \leftarrow \mathcal{A}^{\mathcal{O}_{\text{REKEY}}(\cdot, \cdot, \cdot), \mathcal{O}_{\text{COR}}(\cdot)}(\text{pp}, \{pk^{(i)}\}_{i \in [n]})$	
If $(i^* \in \mathcal{Q}_c)$ or $(\exists j \in \mathcal{Q}_c \text{ s.t. } (i^*, j) \in \mathcal{Q}_{rk})$: Return $b \leftarrow \{0, 1\}$ //avoid TA1 , TA2	
$\beta \leftarrow \{0, 1\}$	
$ct_1^* \leftarrow \text{Enc}_1(pk^{(i^*)}, m_\beta)$	
If $\beta = 0$: $ct_1^* \leftarrow \text{Enc}_1(pk^{(i^*)}, m)$; Else: $ct_1^* \leftarrow \mathcal{C}$	
$\beta' \leftarrow \mathcal{A}^{\mathcal{O}_{\text{REKEY}}(\cdot, \cdot, \cdot), \mathcal{O}_{\text{COR}}(\cdot)}(st, ct_1^*)$	
If $\beta' = \beta$: Return 1; Else Return 0.	
	$\mathcal{O}_{\text{REKEY}}(i, j, f)$: If $(i = i^*)$ and $(j \in \mathcal{Q}_c)$: Return \perp //avoid TA2 $\mathcal{Q}_{rk} := \mathcal{Q}_{rk} \cup \{(i, j)\}$ $rk_{i \rightarrow j}^f \leftarrow \text{FReKGen}(pk^{(i)}, sk^{(i)}, pk^{(j)}, f)$ Return $rk_{i \rightarrow j}^f$
	$\mathcal{O}_{\text{COR}}(i)$: If $(i = i^*)$ or $(i^*, i) \in \mathcal{Q}_{rk}$: Return \perp //avoid TA1 , TA2 $\mathcal{Q}_c := \mathcal{Q}_c \cup \{i\}$ Return $sk^{(i)}$

Fig. 2. The CPA security experiment $\text{Exp}_{\text{FPRE}, \mathcal{A}, n}^{\text{CPA}}$ (with framed boxes) & the Ciphertext Pseudorandomness (CPR) security experiment $\text{Exp}_{\text{FPRE}, \mathcal{A}, n}^{\text{CPR}}$ (with gray boxes) for FPRE, where \mathcal{C} denotes the ciphertext space.

Remark 1 (On the formalization of CPA and CPR securities and discussion on trivial attacks). We formalize the CPA and CPR security notions by defining the experiments $\text{Exp}_{\text{FPRE}, \mathcal{A}, n}^{\text{CPA}}$ and $\text{Exp}_{\text{FPRE}, \mathcal{A}, n}^{\text{CPR}}$, respectively, in Fig. 2. More precisely, we consider a multi-user setting, and the adversary \mathcal{A} is allowed to make two kinds of oracle queries *adaptively*:

- through $\mathcal{O}_{\text{REKEY}}(i, j, f)$ query, \mathcal{A} can get re-encryption keys $rk_{i \rightarrow j}^f$, and
- through $\mathcal{O}_{\text{COR}}(i)$ query, \mathcal{A} can corrupt user i and obtain its secret key $sk^{(i)}$.

We stress that the adversary can issue multiple $\mathcal{O}_{\text{REKEY}}(i, j, f)$ queries, even for the same delegator i and same delegatee j , thus achieving *multiple delegations*. At some point, \mathcal{A} generates an output and receives a challenge ciphertext ct_1^* , which are the only different places in the two experiments. In $\text{Exp}_{\text{FPRE}, \mathcal{A}, n}^{\text{CPA}}$, \mathcal{A} outputs a challenge user index i^* as well as a pair of messages (m_0, m_1) , and receives a challenge ciphertext ct_1^* which encrypts m_β under $pk^{(i^*)}$, where β is the challenge bit that \mathcal{A} aims to guess. This captures the *indistinguishability* of ciphertexts. In $\text{Exp}_{\text{FPRE}, \mathcal{A}, n}^{\text{CPR}}$, \mathcal{A} outputs a challenge user index i^* and a single message m , and receives a challenge ciphertext ct_1^* which either encrypts m under $pk^{(i^*)}$ or is uniformly chosen from the ciphertext space \mathcal{C} , depending on the challenge bit β . This captures the *pseudorandomness* of ciphertexts. Clearly, CPR is stronger than CPA.

Note that CPA (and CPR) only consider the security of the *first-level* ciphertexts. In fact, the CPA security for the *second-level* ciphertexts can be similarly defined: \mathcal{A} outputs a challenge (j^*, m_0, m_1) , receives a challenge ciphertext $ct_2^* \leftarrow \text{Enc}_2(pk^{(j^*)}, m_\beta)$, and aims to guess the challenge bit β . Here we do not capture the CPA security for the second-level ciphertexts in the CPA security definition, but instead, we will formalize it in the security definition of collusion safety (CS) later (cf. Def. 5).

To prevent trivial attacks from \mathcal{A} , we keep track of two sets: \mathcal{Q}_c records the corrupted users, and \mathcal{Q}_{rk} records the tuples (i, j) that \mathcal{A} obtains a re-encryption key $rk_{i \rightarrow j}^f$. Based on that, there are two trivial attacks **TA1-TA2** to obtain information about the plaintext underlying the challenge ciphertext ct_1^* .

TA1: $i^* \in \mathcal{Q}_c$, i.e., \mathcal{A} ever corrupts user i^* and obtains its secret key $sk^{(i^*)}$. In this case, \mathcal{A} can decrypt ct_1^* directly by invoking $\text{Dec}_1(sk^{(i^*)}, ct_1^*)$.

TA2: $\exists j \in \mathcal{Q}_c$, s.t. $(i^*, j) \in \mathcal{Q}_{rk}$, i.e., \mathcal{A} gets a re-encryption key $rk_{i^* \rightarrow j}^f$ starting from the challenge user i^* to some corrupted user j that \mathcal{A} ever obtains its secret key $sk^{(j)}$. In this case, \mathcal{A} can re-encrypt ct_1^* to a ciphertext $ct_2^{(j)}$ encrypted for j via $ct_2^{(j)} \leftarrow_s \text{FReEnc}(rk_{i^* \rightarrow j}^f, ct_1^*)$, then simply decrypt $ct_2^{(j)}$ with $sk^{(j)}$ to obtain a function of the plaintext underlying ct_1^* , which is $f(m_\beta)$ in $\text{Exp}_{\text{FPRE}, \mathcal{A}, n}^{\text{CPA}}$, and is $f(m)$ in the case of $\beta = 0$ in $\text{Exp}_{\text{FPRE}, \mathcal{A}, n}^{\text{CPR}}$.

As such, we exclude the above trivial attacks in both CPA and CPR experiments.

Below, we formalize the property of Unidirectionality (UNID), which basically means that the proxy ability in one direction shouldn't imply the proxy ability in the other direction. Roughly speaking, it requires that given a fine-grained re-encryption key $rk_{j^* \rightarrow i^*}^f$, it is hard for an adversary to come up with a fine-grained re-encryption key $rk_{i^* \rightarrow j^*}^f$ of the other direction.

Definition 3 (Unidirectionality for FPRE). *An FPRE scheme FPRE is unidirectional (UNID), if for any PPT adversary \mathcal{A} and any polynomial n , it holds that $\text{Adv}_{\text{FPRE}, \mathcal{A}, n}^{\text{UNID}}(\lambda) := \Pr[\text{Exp}_{\text{FPRE}, \mathcal{A}, n}^{\text{UNID}} \Rightarrow 1] \leq \text{negl}(\lambda)$, where the experiment $\text{Exp}_{\text{FPRE}, \mathcal{A}, n}^{\text{UNID}}$ is defined in Fig. 3.*

$\text{Exp}_{\text{FPRE}, \mathcal{A}, n}^{\text{UNID}}:$ $\text{pp} \leftarrow_s \text{Setup}$. For $i \in [n]$: $(pk^{(i)}, sk^{(i)}) \leftarrow_s \text{KGen}(\text{pp})$ $\mathcal{Q}_{rk} := \emptyset$ //record re-encryption key queries $\mathcal{Q}_c := \emptyset$ //record corruption queries $i^* := \perp, j^* := \perp$ //record challenge users $(i^*, j^*, f, st) \leftarrow_s \mathcal{A}^{\mathcal{O}_{\text{REKEY}}(\cdot, \cdot), \mathcal{O}_{\text{COR}}(\cdot)}(\text{pp}, \{pk^{(i)}\}_{i \in [n]})$ If $(i^* = j^*)$ or $(i^* \in \mathcal{Q}_c)$ or $((i^*, j^*) \in \mathcal{Q}_{rk})$ or $(\exists j \in \mathcal{Q}_c \text{ s.t. } (i^*, j) \in \mathcal{Q}_{rk})$: Return \perp //avoid TA1', TA2', TA3', TA4' $rk_{j^* \rightarrow i^*}^f \leftarrow_s \text{FReKGen}(pk^{(j^*)}, sk^{(j^*)}, pk^{(i^*)}, f)$ $\mathcal{Q}_{rk} := \mathcal{Q}_{rk} \cup \{(j^*, i^*)\}$ $(rk_{i^* \rightarrow j^*}^f, f') \leftarrow_s \mathcal{A}^{\mathcal{O}_{\text{REKEY}}(\cdot, \cdot), \mathcal{O}_{\text{COR}}(\cdot)}(st, rk_{j^* \rightarrow i^*}^f)$ If f' does not have output diversity: Return \perp //avoid TA5' //check the functionality of $rk_{i^* \rightarrow j^*}^f$ in the following way $m \leftarrow_s \mathcal{M}$, $ct_1^{(i^*)} \leftarrow_s \text{Enc}_1(pk^{(i^*)}, m)$, $ct_2^{(j^*)} \leftarrow_s \text{FReEnc}(rk_{i^* \rightarrow j^*}^f, ct_1^{(i^*)})$ If $\text{Dec}_2(sk^{(j^*)}, ct_2^{(j^*)}) = f'(m)$: Return 1; Else: Return 0	$\mathcal{O}_{\text{REKEY}}(i, j, f):$ If $(i = i^*)$ and $(j = j^* \text{ or } j \in \mathcal{Q}_c)$: Return \perp //avoid TA3', TA4' $\mathcal{Q}_{rk} := \mathcal{Q}_{rk} \cup \{(i, j)\}$ $rk_{i \rightarrow j}^f \leftarrow_s \text{FReKGen}(pk^{(i)}, sk^{(i)}, pk^{(j)}, f)$ Return $rk_{i \rightarrow j}^f$ $\mathcal{O}_{\text{COR}}(i):$ If $(i = i^*)$ or $(i^*, i) \in \mathcal{Q}_{rk}$: Return \perp //avoid TA2', TA4' $\mathcal{Q}_c = \mathcal{Q}_c \cup \{i\}$ Return $sk^{(i)}$
--	--

Fig. 3. The Unidirectionality (UNID) security experiment $\text{Exp}_{\text{FPRE}, \mathcal{A}, n}^{\text{UNID}}$ for FPRE, where “output diversity” is defined as $\Pr[m_0, m_1 \leftarrow_s \mathcal{M} : f'(m_0) \neq f'(m_1)] \geq 1/\text{poly}(\lambda)$ (see Remark 5 in Appendix B.1 for more details).

In Appendix B.1, we give some explanations of the UNID security definition and discuss the trivial attacks **TA1'-TA5'** in Remark 5, and then show that the UNID security is implied by the CPA security in Lemma 6.

Next, we formalize the property of Non-Transitivity (NTR), which essentially requires that given two fine-grained re-encryption keys $rk_{i^* \rightarrow k^*}^{f_1}$ and $rk_{k^* \rightarrow j^*}^{f_2}$ from i^* to k^* and from k^* to j^* , respectively, it is hard for an adversary to compute a fine-grained re-encryption key $rk_{i^* \rightarrow j^*}^{f'}$ from i^* directly to j^* . Below we present the formal definition of NTR security.

Definition 4 (Non-Transitivity for FPRE). *An FPRE scheme FPRE is non-transitive (NTR), if for any PPT adversary \mathcal{A} and any polynomial \mathbf{n} , it holds that $\text{Adv}_{\text{FPRE}, \mathcal{A}, \mathbf{n}}^{\text{NTR}}(\lambda) := \Pr[\text{Exp}_{\text{FPRE}, \mathcal{A}, \mathbf{n}}^{\text{NTR}} \Rightarrow 1] \leq \text{negl}(\lambda)$, where the experiment $\text{Exp}_{\text{FPRE}, \mathcal{A}, \mathbf{n}}^{\text{NTR}}$ is defined in Fig. 4.*

$\text{Exp}_{\text{FPRE}, \mathcal{A}, \mathbf{n}}^{\text{NTR}}:$ <p> $\text{pp} \leftarrow \text{Setup}$. For $i \in [n]$: $(pk^{(i)}, sk^{(i)}) \leftarrow \text{KGen}(\text{pp})$ $\mathcal{Q}_{rk} := \emptyset$ //record re-encryption key queries $\mathcal{Q}_c := \emptyset$ //record corruption queries $i^*, k^*, j^* := \perp$ //record challenge users $(i^*, k^*, j^*, f_1, f_2, st) \leftarrow \mathcal{A}^{\mathcal{O}_{\text{ReKey}}(\cdot, \cdot, \cdot), \mathcal{O}_{\text{Cor}}(\cdot)}(\text{pp}, \{pk^{(i)}\}_{i \in [n]})$ If $(i^* = k^* \text{ or } k^* = j^* \text{ or } j^* = i^*)$ or $(i^* \in \mathcal{Q}_c)$ or $((i^*, j^*) \in \mathcal{Q}_{rk})$ or $(\exists j \in \mathcal{Q}_c \text{ s.t. } (i^*, j) \in \mathcal{Q}_{rk})$: Return \perp //avoid TA1', TA2', TA3', TA4' $rk_{i^* \rightarrow k^*}^{f_1} \leftarrow \text{FReKGen}(pk^{(i^*)}, sk^{(i^*)}, pk^{(k^*)}, f_1)$ $rk_{k^* \rightarrow j^*}^{f_2} \leftarrow \text{FReKGen}(pk^{(k^*)}, sk^{(k^*)}, pk^{(j^*)}, f_2)$ $\mathcal{Q}_{rk} = \mathcal{Q}_{rk} \cup \{(i^*, k^*)\} \cup \{(k^*, j^*)\}$ $(rk_{i^* \rightarrow j^*}^{f'}, f') \leftarrow \mathcal{A}^{\mathcal{O}_{\text{ReKey}}(\cdot, \cdot, \cdot), \mathcal{O}_{\text{Cor}}(\cdot)}(st, rk_{i^* \rightarrow k^*}^{f_1}, rk_{k^* \rightarrow j^*}^{f_2})$ If f' does not have output diversity: Return \perp //avoid TA5' //check the functionality of $rk_{i^* \rightarrow j^*}^{f'}$ in the following way $m \leftarrow \mathcal{M}$, $ct_1^{(i^*)} \leftarrow \text{Enc}_1(pk^{(i^*)}, m)$, $ct_2^{(j^*)} \leftarrow \text{FReEnc}(rk_{i^* \rightarrow j^*}^{f'}, ct_1^{(i^*)})$ If $\text{Dec}_2(sk^{(j^*)}, ct_2^{(j^*)}) = f'(m)$: Return 1; Else: Return 0 </p>	$\mathcal{O}_{\text{ReKey}}(i, j, f):$ <p> If $(i = i^*) \wedge (j = j^* \text{ or } j \in \mathcal{Q}_c)$: Return \perp //avoid TA3', TA4' $\mathcal{Q}_{rk} := \mathcal{Q}_{rk} \cup \{(i, j)\}$ $rk_{i \rightarrow j}^f \leftarrow \text{FReKGen}(pk^{(i)}, sk^{(i)}, pk^{(j)}, f)$ Return $rk_{i \rightarrow j}^f$ </p> $\mathcal{O}_{\text{Cor}}(i):$ <p> If $(i = i^*)$ or $(i^*, i) \in \mathcal{Q}_{rk}$: Return \perp //avoid TA2', TA4' $\mathcal{Q}_c = \mathcal{Q}_c \cup \{i\}$ Return $sk^{(i)}$ </p>
--	--

Fig. 4. The Non-Transitivity (NTR) security experiment $\text{Exp}_{\text{FPRE}, \mathcal{A}, \mathbf{n}}^{\text{NTR}}$ for FPRE, where “output diversity” is defined as $\Pr[m_0, m_1 \leftarrow \mathcal{M} : f'(m_0) \neq f'(m_1)] \geq 1/\text{poly}(\lambda)$ (see Remark 6 in Appendix B.2 for more details).

In Appendix B.2, we give some explanations of the NTR security definition and discuss the trivial attacks **TA1'**-**TA4'** in Remark 6, and then show that the NTR security is implied by the CPA security in Lemma 7.

Now, we formalize the Collusion-Safety (CS) for FPRE. Our CS security definition captures the *CPA security for the second-level ciphertexts*, instead of the master secret security as defined in [3, 16]. Nevertheless, we note that our CS security is at least as strong as theirs. See Appendix B.3 for more discussions. Below we present the formal definition of our CS security.

Definition 5 (Collusion-Safety for FPRE). *An FPRE scheme FPRE is collusion-safe (CS), if for any PPT adversary \mathcal{A} and any polynomial \mathbf{n} , it holds that $\text{Adv}_{\text{FPRE}, \mathcal{A}, \mathbf{n}}^{\text{CS}}(\lambda) := |\Pr[\text{Exp}_{\text{FPRE}, \mathcal{A}, \mathbf{n}}^{\text{CS}} \Rightarrow 1] - \frac{1}{2}| \leq \text{negl}(\lambda)$, where the experiment $\text{Exp}_{\text{FPRE}, \mathcal{A}, \mathbf{n}}^{\text{CS}}$ is defined in Fig. 5.*

$\text{Exp}_{\text{FPRE}, \mathcal{A}, n}^{\text{CS}}:$ $\text{pp} \leftarrow \text{Setup. For } i \in [n]: (pk^{(i)}, sk^{(i)}) \leftarrow \text{KGen}(\text{pp})$ $\mathcal{Q}_c := \emptyset \quad // \text{record corruption queries}$ $i^* := \perp \quad // \text{record challenge user}$ $(i^*, m_0, m_1, st) \leftarrow \mathcal{A}^{\mathcal{O}_{\text{REKEY}}(\cdot, \cdot, \cdot), \mathcal{O}_{\text{COR}}(\cdot)}(\text{pp}, \{pk^{(i)}\}_{i \in [n]})$ $\text{If } i^* \in \mathcal{Q}_c: \text{ Return } \perp \quad // \text{avoid trivial attacks}$ $\beta \leftarrow \{0, 1\}$ $ct_2^* \leftarrow \text{Enc}_2(pk^{(i^*)}, m_\beta)$ $\beta' \leftarrow \mathcal{A}^{\mathcal{O}_{\text{REKEY}}(\cdot, \cdot, \cdot), \mathcal{O}_{\text{COR}}(\cdot)}(st, ct_2^*)$ $\text{If } \beta' = \beta: \text{ Return } 1; \text{ Else Return } 0.$	$\mathcal{O}_{\text{REKEY}}(i, j, f):$ $rk_{i \rightarrow j}^f \leftarrow \text{FREKGen}(pk^{(i)}, sk^{(i)}, pk^{(j)}, f)$ $\text{Return } rk_{i \rightarrow j}^f$ $\mathcal{O}_{\text{COR}}(i):$ $\text{If } i = i^*:$ $\text{Return } \perp \quad // \text{avoid trivial attacks}$ $\mathcal{Q}_c = \mathcal{Q}_c \cup \{i\}$ $\text{Return } sk^{(i)}$
---	--

Fig. 5. The Collusion-Safety (CS) security experiment $\text{Exp}_{\text{FPRE}, \mathcal{A}, n}^{\text{CS}}$ for FPFE.

Remark 2 (On the formalization of CS security). We formalize the CS security by defining the experiment $\text{Exp}_{\text{FPRE}, \mathcal{A}, n}^{\text{CS}}$ in Fig. 5. Similar to previous security notions, we consider a multi-user setting, and the adversary \mathcal{A} is allowed to make $\mathcal{O}_{\text{REKEY}}$ and \mathcal{O}_{COR} queries *adaptively*. At some point, \mathcal{A} outputs a challenge user index i^* as well as a pair of messages (m_0, m_1) , and receives a challenge second-level ciphertext ct_2^* which encrypts m_β under $pk^{(i^*)}$, where β is the challenge bit that \mathcal{A} aims to guess. This captures the *indistinguishability* of second-level ciphertexts. Clearly, to avoid trivial attack, \mathcal{A} cannot obtain the secret key $sk^{(i^*)}$.

In real world, re-encryption relations between ciphertexts often imply the proxy connections between users, therefore it is desirable to hide the relations/connections. We formalize this as the property of *Ciphertext Unlinkability* (CUL), which basically requires that given two ciphertexts (ct_1^*, ct_2^*) , it is hard for an adversary to tell whether ct_2^* is a re-encryption of ct_1^* , or the two ciphertexts are independently and freshly generated. Below we present the formal definition.

Definition 6 (Ciphertext Unlinkability for FPFE). *An FPFE scheme FPFE has ciphertext unlinkability (CUL), if for any PPT adversary \mathcal{A} and any polynomial n , it holds that $\text{Adv}_{\text{FPFE}, \mathcal{A}, n}^{\text{CUL}}(\lambda) := |\Pr[\text{Exp}_{\text{FPFE}, \mathcal{A}, n}^{\text{CUL}} \Rightarrow 1] - \frac{1}{2}| \leq \text{negl}(\lambda)$, where the experiment $\text{Exp}_{\text{FPFE}, \mathcal{A}, n}^{\text{CUL}}$ is defined in Fig. 6.*

$\text{Exp}_{\text{FPFE}, \mathcal{A}, n}^{\text{CUL}}:$ $\text{pp} \leftarrow \text{Setup. For } i \in [n]: (pk^{(i)}, sk^{(i)}) \leftarrow \text{KGen}(\text{pp})$ $\mathcal{Q}_{rk} := \emptyset \quad // \text{record re-encryption key queries}$ $\mathcal{Q}_c := \emptyset \quad // \text{record corruption queries}$ $i^* := \perp, j^* := \perp \quad // \text{record challenge users}$ $(i^*, j^*, (f, m), (m_1, m_2), st) \leftarrow \mathcal{A}^{\mathcal{O}_{\text{REKEY}}(\cdot, \cdot, \cdot), \mathcal{O}_{\text{COR}}(\cdot)}(\text{pp}, \{pk^{(i)}\}_{i \in [n]})$ $\text{If } (i^* \in \mathcal{Q}_c) \text{ or } (j^* \in \mathcal{Q}_c) \text{ or } (\exists j \in \mathcal{Q}_c \text{ s.t. } (i^*, j) \in \mathcal{Q}_{rk}):$ $\text{Return } b \leftarrow \{0, 1\} \quad // \text{avoid TA1'', TA2''}$ $\beta \leftarrow \{0, 1\}$ $\text{If } \beta = 0: \quad // \text{ciphertexts generated with re-encryption relation}$ $ct_1^* \leftarrow \text{Enc}_1(pk^{(i^*)}, m)$ $rk_{i^* \rightarrow j^*}^f \leftarrow \text{FREKGen}(pk^{(i^*)}, sk^{(i^*)}, pk^{(j^*)}, f), ct_2^* \leftarrow \text{FREnc}(rk_{i^* \rightarrow j^*}^f, ct_1^*)$ $\text{If } \beta = 1: \quad // \text{ciphertexts generated independently}$ $ct_1^* \leftarrow \text{Enc}_1(pk^{(i^*)}, m_1), ct_2^* \leftarrow \text{Enc}_2(pk^{(j^*)}, m_2)$ $\beta' \leftarrow \mathcal{A}^{\mathcal{O}_{\text{REKEY}}(\cdot, \cdot, \cdot), \mathcal{O}_{\text{COR}}(\cdot)}(st, ct_1^*, ct_2^*)$ $\text{If } \beta' = \beta: \text{ Return } 1; \text{ Else Return } 0.$	$\mathcal{O}_{\text{REKEY}}(i, j, f):$ $\text{If } (i = i^*) \text{ and } (j \in \mathcal{Q}_c):$ $\text{Return } \perp \quad // \text{avoid TA2''}$ $\mathcal{Q}_{rk} := \mathcal{Q}_{rk} \cup \{(i, j)\}$ $rk_{i \rightarrow j}^f \leftarrow \text{FREKGen}(pk^{(i)}, sk^{(i)}, pk^{(j)}, f)$ $\text{Return } rk_{i \rightarrow j}^f$ $\mathcal{O}_{\text{COR}}(i):$ $\text{If } (i = i^*) \text{ or } (i = j^*) \text{ or } (i^*, i) \in \mathcal{Q}_{rk}:$ $\text{Return } \perp \quad // \text{avoid TA1'', TA2''}$ $\mathcal{Q}_c = \mathcal{Q}_c \cup \{i\}$ $\text{Return } sk^{(i)}$
--	--

Fig. 6. The Ciphertext Unlinkability (CUL) security experiment $\text{Exp}_{\text{FPFE}, \mathcal{A}, n}^{\text{CUL}}$ for FPFE.

Remark 3 (On the formalization of CUL security and discussion on trivial attacks). We formalize the CUL security by defining the experiment $\text{Exp}_{\text{FPRE}, \mathcal{A}, n}^{\text{CUL}}$ in Fig. 6. Similar to previous security notions, we consider a multi-user setting, and the adversary \mathcal{A} is allowed to make $\mathcal{O}_{\text{REKEY}}$ and \mathcal{O}_{COR} queries *adaptively*. At some point, \mathcal{A} outputs a pair of challenge users (i^*, j^*) , a pair of function and message (f, m) as well as a pair of messages (m_1, m_2) , and receives two challenge ciphertexts (ct_1^*, ct_2^*) which are

- (Case $\beta = 0$) *either* two ciphertexts generated with re-encryption relation, namely $ct_1^* \leftarrow_{\$} \text{Enc}_1(pk^{(i^*)}, m)$, $rk_{i^* \rightarrow j^*}^f \leftarrow_{\$} \text{FReKGen}(pk^{(i^*)}, sk^{(i^*)}, pk^{(j^*)}, f)$ and $ct_2^* \leftarrow_{\$} \text{FReEnc}(rk_{i^* \rightarrow j^*}^f, ct_1^*)$,
- (Case $\beta = 1$) *or* two ciphertexts that are generated independently, namely $ct_1^* \leftarrow_{\$} \text{Enc}_1(pk^{(i^*)}, m_1)$ and $ct_2^* \leftarrow_{\$} \text{Enc}_2(pk^{(j^*)}, m_2)$.

\mathcal{A} aims to guess which case it is.

Actually, there are two trivial attacks **TA1''-TA2''** to obtain information about the plaintexts underlying the challenge ciphertexts (ct_1^*, ct_2^*) .

TA1'': $i^* \in \mathcal{Q}_c$ or $j^* \in \mathcal{Q}_c$, i.e., \mathcal{A} ever obtains $sk^{(i^*)}$ or $sk^{(j^*)}$. In this case, \mathcal{A} can decrypt the challenger ciphertext ct_1^* or ct_2^* itself and trivially win.

TA2'': $\exists j \in \mathcal{Q}_c$, s.t. $(i^*, j) \in \mathcal{Q}_{rk}$, i.e., \mathcal{A} gets $sk^{(j)}$ and $rk_{i^* \rightarrow j}^f$ for some user j . In this case, \mathcal{A} can re-encrypt ct_1^* to a ciphertext $ct_2^{(j)}$ encrypted for j via $ct_2^{(j)} \leftarrow_{\$} \text{FReEnc}(rk_{i^* \rightarrow j}^f, ct_1^*)$, then simply decrypt $ct_2^{(j)}$ with $sk^{(j)}$ to obtain a function of the plaintext underlying ct_1^* , which is $f(m)$ in the case of $\beta = 0$ and is $f(m_1)$ in the case of $\beta = 1$.

As such, we exclude the above trivial attacks in the CUL experiment.

Below we show that the CUL security is stronger than the CPA security via Lemma 1, with proof postponed to Appendix B.4 due to space limitations.

Lemma 1 (CUL \Rightarrow CPA). *For any PPT adversary \mathcal{A} breaking the CPA security of FPRE and any polynomial n , there exists a PPT adversary \mathcal{B} breaking the CUL security of FPRE with $\text{Adv}_{\text{FPRE}, \mathcal{B}, n+1}^{\text{CUL}}(\lambda) = \frac{1}{2} \cdot \text{Adv}_{\text{FPRE}, \mathcal{A}, n}^{\text{CPA}}(\lambda)$.*

4 Fine-Grained PRE for Bounded Linear Functions from LWE

In this section, we present a construction of fine-grained PRE (FPRE) scheme $\text{FPRE}_{\text{LWE}}^{\text{lin}}$ for the family of bounded linear functions \mathcal{F}_{lin} (with coefficient of bounded norm). Then based on the LWE assumption, we prove the security of our $\text{FPRE}_{\text{LWE}}^{\text{lin}}$, including CPA security in Theorem 1, ciphertext pseudorandomness (CPR) in Corollary 1, ciphertext unlinkability (CUL) in Theorem 2, and collusion-safety (CS) in Theorem 3. Combining with Lemma 6 and Lemma 7 in Appendix B, our $\text{FPRE}_{\text{LWE}}^{\text{lin}}$ also achieves unidirectionality (UNID) and non-transitivity (NTR).

Parameters. Let $\text{pp}_{\text{LWE}} = (p, q, n, N, \ell, \gamma, \Delta, \chi)$ be LWE-related parameters that meet the following conditions:

- $p, q, n, N, \ell, \gamma, \Delta \in \mathbb{N}$ are integers, where $q := p^2$, $N \geq 2n \log q + 2\omega(\log \lambda)$ and $\gamma \geq O(\sqrt{n \log q}) \cdot \omega(\sqrt{\log n})$;
- χ is a B -bounded distribution, where B satisfies $\gamma \cdot \omega(\log n) \leq B < \min\{p/2, q/(10N)\}$ and $(N+1)(nB + NB + \ell\Delta)B < p/2$.

More precisely, see Table 2 for a concrete parameter choice. For simplicity, we assume that all algorithms of our FPFE scheme take pp_{LWE} as an implicit input.

Table 2. Concrete parameters setting, where λ denotes the security parameter.

Parameters	p	q	n	N	ℓ	γ	Δ	B
Settings	λ^5	λ^{10}	λ	$21\lambda \log \lambda$	λ	$\sqrt{\lambda}(\log \lambda)^2$	λ	$\sqrt{\lambda}(\log \lambda)^4$

Bounded Linear function family. The message space is $\mathcal{M} := \mathbb{Z}_p^\ell$. Define the family of bounded linear functions \mathcal{F}_{lin} from \mathcal{M} to \mathcal{M} over \mathbb{Z}_p as follows:

$$\mathcal{F}_{\text{lin}} = \left\{ f_{\mathbf{M}} : \mathbb{Z}_p^\ell \rightarrow \mathbb{Z}_p^\ell \mid \mathbf{M} \in \mathbb{Z}_p^{\ell \times \ell}, \|\mathbf{M}\|_\infty \leq \Delta \right\}. \quad (3)$$

LWE-based FPFE scheme for \mathcal{F}_{lin} . Let TrapGen, SamplePre, Invert be the PPT algorithms defined in Lemmas 2, 3 and 4 in Appendix A.1, respectively. Our LWE-based FPFE scheme $\text{FPFE}_{\text{LWE}}^{\text{lin}} = (\text{Setup}, \text{KGen}, \text{FReKGen}, \text{Enc}_1, \text{Enc}_2, \text{FReEnc}, \text{Dec}_1, \text{Dec}_2)$ for the linear function family \mathcal{F}_{lin} defined in (3) is shown in Fig. 7.

<p>$\text{pp} \leftarrow_s \text{Setup}$ $\overline{\mathbf{A}'} \leftarrow_s \mathbb{Z}_q^{N \times n}$ Return $\text{pp} := \overline{\mathbf{A}'}$</p> <p>$(pk, sk) \leftarrow_s \text{KGen}(\text{pp})$ $(\overline{\mathbf{A}} \in \mathbb{Z}_q^{N \times n}, \mathbf{T}) \leftarrow_s \text{TrapGen}(1^n, 1^N)$, $\underline{\mathbf{A}} \leftarrow_s \mathbb{Z}_q^{\ell \times n}$ $\mathbf{A} := \begin{pmatrix} \overline{\mathbf{A}} \\ \underline{\mathbf{A}} \end{pmatrix} \in \mathbb{Z}_q^{(N+\ell) \times n}$ $\mathbf{K} \leftarrow_s \{0, 1\}^{\ell \times N}$, $\underline{\mathbf{A}'} := -\mathbf{K}\overline{\mathbf{A}'}$ $pk := (\mathbf{A}, \underline{\mathbf{A}'})$, $sk := (\mathbf{T}, \mathbf{K})$ Return (pk, sk)</p> <p>$rk_{i \rightarrow j}^{f_{\mathbf{M}}} \leftarrow_s \text{FReKGen}(pk^{(i)} = (\mathbf{A}^{(i)}, \underline{\mathbf{A}'}^{(i)}), sk^{(i)} = (\mathbf{T}^{(i)}, \mathbf{K}^{(i)}))$ $pk^{(j)} = (\mathbf{A}^{(j)}, \underline{\mathbf{A}'}^{(j)})$, $f_{\mathbf{M}} \in \mathcal{F}_{\text{lin}}$:</p> <p>$\mathbf{S} \leftarrow_s \chi^{n \times n}$, $\mathbf{E} \leftarrow_s \chi^{(N+\ell) \times n}$ $\mathbf{A}'^{(j)} := \begin{pmatrix} \overline{\mathbf{A}'} \\ \underline{\mathbf{A}'}^{(j)} \end{pmatrix}$ $\mathbf{R} \in \mathbb{Z}^{(N+\ell) \times N} \leftarrow_s \text{SamplePre}(\mathbf{T}^{(i)}, \overline{\mathbf{A}'}^{(i)}, \mathbf{A}'^{(j)}\mathbf{S} + \mathbf{E} - \begin{pmatrix} 0 \\ \mathbf{M} \end{pmatrix} \underline{\mathbf{A}'}^{(i)}, \gamma)$ $rk_{i \rightarrow j}^{f_{\mathbf{M}}} := \begin{pmatrix} \mathbf{R} & \mathbf{0} \\ & \mathbf{M} \end{pmatrix} \in \mathbb{Z}_p^{(N+\ell) \times (N+\ell)}$ // \mathbf{M} is the description of $f_{\mathbf{M}}$ Return $rk_{i \rightarrow j}^{f_{\mathbf{M}}}$</p> <p>$ct_1 \leftarrow_s \text{Enc}_1(pk = (\mathbf{A}, \underline{\mathbf{A}'}), \mathbf{m} \in \mathcal{M})$ $\mathbf{s} \leftarrow_s \chi^n$, $\mathbf{e} \leftarrow_s \chi^{N+\ell}$ $ct_1 := \mathbf{A}\mathbf{s} + \mathbf{e} + \begin{pmatrix} 0 \\ \mathbf{pm} \end{pmatrix} \in \mathbb{Z}_q^{N+\ell}$ Return ct_1</p>	<p>$ct_2 \leftarrow_s \text{Enc}_2(pk = (\mathbf{A}, \underline{\mathbf{A}'}), \mathbf{m} \in \mathcal{M})$ $\mathbf{s} \leftarrow_s \chi^n$, $\mathbf{e} \leftarrow_s \chi^{N+\ell}$ $\mathbf{A}' := \begin{pmatrix} \overline{\mathbf{A}'} \\ \underline{\mathbf{A}'} \end{pmatrix}$ $ct_2 := \mathbf{A}'\mathbf{s} + \mathbf{e} + \begin{pmatrix} 0 \\ \mathbf{pm} \end{pmatrix} \in \mathbb{Z}_q^{N+\ell}$ Return ct_2</p> <p>$ct_2^{(j)} \leftarrow \text{FReEnc}(rk_{i \rightarrow j}^{f_{\mathbf{M}}} \in \mathbb{Z}_p^{(N+\ell) \times (N+\ell)}, ct_1^{(i)} \in \mathbb{Z}_q^{N+\ell})$ $ct_2^{(j)} := rk_{i \rightarrow j}^{f_{\mathbf{M}}} \cdot ct_1^{(i)} \in \mathbb{Z}_q^{N+\ell}$ Return $ct_2^{(j)}$</p> <p>$\mathbf{m} \leftarrow \text{Dec}_1(sk = (\mathbf{T}, \mathbf{K}), ct_1 \in \mathbb{Z}_q^{N+\ell})$ Parse $ct_1 = \begin{pmatrix} \overline{ct_1} \in \mathbb{Z}_q^N \\ ct_1 \in \mathbb{Z}_q^\ell \end{pmatrix}$ $(\mathbf{s}, \mathbf{e}_1) \leftarrow \text{Invert}(\mathbf{T}, ct_1)$ $\tilde{\mathbf{m}} = (\tilde{m}_1, \dots, \tilde{m}_\ell) := \overline{ct_1} - \mathbf{A}\mathbf{s}$ For all $i \in [\ell]$: $m_i := \lceil \tilde{m}_i / p \rceil$ Return $\mathbf{m} = (m_1, \dots, m_\ell)$</p> <p>$\mathbf{m} \leftarrow \text{Dec}_2(sk = (\mathbf{T}, \mathbf{K}), ct_2 \in \mathbb{Z}_q^{N+\ell})$ $\tilde{\mathbf{m}} = (\tilde{m}_1, \dots, \tilde{m}_\ell) := (\mathbf{K} \mid \mathbf{I}_{\ell \times \ell}) \cdot ct_2$ For all $i \in [\ell]$: $m_i := \lceil \tilde{m}_i / p \rceil$ Return $\mathbf{m} = (m_1, \dots, m_\ell)$</p>
---	--

Fig. 7. The LWE-based FPFE scheme $\text{FPFE}_{\text{LWE}}^{\text{lin}}$ for the linear function family \mathcal{F}_{lin} .

Correctness. Let $\text{pp} = \overline{\mathbf{A}'}$, $pk = (\mathbf{A}, \underline{\mathbf{A}'})$ and $sk = (\mathbf{T}, \mathbf{K})$. For a first-level ciphertext ct_1 generated by $\text{Enc}_1(pk, \mathbf{m})$, we have $ct_1 = \begin{pmatrix} \overline{ct_1} \\ \underline{ct_1} \end{pmatrix} = \begin{pmatrix} \overline{\mathbf{A}'\mathbf{s} + \mathbf{e}_1} \\ \underline{\mathbf{A}'\mathbf{s} + \mathbf{e}_2 + p\mathbf{m}} \end{pmatrix}$, where $\mathbf{e}_1 \leftarrow_{\$} \chi^N$, $\mathbf{e}_2 \leftarrow_{\$} \chi^\ell$, and the upper part is an LWE instance of $\overline{\mathbf{A}'}$. Since \mathbf{e}_1 is B -bounded with $B < q/(10N)$, $\|\mathbf{e}_1\| \leq \sqrt{N} \|\mathbf{e}_1\|_\infty \leq \sqrt{N}B < q/(10\sqrt{N})$. Then by Lemma 3 in Appendix A.1, $(\mathbf{s}, \mathbf{e}_1)$ can be correctly recovered via $(\mathbf{s}, \mathbf{e}_1) \leftarrow \text{Invert}(\mathbf{T}, \overline{ct_1})$. Thus according to the decryption algorithm $\text{Dec}_1(sk, ct_1)$, we get $\tilde{\mathbf{m}} = \underline{ct_1} - \underline{\mathbf{A}'\mathbf{s}} = \mathbf{e}_2 + p\mathbf{m}$, and by parsing $\mathbf{e}_2 = (e_{21}, \dots, e_{2\ell})^\top$, we have that $\tilde{m}_i = e_{2i} + pm_i$ for all $i \in [\ell]$. Moreover, since \mathbf{e}_2 is B -bounded with $B < p/2$, each $|e_{2i}| \leq B < p/2$. Consequently, $\lceil \tilde{m}_i/p \rceil = m_i$ and Dec_1 can recover \mathbf{m} correctly from ct_1 .

For a second-level ciphertext ct_2 generated by $\text{Enc}_2(pk, \mathbf{m})$, we have

$$ct_2 = \begin{pmatrix} \overline{\mathbf{A}'\mathbf{s}} + \mathbf{e}_1 \\ \underline{\mathbf{A}'\mathbf{s}} + \mathbf{e}_2 + p\mathbf{m} \end{pmatrix} = \begin{pmatrix} \overline{\mathbf{A}'\mathbf{s}} + \mathbf{e}_1 \\ -\mathbf{K}\overline{\mathbf{A}'\mathbf{s}} + \mathbf{e}_2 + p\mathbf{m} \end{pmatrix},$$

where $\mathbf{e}_1 \leftarrow_{\$} \chi^N$, $\mathbf{e}_2 \leftarrow_{\$} \chi^\ell$. According to the decryption algorithm $\text{Dec}_2(sk, ct_2)$, we get $\tilde{\mathbf{m}} = (\mathbf{K} \mid \mathbf{I}) \cdot ct_2 = \mathbf{K}\mathbf{e}_1 + \mathbf{e}_2 + p\mathbf{m}$, and by defining $\mathbf{e}' = (e'_1, \dots, e'_\ell) := \mathbf{K}\mathbf{e}_1 + \mathbf{e}_2$, we have that $\tilde{\mathbf{m}} = \mathbf{e}' + p\mathbf{m}$ and $\tilde{m}_i = e'_i + pm_i$ for all $i \in [\ell]$. Since $\mathbf{e}_1, \mathbf{e}_2$ are B -bounded with B satisfying $(N+1)(nB + NB + \ell\Delta)B < p/2$ and $\mathbf{K} \in \{0, 1\}^{\ell \times N}$, we have $\|\mathbf{e}'\|_\infty \leq (N+1)B < p/2$ and each $|e'_i| \leq B < p/2$. Consequently, $\lceil \tilde{m}_i/p \rceil = m_i$ and Dec_2 can recover \mathbf{m} correctly from ct_2 .

Fine-Grained One-Hop Correctness. Let $ct_1^{(i)} \leftarrow_{\$} \text{Enc}_1(pk^{(i)}, \mathbf{m})$ and $rk_{i \rightarrow j}^{f_M} \leftarrow_{\$} \text{FReKGen}(pk^{(i)}, sk^{(i)}, pk^{(j)}, f_M)$. For a fine-grained re-encrypted ciphertext $ct_2^{(j)} \leftarrow_{\$} \text{FReEnc}(rk_{i \rightarrow j}^{f_M}, ct_1^{(i)})$, we have

$$\begin{aligned} ct_2^{(j)} &:= \begin{pmatrix} \mathbf{R} & \mathbf{0} \\ \mathbf{M} & \mathbf{0} \end{pmatrix} \cdot ct_1^{(i)} = \begin{pmatrix} \mathbf{R} & \mathbf{0} \\ \mathbf{M} & \mathbf{0} \end{pmatrix} \cdot \left(\begin{pmatrix} \overline{\mathbf{A}^{(i)}} \\ \underline{\mathbf{A}^{(i)}} \end{pmatrix} \mathbf{s} + \begin{pmatrix} \mathbf{e}_1 \\ \mathbf{e}_2 \end{pmatrix} + \begin{pmatrix} \mathbf{0} \\ p\mathbf{m} \end{pmatrix} \right) \\ &= \left(\mathbf{R}\overline{\mathbf{A}^{(i)}} + \begin{pmatrix} \mathbf{0} \\ \mathbf{M} \end{pmatrix} \underline{\mathbf{A}^{(i)}} \right) \cdot \mathbf{s} + \mathbf{R}\mathbf{e}_1 + \begin{pmatrix} \mathbf{0} \\ \mathbf{M}\mathbf{e}_2 \end{pmatrix} + \begin{pmatrix} \mathbf{0} \\ p\mathbf{M}\mathbf{m} \end{pmatrix}. \end{aligned}$$

Since \mathbf{R} is generated by $\mathbf{R} \leftarrow_{\$} \text{SamplePre}(\mathbf{T}^{(i)}, \overline{\mathbf{A}^{(i)}}, \mathbf{A}'^{(j)}\mathbf{S} + \mathbf{E} - \begin{pmatrix} \mathbf{0} \\ \mathbf{M} \end{pmatrix} \underline{\mathbf{A}^{(i)}})$, by Lemma 4 in Appendix A.1, we have $\mathbf{R}\overline{\mathbf{A}^{(i)}} = \mathbf{A}'^{(j)}\mathbf{S} + \mathbf{E} - \begin{pmatrix} \mathbf{0} \\ \mathbf{M} \end{pmatrix} \underline{\mathbf{A}^{(i)}}$. Consequently, we get

$$ct_2^{(j)} = \mathbf{A}'^{(j)} \underbrace{\underbrace{\mathbf{S}\mathbf{s}}_{:=\tilde{\mathbf{s}}} + \mathbf{E}\mathbf{s} + \mathbf{R}\mathbf{e}_1}_{:=\tilde{\mathbf{e}}} + \begin{pmatrix} \mathbf{0} \\ \mathbf{M}\mathbf{e}_2 \end{pmatrix} + \begin{pmatrix} \mathbf{0} \\ p \cdot \underbrace{\mathbf{M}\mathbf{m}}_{=f_M(\mathbf{m})} \end{pmatrix}.$$

By Lemma 4 in Appendix A.1, we know that $\|\mathbf{R}\|_\infty \leq \gamma \cdot \omega(\log n)$, which further yields $\|\mathbf{R}\|_\infty \leq B$ due to $\gamma \cdot \omega(\log n) \leq B$. Now that $\mathbf{E}, \mathbf{s}, \mathbf{R}, \mathbf{e}_1, \mathbf{e}_2$ are all B -bounded and \mathbf{M} is Δ -bounded, so we have $\|\tilde{\mathbf{e}}\|_\infty \leq (nB + NB + \ell\Delta)B$. By a similar argument like the correctness of second-level ciphertexts, since $(N+1)\|\tilde{\mathbf{e}}\|_\infty \leq (N+1)(nB + NB + \ell\Delta)B < p/2$, the decryption algorithm Dec_2 can recover the function value $f_M(\mathbf{m}) = \mathbf{M} \cdot \mathbf{m}$ correctly from the re-encrypted ciphertext $ct_2^{(j)}$.

Next, we show the CPA security of our $\text{FPRE}_{\text{LWE}}^{\text{lin}}$ scheme.

Theorem 1 (CPA Security of $\text{FPRE}_{\text{LWE}}^{\text{lin}}$). *Assume that the $\text{LWE}_{n,q,\chi,N+\ell}$ -assumption holds, then the scheme $\text{FPRE}_{\text{LWE}}^{\text{lin}}$ proposed in Fig. 7 is CPA secure. More precisely, for any PPT adversary \mathcal{A} that makes at most Q times of $\mathcal{O}_{\text{REKEY}}$ queries and for any polynomial n , there exists a PPT algorithm \mathcal{B} against the LWE assumption s.t. $\text{Adv}_{\text{FPRE},\mathcal{A},n}^{\text{CPA}}(\lambda) \leq (n^2nQ + n) \cdot \text{Adv}_{[n,q,\chi,N+\ell],\mathcal{B}}^{\text{LWE}}(\lambda) + \text{negl}(\lambda)$.*

Proof of Theorem 1. We prove the theorem via a sequence of games G_0 – G_5 , where G_0 is the CPA experiment, and in G_5 , \mathcal{A} has a negligible advantage.

Game G_0 : This is the CPA experiment $\text{Exp}_{\text{FPRE},\mathcal{A},n}^{\text{CPA}}$ (cf. Fig. 2). Let Win denote the event that $\beta' = \beta$. By definition, $\text{Adv}_{\text{FPRE},\mathcal{A},n}^{\text{CPA}}(\lambda) = |\Pr_0[\text{Win}] - \frac{1}{2}|$.

Let $\text{pp} = \overline{\mathbf{A}}$ and let $pk^{(i)} = (\mathbf{A}^{(i)}, \underline{\mathbf{A}}'^{(i)})$, $sk^{(i)} = (\mathbf{T}^{(i)}, \mathbf{K}^{(i)})$ denote the public key and secret key of user $i \in [n]$. In this game, the challenger answers \mathcal{A} 's $\mathcal{O}_{\text{REKEY}}$, \mathcal{O}_{COR} queries and generates the challenge ciphertext ct_1^* as follows.

- On receiving a re-encryption key query $\mathcal{O}_{\text{REKEY}}(i, j, f_{\mathbf{M}})$ from \mathcal{A} , the challenger returns \perp to \mathcal{A} directly if trivial attacks ($i = i^*$) and ($j \in \mathcal{Q}_c$) occur. Otherwise, the challenger adds (i, j) to \mathcal{Q}_{rk} , samples $\mathbf{S} \leftarrow_{\$} \chi^{n \times n}$, $\mathbf{E} \leftarrow_{\$} \chi^{(N+\ell) \times n}$, invokes $\mathbf{R} \leftarrow_{\$} \text{SamplePre}\left(\mathbf{T}^{(i)}, \overline{\mathbf{A}}^{(i)}, \mathbf{A}'^{(j)}\mathbf{S} + \mathbf{E} - \begin{pmatrix} \mathbf{0} \\ \mathbf{M} \end{pmatrix} \underline{\mathbf{A}}^{(i)}, \gamma\right)$, where $\mathbf{A}'^{(j)} = \begin{pmatrix} \overline{\mathbf{A}} \\ \underline{\mathbf{A}}'^{(j)} \end{pmatrix}$, and returns $rk_{i \rightarrow j}^{f_{\mathbf{M}}} := (\mathbf{R} \mid \begin{pmatrix} \mathbf{0} \\ \mathbf{M} \end{pmatrix})$ to \mathcal{A} .
- On receiving a corruption query $\mathcal{O}_{\text{COR}}(i)$ from \mathcal{A} , the challenger returns \perp to \mathcal{A} directly if trivial attacks ($i = i^*$) or $(i^*, i) \in \mathcal{Q}_{rk}$ occur. Otherwise, the challenger adds i to \mathcal{Q}_c and returns $sk^{(i)}$ to \mathcal{A} .
- On receiving the challenge tuple $(i^*, \mathbf{m}_0, \mathbf{m}_1)$ from \mathcal{A} , the challenger first checks if trivial attacks ($i^* \in \mathcal{Q}_c$) or $(\exists j \in \mathcal{Q}_c \text{ s.t. } (i^*, j) \in \mathcal{Q}_{rk})$ occur. If yes, the challenger aborts the game with \mathcal{A} by returning a random bit. Otherwise, the challenger chooses a random bit $\beta \leftarrow_{\$} \{0, 1\}$, samples $\mathbf{s} \leftarrow_{\$} \chi^n$, $\mathbf{e} \leftarrow_{\$} \chi^{N+\ell}$, and sends $ct_1^* := \mathbf{A}^{(i^*)}\mathbf{s} + \mathbf{e} + \begin{pmatrix} \mathbf{0} \\ \text{pm}_{\beta} \end{pmatrix}$ to \mathcal{A} .

Game G_1 : It is the same as G_0 , except that, at the beginning of the game, the challenger chooses a random user index $i' \leftarrow_{\$} [n]$ uniformly as the guess of the challenge user i^* , and will abort the game and return a random bit in the following cases.

- **Case 1.** \mathcal{A} issues the challenge tuple $(i^*, \mathbf{m}_0, \mathbf{m}_1)$ but $i' \neq i^*$.
- **Case 2.** \mathcal{A} issues a re-encryption key query $\mathcal{O}_{\text{REKEY}}(i, j, f_{\mathbf{M}})$ such that $(i = i')$ and $(j \in \mathcal{Q}_c)$ before issuing its challenge.
- **Case 3.** \mathcal{A} issues a corruption query $\mathcal{O}_{\text{COR}}(i)$ such that $(i = i')$ or $(i', i) \in \mathcal{Q}_{rk}$ before issuing its challenge.

Case 1 suggests that the challenger's guess is wrong. Now in G_1 , the challenger will abort the game if the guess is wrong. If the guess is correct, i.e., $i' = i^*$, Case 2 and Case 3 are in fact trivial attacks, so they will lead to abort anyway in G_0 and do not contribute to \mathcal{A} 's advantage. Since the challenger will guess i^* correctly with probability $1/n$, we have $|\Pr_0[\text{Win}] - \frac{1}{2}| = \frac{1}{n} |\Pr_1[\text{Win}] - \frac{1}{2}|$.

Game $G_{1,v}, v \in [0, n]$: It is the same as G_1 , except for the reply to \mathcal{A} 's re-encryption key query $\mathcal{O}_{\text{REKEY}}(i, j, f_M)$ which does not lead to any trivial attack.

- If $i = i'$ and $j \leq v$, the challenger uniformly samples $\mathbf{U} \leftarrow_{\$} \mathbb{Z}_q^{(N+\ell) \times n}$ and invokes $\mathbf{R} \leftarrow_{\$} \text{SamplePre}(\mathbf{T}^{(i)}, \overline{\mathbf{A}}^{(i)}, \mathbf{U}, \gamma)$ to get $rk_{i' \rightarrow j}^{f_M} := (\mathbf{R} \mid \begin{smallmatrix} \mathbf{0} \\ \mathbf{M} \end{smallmatrix})$, rather than using $\mathbf{A}'^{(j)}\mathbf{S} + \mathbf{E} - \begin{pmatrix} \mathbf{0} \\ \mathbf{M} \end{pmatrix} \underline{\mathbf{A}}^{(i)}$ with $\mathbf{S} \leftarrow_{\$} \chi^{n \times n}, \mathbf{E} \leftarrow_{\$} \chi^{(N+\ell) \times n}$ as in G_1 .
- Otherwise, the challenger answers the query just like G_1 , that is, $\mathbf{R} \leftarrow_{\$} \text{SamplePre}(\mathbf{T}^{(i)}, \overline{\mathbf{A}}^{(i)}, \mathbf{A}'^{(j)}\mathbf{S} + \mathbf{E} - \begin{pmatrix} \mathbf{0} \\ \mathbf{M} \end{pmatrix} \underline{\mathbf{A}}^{(i)}, \gamma)$ with $\mathbf{S} \leftarrow_{\$} \chi^{n \times n}, \mathbf{E} \leftarrow_{\$} \chi^{(N+\ell) \times n}$.

Clearly, $G_{1,0}$ is identical to G_1 . Thus, we have $\Pr_1[\text{Win}] = \Pr_{1,0}[\text{Win}]$

For $v \in [n]$, let Corr_v denote the event that \mathcal{A} queries v to the corruption oracle \mathcal{O}_{COR} . If Corr_v occurs, \mathcal{A} is not allowed to issue any re-encryption key query of the form $\mathcal{O}_{\text{REKEY}}(i', v, f_M)$ to avoid trivial attacks. Therefore, $G_{1,v-1}$ is identical to $G_{1,v}$ in the case Corr_v occurs, i.e., $\Pr_{1,v-1}[\text{Win} \wedge \text{Corr}_v] = \Pr_{1,v}[\text{Win} \wedge \text{Corr}_v]$. Consequently, we have

$$|\Pr_{1,v-1}[\text{Win}] - \Pr_{1,v}[\text{Win}]| = |\Pr_{1,v-1}[\text{Win} \wedge \neg \text{Corr}_v] - \Pr_{1,v}[\text{Win} \wedge \neg \text{Corr}_v]|. \quad (4)$$

Claim 1. For each $v \in [n]$, $|\Pr_{1,v-1}[\text{Win}] - \Pr_{1,v}[\text{Win}]| \leq \text{Adv}_{[n,q,\chi,N+\ell,\mathcal{B}]}^{nQ\text{-LWE}}(\lambda)$.

Proof. We construct a PPT algorithm \mathcal{B} to break the $nQ\text{-LWE}_{n,q,\chi,N+\ell}$ assumption by simulating $G_{1,v-1}/G_{1,v}$ for \mathcal{A} as follows.

Algorithm \mathcal{B} . Given $(\mathbf{B} \in \mathbb{Z}_q^{(N+\ell) \times n}, \mathbf{Z} \in \mathbb{Z}_q^{(N+\ell) \times nQ})$, \mathcal{B} wants to distinguish $\mathbf{Z} = \mathbf{B}\mathbf{S} + \mathbf{E}$ from $\mathbf{Z} \leftarrow_{\$} \mathbb{Z}_q^{(N+\ell) \times nQ}$, where $\mathbf{B} \leftarrow_{\$} \mathbb{Z}_q^{(N+\ell) \times n}, \mathbf{S} \leftarrow_{\$} \chi^{n \times nQ}, \mathbf{E} \leftarrow_{\$} \chi^{(N+\ell) \times nQ}$. \mathcal{B} parses $\mathbf{B} = \begin{pmatrix} \overline{\mathbf{B}} \\ \underline{\mathbf{B}} \end{pmatrix}$ with $\overline{\mathbf{B}} \in \mathbb{Z}_q^{N \times n}, \underline{\mathbf{B}} \in \mathbb{Z}_q^{\ell \times n}$, and parses $\mathbf{Z} = (\mathbf{Z}_1 \mid \cdots \mid \mathbf{Z}_Q)$ with each $\mathbf{Z}_k \in \mathbb{Z}_q^{(N+\ell) \times n}$. In the case of $\mathbf{Z} = \mathbf{B}\mathbf{S} + \mathbf{E}$, if we parse $\mathbf{S} = (\mathbf{S}_1 \mid \cdots \mid \mathbf{S}_Q)$ with each $\mathbf{S}_k \in \mathbb{Z}_q^{n \times n}$ and parse $\mathbf{E} = (\mathbf{E}_1 \mid \cdots \mid \mathbf{E}_Q)$ with each $\mathbf{E}_k \in \mathbb{Z}_q^{(N+\ell) \times n}$, then we have $\mathbf{Z}_k = \mathbf{B}\mathbf{S}_k + \mathbf{E}_k$. In the case of $\mathbf{Z} \leftarrow_{\$} \mathbb{Z}_q^{(N+\ell) \times nQ}$, we have that \mathbf{Z}_k is uniformly distributed over $\mathbb{Z}_q^{(N+\ell) \times n}$.

\mathcal{B} simulates $G_{1,v-1}/G_{1,v}$ for \mathcal{A} as follows. \mathcal{B} sets $\text{pp} = \overline{\mathbf{A}'} := \overline{\mathbf{B}}$. For the user v , \mathcal{B} invokes $(\overline{\mathbf{A}}^{(v)}, \mathbf{T}^{(v)}) \leftarrow_{\$} \text{TrapGen}(1^n, 1^N)$, samples $\underline{\mathbf{A}}^{(v)} \leftarrow_{\$} \mathbb{Z}_q^{\ell \times n}$, and sets $pk^{(v)} := (\mathbf{A}^{(v)} := \begin{pmatrix} \overline{\mathbf{A}}^{(v)} \\ \underline{\mathbf{A}}^{(v)} \end{pmatrix}, \underline{\mathbf{A}}'^{(v)} := \underline{\mathbf{B}})$. It is clearly that $\mathbf{A}'^{(v)} = \begin{pmatrix} \overline{\mathbf{A}'} \\ \underline{\mathbf{A}}'^{(v)} \end{pmatrix} = \begin{pmatrix} \overline{\mathbf{B}} \\ \underline{\mathbf{B}} \end{pmatrix} = \mathbf{B}$. For all other users $i \in [n] \setminus \{v\}$, \mathcal{B} invokes $\text{KGen}(\text{pp})$ honestly to generate $(pk^{(i)}, sk^{(i)})$. \mathcal{B} sends $(\text{pp}, \{pk^{(i)}\}_{i \in [n]})$ to \mathcal{A} . \mathcal{B} also chooses a random user index $i' \leftarrow_{\$} [n]$ uniformly as the guess of the challenge user i^* .

- On receiving a re-encryption key query $\mathcal{O}_{\text{REKEY}}(i, j, f_M)$ from \mathcal{A} , if $(i = i')$ and $(j \in \mathcal{Q}_c)$, \mathcal{B} aborts the game, just like $G_{1,v-1}$ and $G_{1,v}$. Otherwise, \mathcal{B} replies the query as follows:
 - If $i = i'$ and $j \leq v-1$, \mathcal{B} samples $\mathbf{U} \leftarrow_{\$} \mathbb{Z}_q^{(N+\ell) \times n}$ and invokes $\mathbf{R} \leftarrow_{\$} \text{SamplePre}(\mathbf{T}^{(i)}, \overline{\mathbf{A}}^{(i)}, \mathbf{U}, \gamma)$ to get $rk_{i' \rightarrow j}^{f_M} := (\mathbf{R} \mid \begin{smallmatrix} \mathbf{0} \\ \mathbf{M} \end{smallmatrix})$, the same as $G_{1,v-1}$ and $G_{1,v}$.

- If $i = i'$ and $j = v$, suppose that this is the k -th $\mathcal{O}_{\text{REKEY}}$ query with $k \in [Q]$, \mathcal{B} makes use of \mathbf{Z}_k to invoke $\mathbf{R} \leftarrow_{\$} \text{SamplePre} \left(\mathbf{T}^{(i')}, \overline{\mathbf{A}}^{(i')}, \mathbf{Z}_k - \binom{0}{\mathbf{M}} \underline{\mathbf{A}}^{(v)}, \gamma \right)$ to get $rk_{i' \rightarrow v}^{f_{\mathbf{M}}} := \left(\mathbf{R} \mid \binom{0}{\mathbf{M}} \right)$.

In the case of $\mathbf{Z} = \mathbf{B}\mathbf{S} + \mathbf{E}$, we have $\mathbf{Z}_k = \mathbf{B}\mathbf{S}_k + \mathbf{E}_k = \mathbf{A}'^{(v)}\mathbf{S}_k + \mathbf{E}_k$ for $\mathbf{S}_k \leftarrow_{\$} \chi^{n \times n}$ and $\mathbf{E} \leftarrow_{\$} \chi^{(N+\ell) \times n}$, so \mathcal{B} 's simulation is identical to $\mathbf{G}_{1,v-1}$. In the case of $\mathbf{Z} \leftarrow_{\$} \mathbb{Z}_q^{(N+\ell) \times nQ}$, we have that \mathbf{Z}_k is uniformly distributed over $\mathbb{Z}_q^{(N+\ell) \times n}$, so \mathcal{B} 's simulation is identical to $\mathbf{G}_{1,v}$.

- Otherwise, \mathcal{B} samples $\mathbf{S} \leftarrow_{\$} \chi^{n \times n}$, $\mathbf{E} \leftarrow_{\$} \chi^{(N+\ell) \times n}$ and invokes $\mathbf{R} \leftarrow_{\$} \text{SamplePre} \left(\mathbf{T}^{(i)}, \overline{\mathbf{A}}^{(i)}, \mathbf{A}'^{(j)}\mathbf{S} + \mathbf{E} - \binom{0}{\mathbf{M}} \underline{\mathbf{A}}^{(i)}, \gamma \right)$ to get $rk_{i' \rightarrow j}^{f_{\mathbf{M}}} := \left(\mathbf{R} \mid \binom{0}{\mathbf{M}} \right)$, the same as $\mathbf{G}_{1,v-1}$ and $\mathbf{G}_{1,v}$.
- On receiving a corruption query $\mathcal{O}_{\text{COR}}(i)$ from \mathcal{A} , if $(i = i')$ or $(i', i) \in \mathcal{Q}_{rk}$, \mathcal{B} aborts the game, just like $\mathbf{G}_{1,v-1}$ and $\mathbf{G}_{1,v}$. If $i = v$ (which means that Corr_v occurs), \mathcal{B} also aborts the game. Otherwise, \mathcal{B} returns $sk^{(i)}$ to \mathcal{A} .
- On receiving the challenge tuple $(i^*, \mathbf{m}_0, \mathbf{m}_1)$ from \mathcal{A} , if $i' \neq i^*$, \mathcal{B} aborts the game. Otherwise, \mathcal{B} chooses a random bit $\beta \leftarrow_{\$} \{0, 1\}$ and generates the challenge ciphertext ct_1^* which encrypts \mathbf{m}_β , just like $\mathbf{G}_{1,v-1}$ and $\mathbf{G}_{1,v}$.
- Finally, \mathcal{B} receives a bit β' from \mathcal{A} , and \mathcal{B} outputs 1 to its own challenger if and only if $\beta' = \beta$ and \mathcal{A} never corrupts v (i.e., $\neg \text{Corr}_v$).

Now we analyze the advantage of \mathcal{B} . Overall, if $\mathbf{Z} = \mathbf{B}\mathbf{S} + \mathbf{E}$, \mathcal{B} simulates $\mathbf{G}_{1,v-1}$ perfectly for \mathcal{A} in the case $\neg \text{Corr}_v$, and if $\mathbf{Z} \leftarrow_{\$} \mathbb{Z}_q^{(N+\ell) \times nQ}$, \mathcal{B} simulates $\mathbf{G}_{1,v}$ perfectly for \mathcal{A} in the case $\neg \text{Corr}_v$. Thus, we have

$$\begin{aligned} \text{Adv}_{[n,q,\chi,N+\ell],\mathcal{B}}^{nQ\text{-LWE}}(\lambda) &= \left| \Pr[\mathcal{B}(\mathbf{B}, \mathbf{Z} = \mathbf{B}\mathbf{S} + \mathbf{E}) = 1] - \Pr[\mathcal{B}(\mathbf{B}, \mathbf{Z} \leftarrow_{\$} \mathbb{Z}_q^{(N+\ell) \times nQ}) = 1] \right| \\ &= \left| \Pr_{1,v-1}[\text{Win} \wedge \neg \text{Corr}_v] - \Pr_{1,v}[\text{Win} \wedge \neg \text{Corr}_v] \right|. \end{aligned} \quad (5)$$

Taking (4) and (5) together, Claim 1 follows. \blacksquare

Game \mathbf{G}_2 : It's the same as \mathbf{G}_1 , except for the reply to \mathcal{A} 's re-encryption query $\mathcal{O}_{\text{REKEY}}(i', j, f_{\mathbf{M}})$ when the query leads to no trivial attacks.

- If $i = i'$ (and $j \in [n]$), the challenger uniformly samples $\mathbf{U} \leftarrow_{\$} \mathbb{Z}_q^{(N+\ell) \times n}$ and uses \mathbf{U} to invoke $\mathbf{R} \leftarrow_{\$} \text{SamplePre} \left(\mathbf{T}^{(i')}, \overline{\mathbf{A}}^{(i')}, \mathbf{U}, \gamma \right)$ to obtain $rk_{i' \rightarrow j}^{f_{\mathbf{M}}} := \left(\mathbf{R} \mid \binom{0}{\mathbf{M}} \right)$, and return $rk_{i' \rightarrow j}^{f_{\mathbf{M}}}$ to \mathcal{A} .

Clearly, $\mathbf{G}_2 = \mathbf{G}_{1,n}$ and $\Pr_2[\text{Win}] = \Pr_{1,n}[\text{Win}]$. Thus by Claim 1, we have

$$\left| \Pr_1[\text{Win}] - \Pr_2[\text{Win}] \right| \leq n \cdot \text{Adv}_{[n,q,\chi,N+\ell],\mathcal{B}}^{nQ\text{-LWE}}(\lambda). \quad (6)$$

Game \mathbf{G}_3 : It is the same as \mathbf{G}_2 , except for the reply to \mathcal{A} 's re-encryption query $\mathcal{O}_{\text{REKEY}}(i = i', j, f_{\mathbf{M}})$. If the query does not lead to any trivial attack, then the challenger samples \mathbf{R} by $\mathbf{R} \leftarrow_{\$} D_{\mathbb{Z}^{(N+\ell) \times n}, \gamma}$, instead of invoking $\mathbf{R} \leftarrow_{\$} \text{SamplePre} \left(\mathbf{T}^{(i')}, \overline{\mathbf{A}}^{(i')}, \mathbf{U} \leftarrow_{\$} \mathbb{Z}_q^{(N+\ell) \times n}, \gamma \right)$ as in \mathbf{G}_2 .

Since $\gamma \geq O(\sqrt{n \log q}) \cdot \omega(\sqrt{\log n})$, according to the indistinguishability of preimage-sampling of Lemma 4 in Appendix A.1, \mathbf{G}_3 is statistically close to \mathbf{G}_2 . Thus, $\left| \Pr_2[\text{Win}] - \Pr_3[\text{Win}] \right| \leq \text{negl}(\lambda)$.

Note that in G_3 , the trapdoor $\mathbf{T}^{(i')}$ is not needed any more.

Game G_4 : It is the same as G_3 , except for the generation of $pk^{(i')} = (\mathbf{A}^{(i')}, \underline{\mathbf{A}}'^{(i')})$. In this game, the challenger samples $\mathbf{A}^{(i')} \leftarrow_{\mathfrak{s}} \mathbb{Z}_q^{(N+\ell) \times n}$ uniformly, rather than using the algorithm TrapGen as in G_3 . According to Lemma 2 in Appendix A.1, G_4 is statistically close to G_3 . Thus, $|\Pr_3[\text{Win}] - \Pr_4[\text{Win}]| \leq \text{negl}(\lambda)$.

Game G_5 : It is the same as G_4 , except for the generation of the challenge ciphertext ct_1^* . Now the challenger picks $ct_1^* \leftarrow_{\mathfrak{s}} \mathbb{Z}_q^{N+\ell}$ uniformly, rather than generating it by $ct_1^* := \mathbf{A}^{(i^*)}\mathbf{s} + \mathbf{e} + \begin{pmatrix} \mathbf{0} \\ p\mathbf{m}_\beta \end{pmatrix}$ with $\mathbf{s} \leftarrow_{\mathfrak{s}} \chi^n$, $\mathbf{e} \leftarrow_{\mathfrak{s}} \chi^{N+\ell}$ as in G_4 .

Clearly, the challenge bit β is completely hidden to \mathcal{A} , thus $\Pr_5[\text{Win}] = \frac{1}{2}$. Next we show that G_4 and G_5 are computationally indistinguishable.

Claim 2. $|\Pr_4[\text{Win}] - \Pr_5[\text{Win}]| \leq \text{Adv}_{[n,q,\chi,N+\ell],\mathcal{B}'}^{\text{LWE}}(\lambda)$.

Proof. We construct a PPT algorithm \mathcal{B}' to break the $\text{LWE}_{n,q,\chi,N+\ell}$ assumption by simulating $\mathsf{G}_4/\mathsf{G}_5$ for \mathcal{A} as follows.

Algorithm \mathcal{B}' . Given $(\mathbf{B} \in \mathbb{Z}_q^{(N+\ell) \times n}, \mathbf{z} \in \mathbb{Z}_q^{N+\ell})$, \mathcal{B}' wants to distinguish $\mathbf{z} = \mathbf{B}\mathbf{s} + \mathbf{e}$ from $\mathbf{z} \leftarrow_{\mathfrak{s}} \mathbb{Z}_q^{N+\ell}$, where $\mathbf{B} \leftarrow_{\mathfrak{s}} \mathbb{Z}_q^{(N+\ell) \times n}$, $\mathbf{s} \leftarrow_{\mathfrak{s}} \chi^n$ and $\mathbf{e} \leftarrow_{\mathfrak{s}} \chi^{N+\ell}$.

\mathcal{B}' simulates $\mathsf{G}_4/\mathsf{G}_5$ for \mathcal{A} as follows. \mathcal{B}' invokes Setup honestly to generate $\text{pp} = \overline{\mathbf{A}}'$, and chooses a random user index $i' \leftarrow_{\mathfrak{s}} [n]$ uniformly as the guess of the challenge user i^* . For the user i' , \mathcal{B}' samples $\mathbf{K}^{(i')} \leftarrow_{\mathfrak{s}} \{0,1\}^{\ell \times N}$, and sets $pk^{(i')} = (\mathbf{A}^{(i')} := \mathbf{B}, \underline{\mathbf{A}}'^{(i')} := -\mathbf{K}^{(i')}\overline{\mathbf{A}}')$ and $sk^{(i')} = \perp$. For all other users $i \in [n] \setminus \{i'\}$, \mathcal{B}' invokes $\text{KGen}(\text{pp})$ honestly to generate $(pk^{(i)}, sk^{(i)})$. \mathcal{B}' sends $(\text{pp}, \{pk^{(i)}\}_{i \in [n]})$ to \mathcal{A} .

- On receiving a re-encryption key query $\mathcal{O}_{\text{REKEY}}(i, j, f_{\mathbf{M}})$ from \mathcal{A} , \mathcal{B}' replies \mathcal{A} just like G_4 and G_5 . More precisely, if $(i = i')$ and $(j \in \mathcal{Q}_c)$, \mathcal{B}' aborts the game; otherwise, \mathcal{B}' replies the query as follows:
 - If $i = i'$, \mathcal{B}' samples $\mathbf{R} \leftarrow_{\mathfrak{s}} D_{\mathbb{Z}^{(N+\ell) \times N}, \gamma}$ to get $rk_{i' \rightarrow j}^{f_{\mathbf{M}}} := (\mathbf{R} \mid \begin{smallmatrix} \mathbf{0} \\ \mathbf{M} \end{smallmatrix})$.
 - If $i \neq i'$, \mathcal{B}' invokes $\mathbf{R} \leftarrow_{\mathfrak{s}} \text{SamplePre}(\mathbf{T}^{(i)}, \overline{\mathbf{A}}^{(i)}, \mathbf{A}^{(j)}\mathbf{s} + \mathbf{E}, \gamma)$ to get $rk_{i' \rightarrow j}^{f_{\mathbf{M}}} := (\mathbf{R} \mid \begin{smallmatrix} \mathbf{0} \\ \mathbf{M} \end{smallmatrix})$ using the secret key $sk^{(i)} = (\mathbf{T}^{(i)}, \mathbf{K}^{(i)})$ of user i .
- On receiving a corruption query $\mathcal{O}_{\text{COR}}(i)$ from \mathcal{A} , \mathcal{B}' replies \mathcal{A} just like G_4 and G_5 . More precisely, if $(i = i')$ or $(i', i) \in \mathcal{Q}_{rk}$, \mathcal{B}' aborts the game; otherwise, \mathcal{B}' returns $sk^{(i)}$ to \mathcal{A} .
- On receiving the challenge tuple $(i^*, \mathbf{m}_0, \mathbf{m}_1)$ from \mathcal{A} , if $i' \neq i^*$, \mathcal{B}' aborts the game. Otherwise, \mathcal{B}' chooses a random bit $\beta \leftarrow_{\mathfrak{s}} \{0,1\}$ and computes $ct_1^* := \mathbf{z} + \begin{pmatrix} \mathbf{0} \\ p\mathbf{m}_\beta \end{pmatrix}$. Then \mathcal{B}' sends ct_1^* to \mathcal{A} .

In the case of $\mathbf{z} = \mathbf{B}\mathbf{s} + \mathbf{e}$, $ct_1^* = \mathbf{B}\mathbf{s} + \mathbf{e} + \begin{pmatrix} \mathbf{0} \\ p\mathbf{m}_\beta \end{pmatrix} = \mathbf{A}^{(i^*)}\mathbf{s} + \mathbf{e} + \begin{pmatrix} \mathbf{0} \\ p\mathbf{m}_\beta \end{pmatrix}$, so \mathcal{B}' 's simulation is identical to G_4 . In the case of $\mathbf{z} \leftarrow_{\mathfrak{s}} \mathbb{Z}_q^{N+\ell}$, $ct_1^* = \mathbf{z} + \begin{pmatrix} \mathbf{0} \\ p\mathbf{m}_\beta \end{pmatrix}$ is uniformly distributed, so \mathcal{B}' 's simulation is identical to G_5 .
- Finally, \mathcal{B}' receives a bit β' from \mathcal{A} , and \mathcal{B}' outputs 1 to its own challenger if and only if $\beta' = \beta$.

Now we analyze the advantage of \mathcal{B}' . Overall, if $\mathbf{z} = \mathbf{B}\mathbf{s} + \mathbf{e}$, \mathcal{B}' simulates G_4 perfectly for \mathcal{A} , and if $\mathbf{z} \leftarrow_{\mathfrak{s}} \mathbb{Z}_q^{N+\ell}$, \mathcal{B}' simulates G_5 perfectly for \mathcal{A} . Thus,

$$\begin{aligned} \text{Adv}_{[n,q,\chi,N+\ell],\mathcal{B}'}^{\text{LWE}}(\lambda) &= |\Pr[\mathcal{B}'(\mathbf{B}, \mathbf{z} = \mathbf{B}\mathbf{s} + \mathbf{e}) = 1] - \Pr[\mathcal{B}'(\mathbf{B}, \mathbf{z} \leftarrow_{\mathfrak{s}} \mathbb{Z}_q^{N+\ell}) = 1]| \\ &= |\Pr_4[\text{Win}] - \Pr_5[\text{Win}]|. \quad \blacksquare \end{aligned}$$

Finally, taking all things together, Theorem 1 follows. \square

Note that in the proof of Theorem 1, in G_5 , the challenge ciphertext ct_1^* has been replaced by a random vector in $\mathbb{Z}_q^{N+\ell}$, thus, we have the following corollary.

Corollary 1 (Ciphertext Pseudorandomness of $\text{FPRE}_{\text{LWE}}^{\text{lin}}$). *Assume that the $\text{LWE}_{n,q,\chi,N+\ell}$ -assumption holds, then the scheme $\text{FPRE}_{\text{LWE}}^{\text{lin}}$ proposed in Fig. 7 has ciphertext pseudorandomness (CPR). More precisely, for any PPT adversary \mathcal{A} that makes at most Q times of $\mathcal{O}_{\text{REKEY}}$ queries and for any polynomial \mathbf{n} , there exists a PPT algorithm \mathcal{B} against the LWE assumption such that $\text{Adv}_{\text{FPRE},\mathcal{A},\mathbf{n}}^{\text{CPR}}(\lambda) \leq (2\mathbf{n}^2nQ + \mathbf{n}) \cdot \text{Adv}_{[n,q,\chi,N+\ell],\mathcal{B}}^{\text{LWE}}(\lambda) + \text{negl}(\lambda)$.*

Proof sketch. We can prove the corollary via a sequence of games G_0 – G_9 . Here G_0 – G_5 are similar to those in the proof of Theorem 1, and in particular, in G_5 , the challenge ciphertext ct_1^* is already pseudorandom. The only thing we need to do is to reverse the changes introduced in G_1 – G_4 , and this can be done by the additional games G_6 – G_9 which are symmetric to G_4 – G_0 . \square

Theorem 2 (Ciphertext Unlinkability of $\text{FPRE}_{\text{LWE}}^{\text{lin}}$). *Assume that the $\text{LWE}_{n,q,\chi,N+\ell}$ -assumption holds, then the scheme $\text{FPRE}_{\text{LWE}}^{\text{lin}}$ proposed in Fig. 7 has ciphertext unlinkability (CUL). More precisely, for any PPT adversary \mathcal{A} that makes at most Q times of $\mathcal{O}_{\text{REKEY}}$ queries and for any polynomial \mathbf{n} , there exists a PPT algorithm \mathcal{B} against the LWE assumption such that*

$$\text{Adv}_{\text{FPRE},\mathcal{A},\mathbf{n}}^{\text{CUL}}(\lambda) \leq (3\mathbf{n}^2nQ + \mathbf{n}^2 + 2\mathbf{n}) \cdot \text{Adv}_{[n,q,\chi,N+\ell],\mathcal{B}}^{\text{LWE}}(\lambda) + \text{negl}(\lambda).$$

Proof of Theorem 2. We prove the theorem via a sequence of games G'_0 – G'_9 , where G'_0 is the CUL experiment, and in G'_9 , \mathcal{A} has a negligible advantage.

Game G'_0 : This is the CUL experiment $\text{Exp}_{\text{FPRE},\mathcal{A},\mathbf{n}}^{\text{CUL}}$ (cf. Fig. 6). Let Win denote the event that $\beta' = \beta$. By definition, $\text{Adv}_{\text{FPRE},\mathcal{A},\mathbf{n}}^{\text{CUL}}(\lambda) = |\Pr_0'[\text{Win}] - \frac{1}{2}|$.

Let $\mathbf{pp} = \overline{\mathbf{A}}$ and $pk^{(i)} = (\mathbf{A}^{(i)}, \underline{\mathbf{A}}'^{(i)})$, $sk^{(i)} = (\mathbf{T}^{(i)}, \mathbf{K}^{(i)})$ the public key and secret key of user $i \in [n]$. In this game, the challenger answers \mathcal{A} 's $\mathcal{O}_{\text{REKEY}}$, \mathcal{O}_{COR} queries and generates the challenge ciphertexts (ct_1^*, ct_2^*) as follows.

- On receiving a re-encryption key query $\mathcal{O}_{\text{REKEY}}(i, j, f_{\mathbf{M}})$ from \mathcal{A} , the challenger returns \perp to \mathcal{A} directly if trivial attacks ($i = i^*$) and ($j \in \mathcal{Q}_c$) occur. Otherwise, the challenger adds (i, j) to \mathcal{Q}_{rk} , samples $\mathbf{S} \leftarrow_{\mathfrak{s}} \chi^{n \times n}$, $\mathbf{E} \leftarrow_{\mathfrak{s}} \chi^{(N+\ell) \times n}$, invokes $\mathbf{R} \leftarrow_{\mathfrak{s}} \text{SamplePre}\left(\mathbf{T}^{(i)}, \overline{\mathbf{A}}^{(i)}, \mathbf{A}'^{(j)}\mathbf{S} + \mathbf{E} - \begin{pmatrix} \mathbf{0} \\ \mathbf{M} \end{pmatrix} \underline{\mathbf{A}}^{(i)}, \gamma\right)$, where $\mathbf{A}'^{(j)} = \begin{pmatrix} \overline{\mathbf{A}}^{(j)} \\ \underline{\mathbf{A}}'^{(j)} \end{pmatrix}$, and returns $rk_{i \rightarrow j}^{f_{\mathbf{M}}} := (\mathbf{R} \mid \begin{pmatrix} \mathbf{0} \\ \mathbf{M} \end{pmatrix})$ to \mathcal{A} .

- On receiving a corruption query $\mathcal{O}_{\text{COR}}(i)$ from \mathcal{A} , the challenger returns \perp to \mathcal{A} directly if trivial attacks ($i = i^*$) or ($i = j^*$) or $(i^*, i) \in \mathcal{Q}_{rk}$ occur. Otherwise, the challenger adds i to \mathcal{Q}_c and returns $sk^{(i)}$ to \mathcal{A} .
- On receiving the challenge tuple $(i^*, j^*, (f_{\mathbf{M}}, \mathbf{m}), (\mathbf{m}_1, \mathbf{m}_2))$ from \mathcal{A} , the challenger first checks if trivial attacks ($i^* \in \mathcal{Q}_c$) or ($j^* \in \mathcal{Q}_c$) or $(\exists j \in \mathcal{Q}_c$ s.t. $(i^*, j) \in \mathcal{Q}_{rk})$ occur. If yes, the challenger aborts the game with \mathcal{A} by returning a random bit. Otherwise, the challenger chooses a random bit $\beta \leftarrow_{\$} \{0, 1\}$. In the case $\beta = 0$, the challenger generates ct_1^* by invoking $ct_1^* \leftarrow_{\$} \text{Enc}_1(pk^{(i^*)}, \mathbf{m})$ and generates ct_2^* by invoking $ct_2^* \leftarrow_{\$} \text{FReEnc}(rk_{i^* \rightarrow j^*}^{f_{\mathbf{M}}}, ct_1^*)$ where $rk_{i^* \rightarrow j^*}^{f_{\mathbf{M}}} \leftarrow_{\$} \text{FReKGen}(pk^{(i^*)}, sk^{(i^*)}, pk^{(j^*)}, f_{\mathbf{M}})$. In the case $\beta = 1$, the challenger generates ct_1^* by invoking $ct_1^* \leftarrow_{\$} \text{Enc}_1(pk^{(i^*)}, \mathbf{m}_1)$ and generates ct_2^* by invoking $ct_2^* \leftarrow_{\$} \text{Enc}_2(pk^{(j^*)}, \mathbf{m}_2)$. The challenger sends the challenge ciphertexts (ct_1^*, ct_2^*) to \mathcal{A} .

Game G'_1 : This game is similar to G'_0 , except that, in the case of $\beta = 0$, the challenger picks the first-level challenge ciphertext $ct_1^* \leftarrow_{\$} \mathbb{Z}_q^{N+\ell}$ uniformly at random, rather than generating it by $ct_1^* \leftarrow_{\$} \text{Enc}_1(pk^{(i^*)}, \mathbf{m})$ as in G'_0 .

Claim 3. $|\Pr'_0[\text{Win}] - \Pr'_1[\text{Win}]| \leq \text{Adv}_{\text{FPRE}, \mathcal{B}, n}^{\text{CPR}}(\lambda)$.

Proof. We construct a PPT algorithm \mathcal{B} to break the ciphertext pseudorandomness (CPR) of $\text{FPRE}_{\text{LWE}}^{\text{lin}}$ by simulating G'_0/G'_1 for \mathcal{A} as follows.

Algorithm \mathcal{B} . Algorithm \mathcal{B} is given the public parameter $\text{pp}, \{pk^{(i)}\}_{i \in [n]}$ from its own challenger and has access to its own oracles $\mathcal{O}_{\text{REKEY}}, \mathcal{O}_{\text{COR}}$. \mathcal{B} initializes $\mathcal{Q}_{rk} = \emptyset, \mathcal{Q}_c = \emptyset, i^* = \perp, j^* = \perp$ and sends $\text{pp}, \{pk^{(i)}\}_{i \in [n]}$ to \mathcal{A} .

- On receiving a re-encryption key query $(i, j, f_{\mathbf{M}})$ from \mathcal{A} , \mathcal{B} checks \mathcal{A} 's trivial attack by checking if $(i = i^*)$ and $(j \in \mathcal{Q}_c)$, just like G'_0 and G'_1 . If trivial attacks occur, \mathcal{B} returns \perp to \mathcal{A} , otherwise \mathcal{B} adds (i, j) to \mathcal{Q}_{rk} and queries $(i, j, f_{\mathbf{M}})$ to its own oracle $\mathcal{O}_{\text{REKEY}}$. On receiving $rk_{i \rightarrow j}^{f_{\mathbf{M}}}$ from $\mathcal{O}_{\text{REKEY}}(i, j, f_{\mathbf{M}})$, \mathcal{B} returns $rk_{i \rightarrow j}^{f_{\mathbf{M}}}$ to \mathcal{A} .
- On receiving a corruption query i from \mathcal{A} , \mathcal{B} checks \mathcal{A} 's trivial attack by checking if $(i = i^*)$ or $(i = j^*)$ or $(i^*, i) \in \mathcal{Q}_{rk}$, just like G'_0 and G'_1 . If trivial attacks occur, \mathcal{B} returns \perp to \mathcal{A} , otherwise \mathcal{B} adds i to \mathcal{Q}_c and queries i to its own oracle \mathcal{O}_{COR} . On receiving $sk^{(i)}$ from $\mathcal{O}_{\text{COR}}(i)$, \mathcal{B} returns $sk^{(i)}$ to \mathcal{A} .
- On receiving the challenge tuple $(i^*, j^*, (f_{\mathbf{M}}, \mathbf{m}), (\mathbf{m}_1, \mathbf{m}_2))$ from \mathcal{A} , \mathcal{B} first checks if $(i^* \in \mathcal{Q}_c)$ or $(j^* \in \mathcal{Q}_c)$ or $(\exists j \in \mathcal{Q}_c$ s.t. $(i^*, j) \in \mathcal{Q}_{rk})$ to identify trivial attacks, just like G'_0 and G'_1 . If yes, trivial attacks happen, and \mathcal{B} aborts the game with \mathcal{A} and returns a random bit $b' \leftarrow_{\$} \{0, 1\}$ to its own challenger. Otherwise, \mathcal{B} queries $(i^*, j^*, f_{\mathbf{M}})$ to its own oracle $\mathcal{O}_{\text{REKEY}}$ to obtain $rk_{i^* \rightarrow j^*}^{f_{\mathbf{M}}}$. Moreover, \mathcal{B} sends a challenge tuple (i^*, \mathbf{m}) to its own challenger, and receives a challenge ciphertext \tilde{ct}_1^* from its own challenger, which either encrypts \mathbf{m} under $pk^{(i^*)}$, i.e., $\tilde{ct}_1^* \leftarrow_{\$} \text{Enc}_1(pk^{(i^*)}, \mathbf{m})$, or is uniformly chosen from $\mathbb{Z}_q^{N+\ell}$, i.e., $\tilde{ct}_1^* \leftarrow_{\$} \text{Enc}_1(pk^{(i^*)}, \mathbf{m})$, depending on the challenge bit b that \mathcal{B} 's challenger picks. Then \mathcal{B} chooses a random bit $\beta \leftarrow_{\$} \{0, 1\}$. In the case $\beta = 0$,

- \mathcal{B} sets $ct_1^* := \tilde{ct}_1^*$ and generates ct_2^* by invoking $ct_2^* \leftarrow_s \text{FReEnc}(rk_{i^* \rightarrow j^*}^{f_{\mathbf{M}}}, ct_1^*)$.
 In the case $\beta = 1$, \mathcal{B} generates ct_1^* by invoking $ct_1^* \leftarrow_s \text{Enc}_1(pk^{(i^*)}, \mathbf{m}_1)$ and generates ct_2^* by invoking $ct_2^* \leftarrow_s \text{Enc}_2(pk^{(j^*)}, \mathbf{m}_2)$, just like G'_0 and G'_1 .
 – Finally, \mathcal{B} receives a bit β' from \mathcal{A} , and \mathcal{B} outputs 1 if and only if $\beta' = \beta$.

In the simulation, as long as \mathcal{A} implements trivial attacks $i^* \in \mathcal{Q}_c$ or $j^* \in \mathcal{Q}_c$ or $(\exists j \in \mathcal{Q}_c \text{ s.t. } (i^*, j) \in \mathcal{Q}_{rk})$, \mathcal{B} will abort the experiment, just like G'_0 and G'_1 . Otherwise, no trivial attacks from \mathcal{A} implies that $i^* \notin \mathcal{Q}_c$ and there doesn't exist any re-encryption key query $(i^*, j) \in \mathcal{Q}_{rk} \cup \{(i^*, j^*)\}$ from i^* to $j \in \mathcal{Q}_c$, where $\mathcal{Q}_{rk} \cup \{(i^*, j^*)\}$ is exactly the re-encryption key query set for \mathcal{B} 's challenger. Therefore, \mathcal{B} 's query $(i^*, j^*, f_{\mathbf{M}})$ to its own oracle $\mathcal{O}_{\text{ReKey}}$ does not lead to any trivial attacks in the ciphertext pseudorandomness (CPR) experiment (cf. Fig. 2), and \mathcal{B} 's challenger will answer this query and return $rk_{i^* \rightarrow j^*}^{f_{\mathbf{M}}}$ to \mathcal{B} . So \mathcal{B} 's simulation of $rk_{i^* \rightarrow j^*}^{f_{\mathbf{M}}}$ for \mathcal{A} is perfect.

Now we analyze the advantage of \mathcal{B} . Overall, if the challenge ciphertext \tilde{ct}_1^* that \mathcal{B} received from its own challenger is generated by $\tilde{ct}_1^* \leftarrow_s \text{Enc}_1(pk^{(i^*)}, \mathbf{m})$, \mathcal{B} simulates G'_0 perfectly for \mathcal{A} , and if \tilde{ct}_1^* is uniformly chosen from $\mathbb{Z}_q^{N+\ell}$ by \mathcal{B} 's challenger, \mathcal{B} simulates G'_1 perfectly for \mathcal{A} . Therefore, \mathcal{B} will successfully distinguish $\tilde{ct}_1^* \leftarrow_s \text{Enc}_1(pk^{(i^*)}, \mathbf{m})$ from $\tilde{ct}_1^* \leftarrow_s \mathbb{Z}_q^{N+\ell}$ and break the ciphertext pseudorandomness (CPR) of $\text{FPRE}_{\text{LWE}}^{\text{lin}}$ as long as the probability that Win occurs in G'_0 differs non-negligibly from that in G'_1 , and we have $|\Pr'_0[\text{Win}] - \Pr'_1[\text{Win}]| \leq \text{Adv}_{\text{FPRE}, \mathcal{B}, n}^{\text{CPR}}(\lambda)$. ■

Game G'_2 : It is the same as G'_1 , except that, at the beginning of the game, the challenger chooses $i' \leftarrow_s [n]$ uniformly as the guess of the challenge user i^* , and will abort the game and return a random bit in the following cases.

- **Case 1.** \mathcal{A} issues the challenge tuple $(i^*, j^*, (f_{\mathbf{M}}, \mathbf{m}), (\mathbf{m}_1, \mathbf{m}_2))$ but $i' \neq i^*$.
- **Case 2.** \mathcal{A} issues a re-encryption key query $\mathcal{O}_{\text{ReKey}}(i, j, f_{\mathbf{M}})$ such that $(i = i')$ and $(j \in \mathcal{Q}_c)$ before issuing its challenge.
- **Case 3.** \mathcal{A} issues a corruption query $\mathcal{O}_{\text{Cor}}(i)$ such that $(i = i')$ or $(i = j^*)$ or $(i', i) \in \mathcal{Q}_{rk}$ before issuing its challenge.

Case 1 suggests that the challenger's guess is wrong. Now in G'_2 , the challenger will abort the game if the guess is wrong. If the guess is correct, i.e., $i' = i^*$, Case 2 and Case 3 are in fact trivial attacks, so they will lead to abort anyway in G'_1 and do not contribute to \mathcal{A} 's advantage. Since the challenger will guess i^* correctly with probability $1/n$, we have $|\Pr'_1[\text{Win}] - \frac{1}{2}| = \frac{1}{n} |\Pr'_2[\text{Win}] - \frac{1}{2}|$.

Game G'_3 : It is the same as G'_2 , except for the reply to \mathcal{A} 's re-encryption query $\mathcal{O}_{\text{ReKey}}(i', j, f_{\mathbf{M}})$ which does not lead to any trivial attack.

- If $i = i'$ (and $j \in [n]$), the challenger uniformly samples $\mathbf{U} \leftarrow_s \mathbb{Z}_q^{(N+\ell) \times n}$ and uses \mathbf{U} to invoke $\mathbf{R} \leftarrow_s \text{SamplePre}(\mathbf{T}^{(i')}, \overline{\mathbf{A}}^{(i')}, \mathbf{U}, \gamma)$ to obtain $rk_{i' \rightarrow j}^{f_{\mathbf{M}}} := (\mathbf{R} \mid \mathbf{0})$, and return $rk_{i' \rightarrow j}^{f_{\mathbf{M}}}$ to \mathcal{A} .

Moreover, G'_3 also differs from G'_2 in the generation of the second-level challenge ciphertext ct_2^* in the case of $\beta = 0$. Recall that in G'_2 , in the case of $\beta = 0$, the challenger invokes $rk_{i^* \rightarrow j^*}^{f_M} \leftarrow_s \text{FReKGen}(pk^{(i^*)}, sk^{(i^*)}, pk^{(j^*)}, f_M)$ to compute $ct_2^* \leftarrow_s \text{FReEnc}(rk_{i^* \rightarrow j^*}^{f_M}, ct_1^*)$. Now in this game, the challenger also uniformly samples $\mathbf{U} \leftarrow_s \mathbb{Z}_q^{(N+\ell) \times n}$, uses \mathbf{U} to invoke $\mathbf{R} \leftarrow_s \text{SamplePre}(\mathbf{T}^{(i^*)}, \overline{\mathbf{A}}^{(i^*)}, \mathbf{U}, \gamma)$ to obtain $rk_{i^* \rightarrow j^*}^{f_M} := (\mathbf{R} \mid \mathbf{0}_M)$, then uses $rk_{i^* \rightarrow j^*}^{f_M}$ to compute $ct_2^* \leftarrow_s \text{FReEnc}(rk_{i^* \rightarrow j^*}^{f_M}, ct_1^*)$.

With a similar argument like G_1 - G_2 ($= G_{1.0}$ - $G_{1.n}$) in the proof of Theorem 1, cf. (6), we have $|\Pr'_2[\text{Win}] - \Pr'_3[\text{Win}]| \leq n \cdot \text{Adv}_{[n, q, \chi, N+\ell], \mathcal{B}_1}^{nQ\text{-LWE}}(\lambda)$.

Game G'_4 : It is the same as G'_3 , except for the reply to \mathcal{A} 's re-encryption query $\mathcal{O}_{\text{REKEY}}(i = i', j, f_M)$. If the query does not lead to any trivial attack, then the challenger samples \mathbf{R} by $\mathbf{R} \leftarrow_s D_{\mathbb{Z}^{(N+\ell) \times N}, \gamma}$, instead of invoking $\mathbf{R} \leftarrow_s \text{SamplePre}(\mathbf{T}^{(i')}, \overline{\mathbf{A}}^{(i')}, \mathbf{U} \leftarrow_s \mathbb{Z}_q^{(N+\ell) \times n}, \gamma)$ as in G'_3 .

Moreover, G'_4 also differs from G'_3 in the generation of ct_2^* in the case of $\beta = 0$. Now in this game, in the case of $\beta = 0$, the challenger also samples \mathbf{R} by $\mathbf{R} \leftarrow_s D_{\mathbb{Z}^{(N+\ell) \times N}, \gamma}$, instead of invoking $\mathbf{R} \leftarrow_s \text{SamplePre}(\mathbf{T}^{(i^*)}, \overline{\mathbf{A}}^{(i^*)}, \mathbf{U} \leftarrow_s \mathbb{Z}_q^{(N+\ell) \times n}, \gamma)$ to obtain $rk_{i^* \rightarrow j^*}^{f_M} := (\mathbf{R} \mid \mathbf{0}_M)$, and uses $rk_{i^* \rightarrow j^*}^{f_M}$ to compute $ct_2^* \leftarrow_s \text{FReEnc}(rk_{i^* \rightarrow j^*}^{f_M}, ct_1^*)$.

With a similar argument like G_2 - G_3 in the proof of Theorem 1, we know that G'_4 is statistically close to G'_3 . Thus, $|\Pr'_3[\text{Win}] - \Pr'_4[\text{Win}]| \leq \text{negl}(\lambda)$.

Note that in G'_4 , the trapdoor $\mathbf{T}^{(i')}$ is not needed any more.

Game G'_5 : It is the same as G'_4 , except for the generation of ct_2^* in the case of $\beta = 0$. In this game, in the case of $\beta = 0$, the challenger picks the second-level challenge ciphertext $ct_2^* \leftarrow_s \mathbb{Z}_q^{N+\ell}$ uniformly at random.

Recall that in G'_4 , ct_2^* is generated by invoking $ct_2^* \leftarrow_s \text{FReEnc}(rk_{i^* \rightarrow j^*}^{f_M}, ct_1^*)$, where $ct_1^* \leftarrow_s \mathbb{Z}_q^{N+\ell}$ and $rk_{i^* \rightarrow j^*}^{f_M} := (\mathbf{R} \mid \mathbf{0}_M)$ with $\mathbf{R} \leftarrow_s D_{\mathbb{Z}^{(N+\ell) \times N}, \gamma}$, thus

$$ct_2^* := rk_{i^* \rightarrow j^*}^{f_M} \cdot ct_1^* = \left(\mathbf{R} \mid \mathbf{0}_M \right) \cdot ct_1^* = \mathbf{R} \cdot \overline{ct_1^*} + \begin{pmatrix} \mathbf{0} \\ \mathbf{M} \end{pmatrix} \cdot \underline{ct_1^*}.$$

Since $\mathbf{R} \leftarrow_s D_{\mathbb{Z}, \gamma}^{(N+\ell) \times N}$ with $\gamma \geq O(\sqrt{n \log q}) \cdot \omega(\sqrt{\log n})$ and $\overline{ct_1^*}$ is uniformly distributed over \mathbb{Z}_q^N , according to the indistinguishability of preimage-sampling of Lemma 4 in Appendix A.1, we have that $\mathbf{R} \cdot \overline{ct_1^*}$ is statistically close to the uniform distribution over $\mathbb{Z}_q^{N+\ell}$. Due to the independence between $\mathbf{R} \cdot \overline{ct_1^*}$ and $\begin{pmatrix} \mathbf{0} \\ \mathbf{M} \end{pmatrix} \cdot \underline{ct_1^*}$, we know that the second-level challenge ciphertext $ct_2^* = \mathbf{R} \cdot \overline{ct_1^*} + \begin{pmatrix} \mathbf{0} \\ \mathbf{M} \end{pmatrix} \cdot \underline{ct_1^*}$ generated in G'_4 is statistically indistinguishable from the uniformly chosen $ct_2^* \leftarrow_s \mathbb{Z}_q^{N+\ell}$ in G'_5 . Thus we have $|\Pr'_4[\text{Win}] - \Pr'_5[\text{Win}]| \leq \text{negl}(\lambda)$.

Game G'_6 : It is the same as G'_5 , except for the generation of $pk^{(i')} = (\mathbf{A}^{(i')}, \mathbf{A}'^{(i')})$. Recall that from G'_4 on, the trapdoor $\mathbf{T}^{(i')}$ of $\overline{\mathbf{A}}^{(i')}$ is not needed any more. In G'_6 , the challenger samples $\mathbf{A}^{(i')} \leftarrow_s \mathbb{Z}_q^{(N+\ell) \times n}$ uniformly, rather than using algorithm TrapGen as in G'_5 . According to Lemma 2, G'_6 is statistically close to G'_5 . Thus, $|\Pr'_5[\text{Win}] - \Pr'_6[\text{Win}]| \leq \text{negl}(\lambda)$.

Game G'_7 : It is the same as G'_6 , except that, in the case of $\beta = 1$, the challenger also picks the first-level challenge ciphertext $ct_1^* \leftarrow_{\mathcal{S}} \mathbb{Z}_q^{N+\ell}$ uniformly at random, rather than generating it by $ct_1^* \leftarrow_{\mathcal{S}} \text{Enc}_1(pk^{(i^*)}, \mathbf{m}_1)$ as in G'_6 , i.e., $ct_1^* := \mathbf{A}^{(i^*)}\mathbf{s} + \mathbf{e} + \begin{pmatrix} \mathbf{0} \\ p\mathbf{m}_1 \end{pmatrix}$ with $\mathbf{s} \leftarrow_{\mathcal{S}} \chi^n, \mathbf{e} \leftarrow_{\mathcal{S}} \chi^{N+\ell}$ in G'_6 .

Due to the game changes introduced in G'_2 and G'_6 , we have that $i' = i^*$ and $\mathbf{A}^{(i^*)} = \mathbf{A}^{(i')}$ is uniformly sampled from $\mathbb{Z}_q^{(N+\ell) \times n}$. Then according to the LWE assumption, $\mathbf{A}^{(i^*)}\mathbf{s} + \mathbf{e}$ is computationally indistinguishable from the uniform distribution over $\mathbb{Z}_q^{N+\ell}$. Thus the challenge ciphertext $ct_1^* := \mathbf{A}^{(i^*)}\mathbf{s} + \mathbf{e} + \begin{pmatrix} \mathbf{0} \\ p\mathbf{m}_1 \end{pmatrix}$ generated in G'_6 in the case of $\beta = 1$ is also computationally indistinguishable from the uniformly chosen $ct_1^* \leftarrow_{\mathcal{S}} \mathbb{Z}_q^{N+\ell}$ in G'_7 in the case of $\beta = 1$. Thus we have $|\Pr'_6[\text{Win}] - \Pr'_7[\text{Win}]| \leq \text{Adv}_{[n,q,\chi,N+\ell],\mathcal{B}_2}^{\text{LWE}}(\lambda)$.

Game G'_8 : It is the same as G'_7 , except for the following differences. Firstly, at the beginning of the game, the challenger also chooses a random user index $j' \leftarrow_{\mathcal{S}} [n]$ uniformly as the guess of the challenge user j^* , and will abort the game and return a random bit in the following cases.

- **Case 1.** \mathcal{A} issues the challenge tuple $(i^*, j^*, (f, \mathbf{m}), (\mathbf{m}_1, \mathbf{m}_2))$ but $j' \neq j^*$.
- **Case 2.** \mathcal{A} issues a corruption query $\mathcal{O}_{\text{COR}}(i)$ s.t. $i = j'$ before challenge.

Case 1 suggests that the challenger's guess is wrong. Case 2 will lead to abort anyway and does not contribute to \mathcal{A} 's advantage, just like G'_7 .

Secondly, for the generation of $pk^{(j')} = (\mathbf{A}^{(j')}, \underline{\mathbf{A}}'^{(j')})$, now the challenger generates $\underline{\mathbf{A}}'^{(j')}$ by sampling $\underline{\mathbf{A}}'^{(j')} \leftarrow_{\mathcal{S}} \mathbb{Z}_q^{\ell \times n}$, instead of computing $\underline{\mathbf{A}}'^{(j')} := \mathbf{K}^{(j')} \overline{\mathbf{A}}'$ with $\mathbf{K}^{(j')} \leftarrow_{\mathcal{S}} \{0, 1\}^{\ell \times N}$ as in G'_7 . Note that for each row of $\mathbf{K}^{(j')} \leftarrow_{\mathcal{S}} \{0, 1\}^{\ell \times N}$ and for any q 's prime factor p' , we have that \mathbf{H}_{∞} (each row of $\mathbf{K}^{(j')} \bmod p'$) = N . Given $N \geq 2n \log q + 2\omega(\log \lambda)$, we know that \mathbf{H}_{∞} (each row of $\mathbf{K}^{(j')} \bmod p'$) $\geq 2n \log q + 2\omega(\log \lambda)$. Then according to Lemma 5, $\underline{\mathbf{A}}'^{(j')} := \mathbf{K}^{(j')} \overline{\mathbf{A}}'$ generated in G'_7 is statistically close to the uniform distribution $\underline{\mathbf{A}}'^{(j')} \leftarrow_{\mathcal{S}} \mathbb{Z}_q^{\ell \times n}$ as in G'_8 . Moreover, since j^* is not allowed to be corrupted by \mathcal{A} , it is needless to keep $\mathbf{K}^{(j')}$ as long as $j' = j^*$.

Since the challenger will guess j^* correctly with probability $1/n$, we have $|\Pr'_7[\text{Win}] - \frac{1}{2}| = \frac{1}{n} |\Pr'_8[\text{Win}] - \frac{1}{2}|$.

Game G'_9 : It is the same as G'_8 , except for the generation of ct_2^* in the case of $\beta = 1$. In this game, in the case of $\beta = 1$, the challenger also picks the second-level challenge ciphertext $ct_2^* \leftarrow_{\mathcal{S}} \mathbb{Z}_q^{N+\ell}$ uniformly at random.

Now in G'_9 , both ct_1^* and ct_2^* are independently and uniformly chosen from $\mathbb{Z}_q^{N+\ell}$, regardless of the value of β . Thus, the challenge bit β is completely hidden to \mathcal{A} , and we have $\Pr'_9[\text{Win}] = \frac{1}{2}$. Next we show that G'_8 and G'_9 are computationally indistinguishable.

Recall that in G'_8 , ct_2^* is generated by invoking $ct_2^* \leftarrow_{\mathcal{S}} \text{Enc}_1(pk^{(j^*)}, \mathbf{m}_2)$, i.e., $ct_2^* := \mathbf{A}^{(j^*)}\mathbf{s} + \mathbf{e} + \begin{pmatrix} \mathbf{0} \\ p\mathbf{m}_2 \end{pmatrix}$ with $\mathbf{s} \leftarrow_{\mathcal{S}} \chi^n$ and $\mathbf{e} \leftarrow_{\mathcal{S}} \chi^{N+\ell}$. Due to the game change introduced in G'_8 , we have that $j' = j^*$ and $\mathbf{A}^{(j^*)} = \begin{pmatrix} \overline{\mathbf{A}}' \\ \underline{\mathbf{A}}'^{(j^*)} \end{pmatrix} = \begin{pmatrix} \overline{\mathbf{A}}' \\ \underline{\mathbf{A}}'^{(j')} \end{pmatrix}$ is uniformly sampled from $\mathbb{Z}_q^{(N+\ell) \times n}$. Then according to the LWE assumption,

$\mathbf{A}'^{(j^*)}\mathbf{s} + \mathbf{e}$ is computationally indistinguishable from the uniform distribution over $\mathbb{Z}_q^{N+\ell}$. Thus the challenge ciphertext $ct_2^* := \mathbf{A}'^{(j^*)}\mathbf{s} + \mathbf{e} + \begin{pmatrix} 0 \\ p\mathbf{m}_2 \end{pmatrix}$ generated in \mathbf{G}'_8 in the case of $\beta = 1$ is also computationally indistinguishable from the uniformly chosen $ct_2^* \leftarrow_s \mathbb{Z}_q^{N+\ell}$ in \mathbf{G}'_9 in the case of $\beta = 1$. Thus we have $|\Pr'_8[\text{Win}] - \Pr'_9[\text{Win}]| \leq \text{Adv}_{[n,q,\chi,N+\ell],\mathcal{B}_3}^{\text{LWE}}(\lambda)$.

Finally, taking all things together, Theorem 2 follows. \square

Theorem 3 (Collusion-Safety of $\text{FPRE}_{\text{LWE}}^{\text{lin}}$). *Assume that the $\text{LWE}_{n,q,\chi,N+\ell}$ -assumption holds, then the scheme $\text{FPRE}_{\text{LWE}}^{\text{lin}}$ proposed in Fig. 7 is collusion-safe (CS). More precisely, for any PPT adversary \mathcal{A} and any polynomial \mathbf{n} , there exists a PPT algorithm \mathcal{B} against the LWE assumption such that $\text{Adv}_{\text{FPRE},\mathcal{A},\mathbf{n}}^{\text{CS}}(\lambda) \leq \mathbf{n} \cdot \text{Adv}_{[n,q,\chi,N+\ell],\mathcal{B}}^{\text{LWE}}(\lambda) + \text{negl}(\lambda)$.*

Due to space limitations, we postpone the proof of Theorem 3 to Appendix C.

5 Fine-Grained PRE for Deletion Functions for LWE

In this section, we construct an FPREScheme $\text{FPRE}_{\text{LWE}}^{\text{del}}$ for deletion function family \mathcal{F}_{del} based on the scheme $\text{FPRE}_{\text{LWE}}^{\text{lin}}$ proposed in Sect. 4 for the bounded linear function family \mathcal{F}_{lin} .

Deletion function family. Let $\ell \in \mathbb{N}$ and $\mathcal{M} := \{0, 1, *\}^\ell$ be the message space, where “*” is a special symbol indicating that this bit is invalid or deleted. Given a subset $\mathcal{P} \subseteq [\ell]$, the *deletion function* $f_{\mathcal{P}} : \mathcal{M} \rightarrow \mathcal{M}$ indexed by \mathcal{P} is

$$f_{\mathcal{P}}(m_1, \dots, m_\ell) := (m'_1, \dots, m'_\ell), \text{ where } m'_i := \begin{cases} *, & \text{if } i \in \mathcal{P}; \\ m_i, & \text{if } i \notin \mathcal{P}. \end{cases}$$

That is, $f_{\mathcal{P}}$ will delete the message bits whose indices are contained in the set \mathcal{P} by setting them as the invalid symbol *.

Then we define the family of deletion functions \mathcal{F}_{del} from \mathcal{M} to \mathcal{M} as

$$\mathcal{F}_{\text{del}} = \left\{ f_{\mathcal{P}} : \{0, 1, *\}^\ell \rightarrow \{0, 1, *\}^\ell \mid \mathcal{P} \subseteq [\ell] \right\}.$$

Message encoding and expressing deletion functions in \mathcal{F}_{del} as bounded linear functions in \mathcal{F}_{lin} . In order to construct an FPREScheme $\text{FPRE}_{\text{LWE}}^{\text{del}}$ with message space $\mathcal{M} = \{0, 1, *\}^\ell$ for deletion function family \mathcal{F}_{del} based on the scheme $\text{FPRE}_{\text{LWE}}^{\text{lin}}$ in Fig. 7 for the bounded linear function family \mathcal{F}_{lin} , we will show

- how to encode a message $\mathbf{m} \in \mathcal{M} = \{0, 1, *\}^\ell$ to a message $\tilde{\mathbf{m}} \in \tilde{\mathcal{M}} = \{0, 1\}^{2\ell} \subseteq \mathbb{Z}_p^{2\ell}$ with an encoding algorithm Encode (and also how to decode with an decoding algorithm Decode) so that the symbol “*” of erasure can be encoded by binary bits and later be correctly decoded, and
- how to express a deletion function $f_{\mathcal{P}} \in \mathcal{F}_{\text{del}}$ as a bounded linear function $f_{\mathbf{M}} \in \mathcal{F}_{\text{lin}}$ with a converting function $\Psi : \mathcal{F}_{\text{del}} \rightarrow \mathcal{F}_{\text{lin}}$.

The algorithms Encode and Decode and the converting function Ψ should be *compatible* in the sense that for any $\mathbf{m} \in \mathcal{M} = \{0, 1, *\}^\ell$ and any $f_{\mathcal{P}} \in \mathcal{F}_{\text{del}}$, by setting $f_{\mathbf{M}} := \Psi(f_{\mathcal{P}})$, we have

$$f_{\mathcal{P}}(\mathbf{m}) = \text{Decode}(f_{\mathbf{M}}(\text{Encode}(\mathbf{m}))). \quad (7)$$

The encoding algorithm $\text{Encode} : \{0, 1, *\}^\ell \rightarrow \{0, 1\}^{2\ell}$ and the corresponding decoding algorithm $\text{Decode} : \{0, 1\}^{2\ell} \rightarrow \{0, 1, *\}^\ell$ are defined as follows.

- $\tilde{\mathbf{m}} \in \{0, 1\}^{2\ell} \leftarrow \text{Encode}(\mathbf{m} \in \{0, 1, *\}^\ell)$: Parse $\mathbf{m} = (m_1, \dots, m_\ell)$. For $i \in [\ell]$, set $\tilde{m}_{2i-1}\tilde{m}_{2i} = 00$ if $m_i = *$, set $\tilde{m}_{2i-1}\tilde{m}_{2i} = 01$ if $m_i = 0$, and set $\tilde{m}_{2i-1}\tilde{m}_{2i} = 10$ if $m_i = 1$. Return $\tilde{\mathbf{m}} := (\tilde{m}_1, \dots, \tilde{m}_{2\ell})$.
- $\mathbf{m} \in \{0, 1, *\}^\ell \leftarrow \text{Decode}(\tilde{\mathbf{m}} \in \{0, 1\}^{2\ell})$: Parse $\tilde{\mathbf{m}} = (\tilde{m}_1, \dots, \tilde{m}_{2\ell})$. For $i \in [\ell]$, set $m_i := *$ if $\tilde{m}_{2i-1}\tilde{m}_{2i} = 00$, set $m_i := 0$ if $\tilde{m}_{2i-1}\tilde{m}_{2i} = 01$, and set $m_i := 1$ if $\tilde{m}_{2i-1}\tilde{m}_{2i} = 10$. Return $\mathbf{m} := (m_1, \dots, m_\ell)$.

The converting function $\Psi : \mathcal{F}_{\text{del}} \rightarrow \mathcal{F}_{\text{lin}}$ is defined as follows. On input a deletion function $f_{\mathcal{P}}$ with $\mathcal{P} \subseteq \{0, 1\}^\ell$, $\Psi(f_{\mathcal{P}}) := f_{\mathbf{M}}$, where $\mathbf{M} = (M_{i,j}) \in \{0, 1\}^{2\ell \times 2\ell}$

$$\text{with } \begin{cases} M_{2i-1, 2i-1} = M_{2i, 2i} = 0 & \text{if } i \in \mathcal{P}, i \in [\ell]; \\ M_{2i-1, 2i-1} = M_{2i, 2i} = 1 & \text{if } i \notin \mathcal{P}, i \in [\ell]; \\ M_{i,j} = 0 & \text{if } i \neq j, i, j \in [2\ell]. \end{cases}$$

It is routine to check that the $(\text{Encode}, \text{Decode}, \Psi)$ defined above are compatible, i.e., satisfying (7). More precisely, for any $\mathbf{m} = (m_1, \dots, m_\ell) \in \{0, 1, *\}^\ell$ and any $f_{\mathcal{P}} \in \mathcal{F}_{\text{del}}$, let $\tilde{\mathbf{m}} = (\tilde{m}_1, \dots, \tilde{m}_{2\ell}) := \text{Encode}(\mathbf{m})$ and $f_{\mathbf{M}} := \Psi(f_{\mathcal{P}})$, we have $f_{\mathbf{M}}(\text{Encode}(\mathbf{m})) = \mathbf{M}\tilde{\mathbf{m}}$. Let $\tilde{\mathbf{m}}' = (\tilde{m}'_1, \tilde{m}'_2, \dots, \tilde{m}'_{2\ell}) := \mathbf{M}\tilde{\mathbf{m}}$. We will show that $\text{Decode}(\tilde{\mathbf{m}}')$ successfully recovers $f_{\mathcal{P}}(\mathbf{m})$, i.e., $\text{Decode}(\tilde{m}'_{2i-1}\tilde{m}'_{2i}) = *$ for all $i \in \mathcal{P}$ and $\text{Decode}(\tilde{m}'_{2i-1}\tilde{m}'_{2i}) = m_i$ for all $i \in [\ell] \setminus \mathcal{P}$.

- If $i \in \mathcal{P}$, we have $\tilde{m}'_{2i-1}\tilde{m}'_{2i} = 00$, since $M_{2i-1, j} = M_{2i, j} = 0$ for all $j \in [2\ell]$. Then Decode will result in $m_i = *$, and thus the i -th bit of \mathbf{m} is deleted.
- If $i \in [\ell] \setminus \mathcal{P}$, we have $\tilde{m}'_{2i-1}\tilde{m}'_{2i} = \tilde{m}_{2i-1}\tilde{m}_{2i}$, since $M_{2i-1, 2i-1} = M_{2i, 2i} = 1$, $M_{2i-1, j} = 0$ for all $j \in [2\ell] \setminus \{2i-1\}$ and $M_{2i, j} = 0$ for all $j \in [2\ell] \setminus \{2i\}$. Then Decode keeps m_i unchanged.

Therefore, we have $\text{Decode}(\tilde{\mathbf{m}}') = f_{\mathcal{P}}(\mathbf{m})$ and (7) follows.

Constructing FPRE scheme $\text{FPRE}_{\text{LWE}}^{\text{del}}$ for \mathcal{F}_{del} from $\text{FPRE}_{\text{LWE}}^{\text{lin}}$ for \mathcal{F}_{lin} . Let $\text{FPRE}_{\text{LWE}}^{\text{lin}} = (\text{Setup}, \text{KGen}, \text{FReKGen}, \text{Enc}_1, \text{Enc}_2, \text{FReEnc}, \text{Dec}_1, \text{Dec}_2)$ be the FPRE scheme for the bounded linear function family \mathcal{F}_{lin} as described in Fig. 7 in Sect. 4 with message space $\tilde{\mathbf{m}} \in \tilde{\mathcal{M}} = \{0, 1\}^{2\ell}$, and let $(\text{Encode}, \text{Decode}, \Psi)$ be the encoding algorithm, decoding algorithms and the converting function defined above. We are ready to present the FPRE scheme $\text{FPRE}_{\text{LWE}}^{\text{del}}$ for the deletion function family $\mathcal{F}_{\text{del}} := \{f_{\mathcal{P}} : \{0, 1, *\}^\ell \rightarrow \{0, 1, *\}^\ell \mid \mathcal{P} \subseteq \{0, 1\}^\ell\}$ with message space $\mathcal{M} = \{0, 1, *\}^\ell$. The scheme $\text{FPRE}_{\text{LWE}}^{\text{del}} = (\text{Setup}', \text{KGen}', \text{FReKGen}', \text{Enc}'_1, \text{Enc}'_2, \text{FReEnc}', \text{Dec}'_1, \text{Dec}'_2)$ is described as follows. For the ease of reading, we emphasize the parts related to $(\text{Encode}, \text{Decode}, \Psi)$ in gray boxes.

- $\text{pp} \leftarrow_{\$} \text{Setup}'$: It invokes $\text{pp} \leftarrow_{\$} \text{Setup}$ and returns pp .
- $(pk, sk) \leftarrow_{\$} \text{KGen}'(\text{pp})$: It invokes $(pk, sk) \leftarrow_{\$} \text{KGen}(\text{pp})$ and returns (pk, sk) .
- $rk_{i \rightarrow j}^{f_{\mathcal{P}}} \leftarrow_{\$} \text{FReKGen}'(pk^{(i)}, sk^{(i)}, pk^{(j)}, f_{\mathcal{P}})$: It first computes $f_{\mathcal{M}} := \Psi(f_{\mathcal{P}})$, invokes $rk_{i \rightarrow j}^{f_{\mathcal{M}}} \leftarrow_{\$} \text{FReKGen}(pk^{(i)}, sk^{(i)}, pk^{(j)}, f_{\mathcal{M}})$, and returns $rk_{i \rightarrow j}^{f_{\mathcal{P}}} := rk_{i \rightarrow j}^{f_{\mathcal{M}}}$.
- $ct_1 \leftarrow_{\$} \text{Enc}'_1(pk, \mathbf{m} \in \{0, 1, *\}^{\ell})$: It first encodes $\tilde{\mathbf{m}} \leftarrow \text{Encode}(\mathbf{m})$, then invokes $ct_1 \leftarrow_{\$} \text{Enc}_1(pk, \tilde{\mathbf{m}})$, and returns ct_1 .
- $ct_2 \leftarrow_{\$} \text{Enc}'_2(pk, \mathbf{m} \in \{0, 1, *\}^{\ell})$: It first encodes $\tilde{\mathbf{m}} \leftarrow \text{Encode}(\mathbf{m})$, then invokes $ct_2 \leftarrow_{\$} \text{Enc}_2(pk, \tilde{\mathbf{m}})$, and returns ct_2 .
- $ct_2^{(j)} \leftarrow_{\$} \text{FReEnc}'(rk_{i \rightarrow j}^{f_{\mathcal{P}}}, ct_1^{(i)})$: It invokes $ct_2^{(j)} \leftarrow_{\$} \text{FReEnc}(rk_{i \rightarrow j}^{f_{\mathcal{P}}}, ct_1^{(i)})$ and returns $ct_2^{(j)}$.
- $\mathbf{m} \leftarrow \text{Dec}'_1(sk, ct_1)$: It invokes $\tilde{\mathbf{m}} \leftarrow \text{Dec}_1(sk, ct_1)$, then decodes $\mathbf{m} \leftarrow \text{Decode}(\tilde{\mathbf{m}})$, and returns \mathbf{m} .
- $\mathbf{m} \leftarrow \text{Dec}'_2(sk, ct_2)$: It invokes $\tilde{\mathbf{m}} \leftarrow \text{Dec}_2(sk, ct_2)$, then decodes $\mathbf{m} \leftarrow \text{Decode}(\tilde{\mathbf{m}})$, and returns \mathbf{m} .

We also present a full description of $\text{FPRE}_{\text{LWE}}^{\text{del}}$ in Appendix D for completeness.

Correctness and fine-grained one-hop correctness of $\text{FPRE}_{\text{LWE}}^{\text{del}}$ follow from those of $\text{FPRE}_{\text{LWE}}^{\text{lin}}$ and the compatibility of $(\text{Encode}, \text{Decode}, \Psi)$, i.e., (7).

Remark 4 (Further optimization of $\text{FPRE}_{\text{LWE}}^{\text{del}}$). Note that in our construction of $\text{FPRE}_{\text{LWE}}^{\text{del}}$, we only require the underlying $\text{FPRE}_{\text{LWE}}^{\text{lin}}$ to work with a message space of $\tilde{\mathcal{M}} = \{0, 1\}^{2\ell}$ (rather than $\mathbb{Z}_p^{2\ell}$). This enables us to optimize the $\text{Enc}_1, \text{Enc}_2$ and $\text{Dec}_1, \text{Dec}_2$ algorithms as follows. In $ct_1 := \mathbf{A}\mathbf{s} + \mathbf{e} + \binom{\mathbf{0}}{p\mathbf{m}}$ and $ct_2 := \mathbf{A}'\mathbf{s} + \mathbf{e} + \binom{\mathbf{0}}{p\mathbf{m}}$, the multiplication factor p can be replaced by $\lfloor q/2 \rfloor$, i.e., $ct_1 := \mathbf{A}\mathbf{s} + \mathbf{e} + \binom{\mathbf{0}}{\lfloor q/2 \rfloor \mathbf{m}}$ and $ct_2 := \mathbf{A}'\mathbf{s} + \mathbf{e} + \binom{\mathbf{0}}{\lfloor q/2 \rfloor \mathbf{m}}$. Correspondingly, the decryption algorithms output 0 or 1 depending on the intermediate result is close to 0 or $q/2$. In this way, the parameters q can be much smaller. For example, in the parameter setting in Table 2, q can be set as $q = 2\lambda^5$ instead of $q = \lambda^{10}$.

Acknowledgments. We would like to thank the reviewers for their valuable comments. Yunxiao Zhou, Shengli Liu and Shuai Han were partially supported by the National Key R&D Program of China under Grant 2022YFB2701500, National Natural Science Foundation of China (Grant Nos. 61925207, 62372292, 62002223), Guangdong Major Project of Basic and Applied Basic Research (2019B030302008), and Young Elite Scientists Sponsorship Program by China Association for Science and Technology (YESS20200185). Haibin Zhang was partially supported by the National Key R&D Program of China under Grant 2022YFB2701500, the National Natural Science Foundation of China under 62272043, Major Program of Shandong Provincial Natural Science Foundation for the Fundamental Research under ZR2022ZD03.

References

- [1] Ajtai, M.: Generating hard instances of lattice problems (extended abstract). In: 28th ACM STOC. pp. 99–108. ACM Press (May 1996)
- [2] Applebaum, B., Cash, D., Peikert, C., Sahai, A.: Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 595–618. Springer, Heidelberg (Aug 2009)
- [3] Ateniese, G., Fu, K., Green, M., Hohenberger, S.: Improved proxy re-encryption schemes with applications to secure distributed storage. In: NDSS 2005. The Internet Society (Feb 2005)
- [4] Blaze, M., Bleumer, G., Strauss, M.: Divertible protocols and atomic proxy cryptography. In: Nyberg, K. (ed.) EUROCRYPT’98. LNCS, vol. 1403, pp. 127–144. Springer, Heidelberg (May / Jun 1998)
- [5] Canard, S., Devigne, J., Laguillaumie, F.: Improving the security of an efficient unidirectional proxy re-encryption scheme. *J. Internet Serv. Inf. Secur.* 1(2/3), 140–160 (2011), <https://doi.org/10.22667/JISIS.2011.08.31.140>
- [6] Canetti, R., Hohenberger, S.: Chosen-ciphertext secure proxy re-encryption. In: Ning, P., De Capitani di Vimercati, S., Syverson, P.F. (eds.) ACM CCS 2007. pp. 185–194. ACM Press (Oct 2007)
- [7] Chandran, N., Chase, M., Liu, F.H., Nishimaki, R., Xagawa, K.: Re-encryption, functional re-encryption, and multi-hop re-encryption: A framework for achieving obfuscation-based security and instantiations from lattices. In: Krawczyk, H. (ed.) PKC 2014. LNCS, vol. 8383, pp. 95–112. Springer, Heidelberg (Mar 2014)
- [8] Chandran, N., Chase, M., Vaikuntanathan, V.: Functional re-encryption and collusion-resistant obfuscation. In: Cramer, R. (ed.) TCC 2012. LNCS, vol. 7194, pp. 404–421. Springer, Heidelberg (Mar 2012)
- [9] Chow, S.S.M., Weng, J., Yang, Y., Deng, R.H.: Efficient unidirectional proxy re-encryption. In: Bernstein, D.J., Lange, T. (eds.) AFRICACRYPT 10. LNCS, vol. 6055, pp. 316–332. Springer, Heidelberg (May 2010)
- [10] Cohen, A.: What about bob? The inadequacy of CPA security for proxy re-encryption. In: Lin, D., Sako, K. (eds.) PKC 2019, Part II. LNCS, vol. 11443, pp. 287–316. Springer, Heidelberg (Apr 2019)
- [11] Fan, X., Liu, F.H.: Proxy re-encryption and re-signatures from lattices. In: Deng, R.H., Gauthier-Umaña, V., Ochoa, M., Yung, M. (eds.) ACNS 19. LNCS, vol. 11464, pp. 363–382. Springer, Heidelberg (Jun 2019)
- [12] Fuchsbauer, G., Kamath, C., Klein, K., Pietrzak, K.: Adaptively secure proxy re-encryption. In: Lin, D., Sako, K. (eds.) PKC 2019, Part II. LNCS, vol. 11443, pp. 317–346. Springer, Heidelberg (Apr 2019)
- [13] Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: Ladner, R.E., Dwork, C. (eds.) 40th ACM STOC. pp. 197–206. ACM Press (May 2008)
- [14] Kirshanova, E.: Proxy re-encryption from lattices. In: Krawczyk, H. (ed.) PKC 2014. LNCS, vol. 8383, pp. 77–94. Springer, Heidelberg (Mar 2014)
- [15] Liang, X., Weng, J., Yang, A., Yao, L., Jiang, Z., Wu, Z.: Attribute-based conditional proxy re-encryption in the standard model under LWE. In: Bertino, E., Shulman, H., Waidner, M. (eds.) ESORICS 2021, Part II. LNCS, vol. 12973, pp. 147–168. Springer, Heidelberg (Oct 2021)
- [16] Libert, B., Vergnaud, D.: Unidirectional chosen-ciphertext secure proxy re-encryption. In: Cramer, R. (ed.) PKC 2008. LNCS, vol. 4939, pp. 360–379. Springer, Heidelberg (Mar 2008)

- [17] Miao, P., Patrnanabis, S., Watson, G.J.: Unidirectional updatable encryption and proxy re-encryption from DDH. In: Boldyreva, A., Kolesnikov, V. (eds.) *Public-Key Cryptography - PKC 2023 - 26th IACR International Conference on Practice and Theory of Public-Key Cryptography*, Atlanta, GA, USA, May 7-10, 2023, Proceedings, Part II. *Lecture Notes in Computer Science*, vol. 13941, pp. 368–398. Springer (2023), https://doi.org/10.1007/978-3-031-31371-4_13
- [18] Micciancio, D., Mol, P.: Pseudorandom knapsacks and the sample complexity of LWE search-to-decision reductions. In: Rogaway, P. (ed.) *CRYPTO 2011*. LNCS, vol. 6841, pp. 465–484. Springer, Heidelberg (Aug 2011)
- [19] Micciancio, D., Peikert, C.: Trapdoors for lattices: Simpler, tighter, faster, smaller. In: Pointcheval, D., Johansson, T. (eds.) *EUROCRYPT 2012*. LNCS, vol. 7237, pp. 700–718. Springer, Heidelberg (Apr 2012)
- [20] Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: Gabow, H.N., Fagin, R. (eds.) *37th ACM STOC*. pp. 84–93. ACM Press (May 2005)
- [21] Selvi, S.S.D., Paul, A., Rangan, C.P.: A provably-secure unidirectional proxy re-encryption scheme without pairing in the random oracle model. In: Capkun, S., Chow, S.S.M. (eds.) *CANS 17*. LNCS, vol. 11261, pp. 459–469. Springer, Heidelberg (Nov / Dec 2017)
- [22] Shao, J.: Anonymous ID-based proxy re-encryption. In: Susilo, W., Mu, Y., Seberry, J. (eds.) *ACISP 12*. LNCS, vol. 7372, pp. 364–375. Springer, Heidelberg (Jul 2012)
- [23] Shao, J., Cao, Z.: CCA-secure proxy re-encryption without pairings. In: Jarecki, S., Tsudik, G. (eds.) *PKC 2009*. LNCS, vol. 5443, pp. 357–376. Springer, Heidelberg (Mar 2009)
- [24] Susilo, W., Dutta, P., Duong, D.H., Roy, P.S.: Lattice-based HRA-secure attribute-based proxy re-encryption in standard model. In: Bertino, E., Shulman, H., Waidner, M. (eds.) *ESORICS 2021, Part II*. LNCS, vol. 12973, pp. 169–191. Springer, Heidelberg (Oct 2021)
- [25] Weng, J., Deng, R.H., Ding, X., Chu, C.K., Lai, J.: Conditional proxy re-encryption secure against chosen-ciphertext attack. In: Li, W., Susilo, W., Tupakula, U.K., Safavi-Naini, R., Varadharajan, V. (eds.) *ASIACCS 09*. pp. 322–332. ACM Press (Mar 2009)

Supplementary Material

A Additional Preliminaries

A.1 Lattice Backgrounds

Definition 7 (Discrete Gaussian Distribution). *The Gaussian function with parameter s and center $\mathbf{c} \in \mathbb{R}^n$ is defined as $\rho_{s,\mathbf{c}} : \mathbb{R}^n \rightarrow \mathbb{R}$, $\rho_{s,\mathbf{c}}(\mathbf{x}) := e^{-\pi\|\mathbf{x}-\mathbf{c}\|^2/s^2}$. For a countable set $\mathcal{S} \subset \mathbb{R}^n$, the discrete Gaussian distribution $D_{\mathcal{S},s,\mathbf{c}}$ parameterized with s and \mathbf{c} is defined as $D_{\mathcal{S},s,\mathbf{c}}(\mathbf{x}) := \rho_{s,\mathbf{c}}(\mathbf{x}) / \sum_{\mathbf{x} \in \mathcal{S}} \rho_{s,\mathbf{c}}(\mathbf{x})$ for $\mathbf{x} \in \mathcal{S}$ and $D_{\mathcal{S},s,\mathbf{c}}(\mathbf{x}) := 0$ for $\mathbf{x} \notin \mathcal{S}$. Usually, s is omitted when $s = 1$ and \mathbf{c} is omitted if $\mathbf{c} = \mathbf{0}$.*

Below we recall the LWE and multi-secret LWE assumptions, where both the secret vector and the error vector are sampled from the same distribution (say χ). This version of LWE was formalized by Applebaum et al. [2] and was proved at least as hard as the usual definition of LWE where the secret vector is sampled uniformly at random.

Definition 8 (LWE Assumption [20, 2]). *Let $n, m, q \in \mathbb{N}$ and χ be a distribution over \mathbb{Z}_q . The $\text{LWE}_{n,q,\chi,m}$ -assumption requires that for any PPT adversary \mathcal{A} , its advantage function satisfies $\text{Adv}_{[n,q,\chi,m],\mathcal{A}}^{\text{LWE}}(\lambda) := |\Pr[\mathcal{A}(\mathbf{A}, \mathbf{A}\mathbf{s} + \mathbf{e}) = 1] - \Pr[\mathcal{A}(\mathbf{A}, \mathbf{u}) = 1]| \leq \text{negl}(\lambda)$, where $\mathbf{A} \leftarrow_s \mathbb{Z}_q^{m \times n}$, $\mathbf{s} \leftarrow_s \chi^n$, $\mathbf{e} \leftarrow_s \chi^m$, $\mathbf{u} \leftarrow_s \mathbb{Z}_q^m$.*

For $Q \in \mathbb{N}$, the $Q\text{-LWE}_{n,m,q,\chi}$ -assumption requires that for any PPT \mathcal{A} , its advantage satisfies $\text{Adv}_{[n,q,\chi,m],\mathcal{A}}^{Q\text{-LWE}}(\lambda) := |\Pr[\mathcal{A}(\mathbf{A}, \mathbf{A}\mathbf{S} + \mathbf{E}) = 1] - \Pr[\mathcal{A}(\mathbf{A}, \mathbf{U}) = 1]| \leq \text{negl}(\lambda)$, where $\mathbf{A} \leftarrow_s \mathbb{Z}_q^{m \times n}$, $\mathbf{S} \leftarrow_s \chi^{n \times Q}$, $\mathbf{E} \leftarrow_s \chi^{m \times Q}$ and $\mathbf{U} \leftarrow_s \mathbb{Z}_q^{m \times Q}$.

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

In [1, 19], an algorithm named `TrapGen` is proposed to sample a “nearly” uniform random matrix \mathbf{A} along with a low-norm trapdoor matrix $\mathbf{T}_{\mathbf{A}}$ such that $\mathbf{T}_{\mathbf{A}} \cdot \mathbf{A} = \mathbf{0}$ (cf. Lemma 2). Meanwhile, another algorithm called `Invert` is proposed to make use of $\mathbf{T}_{\mathbf{A}}$ to invert an LWE sample $(\mathbf{A}, \mathbf{A}\mathbf{s} + \mathbf{e})$ to obtain \mathbf{s} and \mathbf{e} (cf. Lemma 3).

Lemma 2 ([1, 19]). *There exists a PPT algorithm `TrapGen` that takes as input positive integers n, q ($q \geq 2$) and a sufficiently large $m = O(n \log q)$, outputs a matrix $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$ and a trapdoor matrix $\mathbf{T}_{\mathbf{A}} \in \mathbb{Z}_q^{m \times m}$ such that \mathbf{A} is statistically close to the uniform distribution, $\mathbf{T}_{\mathbf{A}} \cdot \mathbf{A} = \mathbf{0}$, and $\|\tilde{\mathbf{T}}_{\mathbf{A}}\| \leq O(\sqrt{n \log q})$, where $\tilde{\mathbf{T}}_{\mathbf{A}}$ denotes the Gram-Schmidt orthogonalization of $\mathbf{T}_{\mathbf{A}}$.*

Lemma 3 ([19, Theorem 5.4]). *There exists a deterministic polynomial-time algorithm `Invert` that takes as inputs the trapdoor information $\mathbf{T}_{\mathbf{A}}$ and a vector $\mathbf{v} := \mathbf{A} \cdot \mathbf{s} + \mathbf{e}$ with $\mathbf{s} \in \mathbb{Z}_q^n$ and $\|\mathbf{e}\| \leq q/(10\sqrt{m})$, and outputs \mathbf{s} and \mathbf{e} .*

Lemma 4 ([13]). *Let $n, m, q \in \mathbb{N}$ with $q \geq 2$, and $\gamma \geq O(\sqrt{n \log q}) \cdot \omega(\sqrt{\log n})$.*

- **Preimage-sampling.** Let $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$ be a matrix with a trapdoor $\mathbf{T}_\mathbf{A}$. Let $\mathbf{B} \in \mathbb{Z}_q^{m' \times n}$. There exists a PPT algorithm $\text{SamplePre}(\mathbf{T}_\mathbf{A}, \mathbf{A}, \mathbf{B}, \gamma)$ that outputs a matrix $\mathbf{R} \in \mathbb{Z}_q^{m' \times m}$ which is sampled from a distribution statistically close to $D_{\Lambda_q^\mathbf{B}(\mathbf{A}), \gamma}$ and satisfies $\mathbf{R} \cdot \mathbf{A} = \mathbf{B}$ and $\|\mathbf{R}\|_\infty \leq \gamma \cdot \omega(\log n)$ (except with a negligible probability).
- **Indistinguishability of preimage-sampling.** Let TrapGen be the algorithm defined in Lemma 2. Let $m \geq O(n \log q)$. Then we have $(\mathbf{A}, \mathbf{R}, \mathbf{B}) \approx_s (\mathbf{A}, \mathbf{R}', \mathbf{B}')$, where the probability is over $(\mathbf{A}, \mathbf{T}_\mathbf{A}) \leftarrow_s \text{TrapGen}(n, q, m)$, $\mathbf{B} \leftarrow_s \mathbb{Z}_q^{m' \times n}$, $\mathbf{R} \leftarrow_s \text{SamplePre}(\mathbf{T}_\mathbf{A}, \mathbf{A}, \mathbf{B}, \gamma)$, $\mathbf{R}' \leftarrow_s D_{\mathbb{Z}_q^{m' \times m}, \gamma}$, and $\mathbf{B}' := \mathbf{R}' \cdot \mathbf{A}$.

Lemma 5 (Randomness Extraction, Particular case of [18, Lemma 2.3]). Let $n, m, q \in \mathbb{N}$, $\epsilon \in (0, 1)$. Suppose that \mathbf{r} is chosen from some distribution over \mathbb{Z}_q^m s.t. for q 's prime factor p it holds that $\mathbf{H}_\infty(\mathbf{r} \bmod p) \geq 2n \log q + 2 \log(\frac{1}{\epsilon})$. Then for $\mathbf{A} \leftarrow_s \mathbb{Z}_q^{m \times n}$, $\mathbf{u} \leftarrow_s \mathbb{Z}_q^n$, we have $\Delta((\mathbf{A}, \mathbf{r}^\top \cdot \mathbf{A}), (\mathbf{A}, \mathbf{u}^\top)) \leq \epsilon$.

A.2 Proxy Re-Encryption

We recall the syntax of proxy re-encryption according to [3, 5, 9, 16, 21, 22, 23].

Definition 9 (PRE). A proxy re-encryption (PRE) scheme $\text{PRE} = (\text{Setup}, \text{KGen}, \text{ReKGen}, \text{Enc}_1, \text{Enc}_2, \text{ReEnc}, \text{Dec}_1, \text{Dec}_2)$ with message space \mathcal{M} consists of eight PPT algorithms:

- $\text{pp} \leftarrow_s \text{Setup}$: The setup algorithm outputs a public parameter pp , which serves as an implicit input of other algorithms.
- $(pk, sk) \leftarrow_s \text{KGen}(\text{pp})$: Taking pp as input, the key generation algorithm outputs a pair of public key and secret key (pk, sk) .
- $rk_{i \rightarrow j} \leftarrow_s \text{ReKGen}(pk^{(i)}, sk^{(i)}, pk^{(j)})$: Taking as input a public-secret key pair $(pk^{(i)}, sk^{(i)})$ and another public key $pk^{(j)}$, the re-encryption key generation algorithm outputs a re-encryption key $rk_{i \rightarrow j}$ that allows re-encrypting ciphertexts intended to i into ciphertexts encrypted for j .
- $ct_1 \leftarrow_s \text{Enc}_1(pk, m)$: Taking as input a public key pk and a message $m \in \mathcal{M}$, this algorithm outputs a first-level ciphertext ct_1 that can be further re-encrypted into a second-level ciphertext.
- $ct_2 \leftarrow_s \text{Enc}_2(pk, m)$: Taking as input a public key pk and a message $m \in \mathcal{M}$, this algorithm outputs a second-level ciphertext ct_2 that cannot be re-encrypted anymore.
- $ct_2^{(j)} \leftarrow_s \text{ReEnc}(rk_{i \rightarrow j}, ct_1^{(i)})$: Taking as input a re-encryption key $rk_{i \rightarrow j}$ and a first-level ciphertext intended for i , the re-encryption algorithm outputs a second-level ciphertext re-encrypted for user j .
- $m \leftarrow \text{Dec}_1(sk, ct_1)$: Taking as input a secret key sk and a first-level ciphertext ct_1 , the deterministic decryption algorithm outputs a message m .
- $m \leftarrow \text{Dec}_2(sk, ct_2)$: Taking as input a secret key sk and a second-level ciphertext ct_2 , the deterministic decryption algorithm outputs a message m .

Correctness. For all $m \in \mathcal{M}$, $\text{pp} \leftarrow_s \text{Setup}$, $(pk, sk) \leftarrow_s \text{KGen}(\text{pp})$, $ct_1 \leftarrow_s \text{Enc}_1(pk, m)$ and $ct_2 \leftarrow_s \text{Enc}_2(pk, m)$, it holds that $\text{Dec}_1(sk, ct_1) = m = \text{Dec}_2(sk, ct_2)$.

One-Hop Correctness. For all $m \in \mathcal{M}$, $\text{pp} \leftarrow_s \text{Setup}$, $(pk^{(i)}, sk^{(i)}) \leftarrow_s \text{KGen}(\text{pp})$, $(pk^{(j)}, sk^{(j)}) \leftarrow_s \text{KGen}(\text{pp})$, $rk_{i \rightarrow j} \leftarrow_s \text{ReKGen}(pk^{(i)}, sk^{(i)}, pk^{(j)})$, $ct_1^{(i)} \leftarrow_s \text{Enc}_1(pk^{(i)}, m)$ and $ct_2^{(j)} \leftarrow_s \text{ReEnc}(rk_{i \rightarrow j}, ct_1^{(i)})$, it holds that $\text{Dec}_2(sk^{(j)}, ct_2^{(j)}) = m$.

Note that the above PRE is defined as a *non-interactive* one, since $sk^{(j)}$ is not needed in algorithm ReKGen for the generation of $rk_{i \rightarrow j}$.

B More Discussions on Security Definitions for FPFE in Sect. 3

B.1 More Discussions on UNID Security (Def. 3) and Its Relation to CPA Security

Remark 5 (On the formalization of UNID security and discussion on trivial attacks). We formalize the UNID security by defining the experiment $\text{Exp}_{\text{FPFE}, \mathcal{A}, n}^{\text{UNID}}$ in Fig. 3. Similar to previous security notions, we consider a multi-user setting, and the adversary \mathcal{A} is allowed to make $\mathcal{O}_{\text{REKEY}}$ and \mathcal{O}_{COR} queries *adaptively* to obtain re-encryption keys and secret keys, respectively. At some point, \mathcal{A} outputs a pair of challenge users (i^*, j^*) as well as a function $f \in \mathcal{F}$, and receives a fine-grained re-encryption key $rk_{j^* \rightarrow i^*}^f$. \mathcal{A} continues to make $\mathcal{O}_{\text{REKEY}}$ and \mathcal{O}_{COR} queries, and finally outputs a fine-grained re-encryption key $rk_{i^* \rightarrow j^*}^{f'}$ of the other direction for some $f' \in \mathcal{F}$ (not necessarily the same as f). \mathcal{A} succeeds if $rk_{i^* \rightarrow j^*}^{f'}$ is indeed a fine-grained re-encryption key from i^* to j^* , and the UNID security requires that \mathcal{A} hardly succeeds.

We note that there might not exist a specialized PPT algorithm to check whether $rk_{i^* \rightarrow j^*}^{f'}$ is indeed a fine-grained re-encryption key from i^* to j^* . Thus in $\text{Exp}_{\text{FPFE}, \mathcal{A}, n}^{\text{UNID}}$, we actually check the *functionality* of $rk_{i^* \rightarrow j^*}^{f'}$, i.e., whether it can convert a first-level ciphertext of user i^* that encrypts a randomly chosen message m into a second-level ciphertext of user j^* that encrypts $f'(m)$.

Actually, there are four trivial attacks **TA1'**-**TA4'** to obtain $rk_{i^* \rightarrow j^*}^{f'}$ or obtain the functionality of $rk_{i^* \rightarrow j^*}^{f'}$ for some f' .

TA1': $i^* = j^*$, in this case, \mathcal{A} directly gets $rk_{i^* \rightarrow j^*}^{f'} = rk_{j^* \rightarrow i^*}^f$ for $f' = f$.

TA2': $i^* \in \mathcal{Q}_c$, i.e., \mathcal{A} ever obtains $sk^{(i^*)}$. In this case, \mathcal{A} can use $sk^{(i^*)}$ to generate $rk_{i^* \rightarrow j^*}^{f'}$ itself by invoking $rk_{i^* \rightarrow j^*}^{f'} \leftarrow_s \text{FReKGen}(pk^{(i^*)}, sk^{(i^*)}, pk^{(j^*)}, f')$.

TA3': $(i^*, j^*) \in \mathcal{Q}_{rk}$, i.e., \mathcal{A} directly gets a $rk_{i^* \rightarrow j^*}^{f'}$ from the $\mathcal{O}_{\text{REKEY}}$ oracle.

TA4': $\exists j \in \mathcal{Q}_c$, s.t. $(i^*, j) \in \mathcal{Q}_{rk}$, i.e., \mathcal{A} gets $sk^{(j)}$ and $rk_{i^* \rightarrow j}^{f'}$ for some user j .

In this case, \mathcal{A} can use $sk^{(j)}$ and $rk_{i^* \rightarrow j}^{f'}$ to fulfill the functionality of $rk_{i^* \rightarrow j^*}^{f'}$.

To see this, given a first-level ciphertext $ct_1^{(i^*)}$ that encrypts a message m , it can firstly use $rk_{i^* \rightarrow j}^{f'}$ to re-encrypt $ct_1^{(i^*)}$ to a second-level ciphertext $ct_2^{(j)}$

that encrypts $f'(m)$ by invoking $ct_2^{(j)} \leftarrow_s \text{FReEnc}(rk_{i^* \rightarrow j}^{f'}, ct_1^{(i^*)})$, then use $sk^{(j)}$ to decrypt $ct_2^{(j)}$ via $f'(m) \leftarrow \text{Dec}_2(sk^{(j)}, ct_2^{(j)})$ to recover $f'(m)$, and finally encrypt $f'(m)$ under $pk^{(j^*)}$ to obtain a second-level ciphertext $ct_2^{(j^*)}$ via $ct_2^{(j^*)} \leftarrow_s \text{Enc}_2(pk^{(j^*)}, f'(m))$.

As such, we exclude the above trivial attacks in the UNID experiment.

Moreover, there is an additional trivial attack **TA5'** to obtain the *functionality* of $rk_{i^* \rightarrow j^*}^{f'}$ for certain f' .

TA5': The function f' is a constant function or an almost constant function, i.e., f' maps (almost) all messages $m \in \mathcal{M}$ to a constant $c = f'(m) \in \mathcal{M}$. In this case, \mathcal{A} trivially obtains the functionality of $rk_{i^* \rightarrow j^*}^{f'}$, since it can simply encrypt the constant c via $\text{Enc}_2(pk^{(j^*)}, c)$ to produce a second-level ciphertext of user j^* that encrypts $c = f'(m)$.

To exclude this additional trivial attack, we require that the function f' for which \mathcal{A} produces $rk_{i^* \rightarrow j^*}^{f'}$ satisfies the property of *output diversity*, i.e.,

$$\Pr[m_0, m_1 \leftarrow_s \mathcal{M} : f'(m_0) \neq f'(m_1)] \geq 1/\text{poly}(\lambda). \quad (8)$$

The output diversity of f' can be checked efficiently as follows: pick $m_{\ell 0}, m_{\ell 1} \leftarrow_s \mathcal{M}$ randomly for $\ell \in [\lambda \cdot \text{poly}(\lambda)]$, and return success if $\exists \ell$ s.t. $f'(m_{\ell 0}) \neq f'(m_{\ell 1})$ and return failure otherwise. If f' has output diversity, the above procedure returns success with an overwhelming probability $1 - (1 - 1/\text{poly}(\lambda))^{\lambda \cdot \text{poly}(\lambda)} \approx 1 - e^{-\lambda}$; otherwise, the above procedure returns failure with an overwhelming probability $(1 - \text{negl}(\lambda))^{\lambda \cdot \text{poly}(\lambda)} \geq 1 - \lambda \cdot \text{poly}(\lambda) \cdot \text{negl}(\lambda) = 1 - \text{negl}(\lambda)$.

Below we show that the UNID security is implied by the CPA security.

Lemma 6 (CPA \Rightarrow UNID). *For any PPT adversary \mathcal{A} breaking the UNID security of FPRE , there exists a PPT adversary \mathcal{B} breaking the CPA security of FPRE with $\text{Adv}_{\text{FPRE}, \mathcal{B}, n}^{\text{CPA}}(\lambda) = 1/2 \cdot \text{Adv}_{\text{FPRE}, \mathcal{A}, n}^{\text{UNID}}(\lambda)$.*

Proof. We construct \mathcal{B} to break the CPA security by simulating the UNID experiment $\text{Exp}_{\text{FPRE}, \mathcal{A}, n}^{\text{UNID}}$ for \mathcal{A} as follows.

Algorithm \mathcal{B} . Algorithm \mathcal{B} is given the public parameter $\text{pp}, \{pk^{(i)}\}_{i \in [n]}$ from its own challenger and has access to its own oracles $\mathcal{O}_{\text{REKEY}}, \mathcal{O}_{\text{COR}}$.

- (1) \mathcal{B} initializes $\mathcal{Q}_{rk} = \emptyset, \mathcal{Q}_c = \emptyset, i^* = \perp, j^* = \perp$ and sends $\text{pp}, \{pk^{(i)}\}_{i \in [n]}$ to \mathcal{A} .
 - On receiving a re-encryption key query (i, j, f) from \mathcal{A} , \mathcal{B} checks \mathcal{A} 's trivial attacks by checking if $(i = i^*)$ and $(j = j^* \text{ or } j \in \mathcal{Q}_c)$, just like $\text{Exp}_{\text{FPRE}, \mathcal{A}, n}^{\text{UNID}}$. If trivial attacks occur, \mathcal{B} returns \perp to \mathcal{A} , otherwise \mathcal{B} adds (i, j) to \mathcal{Q}_{rk} and queries (i, j, f) to its own oracle $\mathcal{O}_{\text{REKEY}}$. On receiving $rk_{i \rightarrow j}^f$ from $\mathcal{O}_{\text{REKEY}}(i, j, f)$, \mathcal{B} returns $rk_{i \rightarrow j}^f$ to \mathcal{A} .
 - On receiving a corruption query i from \mathcal{A} , \mathcal{B} checks \mathcal{A} 's trivial attacks by checking if $(i = i^*)$ or $(i^*, i) \in \mathcal{Q}_{rk}$, just like $\text{Exp}_{\text{FPRE}, \mathcal{A}, n}^{\text{UNID}}$. If trivial attacks occur, \mathcal{B} returns \perp to \mathcal{A} , otherwise \mathcal{B} adds i to \mathcal{Q}_c and queries i to its own oracle \mathcal{O}_{COR} . On receiving $sk^{(i)}$ from $\mathcal{O}_{\text{COR}}(i)$, \mathcal{B} returns $sk^{(i)}$ to \mathcal{A} .

- On receiving the challenge tuple (i^*, j^*, f) from \mathcal{A} , \mathcal{B} first checks if $(i^* \in \mathcal{Q}_c)$ or $((i^*, j^*) \in \mathcal{Q}_{rk})$ or $(\exists j \in \mathcal{Q}_c \text{ s.t. } (i^*, j) \in \mathcal{Q}_{rk})$ to identify trivial attacks, just like $\text{Exp}_{\text{FPRE}, \mathcal{A}, n}^{\text{UNID}}$. If yes, \mathcal{B} aborts the experiment with \mathcal{A} and returns a random bit $\beta' \leftarrow_{\$} \{0, 1\}$ to its own challenger. Otherwise, \mathcal{B} adds (i^*, j^*) to \mathcal{Q}_{rk} , and queries j^* to its own oracle \mathcal{O}_{COR} . On receiving $sk^{(j^*)}$ from $\mathcal{O}_{\text{COR}}(j^*)$, \mathcal{B} invokes $rk_{j^* \rightarrow i^*}^f \leftarrow_{\$} \text{FReKGen}(pk^{(j^*)}, sk^{(j^*)}, pk^{(i^*)}, f)$ and return $rk_{j^* \rightarrow i^*}^f$ to \mathcal{A} .
- (2) Finally, on receiving \mathcal{A} 's answer $(f', rk_{i^* \rightarrow j^*}^{f'})$, \mathcal{B} checks whether f' has output diversity efficiently. If f' does not have output diversity, \mathcal{B} aborts the experiment with \mathcal{A} and returns a random bit $\beta' \leftarrow_{\$} \{0, 1\}$ to its own challenger. Otherwise, \mathcal{B} chooses $m_0, m_1 \leftarrow_{\$} \mathcal{M}$ s.t. $f'(m_0) \neq f'(m_1)$, and sends challenge tuple (i^*, m_0, m_1) to its own challenger.

On receiving ct_1^* from its own challenger, \mathcal{B} invokes $ct_2^{(j^*)} \leftarrow_{\$} \text{FReEnc}(rk_{i^* \rightarrow j^*}^{f'}, ct_1^*)$ using the $rk_{i^* \rightarrow j^*}^{f'}$ produced by \mathcal{A} and computes $m' := \text{Dec}_2(sk^{(j^*)}, ct_2^{(j^*)})$. If $m' = f'(m_0)$, \mathcal{B} sets $\beta' = 0$, and if $m' = f'(m_1)$, \mathcal{B} sets $\beta' = 1$, otherwise, \mathcal{B} picks a random bit $\beta' \leftarrow_{\$} \{0, 1\}$. \mathcal{B} returns β' to its own challenger.

In the simulation, if \mathcal{A} implements trivial attacks **TA1'-TA5'**, \mathcal{B} will abort the experiment, just like $\text{Exp}_{\text{FPRE}, \mathcal{A}, n}^{\text{UNID}}$. Otherwise, no trivial attacks from \mathcal{A} implies that $i^* \notin \mathcal{Q}_c$ and there does not exist any re-encryption key from i^* to $j \in \mathcal{Q}_c \cup \{j^*\}$, while $\mathcal{Q}_c \cup \{j^*\}$ is exactly the corrupted users set for \mathcal{B} 's challenger. Thus, \mathcal{B} never issue queries leading to trivial attacks **TA1** and **TA2**. So \mathcal{B} simulates $\text{Exp}_{\text{FPRE}, \mathcal{A}, n}^{\text{UNID}}$ perfectly for \mathcal{A} .

Now we analyze the advantage of \mathcal{B} . Note that \mathcal{A} wins in $\text{Exp}_{\text{FPRE}, \mathcal{A}, n}^{\text{UNID}}$ means that the $rk_{i^* \rightarrow j^*}^{f'}$ produced by \mathcal{A} passes the check of functionality. Therefore, in the case of \mathcal{A} wins, for the challenge ciphertext ct_1^* that encrypts the randomly chosen message m_β , the re-encrypted ciphertext $ct_2^{(j^*)} \leftarrow_{\$} \text{FReEnc}(rk_{i^* \rightarrow j^*}^{f'}, ct_1^*)$ using the $rk_{i^* \rightarrow j^*}^{f'}$ produced by \mathcal{A} will decrypt to $m' := \text{Dec}_2(sk^{(j^*)}, ct_2^{(j^*)}) = f(m_\beta)$, and thus \mathcal{B} can guess β correctly with probability 1. Otherwise, \mathcal{B} will submit a random bit β' to its own challenger, and thus guess β correctly with probability 1/2. Overall,

$$\begin{aligned}
\text{Adv}_{\text{FPRE}, \mathcal{B}, n}^{\text{CPA}}(\lambda) &= |\Pr[\beta' = \beta] - \frac{1}{2}| \\
&= |\Pr[\mathcal{A} \text{ wins}] \cdot \Pr[\beta' = \beta \mid \mathcal{A} \text{ wins}] + \Pr[\neg \mathcal{A} \text{ wins}] \cdot \Pr[\beta' = \beta \mid \neg \mathcal{A} \text{ wins}] - \frac{1}{2}| \\
&= |\Pr[\mathcal{A} \text{ wins}] \cdot 1 + (1 - \Pr[\mathcal{A} \text{ wins}]) \cdot \frac{1}{2} - \frac{1}{2}| = \frac{1}{2} \cdot \Pr[\mathcal{A} \text{ wins}] = \frac{1}{2} \cdot \text{Adv}_{\text{FPRE}, \mathcal{A}, n}^{\text{UNID}}(\lambda). \square
\end{aligned}$$

B.2 More Discussions on NTR Security (Def. 4) and Its Relation to CPA Security

Remark 6 (On the formalization of NTR security and discussion on trivial attacks). We formalize the NTR security by defining the experiment $\text{Exp}_{\text{FPRE}, \mathcal{A}, n}^{\text{NTR}}$ in Fig. 4. Similar to previous security notions, we consider a multi-user setting, and the adversary \mathcal{A} is allowed to make $\mathcal{O}_{\text{REKEY}}$ and \mathcal{O}_{COR} queries *adaptively*. At some point, \mathcal{A} outputs a triple of challenge users (i^*, k^*, j^*) as well as a pair of

functions (f_1, f_2) , and receives two fine-grained re-encryption keys $rk_{i^* \rightarrow k^*}^{f_1}$ and $rk_{k^* \rightarrow j^*}^{f_2}$. \mathcal{A} continues to make $\mathcal{O}_{\text{REKEY}}$ and \mathcal{O}_{COR} queries, and finally outputs a fine-grained re-encryption key $rk_{i^* \rightarrow j^*}^{f'}$ from i^* directly to j^* for some $f' \in \mathcal{F}$. \mathcal{A} succeeds if $rk_{i^* \rightarrow j^*}^{f'}$ is indeed a fine-grained re-encryption key from i^* to j^* , and the NTR security requires that \mathcal{A} hardly succeeds. Similar to Remark 5, in $\text{Exp}_{\text{FPRE}, \mathcal{A}, n}^{\text{NTR}}$, we also check the *functionality* of $rk_{i^* \rightarrow j^*}^{f'}$, and we also exclude the five trivial attacks **TA1'**-**TA5'** as defined in Remark 5 (with a slight change in **TA1'** that i^*, k^*, j^* should be distinct) which can trivially obtain $rk_{i^* \rightarrow j^*}^{f'}$ or obtain the functionality of $rk_{i^* \rightarrow j^*}^{f'}$.

Below we show that the NTR security is implied by the CPA security.

Lemma 7 (CPA \Rightarrow NTR). *For any PPT adversary \mathcal{A} breaking the NTR security of FPRE, there exists a PPT adversary \mathcal{B} breaking the CPA security of FPRE with $\text{Adv}_{\text{FPRE}, \mathcal{B}, n}^{\text{CPA}}(\lambda) = 1/2 \cdot \text{Adv}_{\text{FPRE}, \mathcal{A}, n}^{\text{NTR}}(\lambda)$.*

Proof. The proof is similar to that of Lemma 6. We construct \mathcal{B} to break the CPA security by simulating the NTR experiment $\text{Exp}_{\text{FPRE}, \mathcal{A}, n}^{\text{NTR}}$ for \mathcal{A} as follows.

Algorithm \mathcal{B} . Algorithm \mathcal{B} is given the public parameter $\text{pp}, \{pk^{(i)}\}_{i \in [n]}$ from its own challenger and has access to its own oracles $\mathcal{O}_{\text{REKEY}}, \mathcal{O}_{\text{COR}}$.

- (1) \mathcal{B} initializes $\mathcal{Q}_{rk} = \emptyset, \mathcal{Q}_c = \emptyset, i^*, k^*, j^* = \perp$ and sends $\text{pp}, \{pk^{(i)}\}_{i \in [n]}$ to \mathcal{A} .
 - On receiving a re-encryption key query (i, j, f) from \mathcal{A} , \mathcal{B} checks \mathcal{A} 's trivial attacks by checking if $(i = i^*)$ and $(j = j^* \text{ or } j \in \mathcal{Q}_c)$, just like $\text{Exp}_{\text{FPRE}, \mathcal{A}, n}^{\text{NTR}}$. If trivial attacks occur, \mathcal{B} returns \perp to \mathcal{A} , otherwise \mathcal{B} adds (i, j) to \mathcal{Q}_{rk} and queries (i, j, f) to its own oracle $\mathcal{O}_{\text{REKEY}}$. On receiving $rk_{i \rightarrow j}^f$ from $\mathcal{O}_{\text{REKEY}}(i, j, f)$, \mathcal{B} returns $rk_{i \rightarrow j}^f$ to \mathcal{A} .
 - On receiving a corruption query i from \mathcal{A} , \mathcal{B} checks \mathcal{A} 's trivial attacks by checking if $(i = i^*)$ or $(i^*, i) \in \mathcal{Q}_{rk}$, just like $\text{Exp}_{\text{FPRE}, \mathcal{A}, n}^{\text{NTR}}$. If trivial attacks occur, \mathcal{B} returns \perp to \mathcal{A} , otherwise \mathcal{B} adds i to \mathcal{Q}_c and queries i to its own oracle \mathcal{O}_{COR} . On receiving $sk^{(i)}$ from $\mathcal{O}_{\text{COR}}(i)$, \mathcal{B} returns $sk^{(i)}$ to \mathcal{A} .
 - On receiving the challenge tuple $(i^*, k^*, j^*, f_1, f_2)$ from \mathcal{A} , \mathcal{B} first checks whether $(i^* \in \mathcal{Q}_c)$ or $((i^*, j^*) \in \mathcal{Q}_{rk})$ or $(\exists j \in \mathcal{Q}_c \text{ s.t. } (i^*, j) \in \mathcal{Q}_{rk})$ to identify trivial attacks, just like $\text{Exp}_{\text{FPRE}, \mathcal{A}, n}^{\text{NTR}}$. If yes, \mathcal{B} aborts the experiment with \mathcal{A} and returns a random bit $\beta' \leftarrow_{\$} \{0, 1\}$ to its own challenger. Otherwise, \mathcal{B} adds (i^*, k^*) and (k^*, j^*) to \mathcal{Q}_{rk} , queries $(i^*, k^*, f_1), (k^*, j^*, f_2)$ to its own oracle $\mathcal{O}_{\text{REKEY}}$ to obtain $rk_{i^* \rightarrow k^*}^{f_1}, rk_{k^* \rightarrow j^*}^{f_2}$ and sends them to \mathcal{A} .
- (2) Finally, on receiving \mathcal{A} 's answer $(f', rk_{i^* \rightarrow j^*}^{f'})$, \mathcal{B} checks whether f' has output diversity efficiently. If f' does not have output diversity, \mathcal{B} aborts the experiment with \mathcal{A} and returns a random bit $\beta' \leftarrow_{\$} \{0, 1\}$ to its own challenger. Otherwise, \mathcal{B} chooses $m_0, m_1 \leftarrow_{\$} \mathcal{M}$ s.t. $f'(m_0) \neq f'(m_1)$, and sends challenge tuple (i^*, m_0, m_1) to its own challenger.

On receiving ct_1^* from its own challenger, \mathcal{B} first queries j^* to its own oracle \mathcal{O}_{COR} to obtain $sk^{(j^*)}$ from $\mathcal{O}_{\text{COR}}(j^*)$. Then \mathcal{B} invokes $ct_2^{(j^*)} \leftarrow_{\$} \text{FReEnc}(rk_{i^* \rightarrow j^*}^{f'},$

ct_1^*) using the $rk_{i^* \rightarrow j^*}^{f'}$ produced by \mathcal{A} and computes $m' := \text{Dec}_2(sk^{(j^*)}, ct_2^{(j^*)})$. If $m' = f'(m_0)$, \mathcal{B} sets $\beta' = 0$, and if $m' = f'(m_1)$, \mathcal{B} sets $\beta' = 1$, otherwise, \mathcal{B} picks a random bit $\beta' \leftarrow_{\$} \{0, 1\}$. \mathcal{B} returns β' to its own challenger.

In the simulation, if \mathcal{A} implements trivial attacks **TA1'**-**TA5'**, \mathcal{B} will abort the experiment, just like $\text{Exp}_{\text{FPRE}, \mathcal{A}, n}^{\text{NTR}}$. Otherwise, no trivial attacks from \mathcal{A} implies that $i^* \notin \mathcal{Q}_c$ and there doesn't exist any re-encryption key from i^* to $j \in \mathcal{Q}_c \cup \{j^*\}$, while $\mathcal{Q}_c \cup \{j^*\}$ is exactly the corrupted users set for \mathcal{B} 's challenger. Thus, \mathcal{B} never issue queries leading to trivial attacks **TA1** and **TA2**. So \mathcal{B} simulates $\text{Exp}_{\text{FPRE}, \mathcal{A}, n}^{\text{NTR}}$ perfectly for \mathcal{A} .

Now we analyze the advantage of \mathcal{B} . Note that \mathcal{A} wins in $\text{Exp}_{\text{FPRE}, \mathcal{A}, n}^{\text{NTR}}$ means that the $rk_{i^* \rightarrow j^*}^{f'}$ produced by \mathcal{A} passes the check of functionality. Therefore, in the case of \mathcal{A} wins, for the challenge ciphertext ct_1^* that encrypts the randomly chosen message m_β , the re-encrypted ciphertext $ct_2^{(j^*)} \leftarrow_{\$} \text{FReEnc}(rk_{i^* \rightarrow j^*}^{f'}, ct_1^*)$ using the $rk_{i^* \rightarrow j^*}^{f'}$ produced by \mathcal{A} will decrypt to $m' := \text{Dec}_2(sk^{(j^*)}, ct_2^{(j^*)}) = f(m_\beta)$, and thus \mathcal{B} can guess β correctly with probability 1. Otherwise, \mathcal{B} will submit a random bit β' to its own challenger, and thus guess β correctly with probability 1/2. Overall,

$$\begin{aligned} \text{Adv}_{\text{FPRE}, \mathcal{B}, n}^{\text{CPA}}(\lambda) &= |\Pr[\beta' = \beta] - \frac{1}{2}| \\ &= |\Pr[\mathcal{A} \text{ wins}] \cdot \Pr[\beta' = \beta \mid \mathcal{A} \text{ wins}] + \Pr[\neg \mathcal{A} \text{ wins}] \cdot \Pr[\beta' = \beta \mid \neg \mathcal{A} \text{ wins}] - \frac{1}{2}| \\ &= |\Pr[\mathcal{A} \text{ wins}] \cdot 1 + (1 - \Pr[\mathcal{A} \text{ wins}]) \cdot \frac{1}{2} - \frac{1}{2}| = \frac{1}{2} \cdot \Pr[\mathcal{A} \text{ wins}] = \frac{1}{2} \cdot \text{Adv}_{\text{FPRE}, \mathcal{A}, n}^{\text{NTR}}(\lambda). \square \end{aligned}$$

B.3 More Discussions on CS Security (Def. 5) and Its Relation to Existing Formalizations

Traditionally, Collision-Safety (CS) requires that given a fine-grained re-encryption key $rk_{i^* \rightarrow j^*}^f$ starting from i^* to some j^* and a secret key $sk^{(j^*)}$ of j^* , it is hard for an adversary to compute a secret key $sk^{(i^*)}$ of i^* (or obtain the functionality of $sk^{(i^*)}$). See [3, 16] for example. This notion is also called master secret security in [3]. In fact, the secret key $sk^{(i^*)}$ of i^* has three functionalities/abilities, and two of them can already be fulfilled with $(rk_{i^* \rightarrow j^*}^f, sk^{(j^*)})$, so the CS security essentially stipulates the hardness for the adversary to obtain the third functionality of $sk^{(i^*)}$.

- (1) The first is the ability of $sk^{(i^*)}$ to generate fine-grained re-encryption keys $rk_{i^* \rightarrow k}^f$ starting from i^* to any user k via $rk_{i^* \rightarrow k}^f \leftarrow_{\$} \text{FReKGen}(pk^{(i^*)}, sk^{(i^*)}, pk^{(k)}, f)$, and the generated $rk_{i^* \rightarrow k}^f$ has the functionality of re-encrypting any first-level ciphertext $ct_1^{(i^*)}$ of user i^* into a second-level ciphertext $ct_2^{(k)}$ of user k . In fact, this functionality can also be fulfilled with $(rk_{i^* \rightarrow j^*}^f, sk^{(j^*)})$: given $ct_1^{(i^*)}$, one can firstly re-encrypt $ct_1^{(i^*)}$ to a second-level ciphertext $ct_2^{(j^*)}$ via $ct_2^{(j^*)} \leftarrow_{\$} \text{FReEnc}(rk_{i^* \rightarrow j^*}^f, ct_1^{(i^*)})$, then decrypt $ct_2^{(j^*)}$ via $\text{Dec}_2(sk^{(j^*)}, ct_2^{(j^*)})$

to recover a function $f(m)$ of the plaintext underlying $ct_1^{(i^*)}$, and finally encrypt $f(m)$ under $pk^{(k)}$ to obtain a second-level ciphertext $ct_2^{(k)}$ encrypted for k via $ct_2^{(k)} \leftarrow_s \text{Enc}_2(pk^{(k)}, f(m))$.

- (2) The second is the ability of $sk^{(i^*)}$ to decrypt first-level ciphertexts $ct_1^{(i^*)}$ intended for i^* via $\text{Dec}_1(sk^{(i^*)}, ct_1^{(i^*)})$. This functionality can also be fulfilled with $(rk_{i^* \rightarrow j^*}^f, sk^{(j^*)})$: given $ct_1^{(i^*)}$, one can firstly re-encrypt $ct_1^{(i^*)}$ to a second-level ciphertext $ct_2^{(j^*)}$ encrypted for j^* via $ct_2^{(j^*)} \leftarrow_s \text{FReEnc}(rk_{i^* \rightarrow j^*}^f, ct_1^{(i^*)})$, then simply decrypt $ct_2^{(j^*)}$ via $\text{Dec}_2(sk^{(j^*)}, ct_2^{(j^*)})$ to learn a function of the plaintext underlying $ct_1^{(i^*)}$.
- (3) The third is the ability of $sk^{(i^*)}$ to decrypt second-level ciphertexts $ct_2^{(i^*)}$ intended for i^* via $m \leftarrow \text{Dec}_2(sk^{(i^*)}, ct_2^{(i^*)})$.

Consequently, CS security essentially characterizes the hardness to obtain the decryption ability of second-level ciphertexts $ct_2^{(i^*)}$ intended for i^* , and accordingly, we will formalize our CS security as the *CPA security for the second-level ciphertexts*. We stress that our CS security is at least as strong as the existing collusion-safety formalizations like [3, 16], since the adversary can easily decrypt second-level ciphertexts intended for i^* if it is able to compute $sk^{(i^*)}$, but not vice versa.

B.4 Proof of Lemma 1 (CUL \Rightarrow CPA)

Lemma 1 (CUL \Rightarrow CPA) *For any PPT adversary \mathcal{A} breaking the CPA security of FPFE and any polynomial n , there exists a PPT adversary \mathcal{B} breaking the CUL security of FPFE with $\text{Adv}_{\text{FPFE}, \mathcal{B}, n+1}^{\text{CUL}}(\lambda) = \frac{1}{2} \cdot \text{Adv}_{\text{FPFE}, \mathcal{A}, n}^{\text{CPA}}(\lambda)$.*

Proof. We construct \mathcal{B} to break the CUL security by simulating the CPA experiment $\text{Exp}_{\text{FPFE}, \mathcal{A}, n}^{\text{CPA}}$ for \mathcal{A} as follows.

Algorithm \mathcal{B} . Algorithm \mathcal{B} is given the public parameter pp , $\{pk^{(i)}\}_{i \in [n+1]}$ from its own challenger and has access to its own oracles $\mathcal{O}_{\text{REKEY}}, \mathcal{O}_{\text{COR}}$.

- (1) \mathcal{B} initializes $\mathcal{Q}_{rk} = \emptyset, \mathcal{Q}_c = \emptyset, i^* = \perp$ and sends $\text{pp}, \{pk^{(i)}\}_{i \in [n]}$ to \mathcal{A} .
 - On receiving a re-encryption key query (i, j, f) from \mathcal{A} , \mathcal{B} checks \mathcal{A} 's trivial attacks by checking if $(i = i^*)$ and $(j \in \mathcal{Q}_c)$, just like $\text{Exp}_{\text{FPFE}, \mathcal{A}, n}^{\text{CPA}}$. If trivial attacks occur, \mathcal{B} returns \perp to \mathcal{A} , otherwise \mathcal{B} adds (i, j) to \mathcal{Q}_{rk} and queries (i, j, f) to its own oracle $\mathcal{O}_{\text{REKEY}}$. On receiving $rk_{i \rightarrow j}^f$ from $\mathcal{O}_{\text{REKEY}}(i, j, f)$, \mathcal{B} returns $rk_{i \rightarrow j}^f$ to \mathcal{A} .
 - On receiving a corruption query i from \mathcal{A} , \mathcal{B} checks \mathcal{A} 's trivial attacks by checking if $(i = i^*)$ or $(i^*, i) \in \mathcal{Q}_{rk}$, just like $\text{Exp}_{\text{FPFE}, \mathcal{A}, n}^{\text{CPA}}$. If trivial attacks occur, \mathcal{B} returns \perp to \mathcal{A} , otherwise \mathcal{B} adds i to \mathcal{Q}_c and queries i to its own oracle \mathcal{O}_{COR} . On receiving $sk^{(i)}$ from $\mathcal{O}_{\text{COR}}(i)$, \mathcal{B} returns $sk^{(i)}$ to \mathcal{A} .
 - On receiving the challenge tuple (i^*, m_0, m_1) from \mathcal{A} , \mathcal{B} first checks if $(i^* \in \mathcal{Q}_c)$ or $(\exists j \in \mathcal{Q}_c \text{ s.t. } (i^*, j) \in \mathcal{Q}_{rk})$ to identify trivial attacks, just like

- $\text{Exp}_{\text{FPRE}, \mathcal{A}, n}^{\text{CPA}}$. If yes, \mathcal{B} aborts the experiment with \mathcal{A} and returns a random bit $\beta' \leftarrow_{\text{s}} \{0, 1\}$ to its own challenger. Otherwise, \mathcal{B} chooses $m_2 \leftarrow_{\text{s}} \mathcal{M}$, $f \leftarrow_{\text{s}} \mathcal{F}$ and sends challenge tuple $(i^*, j^* := n+1, (f, m_0), (m_1, m_2))$ to its own challenger. On receiving (ct_1^*, ct_2^*) from its own challenger, \mathcal{B} returns ct_1^* to \mathcal{A} .
- (2) Finally, on receiving \mathcal{A} 's answer β' , \mathcal{B} returns β' to its own challenger.

In the simulation, if \mathcal{A} implements trivial attacks **TA1-TA2**, \mathcal{B} will abort the experiment, just like $\text{Exp}_{\text{FPRE}, \mathcal{A}, n}^{\text{CPA}}$. Otherwise, no trivial attacks from \mathcal{A} implies that $i^* \notin \mathcal{Q}_c$ and there does not exist any re-encryption key from i^* to $j \in \mathcal{Q}_c$, while \mathcal{Q}_c is exactly the corrupted users set for \mathcal{B} 's challenger. Moreover, since \mathcal{B} sets $j^* := n+1$ and user $n+1$ is invisible to \mathcal{A} , we have $j^* \notin \mathcal{Q}_c$. Thus, \mathcal{B} never issue queries leading to trivial attacks **TA1''** and **TA2''**. So \mathcal{B} simulates $\text{Exp}_{\text{FPRE}, \mathcal{A}, n}^{\text{CPA}}$ perfectly for \mathcal{A} .

Now we analyze the advantage of \mathcal{B} . We denote the challenge bit chosen by \mathcal{B} 's challenger by β_C . Note that \mathcal{A} wins in $\text{Exp}_{\text{FPRE}, \mathcal{A}, n}^{\text{CPA}}$ means that $\beta' = \beta$, where \mathcal{B} implicitly sets $\beta := \beta_C$. If $\beta_C = 0$, ct_1^* is generated by $ct_1^* \leftarrow_{\text{s}} \text{Enc}_1(pk^{(i^*)}, m_0)$, which implies that $\beta = 0$. If $\beta_C = 1$, ct_1^* is generated by $ct_1^* \leftarrow_{\text{s}} \text{Enc}_1(pk^{(i^*)}, m_1)$, which implies that $\beta = 1$. Thus, if $\beta' = \beta$, we have $\beta' = \beta_C$ and \mathcal{B} can guess β_C correctly with probability 1. Otherwise, \mathcal{B} will submit a random bit β' to its own challenger, and thus guess β_C correctly with probability $1/2$. Overall,

$$\begin{aligned} \text{Adv}_{\text{FPRE}, \mathcal{B}, n+1}^{\text{CUL}}(\lambda) &= |\Pr[\beta' = \beta_C] - \frac{1}{2}| \\ &= |\Pr[\mathcal{A} \text{ wins}] \cdot \Pr[\beta' = \beta_C | \mathcal{A} \text{ wins}] + \Pr[\neg \mathcal{A} \text{ wins}] \cdot \Pr[\beta' = \beta_C | \neg \mathcal{A} \text{ wins}] - \frac{1}{2}| \\ &= |\Pr[\mathcal{A} \text{ wins}] \cdot 1 + (1 - \Pr[\mathcal{A} \text{ wins}]) \cdot \frac{1}{2} - \frac{1}{2}| = \frac{1}{2} \cdot \Pr[\mathcal{A} \text{ wins}] = \frac{1}{2} \cdot \text{Adv}_{\text{FPRE}, \mathcal{A}, n}^{\text{CPA}}(\lambda). \square \end{aligned}$$

C Proof of Theorem 3 (Collusion-Safety of $\text{FPRE}_{\text{LWE}}^{\text{lin}}$)

Theorem 3 (Collusion-Safety of $\text{FPRE}_{\text{LWE}}^{\text{lin}}$) Assume that the $\text{LWE}_{n, q, \chi, N+\ell}$ -assumption holds, then the scheme $\text{FPRE}_{\text{LWE}}^{\text{lin}}$ proposed in Fig. 7 is collusion-safe (CS). More precisely, for any PPT adversary \mathcal{A} and any polynomial n , there exists a PPT algorithm \mathcal{B} against the LWE assumption such that $\text{Adv}_{\text{FPRE}, \mathcal{A}, n}^{\text{CS}}(\lambda) \leq n \cdot \text{Adv}_{[n, q, \chi, N+\ell], \mathcal{B}}^{\text{LWE}}(\lambda) + \text{negl}(\lambda)$.

Proof of Theorem 3. We prove the theorem via a sequence of games G_0'' - G_3'' , where G_0'' is the CS experiment, and in G_3'' , \mathcal{A} has a negligible advantage.

Game G_0'' : This is the CS experiment $\text{Exp}_{\text{FPRE}, \mathcal{A}, n}^{\text{CS}}$ (cf. Fig. 5). Let Win denote the event that $\beta' = \beta$. By definition, $\text{Adv}_{\text{FPRE}, \mathcal{A}, n}^{\text{CS}}(\lambda) = |\Pr_0''[\text{Win}] - \frac{1}{2}|$.

Let $\text{pp} = \overline{\mathbf{A}'}^i$ and let $pk^{(i)} = (\mathbf{A}^{(i)}, \underline{\mathbf{A}}'^{(i)})$, $sk^{(i)} = (\mathbf{T}^{(i)}, \mathbf{K}^{(i)})$ denote the public key and secret key of user $i \in [n]$. In this game, the challenger answers \mathcal{A} 's $\mathcal{O}_{\text{REKEY}}$, \mathcal{O}_{COR} queries and generates the challenge ciphertext ct_2^* as follows.

- On receiving a re-encryption key query $\mathcal{O}_{\text{REKEY}}(i, j, f_{\mathbf{M}})$ from \mathcal{A} , the challenger invokes $\mathbf{R} \leftarrow_{\text{s}} \text{SamplePre}(\mathbf{T}^{(i)}, \overline{\mathbf{A}}'^{(i)}, \mathbf{A}'^{(j)} \mathbf{S} + \mathbf{E} - \binom{\mathbf{0}}{\mathbf{M}} \underline{\mathbf{A}}^{(i)}, \gamma)$, where $\mathbf{A}'^{(j)} = \binom{\overline{\mathbf{A}}'^{(j)}}{\underline{\mathbf{A}}'^{(j)}}$, and returns $rk_{i \rightarrow j}^{f_{\mathbf{M}}} := \left(\mathbf{R} \mid \binom{\mathbf{0}}{\mathbf{M}} \right)$ to \mathcal{A} .

- On receiving a corruption query $\mathcal{O}_{\text{Cor}}(i)$ from \mathcal{A} , the challenger returns \perp to \mathcal{A} directly if trivial attack $i = i^*$ occurs. Otherwise, the challenger adds i to \mathcal{Q}_c and returns $sk^{(i)}$ to \mathcal{A} .
- On receiving the challenge tuple $(i^*, \mathbf{m}_0, \mathbf{m}_1)$ from \mathcal{A} , the challenger first checks if trivial attack $i^* \in \mathcal{Q}_c$ occurs. If yes, the challenger aborts the game with \mathcal{A} by returning a random bit. Otherwise, the challenger chooses a random bit $\beta \leftarrow_{\$} \{0, 1\}$, samples $\mathbf{s} \leftarrow_{\$} \chi^n$, $\mathbf{e} \leftarrow_{\$} \chi^{N+\ell}$, sets $\mathbf{A}'^{(i^*)} := (\underline{\overline{\mathbf{A}'^{(i^*)}}})$, and sends $ct_2^* := \mathbf{A}'^{(i^*)}\mathbf{s} + \mathbf{e} + \binom{\mathbf{0}}{p\mathbf{m}_\beta}$ to \mathcal{A} .

Game G_1'' : It is the same as G_0'' , except that, at the beginning of the game, the challenger chooses a random user index $i' \leftarrow_{\$} [n]$ uniformly as the guess of the challenge user i^* , and will abort the game if \mathcal{A} issues the challenge tuple $(i^*, \mathbf{m}_0, \mathbf{m}_1)$ but $i' \neq i^*$. Since the challenger will guess i^* correctly with probability $1/n$, we have $|\Pr_0''[\text{Win}] - \frac{1}{2}| = \frac{1}{n} |\Pr_1''[\text{Win}] - \frac{1}{2}|$.

Game G_2'' : It is the same as G_1'' , except that, for the generation of $pk^{(i')} = (\mathbf{A}^{(i')}, \underline{\mathbf{A}'}^{(i')})$, now the challenger generates $\underline{\mathbf{A}'}^{(i')}$ by sampling $\underline{\mathbf{A}'}^{(i')} \leftarrow_{\$} \mathbb{Z}_q^{\ell \times n}$, instead of computing $\underline{\mathbf{A}'}^{(i')} := \mathbf{K}^{(i')}\overline{\mathbf{A}}$ with $\mathbf{K}^{(i')} \leftarrow_{\$} \{0, 1\}^{\ell \times N}$ as in G_1'' . Note that for each row of $\mathbf{K}^{(i')} \leftarrow_{\$} \{0, 1\}^{\ell \times N}$ and for any q 's prime factor p' , we have that \mathbf{H}_∞ (each row of $\mathbf{K}^{(i')} \bmod p'$) = $N \geq 2n \log q + 2\omega(\log \lambda)$. Then according to Lemma 5, $\underline{\mathbf{A}'}^{(i')} := \mathbf{K}^{(i')}\overline{\mathbf{A}}$ generated in G_1'' is statistically close to the uniform distribution $\underline{\mathbf{A}'}^{(i')} \leftarrow_{\$} \mathbb{Z}_q^{\ell \times n}$ as in G_2'' . Moreover, since i^* is not allowed to be corrupted by \mathcal{A} , it is needless to keep $\mathbf{K}^{(i')}$ as long as $i' = i^*$. Therefore, G_2'' is statistically close to G_1'' , and we have $|\Pr_1''[\text{Win}] - \Pr_2''[\text{Win}]| \leq \text{negl}(\lambda)$.

Game G_3'' : It is the same as G_2'' , except for the generation of ct_2^* . In this game, the challenger picks $ct_2^* \leftarrow_{\$} \mathbb{Z}_q^{N+\ell}$ uniformly, rather than generating it by $ct_2^* := \mathbf{A}'^{(i^*)}\mathbf{s} + \mathbf{e} + \binom{\mathbf{0}}{p\mathbf{m}_\beta}$ with $\mathbf{s} \leftarrow_{\$} \chi^n$, $\mathbf{e} \leftarrow_{\$} \chi^{N+\ell}$, $\mathbf{A}'^{(i^*)} := (\underline{\overline{\mathbf{A}'^{(i^*)}}})$ as in G_2'' .

Clearly, the challenge bit β is completely hidden to \mathcal{A} , thus $\Pr_3''[\text{Win}] = \frac{1}{2}$. Next we show that G_2'' and G_3'' are computationally indistinguishable.

Due to the game changes introduced in G_1'' and G_2'' , we have that $i' = i^*$ and $\mathbf{A}'^{(i^*)} = (\underline{\overline{\mathbf{A}'^{(i^*)}}}) = (\underline{\overline{\mathbf{A}'^{(i')}}})$ is uniformly sampled from $\mathbb{Z}_q^{(N+\ell) \times n}$. Then according to the LWE assumption, $\mathbf{A}'^{(i^*)}\mathbf{s} + \mathbf{e}$ is computationally indistinguishable from the uniform distribution over $\mathbb{Z}_q^{N+\ell}$. Due to the independence between $\mathbf{A}'^{(i^*)}\mathbf{s} + \mathbf{e}$ and $\binom{\mathbf{0}}{p\mathbf{m}_\beta}$, we know that the challenge ciphertext $ct_2^* := \mathbf{A}'^{(i^*)}\mathbf{s} + \mathbf{e} + \binom{\mathbf{0}}{p\mathbf{m}_\beta}$ generated in G_2'' is also computationally indistinguishable from the uniformly chosen $ct_2^* \leftarrow_{\$} \mathbb{Z}_q^{N+\ell}$ in G_3'' . Thus we have $|\Pr_2''[\text{Win}] - \Pr_3''[\text{Win}]| \leq \text{Adv}_{[n, q, \chi, N+\ell], \mathcal{B}}^{\text{LWE}}(\lambda)$.

Finally, taking all things together, Theorem 3 follows. \square

D Description of $\text{FPRE}_{\text{LWE}}^{\text{del}}$ for Deletion Functions

<p>$\text{pp} \leftarrow \text{Setup}'$: $\overline{\mathbf{A}}' \leftarrow \mathbb{Z}_q^{N \times n}$ Return $\text{pp} := \overline{\mathbf{A}}'$</p> <p>$(pk, sk) \leftarrow \text{KGen}'(\text{pp})$: $(\overline{\mathbf{A}} \in \mathbb{Z}_q^{N \times n}, \mathbf{T}) \leftarrow \text{TrapGen}(1^n, 1^N)$, $\underline{\mathbf{A}} \leftarrow \mathbb{Z}_q^{2\ell \times n}$ $\mathbf{A} := \begin{pmatrix} \overline{\mathbf{A}} \\ \underline{\mathbf{A}} \end{pmatrix} \in \mathbb{Z}_q^{(N+2\ell) \times n}$ $\mathbf{K} \leftarrow \{0, 1\}^{2\ell \times N}$, $\underline{\mathbf{A}}' := -\mathbf{K}\overline{\mathbf{A}}'$ $pk := (\mathbf{A}, \underline{\mathbf{A}})$, $sk := (\mathbf{T}, \mathbf{K})$ Return (pk, sk)</p> <p>$rk_{i \rightarrow j}^{\text{FP}} \leftarrow \text{FReKGen}'(pk^{(i)} = (\mathbf{A}^{(i)}, \underline{\mathbf{A}}'^{(i)}), sk^{(i)} = (\mathbf{T}^{(i)}, \mathbf{K}^{(i)}),$ $\overline{pk}^{(j)} = (\mathbf{A}^{(j)}, \underline{\mathbf{A}}'^{(j)}), f_P \in \mathcal{F}_{\text{del}})$:</p> <p>$\mathbf{M} = (M_{i,j}) := \mathbf{0}_{2\ell \times 2\ell}$ For $i \in [\ell]$: If $i \notin \mathcal{P}$: $M_{2i-1, 2i-1} := 1, M_{2i, 2i} := 1$ // not delete $\mathbf{S} \leftarrow \chi^{n \times n}$, $\mathbf{E} \leftarrow \chi^{(N+2\ell) \times n}$ $\mathbf{A}'^{(j)} := \begin{pmatrix} \overline{\mathbf{A}}'^{(j)} \\ \underline{\mathbf{A}}'^{(j)} \end{pmatrix}$ $\mathbf{R} \in \mathbb{Z}^{(N+2\ell) \times N} \leftarrow \text{SamplePre}(\mathbf{T}^{(i)}, \overline{\mathbf{A}}'^{(i)}, \mathbf{A}'^{(j)}\mathbf{S} + \mathbf{E} - \begin{pmatrix} \mathbf{0} \\ \mathbf{M} \end{pmatrix} \underline{\mathbf{A}}'^{(i)}, \gamma)$ $rk_{i \rightarrow j}^{\text{FP}} := \begin{pmatrix} \mathbf{R} & \mathbf{0} \\ & \mathbf{M} \end{pmatrix} \in \mathbb{Z}^{(N+2\ell) \times (N+2\ell)}$ Return $rk_{i \rightarrow j}^{\text{FP}}$</p> <p>$ct_1 \leftarrow \text{Enc}'_1(pk = (\mathbf{A}, \underline{\mathbf{A}}), \mathbf{m} \in \{0, 1, *\}^\ell)$: $\mathbf{m}' \in \{0, 1\}^{2\ell} \leftarrow \text{Encode}(\mathbf{m})$ $\mathbf{s} \leftarrow \chi^n$, $\mathbf{e} \leftarrow \chi^{N+2\ell}$ $ct_1 := \mathbf{A}\mathbf{s} + \mathbf{e} + \begin{pmatrix} \mathbf{0} \\ \text{pm}' \end{pmatrix} \in \mathbb{Z}_q^{N+2\ell}$ Return ct_1</p>	<p>$ct_2 \leftarrow \text{Enc}'_2(pk = (\mathbf{A}, \underline{\mathbf{A}}), \mathbf{m} \in \{0, 1, *\}^\ell)$: $\mathbf{s} \leftarrow \chi^n$, $\mathbf{e} \leftarrow \chi^{N+2\ell}$ $\mathbf{A}' := \begin{pmatrix} \overline{\mathbf{A}}' \\ \underline{\mathbf{A}}' \end{pmatrix}$ $\mathbf{m}' \in \{0, 1\}^{2\ell} \leftarrow \text{Encode}(\mathbf{m})$ $ct_2 := \mathbf{A}'\mathbf{s} + \mathbf{e} + \begin{pmatrix} \mathbf{0} \\ \text{pm}' \end{pmatrix} \in \mathbb{Z}_q^{N+2\ell}$ Return ct_2</p> <p>$ct_2^{(j)} \leftarrow \text{FReEnc}'(rk_{i \rightarrow j}^{\text{FP}} \in \mathbb{Z}^{(N+2\ell) \times (N+2\ell)},$ $\overline{ct_1}^{(i)} \in \mathbb{Z}_q^{N+2\ell})$: $ct_2^{(j)} := rk_{i \rightarrow j}^{\text{FP}} \cdot \overline{ct_1}^{(i)} \in \mathbb{Z}_q^{N+2\ell}$ Return $ct_2^{(j)}$</p> <p>$\mathbf{m} \leftarrow \text{Dec}'_1(sk = (\mathbf{T}, \mathbf{K}), ct_1 \in \mathbb{Z}_q^{N+2\ell})$: Parse $ct_1 = \begin{pmatrix} ct_1 \in \mathbb{Z}_q^N \\ ct_1 \in \mathbb{Z}_q^{2\ell} \end{pmatrix}$ $(\mathbf{s}, \mathbf{e}) \leftarrow \text{Invert}(\mathbf{T}, ct_1)$ $\overline{\mathbf{m}} = (\overline{m}_1, \dots, \overline{m}_{2\ell}) := ct_1 - \mathbf{A}\mathbf{s}$ For all $i \in [2\ell]$: $m'_i := \lceil \overline{m}_i / p \rceil$ $\mathbf{m} \in \{0, 1, *\}^\ell \leftarrow \text{Decode}(\mathbf{m}' = (m'_1, \dots, m'_{2\ell}))$ Return \mathbf{m}</p> <p>$\mathbf{m} \leftarrow \text{Dec}'_2(sk = (\mathbf{T}, \mathbf{K}), ct_2 \in \mathbb{Z}_q^{N+2\ell})$: $\overline{\mathbf{m}} = (\overline{m}_1, \dots, \overline{m}_{2\ell}) := (\mathbf{K} \mid \mathbf{I}_{2\ell \times 2\ell}) \cdot ct_2$ For all $i \in [2\ell]$: $m'_i := \lceil \overline{m}_i / p \rceil$ $\mathbf{m} \in \{0, 1, *\}^\ell \leftarrow \text{Decode}(\mathbf{m}' = (m'_1, \dots, m'_{2\ell}))$ Return \mathbf{m}</p>
--	---

Fig. 8. The LWE-based FPRE scheme $\text{FPRE}_{\text{LWE}}^{\text{del}}$ for the deletion function family \mathcal{F}_{del} .