Proxy Re-Encryption Schemes with Key Privacy from LWE ⋆

Le Trieu Phong Lihua Wang Yoshinori Aono Manh Ha Nguyen Xavier Boyen

Abstract. Proxy re-encryption (PRE) is a cryptographic primitive in which a proxy can transform Alice's ciphertexts into ones decryptable by Bob. Key-private PRE specifies an additional level of security, requiring that proxy keys leak no information on the identities of Alice and Bob. In this paper, we build two key-private PRE schemes: (1) we propose a CPA-secure key-private PRE scheme in the standard model, and (2) we then transform it into a CCA-secure scheme in the random oracle model. Both schemes enjoy following properties: both are uni-directional and the CPA-secure one is a multi-hop scheme. In addition, the security of our schemes is based on the hardness of the standard Learning-With-Errors (LWE) problem, itself reducible from worst-case lattice hard problems that are conjectured immune to quantum cryptanalysis, or "post-quantum". We implement the CPA-secure scheme and point out that, among many applications, it can be sufficiently used for the practical task of key rotation over encrypted data.

Keywords: proxy re-encryption, key privacy, learning with errors, chosen ciphertext security, key rotation over encrypted data

1 Introduction

Proxy re-encryption (PRE), introduced in [11], is a type of public key encryption, featuring a kind of delegation via "proxy re-encryption keys": Alice and Bob can set up a proxy key $rk_{A\to B}$ transforming Alice's ciphertexts to those Bob can decrypt. The proxy key $rk_{A\to B}$ is usually put in a semi-trusted server whose task is to do the ciphertext transformation, and not being able to decrypt.

Key-private PRE. When anonymity is a concern, it is required that neither the delegator (Alice) nor the delegatee (Bob) be identifiable from the proxy re-encryption key $rk_{A\to B}$ that they set up. We speak of *key-private* PRE — which, informally, requires that the key $rk_{A\to B}$ looks random from the viewpoint of the proxy server.

Key-private PRE (a.k.a., anonymous PRE), as summarized in [6], can be useful in various applications as PRE, while additionally providing anonymity, such as distributed file systems, digital rights management, credential system, and forwarding encrypted emails. Later in this section, we will point out that key-private PRE can also be useful for the task of asymmetric key rotation in encrypted data storage.

LWE. "Learning With Errors" or LWE [30] is rapidly emerging as a hardness assumption of choice when long-term security is an issue, both classical and quantum. As shown in [30, 13], the LWE assumption theoretically has strong connection to lattice hardness assumptions, which are conjectured very safe in many respects. In practice, however, there are certain efficient "attacks" on LWE, such as those illustrated in [26, 29, 24, 25, 3]. Although none of those attacks comes anywhere close to breaking the LWE assumption in a theoretical sense, they force us to be careful in our choice of LWE parameters especially in the implementation of LWE-based schemes.

⋆ Abridged parts of this paper was presented in [3]. L. T. Phong, L. Wang, and Y. Aono are with NICT, Japan. M. H. Nguyen was at Tokyo Institute of Technology. X. Boyen is with Queensland Institute of Technology. Emails: {phong, wlh, aono}@nict.go.jp.

1.1 Our Contributions

We first construct key-private PRE schemes secure under the LWE assumption. We then propose concrete parameters for the schemes, and then make a testing implementation to demonstrate the efficiency. Finally, we show how to use our schemes for the task of key rotation. Details are as follows.

• (**CPA-secure scheme**) We design a key-private PRE against chosen plaintext attacks (CPA). This is achieved by transforming the public key encryption scheme of [24] in ways that are reminiscent of certain techniques developed in the context of fully homomorphic encryption schemes such as [12], notwithstanding certain impossibility results shown in [6]. At a very high level, we exploit two (new) facts about (our variant of) the encryption scheme of [24]: recipient anonymity, and additive homomorphism. We use those facts to show that our PRE scheme is CPA-secure under the LWE assumption, in the standard model. See Section 3.

• (**CCA-secure scheme**) We then show that our scheme is eligible for conversion into a CCA-secure encryption scheme, using the well-known Fujisaki-Okamoto method [18, 19], without losing the key-private PRE functionality. Consequently, this scheme is CCA-secure under the LWE assumption, in the random oracle model [10]. See Section 4.

• (**Implementation**) The CPA-secure scheme is implemented in Section 3.5. While requiring some storage for keys and ciphertexts, the scheme has extremely high speed in encryption and decryption. The CCA-secure scheme is only a little slower.

• (**Key rotation using key-private PRE**) Key rotation is the process of re-keying encrypted data from an old key to a new one. Formally, on the federal side, it is recommended by NIST [27] via the concept of cryptoperiods of keys, namely the time spans during which they are used. As suggested in [27, Table 1], cryptoperiods are on the order of 1-2 years. On the industrial side, it is required by the Payment Card Industry Data Security Standard (PCI DSS) [1], and is recommended by the Open Web Application Security Project (OWASP) [2], specifying that "*key rotation is a must as all good keys do come to an end either through expiration or revocation. So a developer will have to deal with rotating keys at some point – better to have a system in place now rather than scrambling later.*"

We find that key-private PRE can be useful for the purpose of asymmetric key rotation. More precisely, let C be the encryption of a plaintext M under a public key pk_{old}, namely $C_{\mathsf{old}} = \mathsf{Enc}(pk_{\mathsf{old}}, M)$. We want to obtain an encryption of the *same* M under *another* public key pk_{new}, namely we want to compute $C_{\mathsf{new}} = \mathsf{Enc}(pk_{\mathsf{new}}, M)$. Using a key-private PRE scheme, the task can be accomplished by an outsourced server holding the re-encryption key $rk_{\mathsf{old}\to\mathsf{new}}$. The key privacy of $rk_{\mathsf{old}\to\mathsf{new}}$ will ensure that the key is psuedo-random, so that the outsourced server will not obtain any useful information.

More specifically, consider an encrypted file system using key-private PRE. Whenever key rotation is necessary, a data holder generates $(pk_{\mathsf{new}}, sk_{\mathsf{new}})$. He then uses $(pk_{\mathsf{new}}, sk_{\mathsf{new}})$ and his previously generated $(pk_{\mathsf{old}}, sk_{\mathsf{old}})$ to create the re-encryption key $rk_{\mathsf{old}\to\mathsf{new}}$, which is sent to the file system server. Using re-encryption, all data encrypted under pk_{old} can be transformed into those under pk_{new}. Finally, the data holder discards $(pk_{\mathsf{old}}, sk_{\mathsf{old}})$, and the file system erases ciphertexts under pk_{old}.

With our proposed schemes, the task of asymmetric key rotation can be accomplished efficiently. Specifically, by the implementation in Section 3.5, one thousand ciphertexts can be rotated in a few seconds, using even a laptop.

1.2 Discussion on two intuitions of key privacy

We realize that there are two intuitions on key privacy existing in the literature, categorized below.

Table 1. Classifications of used notions of key privacy.

Intuition 1 (no *identity* revelation from rk)	**Intuition 2** (no *public key* revelation from rk)
this work, [6]	[35, 28]

- (**Intuition 1**) The proxy key $rk_{A\to B}$ is required to leak no information on the *identities* of parties A and B.
- (**Intuition 2**) The proxy key $rk_{A\to B}$ is required to leak no information on the *public keys* of parties A and B.

In any setting where the link (e.g., via signatures as in PKI) between public key and identity are ensured, then the above intuitions are identical. However, when there is no such link, they are different.

In Table 1, we classify papers using the two intuitions. Our work uses Intuition 1.

In the literature, the paper [6] has used Intuition 1. Indeed, quoting from the abstract of [6], "*... use the proxy to help re-encrypt sensitive information without revealing to the proxy the identity of the recipients.*" Formally, [6, full version, Definition 2.5], the challenge oracle gets the challenge input as identities (i, j), not public keys (pk_i, pk_j).

On the other hand, in the literature, the paper [35] has used Intuition 2, which is clear from their security definitions, since the challenge oracle takes public keys as the challenge input. We classify [28] as using Intuition 2 because that paper makes an effort to hide the delegatee's public key by re-randomization.

Choosing which intuition depends on concrete applications. Some examples supporting Intuition 1 are as follows.

• **Example 1:** In the application of key rotation in Section 1.1, Intuition 1 is enough, as it sufficiently ensures that the proxy server is harmless.

• **Example 2:** In email forwarding [6], "*Alice can hide the fact that Bob is a delegatee by instructing the server to convert her encrypted emails via a key-private PRE scheme and to forward the results to an anonymous (or group) email address (i.e., an address reachable by Bob but that does not contain any identifiable information on Bob, like a P.O. Box address indeed).*"

Therefore, Intuition 1 makes sense because it is assumed above that there is no link between the anonymous e-mail address (of the delegatee) and his/her identity.

• **Example 3:** In distributed file system [6], "*Alice may want Bob to read some of her encrypted files, thus she instructs the file system to convert those files using a proxy re-encryption key from Alice to Bob. In a distributed file system, anyone can access those files but only Bob can read them. If the PRE scheme employed is key-private, nobody can even tell who can access and read any file in the system.*"

Intuition 1 applies because of the word "*who*". To be complete, in this application, the IP address of Bob must also be anonymous (e.g., via anonymous routing) when accessing the distributed file system, regardless of concrete schemes employing Intuition 1 or Intuition 2.

1.3 Related works

Achieving key-private PRE has been acknowledged as a difficult task. In fact, many PRE schemes including [11, 7, 14, 20, 23] are not key-private, as shown in [6]. There exist a few secure schemes with key privacy up to now: (1) the CPA-secure schemes in [6, 28]; and (2) the CCA-secure in the random oracle model in [35] under the decisional Diffie-Hellman assumption (and hence different from ours). Another PRE scheme had

been claimed CCA-secure and key-private, in [36], but unfortunately its CCA security was recently broken as shown in [21].

There have also been a lot of works trying to achieve CCA security for PRE: we mention [23] achieving the weaker notion of Replayable CCA from LWE per the corrected proof in [31], as well as [38, 33, 34, 21] and [16] achieving full CCA respectively in bilinear groups and in groups where decisional Diffie-Hellman assumption holds. None of the schemes is key-private.

Additionally, some works have looked at identity-based PRE: see for instance [32] for a key-private scheme using pairings in the random oracle model, and [37] for a construction focused on collusion safeness, to list just some recent ones.

Nishimaki and Xagawa [28] consider notions of key privacy of Intuition 2 (see Table 1), and hence different from this work. The paper [28] do not consider CCA security.

Also differing from ours, Watanabe and Yoshino [39] tackle the task of key rotation by constructing a system using all-or-nothing transform.

1.4 Differences with the conference version [3]

Abridged parts of this paper have been presented in [3]. This paper improves those parts in several ways:

- We implement one scheme in Section 3.5 to demonstrate its efficiency, which is in orders of magnitude faster than proxy re-encryption schemes based on pairings. This shows that our scheme is very suitable for applications requiring high speed such as asymmetric key rotations discussed in Section 1.1.
- We clarify and correct many technical points, most significantly in Section 4 concerning CCA security. As a byproduct, we also provide in A a public key encryption scheme CCA-secure under the LWE assumption in the random oracle model.

We also realize that our schemes solve an open problem posted in previous work [6]. See the discussion in Section 2.3.

This paper does not contain results on the practical hardness of LWE assumption presented in [3]. Those results (with refined analysis) are treated independently in a separate paper [4].

2 Preliminaries and definitions

We use $\overset{\text{c}}{\approx}$ for computational indistinguishability.

LWE assumption. Succinctly, the assumption LWE(n, α, q) asserts that

$$(A, Ax + e) \overset{\text{c}}{\approx} (A, r)$$

in which

- $A \in \mathbb{Z}_q^{m\times n}$ and $r \in \mathbb{Z}_q^{m\times 1}$ are randomly chosen.
- $x \in \psi_{\alpha q}^{n\times 1}$, $e \in \psi_{\alpha q}^{m\times 1}$, and $Ax + e$ is computed over $\mathbb{Z}_q$. Moreover $\psi_{\alpha q}$ is the Gaussian distribution over the integers $\mathbb{Z}$, with mean 0 and deviation αq. (Originally, x is chosen randomly from $\mathbb{Z}_q^{n\times 1}$ in [30]. However, as showed in [5, 26], one can take $x \in \psi_{\alpha q}^{n\times 1}$ as we do here.)

More precisely, the LWE(n, α, q) assumption for n, α, q depending on security parameter λ asserts that the advantage

$$\mathbf{Adv}_{\mathcal{D}}^{\text{LWE}(n,\alpha,q)}(\lambda) = |\Pr[\mathcal{D}(A, Ax + e) \to 1] - \Pr[\mathcal{D}(A, r) \to 1]|$$

is negligible in λ for all poly-time distinguisher $\mathcal{D}$.

By a standard hybrid argument over columns of $X \in \psi_{\alpha q}^{n \times l}$, we have under the LWE assumption

$$(A, AX + E) \overset{c}{\approx} (A, R)$$

for random $R \in \mathbb{Z}_q^{m \times l}$ and Gaussian noise $E \in \psi_s^{m \times l}$. This fact will be used in our security proofs.

Syntax of "interactive" PRE. The scheme consists of algorithms (ParamsGen, KeyGen, ReKeyGen, Enc, ReEnc, Dec), described as follows.

ParamsGen(λ)	returns public parameters pp according to security parameter λ.
KeyGen(pp, λ)	returns public-secret key pairs (pk, sk).
ReKeyGen(pp, sk_i, sk_j)	returns the re-encryption key $rk_{i \to j}$.
Enc(pp, pk, m)	returns a ciphertext CT.
ReEnc(pp, $rk_{i \to j}$, CT_i)	transforms ciphertext CT_i of party i into a ciphertext that party j can decrypt.
Dec(pp, sk, CT)	recovers a message m.

Above, "interactive" means both secrets sk_i and sk_j are needed in creating the re-encryption key $rk_{i \to j}$. For readability, we omit some inputs that are publicly available in the above description, e.g., ReKeyGen and ReEnc may get additional inputs pk_i and pk_j.

2.1 CPA definitions

In following definitions, entities are indexed by integers. Namely, if there are N entities in the system, they are represented by the set $\{1, \ldots, N\}$ for their identities. In the real usage of our PRE scheme, one user can have many identities.

Definition 1 (CPA security of PRE, [6]). *Consider following interactions between an adversary $\mathcal{A}$ and a challenger C.*
Phase 1:

- *C generates public parameters $pp \leftarrow$ ParamsGen(λ) and gives them to $\mathcal{A}$.*
- *Uncorrupted key generation: C generates $(pk, sk) \leftarrow$ KeyGen(pp, λ) and gives $\mathcal{A}$ public key pk upon request. $\mathcal{A}$ can request many pk, and let Γ_H be the set of uncorrupted (honest) indexes.*
- *Corrupted key generation: In this process, C generates key pair $(pk, sk) \leftarrow$ KeyGen(pp, λ) and $\mathcal{A}$ is given (pk, sk). $\mathcal{A}$ can request many times, and let Γ_C be the set of corrupted indexes.*

Phase 2:

- *Re-encryption key generation: $\mathcal{A}$ submits a pair (i, j) to get the re-encryption key $rk_{i \to j} \leftarrow$ ReKeyGen(pp, sk_i, sk_j). All requests where $i \in \Gamma_H$, $j \in \Gamma_C$ are ignored.*
- *Re-encryption: $\mathcal{A}$ submits (i, j, C_i). The challenger generates the re-encryption key $rk_{i \to j} \leftarrow$ ReKeyGen (pp, sk_i, sk_j) if it is not yet created, and returns the re-encrypted ciphertext $C_j \leftarrow$ ReEnc(pp, $rk_{i \to j}, C_i$). All requests where $i \in \Gamma_H$, $j \in \Gamma_C$ are ignored.*
- *Challenge: $\mathcal{A}$ submits (i^*, m_0, m_1). The challenger then chooses random bit $b \in \{0, 1\}$, and then returns $C_{i^*} =$ Enc(pp, pk_{i^*}, m_b). This is done only once, and it is required that $i^* \in \Gamma_H$.*

$\mathcal{A}$ finally outputs $b' \in \{0,1\}$ as a guess of b. Define $\mathcal{A}$'s advantage as

$$\mathbf{Adv}^{cpa}_{\mathcal{A}}(\lambda) = \left|\Pr[b' = b] - \frac{1}{2}\right|.$$

The PRE scheme is CPA-secure if this advantage is negligible for all poly-time adversary $\mathcal{A}$.

Definition 2 (Re-encryption key privacy). *Consider following interactions of an adversary $\mathcal{A}$. Phase 1 is the same as in Definition 1.*
Phase 2:

- *Re-encryption key generation: On input $(i,\ j)$ where $i \in \Gamma_H \cup \Gamma_C$, $j \in \Gamma_H \cup \Gamma_C$, return the key $rk_{i\to j}$.*
- *Re-encryption queries (i, j, C_i): The challenger returns $C_j = \mathsf{ReEnc}(pp, rk_{i\to j}, C_i)$ as the re-encrypted ciphertext. If the re-encryption key $rk_{i\to j}$ was not generated yet, the challenger creates it.*
- *Challenge: This can only be queried once. On input (i^*, j^*), the challenger takes a bit b randomly, returns $rk^* = rk_{i^*\to j^*} = \mathsf{ReKeyGen}(pp, sk_{i^*}, sk_{j^*})$ if $b = 1$ or returns a random key rk^* in the key space if $b = 0$. The constraints are: (1) $rk_{i^*\to j^*}$ was not given out before, (2) there is no chain of re-encryption keys from j^* to any $k \in \Gamma_C$, (3) $j^* \in \Gamma_H$. Note there is no limitation on i^*, namely $i^* \in \Gamma_H \cup \Gamma_C$.*

Eventually, $\mathcal{A}$ outputs a bit b', and its advantage is defined as

$$\mathbf{Adv}^{kp-cpa}_{\mathcal{A}}(\lambda) = \left|\Pr[b' = b] - \frac{1}{2}\right|.$$

The PRE scheme is key-private if this advantage is negligible for all poly-time adversary $\mathcal{A}$.

The above definition differs from its counterpart in [6] in some ways. The condition (3) with $j^* \in \Gamma_H$ while i^* is free in $\Gamma_H \cup \Gamma_C$ means that, regardless delegators, *honest* delegatees are enough to provide key privacy. This is stronger than [6] which requires both $i^*, j^* \in \Gamma_H$.

Our definition also removes a "collusion-safe" condition implicitly incorporated in that of [6] by requiring condition (2). This is for good because we think that collusion-safeness and key privacy are separate issues in applications.

2.2 CCA definitions

We will consider CCA-secure PRE schemes with one hop, namely each ciphertext is re-encrypted only once. The syntax is different from that of the CPA-secure schemes in having two separate decryption algorithms:

$\mathsf{Dec}(pp, sk, CT)$	decrypting ciphertexts CT outputted by Enc
$\mathsf{DecR}(pp, sk, CT', pi)$	decrypting ciphertexts CT' outputted by ReEnc, where pi is some previously published information

The auxiliary published information pi will be the used public and re-encryption keys in our construction in Section 4.

Definition 3 (CCA security of ciphertexts outputted by Enc). *This is the same as Definition 1, with following additional decryption queries by adversary $\mathcal{A}$ in Phase 2:*

- *$\mathcal{A}$ repeatedly submits (pk_i, C) to the challenger, who in turn returns the decryption $\mathsf{Dec}(pp, sk_i, C)$. The trivial constraints here are: $i \in \Gamma_H$ and $(pk_i, C) \neq (pk_{i^*}, C_{i^*})$.*

– *$\mathcal{A}$ repeatedly submits (pi, C) where $pi = (pk_i, pk_j, rk_{i\to j})$ to the challenger, who returns the decryption* $\mathsf{DecR}(pp, sk_j, C, pi)$. *The necessary constraints in this case are: $j \in \Gamma_H$ and*

$$(pi, C) \neq ((pk_{i^*}, pk_j, rk_{i^*\to j}), \mathsf{ReEnc}(pp, rk_{i^*\to j}, C_{i^*}))$$

where $rk_{i^*\to j}$ is some re-encryption key from i^* to j. In more details,

- *If $pi = (pk_{i^*}, pk_j, rk_{i^*\to j})$, then $C \neq \mathsf{ReEnc}(pp, rk_{i^*\to j}, C_{i^*})$, namely C is not a ciphertext returned by* ReEnc *with inputs $pp, rk_{i^*\to j}, C_{i^*}$. Also, $rk_{i^*\to j}$ in pi tells us that $\mathcal{A}$ holds the re-encryption key.*
- *If $pi \neq (pk_{i^*}, pk_j, rk_{i^*\to j})$, then C can be $\mathsf{ReEnc}(pp, rk_{i^*\to j}, C_{i^*})$ obtained by $\mathcal{A}$ via the re-encryption oracle.*

Define $\mathcal{A}$'s advantage as

$$\mathbf{Adv}^{\mathsf{Enc},cca}_{\mathcal{A}}(\lambda) = \left|\Pr[b' = b] - \frac{1}{2}\right|.$$

The PRE scheme is CCA-secure with respect to Enc *if this advantage is negligible for all poly-time adversary $\mathcal{A}$.*

Definition 4 (CCA security of ciphertexts outputted by ReEnc**).** *This security notion is the same as Definition 1, with following change in challenge in Phase 2:*

– *Challenge: $\mathcal{A}$ submits pk_{i^*} and $pk_{i_{\mathcal{A}}}$ for $i^* \in \Gamma_H$, $i_{\mathcal{A}} \in \Gamma_H \cup \Gamma_C$, and two messages m_0, m_1. The challenger takes a random bit $b \in \{0, 1\}$ and returns the challenge ciphertext $C_{i^*} = \mathsf{ReEnc}(pp, rk_{i_{\mathcal{A}}\to i^*}, CT_b)$ to $\mathcal{A}$, where $CT_b = \mathsf{Enc}(pp, pk_{i_{\mathcal{A}}}, m_b)$. It is worth noting that $rk_{i_{\mathcal{A}}\to i^*}$ may be known to $\mathcal{A}$ via a query for re-encryption key.*

Also, the decryption queries are added in Phase 2 as follows:

– *$\mathcal{A}$ repeatedly submits (pk_i, C) to the challenger, who in turn returns the decryption $\mathsf{Dec}(pp, sk_i, C)$. The trivial constraints here is $i \in \Gamma_H$.*
– *$\mathcal{A}$ repeatedly submits (pi, C) where $pi = (pk_i, pk_j, rk_{i\to j})$ to the challenger, who returns the decryption $\mathsf{DecR}(pp, sk_j, C, pi)$. The necessary constraints in this case are: $i \in \Gamma_H$ and $(pi, C) \neq (pk_{i_{\mathcal{A}}}, pk_{i^*}, rk_{i_{\mathcal{A}}\to i^*}, C_{i^*})$.*

Define $\mathcal{A}$'s advantage as

$$\mathbf{Adv}^{\mathsf{ReEnc},cca}_{\mathcal{A}}(\lambda) = \left|\Pr[b' = b] - \frac{1}{2}\right|.$$

The PRE scheme is CCA-secure with respect to ReEnc *if this advantage is negligible for all poly-time adversary $\mathcal{A}$.*

Definition 5 (Re-encryption key privacy, CCA setting). *Phase 1, Phase 2 are the same as in Definition 2, except that in Phase 2, $\mathcal{A}$ can access to additional oracles for decryption:*

– *$\mathcal{A}$ submits query of form (pk_i, C) for $i \in \Gamma_H$. The challenger returns the decryption $\mathsf{Dec}(pp, sk_i, C)$.*
– *$\mathcal{A}$ submits query of form (pi, C) where $pi = (pk_i, pk_j, rk_{i\to j})$. The challenger returns the decryption of re-encrypted ciphertext $\mathsf{DecR}(pp, sk_j, C, pi)$. The query cannot simultaneously satisfy: (1) $pk_j = pk_{j^*}$ (the challenge index), and (2) C is a re-encrypted ciphertext computed via the challenge rk^*, which is honest $rk_{i^*\to j^*}$ or random, formed as in Definition 2, namely $C = \mathsf{ReEnc}(pp, pk_{i^*}, pk_{j^*}, rk^*, C')$ for some adversarial C'.*

Define $\mathcal{A}$'s advantage as

$$\mathbf{Adv}^{kp-cca}_{\mathcal{A}}(\lambda) = \left|\Pr[b' = b] - \frac{1}{2}\right|.$$

The PRE scheme is key-private in CCA sense if this advantage is negligible for all poly-time adversary $\mathcal{A}$.

2.3 Discussion: possible multiple keys per entity pair

The definition of key privacy in [6] allows only one re-encryption key per pair of entities. Moreover, the PRE scheme given in [6] is insecure in case two (or more) re-encryption keys exist per pair, as discussed in [6, Remark 3.5, Eprint version]. The authors of [6] asked for definitions where multiple re-encryption keys are permitted per pair, and that is exactly captured by our Definitions 2 (CPA case) and 5 (CCA case). Our PRE schemes meet these notions, and hence are still secure when many re-encryption keys are shared between two entities.

3 Our key-private, CPA-secure PRE

3.1 Description

Let us first recall some notions, originally used in fully homomorphic encryption.

Functions Bits(·) and Power2(·). The functions $\mathsf{Power2}(\cdot)$ and $\mathsf{Bits}(\cdot)$ are described as follows. Let $v \in \mathbb{Z}_q^n$ and $\kappa = \lceil \lg q \rceil$ where $\lg(\cdot)$ is logarithm of base 2. Then there are bit vectors $v_i \in \{0,1\}^n$ such that $v = \sum_{i=0}^{\kappa-1} 2^i v_i$. Then

$$\mathsf{Bits}(v) = [v_0 | \cdots | v_{\kappa-1}] \in \{0,1\}^{1 \times n\kappa}.$$

Let $W = [W_1 | \cdots | W_l] \in \mathbb{Z}_q^{n \times l}$ where W_i are column vectors. Then

$$\mathsf{Power2}(W) = \begin{bmatrix} W_1 & \cdots & W_l \\ 2W_1 & \cdots & 2W_l \\ \vdots & & \vdots \\ 2^{\kappa-1}W_1 & \cdots & 2^{\kappa-1}W_l \end{bmatrix} \in \mathbb{Z}_q^{n\kappa \times l}.$$

It is easy to check that

$$\mathsf{Bits}(v)\mathsf{Power2}(W) = vW \in \mathbb{Z}_q^{1 \times l}.$$

This equality will be useful in checking the correctness of our schemes. Following PRE scheme is based on the public key encryption scheme given in [24]. In particular, the first four algorithms are basically the same as those in [24], while the last two are ours.

Parameters generation $\mathsf{ParamsGen}^{cpa}(\lambda)$**:** Choose positive integers q, n, and take matrix $A \in \mathbb{Z}_q^{n \times n}$ randomly. Return $pp = (q, n, A)$, which is the input to all algorithms below.

Key generation $\mathsf{KeyGen}^{cpa}(pp, \lambda)$**:** Let $s = \alpha q$ for $0 < \alpha < 1$. Take Gaussian noise matrices $R, S \in \psi_s^{n \times l}$. The public key is $pk = P$ for $P = R - AS \in \mathbb{Z}_q^{n \times l}$, and the secret key is $sk = S$. Here, l is the message length in bits, while n is the key dimension. Return (pk, sk).

Encryption $\mathsf{Enc}_{pk}^{cpa}(m; \mathsf{randomness})$**:** To encrypt $m \in \{0,1\}^l$, use $\mathsf{randomness}$ to take Gaussian noise vectors $e_1, e_2 \in \psi_s^{1 \times n}$, and $e_3 \in \psi_s^{1 \times l}$, and return ciphertext $c = (c_1, c_2) \in \mathbb{Z}_q^{1 \times (n+l)}$ where

$$c_1 = e_1 A + e_2 \in \mathbb{Z}_q^{1 \times n}, c_2 = e_1 P + e_3 + m \cdot \left\lfloor \frac{q}{2} \right\rfloor \in \mathbb{Z}_q^{1 \times l}.$$

Decryption $\mathsf{Dec}_{sk}^{cpa}(c)$**:** To decrypt $c = (c_1, c_2) \in \mathbb{Z}_q^{1 \times (n+l)}$ by secret key S, compute $\overline{m} = c_1 S + c_2 \in \mathbb{Z}_q^l$. Let $\overline{m} = (\overline{m}_1, \ldots, \overline{m}_l)$. If $\overline{m}_i \in [-\lfloor \frac{q}{4} \rfloor, \lfloor \frac{q}{4} \rfloor] \subset \mathbb{Z}_q$, let $m_i = 0$; otherwise $m_i = 1$.

Proxy key generation $\mathsf{ReKeyGen}(pk_A, sk_A, pk_B, sk_B)$**:** Alice with keys ($pk_A = P_A$, $sk_A = S_A$) and Bob with keys ($pk_B = P_B$, $sk_B = S_B$) want to set up proxy key $rk_{A\to B}$. The proxy key consists of P_B and

$$rk_{A\to B} = \begin{bmatrix} X & -XS_B + E + \mathsf{Power2}(S_A) \\ \mathbf{0}_{l\times n} & \mathbf{I}_{l\times l} \end{bmatrix}$$

in which matrices $X \in \mathbb{Z}_q^{n\kappa\times n}$ ($\kappa = \lceil \lg q \rceil$) is chosen uniformly random and noise matrix E is chosen from $\psi_s^{n\kappa\times l}$.

Re-encryption $\mathsf{ReEnc}^{cpa}(pk_B, rk_{A\to B}, (c_1, c_2))$**:** Parse the proxy key as P_B and $rk_{A\to B}$. To transform Alice's ciphertext $(c_1, c_2) \in \mathbb{Z}_q^{1\times(n+l)}$ into Bob's ciphertext, return

$$\underbrace{f_1[A|P_B] + [f_2|f_3]}_{\text{re-randomization part}} + [\mathsf{Bits}(c_1)|c_2] \cdot rk_{A\to B} \in \mathbb{Z}_q^{1\times(n+l)}$$

in which $f_1, f_2 \in \psi_s^{1\times n}$, and $f_3 \in \psi_s^{1\times l}$ are chosen by the proxy.

3.2 Intuition on key privacy

The re-encryption key contains following information on Bob's secret $sk_B = S_B$: $(A, P_B = R_B - AS_B)$ and $(X, -XS_B + E)$. These together can be written in LWE form

$$\begin{bmatrix} A \\ X \end{bmatrix}, -\begin{bmatrix} A \\ X \end{bmatrix} S_B + \begin{bmatrix} R_B \\ E \end{bmatrix}$$

and thus $rk_{A\to B}$ is pseudo-random if S_B is kept secret, namely if Bob is not corrupted. This also implies that public key P_B is pseudo-random and unrelated to Bob, which is the recipient-anonymous property of [24].

However, the above is still insufficient! If re-encryption is deterministic, for example using the second-half formula $[\mathsf{Bits}(c_1)|c_2] \cdot rk_{A\to B}$ (and P_B is unused), there is an attack on key privacy as follows:

1. Adversary $\mathcal{A}$ creates ciphertext (c_1, c_2) using Alice's public key pk_A.
2. $\mathcal{A}$ asks for challenge rk^* between Alice and Bob, which is either honestly generated $rk_{A\to B}$ or random.
3. $\mathcal{A}$ asks for re-encryption (c_1', c_2') of (c_1, c_2) from Alice to Bob, namely $(c_1', c_2') = [\mathsf{Bits}(c_1)|c_2] \cdot rk_{A\to B}$.
4. $\mathcal{A}$ checks whether $(c_1', c_2') = [\mathsf{Bits}(c_1)|c_2] \cdot rk^*$.
   If the comparison holds true, $\mathcal{A}$ decides that rk^* is the re-encryption key between Alice and Bob. The idea of this attack is originated in [6, Lemma 2.7], and works well under the condition that re-encryption is deterministic.

To deal with the attack, we add the term $f_1[A|P_B]+[f_2|f_3]$ into the re-encryption. This is exactly an encryption of zero vector under the public key of Bob, so that decryption by Bob will not be affected by this term. Thus re-encryption is randomized, and we succeed in proving that the PRE scheme is key-private.

3.3 Some useful properties

Multi-hop property. A ciphertext $(c_1, c_2) \in \mathbb{Z}_q^{1\times(n+l)}$, after re-encryption, is changed to a ciphertext in $\mathbb{Z}_q^{1\times(n+l)}$. Namely, re-encryption does not change the format of ciphertexts. Original ciphertexts and transformed ciphertexts are decrypted by the same decryption algorithm (with different secret keys, of course). Thus our scheme is multi-hop, namely a ciphertext can be re-encrypted many times as long as the incurred noise is kept small enough.

Uni-directional property. This property ensures that Alice and proxy together cannot decrypted Bob's ciphertexts. This is intuitively true because Alice and proxy can only get $(X, -XS_B + E)$ where X and E are chosen by Bob. The tuple is pseudo-random under the LWE assumption, so the information is useless.

3.4 Correctness

First, we check that normal ciphertext $c_1 = e_1A + e_2$, $c_2 = e_1P + e_3 + m \cdot \lfloor \frac{q}{2} \rfloor$ can be decrypted by secret S via the formula $c_1S + c_2$. Indeed,

$$\begin{aligned} c_1S + c_2 &= [c_1|c_2] \begin{bmatrix} S \\ \mathbf{I}_{l\times l} \end{bmatrix} \\ &= \left(e_1[A|P] + [e_2|e_3] + [\mathbf{0}|m \cdot \lfloor \frac{q}{2} \rfloor] \right) \begin{bmatrix} S \\ \mathbf{I}_{l\times l} \end{bmatrix} \\ &= e_1(AS + P) + (e_2S + e_3) + m \cdot \left\lfloor \frac{q}{2} \right\rfloor \\ &= e_1R + e_2S + e_3 + m \cdot \left\lfloor \frac{q}{2} \right\rfloor \end{aligned}$$

will yield m if the noise $e_1R + e_2S + e_3$ is small enough. Second, we check a transformed ciphertext can be decrypted by Bob. Namely, decryption of ciphertext $f_1[A|P_B] + [f_2|f_3] + [\mathsf{Bits}(c_1)|c_2] \cdot Q$ by Bob's secret S_B is the same as Alice's decryption on (c_1, c_2) with S_A. Indeed,

$$\begin{aligned} (f_1[A|P_B] + [f_2|f_3] + [\mathsf{Bits}(c_1)|c_2] \cdot Q) \begin{bmatrix} S_B \\ \mathbf{I}_{l\times l} \end{bmatrix} &= f_1(AS_B + P_B) + f_2S_B + f_3 \\ &\quad + [\mathsf{Bits}(c_1)|c_2] \begin{bmatrix} E + \mathsf{Power2}(S_A) \\ \mathbf{I}_{l\times l} \end{bmatrix} \\ &= f_1R_B + f_2S_B + f_3 + \mathsf{Bits}(c_1)\,(E + \mathsf{Power2}(S_A)) + c_2 \\ &= \underbrace{f_1R_B + f_2S_B + f_3 + \mathsf{Bits}(c_1)E}_{\text{noise}} + c_1S_A + c_2 \end{aligned}$$

will yield Alice's decryption on (c_1, c_2) since the incurred noise is sufficiently small. Technical details are provided below.

We will use following lemmas, whose proofs can be derived from [8, 9]. Below $|\cdot|$ stands for either the Euclidean norm of a vector or the absolute value; $\langle \cdot, \cdot \rangle$ for inner product. Writing $|\psi_s^n|$ is a short hand for taking a vector from the distribution and computing its norm.

Lemma 1. *Let $c \geq 1$ and $C = c \cdot \exp(\frac{1-c^2}{2})$. Then for any real $s > 0$ and any integer $n \geq 1$, we have*

$$\Pr\left[|\psi_s^n| \geq \frac{c \cdot s\sqrt{n}}{\sqrt{2\pi}} \right] \leq C^n.$$

Lemma 2. *For any real $s > 0$ and $T > 0$, and any $x \in \mathbb{R}^n$, we have*

$$\Pr\left[|\langle x, \psi_s^n \rangle| \geq T\, s|x| \right] < 2\exp(-\pi T^2).$$

Theorem 1 (Correctness). *Let q be the modulus as in the scheme. For correctness of our PRE, we need*

$$s \leq \frac{1}{\sqrt{A}} \left(\sqrt{\frac{q}{4} + \left(\frac{B}{2\sqrt{A}} \right)^2} - \frac{B}{2\sqrt{A}} \right)$$

in which A and B are given in equation (2) in the proof.

Proof. It suffices to check correctness of the transformed ciphertexts, since the noise is bigger than that in original ones. Continuing the text above, let us now check the decryption of transformed ciphertexts of (c_1, c_2), which is

$$\begin{aligned} & f_1 R_B + f_2 S_B + f_3 + \mathsf{Bits}(c_1)E + c_1 S_A + c_2 \\ &= \underbrace{f_1 R_B + f_2 S_B + f_3 + \mathsf{Bits}(c_1)E}_{\text{noise in } \underline{\text{one}} \text{ re-encryption}} + \underbrace{e_1 R + e_2 S + e_3}_{\text{original noise}} + m \cdot \left\lfloor \frac{q}{2} \right\rfloor. \end{aligned}$$

Generally, the noise after h hops is thus can be written as

$$\sum_{i=1}^{h} \left(f_1^{(i)} R_B + f_2^{(i)} S_B + f_3^{(i)} + \mathsf{Bits}(c_1) E^{(i)} \right) + e_1 R + e_2 S + e_3 \in \mathbb{Z}_q^{1 \times l}.$$

Suppose $\mathsf{Bits}(c_1)$ contains all 1's, namely $\mathsf{Bits}(c_1) = \mathbf{1}_{1 \times n\kappa}$. Each component in $\mathbb{Z}_q$ of the above noise vector can be written as the inner product of two vectors of form

$$\begin{aligned} e &= \left(f_1^{(1)}, f_2^{(1)}, f_3^{(1)} \quad , \ldots, f_1^{(h)}, f_2^{(h)}, f_3^{(h)}, e^{(1)}, \ldots, e^{(h)}, \;\; e_1, e_2, e_3 \right) \\ x &= \left(r_B^{(1)}, s_B^{(1)}, \mathbf{0}\mathbf{1}\mathbf{0}_{1 \times l}, \ldots, r_B^{(h)}, s_B^{(h)}, \mathbf{0}\mathbf{1}\mathbf{0}_{1 \times l}, \underbrace{\mathbf{1}_{1 \times n\kappa}}_{1 \le i \le h}, r, r', \mathbf{0}\mathbf{1}\mathbf{0}_{1 \times l} \right) \end{aligned}$$

where, for all $1 \le i \le h$, the notations are as follows.

- Vectors $f_1^{(i)}, f_2^{(i)} \in \psi_s^{1 \times n}$, and $f_3^{(i)} \in \psi_s^{1 \times l}$. Vectors $e^{(i)} \in \psi_s^{1 \times n\kappa}$ represents one column in matrix $E^{(i)}$. Vectors $e_1 \in \psi_s^{1 \times n}, e_2 \in \psi_s^{1 \times n}$, and $e_3 \in \psi_s^{1 \times l}$ are the noises in the original ciphertext.
- Vectors $r_B^{(i)}, s_B^{(i)} \in \psi_s^{1 \times n}$, and $\mathbf{0}\mathbf{1}\mathbf{0}_{1 \times l}$ stands for a vector of length l with all 0's except one 1; $\mathbf{1}_{1 \times n\kappa}$ for a vector of length $n\kappa$ with all 1's. Vectors $r, r' \in \psi_s^{1 \times n}$ represent corresponding columns in matrices R, S.

We have

$$\begin{aligned} e &\in \mathbb{Z}_s^{1 \times \left(\sum_{i=1}^{h} (2n + l + n\kappa) + 2n + l \right)} \\ \|x\| &\le \|(r_2^{(1)}, s_2^{(1)}, \ldots, r_2^{(h)}, s_2^{(h)}, r, r')\| + \sqrt{\kappa h n + h + 1} \end{aligned}$$

where $(r_2^{(1)}, s_2^{(1)}, \ldots, r_2^{(h)}, s_2^{(h)}, r, r') \in \psi_s^{1 \times (2hn + 2n)}$. Applying Lemma 1 for vector of length $2hn + 2n$, with high probability of

$$1 - C^{2hn+2n} \ge 1 - 2^{-40} \text{ (even if } h = 1)$$

we have

$$\|x\| \le \frac{c \cdot s \sqrt{2hn + 2n}}{\sqrt{2\pi}} + \sqrt{\kappa h n + h + 1}.$$

We now use Lemma 2 with vectors x and e. Let ρ be the error per message symbol in decryption, we set $2\exp(-\pi T^2) = \rho$, so $T = \sqrt{\ln(2/\rho)}/\sqrt{\pi}$. For correctness, we need $T \cdot s \cdot \|x\| \le q/4$, which holds true provided that

$$\frac{\sqrt{\ln(2/\rho)}}{\sqrt{\pi}} \cdot s \cdot \left(\frac{cs\sqrt{2n(h+1)}}{\sqrt{2\pi}} + \sqrt{(\kappa n + 1)h + 1} \right) \le \frac{q}{4}$$

which can be shortly written in form $As^2 + Bs - \frac{q}{4} \le 0$. Therefore,

$$s \le \frac{1}{\sqrt{A}} \left(\sqrt{\frac{q}{4} + \left(\frac{B}{2\sqrt{A}} \right)^2} - \frac{B}{2\sqrt{A}} \right) \tag{1}$$

in which

$$A = \frac{\sqrt{\ln(2/\rho)} \cdot c \cdot \sqrt{n(h+1)}}{\pi}, B = \sqrt{\frac{\ln(2/\rho)}{\pi}} \cdot \sqrt{(\kappa n + 1)h + 1} \tag{2}$$

which will be used in Section 3.5 below for testing implementations.

3.5 Testing implementations

We use parameters described in Table 2 in experiments in this section. When $q = 16381$, $n = 450$, $s = 3$, the bit security of (search) LWE is about 135 due to the cryptanalysis in [3, 4].

Table 2. Parameters in experiments. The constant c is for the use of Lemma 1. The number of hop h in this table is theoretically chosen in formula (2) to help fixing the noise deviation s given in formula (1), while ρ is the error per bit in decryption.

q	$\kappa = \lceil \lg q \rceil$	n	c
16381	14	450	1.13

ρ	h in (2)	s by (1)	bit sec. (via [3, 4])
1/200	10	≤ 5.08	≈ 161
2^{-128}	4	≤ 3.05	≈ 135

Experimental number of hops and ρ. Recall that ρ is the error per bit in decrypted message. We take two choices of this parameter for experiment:

- $\rho = \frac{1}{200}$, namely there is a certain error per bit, so that some error correcting code (ECC) must be needed to recover the message. The concrete ECC will be discussed below.
- $\rho = 2^{-128}$, namely the error per bit is negligible, so that no ECC is necessary.

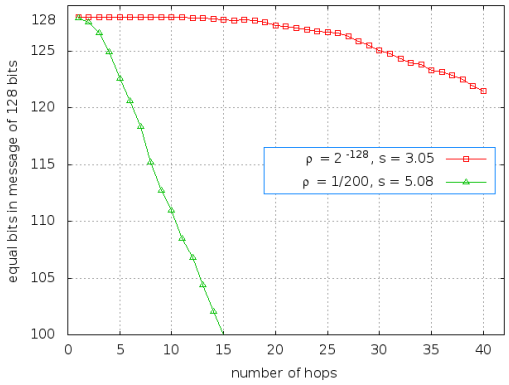

Fig. 1. Correctness of 128-bit message when the error per bit $\rho = 2^{-128}$ or $\rho = 1/200$. The data is produced via 40 hops of re-encryption, with decryption after each hop to count the correct bits. The whole process is independently repeated 50 times to take the average reported in the graph.

The correctness of 128-bit decrypted messages is reported in Figure 1. As expected, correctly recovered bits decrease when the number of hops increases due to the noises added in re-encryption. Nevertheless, some notes are as follows.

– When $\rho = \frac{1}{200}$, and we want $h = 10$, the we need an ECC being able to correct about $128 - 110 = 18$ bits. Therefore, the 18-error-correcting-code BCH(255, 131, 18) can be used, in which 255 is the code length in bit, 131 is the data length in bit. Since we just need to encode 128 bits, in fact $131 - 128 = 3$ bits are redundant. In this case, the message length l is the code length, $l = 255$.
– When $\rho = 2^{-128}$, one can see in Figure 1 that correctness over 128 bits are ensured until 10 hops, which is larger than the theoretical value of $h = 4$ set in Table 2. This is due to the fact that the proof of Theorem 1 deals with maximum noises which may not happen frequently. In this case, the message length $l = 128$.

Speed. We implement the pairing-based scheme in [6] using the PBC 0.5.13 library, and our CPA-secure scheme using the Eigen 3.1.4 library [17] for linear algebra to compare the performance. All experiments are over a laptop (Intel 2.0GHz CPU, 8GB RAM) running Ubuntu 12.04 LTS. The C compiler is `g++` 4.6.3 with compiling options `-O3 -funroll-loops -ffast-math` when running the code for our scheme and additionally `-lbpc -lgmp -lm` for [6].

Neglecting time for key generation in Table 3, we remark that the running speeds of Enc, ReEnc, and Dec in our proposal beats the corresponding algorithms of [6] at a margin of multiples or even hundred times. Indeed, at approximately the same security level of 112-bit, ours are 280.75 times faster in Enc, 94.98 times faster in ReEnc, and at least 146.44 times faster in decryption. These speedups are due to the fact that linear algebra operations are faster than exponentiations and pairings.

Table 3. Performance comparison between [6] and ours. Running times are given in milliseconds, averaged over 100 executions of each algorithm. Type a and a1 parings are of around 80- and 112- bit security correspondingly, and type a is the fastest pairing in the library. Dec1 and Dec2 in[6] are the decryption algorithms for first-level (not re-encryptable) and second-level (re-encryptable) ciphertexts.

The CPA-secure scheme in [6]

Pairing type	One pairing cost	KeyGen	Enc	ReKeyGen	ReEnc	Dec2	Dec1
a	5.99	6.24	7.55	11.73	16.44	7.38	0.47
a1	92.5	60	112.3	201.9	352.4	170.2	6.59

Our CPA-secure scheme, noise deviation s = 3.05, LWE's bit security ≈ 135

N/A	N/A	26	0.4	380	3.71	0.045	0.045

Key and ciphertext sizes. When $s = 3.05$, we can take $l = 128$ as no ECC is necessary as discussed above. The public key is $P \in \mathbb{Z}_{16381}^{450\times 128}$, so takes storage of about

$$\frac{450 \cdot 128 \cdot \lg 16381}{10^3 \cdot 8} \approx 100.8 \text{ (kilobytes)}.$$

The secret key is also of the same size. A ciphertext is in $\mathbb{Z}_{16381}^{450+128}$, so has size $(450 + 128)\lg 16381$ (bits), which is around 1.01 kilobytes. The re-encryption matrix $rk_{A\to B}$ contains matrices in $\mathbb{Z}_{16381}^{450\cdot 14\times 450}$ and $\mathbb{Z}_{16381}^{450\cdot 14\times 128}$, so requires storage of

$$\frac{(450 \cdot 14 \cdot 450 + 450 \cdot 14 \cdot 128)\lg 16381}{10^6 \cdot 8} \approx 6.37 \text{ (megabytes)}.$$

The summary of these computations is in Table 4.

Table 4. Sizes computed with parameters in Table 2. Message length $|m| = l = 128$.

$pk = P$	$sk = S$	CT	$\|CT\|/\|m\|$	rk
100.8 (kilobytes)	100.8 (kilobytes)	1.01 kilobytes	63.21	6.37 (megabytes)

3.6 Security theorems

Theorem 2 (CPA security). *Under the LWE(n, α, q) assumption, the above PRE scheme is CPA-secure. Specifically, for a poly-time adversary $\mathcal{A}$, there exists a poly-time distinguisher $\mathcal{D}$ such that*

$$\mathbf{Adv}^{cpa}_{\mathcal{A}}(\lambda) \leq (NQ_{rk}l + NQ_{re} + 1) \cdot \mathbf{Adv}^{\mathrm{LWE}(n,\alpha,q)}_{\mathcal{D}}(\lambda)$$

where Q_{rk} and Q_{re} are correspondingly the number of re-encryption key queries and re-encryption queries, N is the number of honest entities, and l is the message length.

Proof. Consider an adversary $\mathcal{A}$ against the PRE. Let Game_0 be the interactions between $\mathcal{A}$ and a challenger as in Definition 1. In this initial game, $pp = (q, n, A)$, Γ_H is the set of honest entities, Γ_C is the set of corrupted entities. A key pair (P_i, S_i) satisfying $P_i = R_i - AS_i$ for Gaussian noise matrix R_i, S_i. The re-encryption key from party i to party j is (P_j, Q_{ij}) in which

$$Q_{ij} = \begin{bmatrix} X_{ij} & -X_{ij}S_j + E_{ij} + \mathsf{Power2}(S_i) \\ \mathbf{0}_{l\times n} & \mathbf{I}_{l\times l} \end{bmatrix}$$

in which X_{ij}, E_{ij} are generated by party j. The challenge ciphertext related to party i^* is (c_1^*, c_2^*) where

$$c_1^* = e_1^* A + e_2^* \text{ and } c_2^* = e_1^* P^* + e_3^* + m_b \cdot \lfloor \frac{q}{2} \rfloor$$

in which $b \in \{0, 1\}$ is the challenge bit, (e_1^*, e_2^*, e_3^*) are Gaussian noise, and P^* is the challenge public key.

For notational convenience, let $\Gamma_H = \{1, \ldots, N\}$. Following $\mathrm{Game}_{1\leq k\leq N}$ corresponds to honest party $k \in \Gamma_H$. Game_k is identical to Game_{k-1}, except the following changes:

- P_k ($= R_k - AS_k$ in Game_{k-1}) is changed into a random matrix P'_k.
- Re-encryption key query (i, k): return $rk_{i\to k} = (P'_k, Q_{ik})$ in which

$$Q_{ik} = \begin{bmatrix} X_{ik} & R_{ik} \\ \mathbf{0}_{l\times n} & \mathbf{I}_{l\times l} \end{bmatrix}$$

  where R_{ik} is freshly random.
- Re-encryption query (i, k, C_i): return a random vector in $\mathbb{Z}_q^{1\times(n+l)}$ to $\mathcal{A}$.

$\mathrm{Game}_{\mathrm{final}}$ is identical to Game_N except that challenge ciphertext is

$$c_1^* = r_1^* \text{ and } c_2^* = r_2^* + m_b \cdot \left\lfloor \frac{q}{2} \right\rfloor$$

in which r_1^*, r_2^* are freshly random vectors over $\mathbb{Z}_q$ of proper lengths. By this change, the challenge bit b is information-theoretically hidden from $\mathcal{A}$, so $\Pr[b' = b] = \frac{1}{2}$, and hence $\mathcal{A}$'s advantage in Game_1 is 0.

We now need to prove that the games are indistinguishable from the view of $\mathcal{A}$, under the LWE assumption. The change from Game_{k-1} to Game_k involves turning

$$P_k = R_k - AS_k, R_{ik} = E_{ik} - X_{ik}S_k$$

into random matrices. This is ensured by LWE with secret S_k of the form

$$\begin{bmatrix} A \\ \vdots \\ X_{ik} \\ \vdots \end{bmatrix}_i, - \begin{bmatrix} A \\ \vdots \\ X_{ik} \\ \vdots \end{bmatrix}_i S_k + \begin{bmatrix} R_k \\ \vdots \\ X_{ik} \\ \vdots \end{bmatrix}_i$$

where index i corresponds to all re-encryption key queries (i, k). Here we rely on the LWE(n, α, q) assumption over l column vectors of S_k, illustrating the loss factor $Q_{rk}l$ in reduction. The change also relies on the fact that $f_1[A|P'_k] + [f_2|f_3]$ is pseudo-random under LWE(n, α, q) when dealing with re-encryption queries, illustrating the loss factor Q_{re} in reduction.

The change from Game_N to $\text{Game}_{\text{final}}$ involves turning $e_1^*A + e_2^*$ and $c_2^* = e_1^*P^* + e_3^*$ into random vectors. This is ensured by LWE with secret $(e_1^*)^T$ (the transpose of e_1^*) of the form

$$[A|P^*]^T, (e_1^*[A|P^*] + [e_2^*|e_3^*])^T$$

where P^* is random by one of previous games. The assumption parameter is also LWE(n, α, q), illustrating the final loss factor 1 in the theorem statement.

Theorem 3 (Key privacy). *Under the LWE(n, α, q) assumption, the above PRE scheme is key-private. Specifically, for a poly-time adversary $\mathcal{A}$, there exists a poly-time distinguisher $\mathcal{D}$ such that*

$$\mathbf{Adv}_{\mathcal{A}}^{kp-cpa}(\lambda) \leq N(Q_{rk}l + Q_{re}) \cdot \mathbf{Adv}_{\mathcal{D}}^{\text{LWE}(n,\alpha,q)}(\lambda)$$

where Q_{rk} and Q_{re} are correspondingly the number of re-encryption key queries and re-encryption queries, N is the number of honest entities, and l is the message length.

Proof. Let Game_0 be the attack game as in Definition 2. In the game, the challenge re-encryption key from i^* to j^* is $(P_{j^*}, Q_{i^*j^*})$ where

$$Q_{i^*j^*} = \begin{bmatrix} X_{i^*j^*} & -X_{i^*j^*}S_{j^*} + E_{i^*j^*} + \mathsf{Power2}(S_{i^*}) \\ \mathbf{0}_{l\times n} & \mathbf{I}_{l\times l} \end{bmatrix}$$

for $j^* \in \Gamma_H$. For notational convenience, let $\Gamma_H = \{1, \ldots, N\}$. Following $\text{Game}_{1\leq k\leq N}$ corresponds to honest party $k \in \Gamma_H$. Game_k is identical to Game_{k-1}, except that re-encryption key from i to k for any $i \in \Gamma_H \cup \Gamma_C$ is set to

$$\left(P_k, \begin{bmatrix} X_{ik} & R_{ik} \\ \mathbf{0}_{l\times n} & \mathbf{I}_{l\times l} \end{bmatrix}\right)$$

where P_k, R_{ik} are freshly random matrices over $\mathbb{Z}_q$ of proper sizes. Since $j^* \in \Gamma_H$ by the constraint in definition, the challenge re-encryption key from i^* to j^* is changed in Game_{j^*} into

$$\left(P_{j^*}, \begin{bmatrix} X_{i^*j^*} & R_{i^*j^*} \\ \mathbf{0}_{l\times n} & \mathbf{I}_{l\times l} \end{bmatrix}\right)$$

for random matrices $P_{j^*}, R_{i^*j^*}$.

Also, in each $\text{Game}_{1\le k\le N}$, re-encryption queries (i, k, C_i) is answered by random vectors of length $\mathbb{Z}_q^{1\times(n+l)}$ for all index i. Thus in Game_N, the challenge that $\mathcal{A}$ gets is random in both cases $b = 0$ and $b = 1$, and hence $\mathcal{A}$'s advantage is 0.

We now show that all games are indistinguishable under the LWE assumption. Specifically, the games Game_k and Game_{k-1} are indistinguishable under LWE with secret S_k of the form

$$\begin{bmatrix} A \\ \vdots \\ X_{ik} \\ \vdots \end{bmatrix}_i, - \begin{bmatrix} A \\ \vdots \\ X_{ik} \\ \vdots \end{bmatrix}_i S_k + \begin{bmatrix} R_k \\ \vdots \\ E_{ik} \\ \vdots \end{bmatrix}_i$$

where i depends on the re-encryption key queries, and LWE of form $f_1[A|P_k] + [f_2|f_3]$ for random matrix P_k and secret Gaussian noise vectors f_1, f_2, f_3. Thus all games above are indistinguishable to $\mathcal{A}$ under the $\text{LWE}(n, \alpha, q)$ assumption.

In theorem statement, the loss factor N is due to N games in consideration, and in each game there are Q_{rk} re-encryption key queries and Q_{re} re-encryption queries. The factor l comes from the LWE secrets corresponding to l columns of S_k above.

4 Our key-private, CCA-secure PRE

4.1 Bugs in the conference version

The scheme for CCA security in [3] unfortunately contains flaws in correctness and security: re-encrypted ciphertexts cannot be decrypted rightly, and security of original ciphertexts can be violated. To recall, for original ciphertexts of the form (c_1, c_2, c_s), the re-encrypted ones is of the form (c_1', c_2', c_s) with

$$(c_1', c_2') = \mathsf{ReEnc}^{cpa}(pk_B, rk_{A\to B}, c_1, c_2) \tag{3}$$

where ReEnc^{cpa} was described in Section 3.

- Correctness flaw: in [3], the decryption algorithm for (c_1, c_2, c_s) is also used to decrypt (c_1', c_2', c_s). The former returns $\perp$ immediately if

$$(c_1, c_2) \neq \mathsf{Enc}^{cpa}_{pk_A}(\sigma; H(\sigma, c_s)) \tag{4}$$

  where $\sigma = \mathsf{Dec}^{cpa}_{sk_A}(c_1, c_2)$. Since condition (4) won't be satisfied when (c_1, c_2) is replaced by (c_1', c_2') due to the re-randomization in ReEnc^{cpa}, (c_1', c_2', c_s) will almost always get rejected.
- Potential security flaw: now suppose (c_1, c_2, c_s) is the challenge ciphertext. The adversary then submits query $(c_1, c_2, 0^{|c_s|})$ for re-encryption to obtain $(c_1', c_2', 0^{|c_s|})$ where $0^{|c_s|}$ is a vector of all zeros of length equal to c_s, and (c_1', c_2') is as in (3). The adversary then submit query (c_1', c_2', c_s) for decryption, and if the decryption algorithm worked correctly, the adversary would obtain the challenge hidden bit.
  The reason for the possible security bug is in the fact that c_s is not glued together with (c_1', c_2'), as in (3) there is no role for c_s.

In the below revised scheme, we fix the above bugs.

4.2 The revised CCA-secure scheme

We utilize Fujisaki-Okamoto idea in [18, 19]: intuitively, $\mathsf{Enc}^{cpa}_{pk}(M; H(M))$ for a hash function H provides not only secrecy but also *authenticity* on message M, as any change on M will yield a "fresh randomness" $H(M)$. Moreover, one can put other inputs into H to get authenticity on those elements as well.

In more details, the encryption is as follows

$$\mathsf{Enc}^{cca}_{pk}(m;\sigma) = \Big(\mathsf{Enc}^{cpa}_{pk}\big(\sigma; H(\sigma, c_s)\big), \underbrace{\mathsf{SE}_{G(\sigma)}(m)}_{c_s}\Big)$$

in which

- σ is random; H and G are hash functions modeled as random oracles.
- $c_s = \mathsf{SE}_{G(\sigma)}(m)$ is the symmetric encryption of m under the key $G(\sigma)$.

Below is the detailed description of our CCA-secure scheme. Let $(\mathsf{SE}, \mathsf{SD})$ be a symmetric encryption scheme, which is one-time secure. (One example of $(\mathsf{SE}, \mathsf{SD})$ is the one-time-pad.) Let G, H are random oracles where $G : \{0,1\}^l \to \{0,1\}^{l_{sym}}$ for l_{sym} being the key length of the symmetric encryption scheme; $H : \{0,1\}^* \to \{0,1\}^{l_{gau}}$ for l_{gau} being the bit-length of the seed used in generating Gaussian noises.

Algorithms for parameters generation, key generation, proxy key generation are identical to the CPA-secure scheme in Section 3. The differences are in following algorithms.

Encryption $\mathsf{Enc}^{cca}_{pk}(m)$**:** Choose random $\sigma \in \{0,1\}^l$. Symmetrically encrypt message $m \in \{0,1\}^*$ by letting $c_s = \mathsf{SE}_{G(\sigma)}(m)$. Let $h = H(\sigma, c_s)$.

Encrypt σ: use randomness h, take Gaussian noise vectors $e_1, e_2 \in \psi_s^{1\times n}$, and $e_3 \in \psi_s^{1\times l}$, and return ciphertext $c = (c_1, c_2) \in \mathbb{Z}_q^{1\times(n+l)}$ where

$$c_1 = e_1 A + e_2 \in \mathbb{Z}_q^{1\times n}, c_2 = e_1 P + e_3 + \sigma \cdot \left\lfloor \frac{q}{2} \right\rfloor \in \mathbb{Z}_q^{1\times l}. \tag{5}$$

Return $ct^{(0)} = (c_1, c_2, c_s)$.

Re-encryption $\mathsf{ReEnc}^{cca}(pk_A, pk_B, rk_{A\to B}, ct^{(0)})$**:** do the re-encryption on $ct^{(0)} = (c_1, c_2, c_s)$ as follows. Below, $H_1 : \{0,1\}^* \to \{0,1\}^{l_{seedEnc}}$ and $G_1 : \{0,1\}^l \to \{0,1\}^{l_{seedEnc}}$ for $l_{seedEnc}$ being the seed length in encryption. Both G_1 and H_1 are modeled as random oracles.

1. Randomization: take random $\tau \in \{0,1\}^l$ and compute

$$\begin{aligned}(c_1', c_2') = \mathsf{Enc}^{cpa}_{pk_B}\Big(0; H_1(\tau, \underbrace{pk_A, pk_B, ct^{(0)}, rk_{A\to B}}_{\text{for integrity}})\Big)\\ + \quad [\mathsf{Bits}(c_1)|c_2] \cdot rk_{A\to B}.\end{aligned} \tag{6}$$

   where $\mathsf{Enc}^{cpa}_{pk_B}$ is the encryption algorithm in the CPA-secure scheme in Section 3 used with public key pk_B.

2. Asymmetrically encrypting τ using pk_B:

$$(d_1, d_2) = \mathsf{Enc}^{cpa}_{pk_B}(\tau; G_1(\tau)). \tag{7}$$

Return ciphertext $ct^{(1)} = (c_1', c_2', d_1, d_2, c_s)$.

Decryption of original ciphertexts $\mathsf{Dec}^{cca}_{sk_A}(ct^{(0)})$**:** To decrypt $c = (c_1, c_2, c_s)$ by secret key $sk_A = S$, execute following steps.

1. (Reconstruction) Compute $\overline{\sigma} = c_1 S + c_2 \in \mathbb{Z}_q^l$. Let $\overline{\sigma} = (\overline{\sigma}_1, \ldots, \overline{\sigma}_l)$. If $\overline{\sigma}_i \in [-\lfloor \frac{q}{4} \rfloor, \lfloor \frac{q}{4} \rfloor) \subset \mathbb{Z}_q$, let $\sigma'_i = 0$; otherwise $\sigma'_i = 1$. Let $\sigma' = \sigma'_1 \cdots \sigma'_l$ and $h' = H(\sigma', c_s)$.
2. (Integrity check) Using σ', h', check $(c_1, c_2) = \mathsf{Enc}^{cpa}_{pk_A}(\sigma'; h')$. Namely compute $(c'_1, c'_2) = \mathsf{Enc}^{cpa}_{pk_A}(\sigma'; h')$ and if $(c'_1, c'_2) \neq (c_1, c_2)$, return $\perp$; otherwise return $\mathsf{SD}_{G(\sigma')}(c_s)$ as the message.

Decryption $\mathsf{DecR}^{cca}_{sk_B}(ct^{(1)}, pi)$ with auxiliary public information $pi = (pk_A, pk_B, rk_{A\to B})$:

1. (Ensure τ is not malformed) Use sk_B to obtain τ. Check equation (7), by re-encrypting τ under pk_B with randomness $G_1(\tau)$.
2. (Ensure σ is not malformed) Use sk_B to obtain σ from (c'_1, c'_2). Then check equation (6) in which on the right hand side (c_1, c_2) is replaced by $\mathsf{Enc}^{cpa}_{pk_A}(\sigma; H(\sigma, c_s))$
3. If all checks go through, return $\mathsf{SD}_{G(\sigma)}(c_s)$ as the plaintext.

Bugs described in Section 4.1 solved. In equation (6), $ct^{(0)} = (c_1, c_2, c_s)$ is authenticated by the hash functions H_1. This glues $ct^{(0)}$ and in particular c_s with (c'_1, c'_2) of $ct^{(1)}$.

The randomness τ is for re-randomization process. To re-check (6) in decryption of re-encrypted ciphertexts, τ is sent via (d_1, d_2) which has both secrecy and authenticity.

In particular, it is now clear that re-encrypted ciphertexts can be decrypted correctly. Moreover, suppose (c_1, c_2, c_s) is the challenge ciphertext, and $(c_1, c_2, 0^{|c_s|})$ is re-encrypted to $ct^{(1)} = (c'_1, c'_2, d_1, d_2, 0^{|c_s|})$ via $pi = (pk_A, pk_B, rk_{A\to B})$. Then query $ct^{(1)}_* = (c'_1, c'_2, d_1, d_2, c_s)$ to $\mathsf{DecR}^{cca}_{sk_B}(\cdot, pi)$ will get $\perp$, as equation (6) won't be satisfied.

Theorem 4 (Security of Enc's ciphertexts). *Ciphertexts directly outputted by algorithm* Enc *is CCA-secure under the LWE assumption, if G, H, G_1, H_1 are random oracles. Specifically, for a poly-time adversary $\mathcal{A}$, there exists a poly-time distinguishers $\mathcal{D}, \mathcal{D}'$ such that*

$$\begin{aligned}\mathbf{Adv}^{\mathsf{Enc},cca}_{\mathcal{A}}(\lambda) \leq (NQ_{rk}l + NQ_{re} + 1) \cdot \mathbf{Adv}^{\mathrm{LWE}(n,\alpha,q)}_{\mathcal{D}}(\lambda) \\ +(Q_{\mathsf{Dec}} + Q_{\mathsf{DecR}}) \cdot \mathbf{Adv}^{\mathrm{LWE}(n,\alpha,q)}_{\mathcal{D}'}(\lambda)\end{aligned}$$

where Q_{rk} and Q_{re} are correspondingly the number of re-encryption key queries and re-encryption queries, N is the number of honest entities, and l is the message length. Q_{Dec} and Q_{DecR} are the number of decryption queries to oracles using algorithms Dec *and* DecR *correspondingly.*

Proof. At a high level, in the following, Game_0 is the original CCA game, in which secret keys of entities are used in simulation for decryption oracles. We will transform that game into Game_1 and Game_2, in which the simulation is done without the secret keys by utilizing random oracles H and G. We then show that both games are indistinguishable. This shows decryption oracles are useless to the adversary, so that the arguments after Game_1 are essentially identical to the CPA security proof. More details are as follows.

Since Game_0 is the original CCA game as in Definition 3, the decryption oracle using $\mathsf{Dec}(pp, sk_i = S_i, \cdot)$ works as follows:

1. Get input (pk_i, C) for $C = (c_1, c_2, c_s)$. Decrypt (c_1, c_2) using $sk_i = S_i$ to get $\sigma' \in \{0,1\}^l$.
2. Return $\perp$ if $(c_1, c_2) \neq \mathsf{Enc}^{cpa}(\sigma'; h')$ for $h' = H(\sigma', c_s)$. Otherwise return $\mathsf{SD}_{G(\sigma')}(c_s)$.

Moreover, let H_{list} and G_{list} store hash queries and corresponding answers either from $\mathcal{A}$ or from the algorithms called by the challenger. In particular, H_{list} contains tuple of forms $(\sigma, c_s, h = H(\sigma, c_s))$.

Game_1 is the same as Game_0, except that the decryption oracle $\mathsf{Dec}(pp, sk_i = S_i, \cdot)$ works, without using sk_i, as follows:

1. Get input (pk_i, C) for $C = (i, c_1, c_2, c_s)$. Using fixed c_s, check H_{list} to find tuple (σ, c_s, h) satisfying $\sigma \in \{0,1\}^l$ and $(c_1, c_2) = \mathsf{Enc}^{cpa}_{pk_i}(\sigma; h)$.
2. Return $\perp$ if there is no such tuple. Otherwise return $\mathsf{SD}_{G(\sigma)}(c_s)$.

We now argue Game_0 and Game_1 are indistinguishable, by examining the behavior of the decryption oracle in the games. If $\perp$ is returned by the oracles in both games, there is obviously no difference, so let us consider the remaining cases on query (pk_i, c_1, c_2, c_s):

- Case 1: the decryption in Game_0 returns $m' = \mathsf{SD}_{G(\sigma')}(c_s)$ while in Game_1 returns $m = \mathsf{SD}_{G(\sigma)}(c_s)$. Note
  $(c_1, c_2) = \mathsf{Enc}^{cpa}_{pk_i}(\sigma'; H(\sigma', c_s))$ in Game_0, and
  $(c_1, c_2) = \mathsf{Enc}^{cpa}_{pk_i}(\sigma; H(\sigma, c_s))$ in Game_1,
  so that $\sigma' = \sigma$, and hence $m' = m$. Thus there is no difference of the games in this case.
- Case 2: the decryption in Game_0 returns $\perp$ while in Game_1 returns $m = \mathsf{SD}_{G(\sigma)}(c_s)$. This means
  $\sigma' = \mathsf{Dec}^{cpa}_{S_i}(c_1, c_2)$, and
  $(c_1, c_2) \neq \mathsf{Enc}^{cpa}_{pk_i}(\sigma'; h')$ for $h' = H(\sigma', c_s)$,
  and yet there is a tuple tuple (σ, c_s, h) satisfying $(c_1, c_2) = \mathsf{Enc}^{cpa}_{pk_i}(\sigma; h)$. By the correctness of the CPA-secure PRE, $\sigma' = \sigma$, and hence $h = h'$, leading to a contradiction since $\mathsf{Enc}^{cpa}_{pk_i}(\sigma; h) \neq \mathsf{Enc}^{cpa}_{pk_i}(\sigma; h)$. Thus this case cannot be happened.
- Case 3: the decryption in Game_0 returns $m' = \mathsf{SD}_{G(\sigma')}(c_s)$ while in Game_1 returns $\perp$. The former means

$$\sigma' = \mathsf{Dec}^{cpa}_{S_i}(c_1, c_2), (c_1, c_2) = \mathsf{Enc}^{cpa}_{pk_i}(\sigma'; h')$$

  for $h' = H(\sigma', c_s)$.
  The latter means there is no tuple $(\sigma, c_s, h) \in H_{list}$ satisfying $(c_1, c_2) = \mathsf{Enc}^{cpa}_{pk_i}(\sigma; h)$. This implies tuple $(\sigma', c_s, h' = H(\sigma', c_s))$ corresponding to m' is not yet in H_{list}. Therefore $h' = H(\sigma', c_s)$ is freshly random from the viewpoint of $\mathcal{A}$. The condition $(c_1, c_2) = \mathsf{Enc}^{cpa}_{pk_i}(\sigma'; h')$ holds with negligible probability since, as proved in Theorem 2, the right hand side $\mathsf{Enc}^{cpa}_{pk_i}(\sigma'; h')$ is unpredictable under the LWE assumption.

Thus Game_0 and Game_1 are indistinguishable under the LWE assumption.

In Game_1, the decryption oracle for re-encrypted ciphertexts handles queries of form (pi, C) where C can be parsed as $(c'_1, c'_2, d_1, d_2, c_s)$ and pi as $(pk_i, pk_j, rk_{i\to j})$. This oracle works as follows.

1. Use $sk_j = S_j$ to decrypt (d_1, d_2), namely compute $\tau = \mathsf{Dec}^{cpa}_{sk_j}(d_1, d_2)$.
2. If $(d_1, d_2) \neq \mathsf{Enc}^{cpa}_{pk_j}(\tau; G_1(\tau))$, return $\perp$ immediately.
3. Use sk_j to decrypt (c'_1, c'_2), namely compute $\sigma = \mathsf{Dec}^{cpa}_{sk_j}(c'_1, c'_2)$.
4. Let $(c_1, c_2) = \mathsf{Enc}^{cpa}_{pk_i}(\sigma; h)$ where $h = H(\sigma, c_s)$ and $ct^{(0)} = (c_1, c_2, c_s)$.
5. Immediately return $\perp$ if

$$(c'_1, c'_2) \neq \mathsf{Enc}^{cpa}_{pk_j}\Big(0; H_1(\tau, ct^{(0)}, pi)\Big) + [\mathsf{Bits}(c_1)|c_2] \cdot rk_{i\to j}.$$

6. Return $\mathsf{SD}_{G(\sigma)}(c_s)$.

In Game_2, the decryption oracle handles queries of form (pi, C) where C can be parsed as $(c'_1, c'_2, d_1, d_2, c_s)$ and pi as $(pk_i, pk_j, rk_{i\to j})$ as follows.

1. Check G_1-list to find τ satisfying $(d_1, d_2) = \mathsf{Enc}^{cpa}_{pk_j}(\tau; G_1(\tau))$. If there is no such τ, return $\perp$.

2. Use fixed τ and pi and c_s, check H_1-list to find $ct^{(0)} = (c_1, c_2, c_s)$ satisfying

$$(c'_1, c'_2) = \mathsf{Enc}^{cpa}_{pk_j}\Big(0; H_1(\tau, ct^{(0)}, pi)\Big) + [\mathsf{Bits}(c_1)|c_2] \cdot rk_{i\to j}$$

and return $\perp$ immediately if there is no such $ct^{(0)}$.
3. For fixed c_s and given (c_1, c_2) above, check H-list to find σ satisfying

$$(c_1, c_2) = \mathsf{Enc}^{cpa}_{pk_i}(\sigma; H(\sigma, c_s))$$

and if there is no such σ, return $\perp$ immediately.
4. Having above σ, return $\mathsf{SD}_{G(\sigma)}(c_s)$.

We now consider following cases.

- Case 1: the decryption in Game_1 returns m_1 ($\neq \perp$) while in Game_2 returns m_2 ($\neq \perp$). Viewing steps 6 (Game_1) and 4 (Game_2), to prove $m_1 = m_2$, it suffices to show that σ is identical in the both games. Thus we need $\mathsf{Dec}^{cpa}_{sk_j}(c'_1, c'_2)$ in Game_1 is the same as $\mathsf{Dec}^{cpa}_{sk_i}(c_1, c_2)$ in Game_2, which holds true since in the latter game

$$\begin{aligned}&(c'_1, c'_2) = \\ &\mathsf{Enc}^{cpa}_{pk_j}\Big(0; H_1(\tau, (c_1, c_2, c_s), pi)\Big) + [\mathsf{Bits}(c_1)|c_2] \cdot rk_{i\to j}\end{aligned}$$

and

$$[\mathsf{Bits}(c_1)|c_2] \cdot rk_{i\to j} \cdot \begin{bmatrix} sk_j \\ \mathbf{I}_{l\times l} \end{bmatrix} = \mathsf{smallnoise} + c_1 sk_i + c_2$$

so that $\mathsf{Dec}^{cpa}_{sk_j}(c'_1, c'_2)$ (in both Game_1 and Game_2) is the same as $\mathsf{Dec}^{cpa}_{sk_i}(c_1, c_2)$ in Game_2 as required.
- Case 2: the decryption in Game_1 returns m ($\neq \perp$) while in Game_2 returns $\perp$. We will show this case happens with negligible probability. The former tells us that all integrity checks pass. The latter's $\perp$ occurs due to there is no τ or $ct^{(0)}$ or σ in G_1 list, or H_1-list, or H-list correspondingly. Therefore, in Game_1, following equations

$$\begin{aligned}(d_1, d_2) &= \mathsf{Enc}^{cpa}_{pk_j}(\tau; G_1(\tau)) \\ (c_1, c_2) &= \mathsf{Enc}^{cpa}_{pk_i}(\sigma; H(\sigma, c_s)) \\ (c'_1, c'_2) &= \mathsf{Enc}^{cpa}_{pk_j}\Big(0; H_1(\tau, ct^{(0)}, pi)\Big) + [\mathsf{Bits}(c_1)|c_2] \cdot rk_{i\to j}\end{aligned}$$

are met with negligible probability since the randomness values used in $\mathsf{Enc}^{cpa}_{pk_j}$ of the right side are fresh. More precisely, this case happens with probability less than the probability of solving the LWE problem.
- Case 3: the decryption in Game_1 returns $\perp$ while in Game_2 returns m ($\neq \perp$). The former means either

$$(d_1, d_2) \neq \mathsf{Enc}^{cpa}_{pk_j}(\tau; G_1(\tau))$$

or

$$(c'_1, c'_2) \neq \mathsf{Enc}^{cpa}_{pk_j}\Big(0; H_1(\tau, ct^{(0)}, pi)\Big) + [\mathsf{Bits}(c_1)|c_2] \cdot rk_{i\to j}$$

and yet the latter ensures equality, yielding a contradiction. Thus this case cannot happen.

Let us now look at the challenge ciphertext which hides a bit b

$$(c_1^*, c_2^*) = \mathsf{Enc}_{pk_{i^*}}^{cpa}(\sigma^*; h^*), c_s^* = \mathsf{SE}_{k^*}(m_b)$$

for random $\sigma^* \in \{0,1\}^l$, and $h^* = H(\sigma^*, c_s^*)$, $k^* = G(\sigma^*)$. Since H and G are random oracles, h^* and k^* are random provided that σ^* is not given out to $\mathcal{A}$, which holds true since $\mathsf{Enc}_{pk_{i^*}}^{cpa}$ is in CPA-secure PRE. (More precisely, only onewayness is needed here.) By random k^*, $c_s^* = \mathsf{SE}(m_b)$ leaks no information on b due to the security of the symmetric encryption scheme. We omit other details similar to the proof of Theorem 2 $\square$.

Theorem 5 (Security of ReEnc's ciphertexts). *Ciphertexts directly outputted by algorithm* ReEnc *is CCA-secure under the LWE assumption, if G, H, G_1, H_1 are random oracles. Specifically, for a poly-time adversary $\mathcal{A}$, there exists a poly-time distinguishers $\mathcal{D}, \mathcal{D}'$ such that*

$$\begin{aligned}\mathbf{Adv}_{\mathcal{A}}^{\mathsf{ReEnc},cca}(\lambda) \leq (NQ_{rk}l + NQ_{re} + 1) \cdot \mathbf{Adv}_{\mathcal{D}}^{\mathrm{LWE}(n,\alpha,q)}(\lambda) \\ + \quad (Q_{\mathsf{Dec}} + Q_{\mathsf{DecR}}) \cdot \mathbf{Adv}_{\mathcal{D}'}^{\mathrm{LWE}(n,\alpha,q)}(\lambda)\end{aligned}$$

where Q_{rk} and Q_{re} are correspondingly the number of re-encryption key queries and re-encryption queries, N is the number of honest entities, and l is the message length. Q_{Dec} and Q_{DecR} are the number of decryption queries to oracles using algorithms Dec *and* DecR *correspondingly.*

Proof. Simulation for re-encryption queries and re-encryption key queries is the same as in the the proof of Theorem 2. Simulation for decryption oracles goes along the lines of the proof of Theorem 4. The challenge ciphertext is $C_{i^*} = (c_1', c_2', d_1, d_2, c_s)$ where, as in equation (6), (c_1', c_2') is computed using

$$\begin{aligned}&\underbrace{\mathsf{Enc}_{pk_{i^*}}^{cpa}\Big(0; H_1(\tau^*, pk_{i_{\mathcal{A}}}, pk_{i^*}, CT_b, rk_{i_{\mathcal{A}} \to i^*})\Big)}_{\text{re-randomization part}} \\ &+ \quad \Big[\mathsf{Bits}([CT_b]_1) \Big| [CT_b]_2\Big] \cdot rk_{i_{\mathcal{A}} \to i^*}\end{aligned}$$

and $(d_1, d_2) = \mathsf{Enc}_{pk_{i^*}}^{cpa}(\tau^*, G_1(\tau^*))$ where τ^* is randomly chosen by the challenger. Thus the bit b is hidden thanks to the re-randomization part, which is indistinguishable from random under the LWE assumption.

Theorem 6 (Key privacy). *The above PRE scheme is key-private in CCA setting under the LWE assumption, if G, H, G_1, H_1 are random oracles. Specifically, for a poly-time adversary $\mathcal{A}$, there exists a poly-time distinguishers $\mathcal{D}, \mathcal{D}'$ such that*

$$\begin{aligned}\mathbf{Adv}_{\mathcal{A}}^{kp-cca}(\lambda) \leq N(Q_{rk}l + Q_{re}) \cdot \mathbf{Adv}_{\mathcal{D}}^{\mathrm{LWE}(n,\alpha,q)}(\lambda) \\ + \quad (Q_{\mathsf{Dec}} + Q_{\mathsf{DecR}}) \cdot \mathbf{Adv}_{\mathcal{D}'}^{\mathrm{LWE}(n,\alpha,q)}(\lambda)\end{aligned}$$

where Q_{rk} and Q_{re} are correspondingly the number of re-encryption key queries and re-encryption queries, N is the number of honest entities, and l is the message length. Q_{Dec} and Q_{DecR} are the number of decryption queries to oracles using algorithms Dec *and* DecR *correspondingly.*

Proof. Simulation for re-encryption queries and re-encryption key queries is the same as in the the proof of Theorem 3, yielding the loss factor $N(Q_{rk}l + Q_{re})$ to LWE. Simulation for decryption oracles goes along the lines of the proof of Theorem 4, making the loss factor $(Q_{\mathsf{Dec}} + Q_{\mathsf{DecR}})$. The challenge re-encryption key is changed to random matrices, as in Game_{j^*} of the proof of Theorem 3, and hence leaks no information on the bit b under the LWE assumption.

Acknowledgment

Lihua Wang and Yoshinori Aono are partially supported respectively by JSPS Grants No. 15K00028 and No. 26730069 . Xavier Boyen gratefully acknowledges support from the Australian Research Council under ARC Grant DP-140103885, and the hospitality of NICT.

References

1. PCI DSS. https://www.pcisecuritystandards.org/documents/Prioritized_Approach_V2.0.pdf.
2. OWASP. https://www.owasp.org/index.php/Cryptographic_Storage_Cheat_Sheet.
3. Y. Aono, X. Boyen, L. T. Phong, and L. Wang. Key-private proxy re-encryption under LWE. In G. Paul and S. Vaudenay, editors, *INDOCRYPT*, volume 8250 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2013.
4. Y. Aono, L. T. Phong, and L. Wang. Hardness Estimation of LWE via Band Pruning. Available at http://eprint.iacr.org/2015/1026, 2015.
5. B. Applebaum, D. Cash, C. Peikert, and A. Sahai. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In S. Halevi, editor, *CRYPTO*, volume 5677 of *Lecture Notes in Computer Science*, pages 595–618. Springer, 2009.
6. G. Ateniese, K. Benson, and S. Hohenberger. Key-private proxy re-encryption. In M. Fischlin, editor, *CT-RSA*, volume 5473 of *Lecture Notes in Computer Science*, pages 279–294. Springer, 2009. Full version at http://eprint.iacr.org/2008/463.
7. G. Ateniese, K. Fu, M. Green, and S. Hohenberger. Improved proxy re-encryption schemes with applications to secure distributed storage. *ACM Trans. Inf. Syst. Secur.*, 9(1):1–30, 2006.
8. W. Banaszczyk. New bounds in some transference theorems in the geometry of numbers. *Mathematische Annalen*, 296(1):625–635, 1993.
9. W. Banaszczyk. Inequalities for convex bodies and polar reciprocal lattices in $\mathbb{R}^n$. *Discrete & Computational Geometry*, 13(1):217–231, 1995.
10. M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In D. E. Denning, R. Pyle, R. Ganesan, R. S. Sandhu, and V. Ashby, editors, *ACM Conference on Computer and Communications Security*, pages 62–73. ACM, 1993.
11. M. Blaze, G. Bleumer, and M. Strauss. Divertible protocols and atomic proxy cryptography. In K. Nyberg, editor, *EUROCRYPT*, volume 1403 of *Lecture Notes in Computer Science*, pages 127–144. Springer, 1998.
12. Z. Brakerski. Fully homomorphic encryption without modulus switching from classical gapsvp. In R. Safavi-Naini and R. Canetti, editors, *CRYPTO*, volume 7417 of *Lecture Notes in Computer Science*, pages 868–886. Springer, 2012.
13. Z. Brakerski, A. Langlois, C. Peikert, O. Regev, and D. Stehlé. Classical hardness of learning with errors. In D. Boneh, T. Roughgarden, and J. Feigenbaum, editors, *STOC*, pages 575–584. ACM, 2013.
14. R. Canetti and S. Hohenberger. Chosen-ciphertext secure proxy re-encryption. In P. Ning, S. D. C. di Vimercati, and P. F. Syverson, editors, *ACM Conference on Computer and Communications Security*, pages 185–194. ACM, 2007.
15. E. Dawson, editor. *Topics in Cryptology - CT-RSA 2013 - The Cryptographers'Track at the RSA Conference 2013, San Francisco, CA, USA,February 25-March 1, 2013. Proceedings*, volume 7779 of *Lecture Notes in Computer Science*. Springer, 2013.
16. R. H. Deng, J. Weng, S. Liu, and K. Chen. Chosen-ciphertext secure proxy re-encryption without pairings. In M. K. Franklin, L. C. K. Hui, and D. S. Wong, editors, *CANS*, volume 5339 of *Lecture Notes in Computer Science*, pages 1–17. Springer, 2008.
17. Eigen library. Website: http://eigen.tuxfamily.org/index.php?title=Main_Page#Overview.
18. E. Fujisaki and T. Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In M. J. Wiener, editor, *CRYPTO*, volume 1666 of *Lecture Notes in Computer Science*, pages 537–554. Springer, 1999.
19. E. Fujisaki and T. Okamoto. Secure integration of asymmetric and symmetric encryption schemes. *J. Cryptology*, 26(1):80–101, 2013.
20. S. Hohenberger, G. N. Rothblum, A. Shelat, and V. Vaikuntanathan. Securely obfuscating re-encryption. In S. P. Vadhan, editor, *TCC*, volume 4392 of *Lecture Notes in Computer Science*, pages 233–252. Springer, 2007.
21. T. Isshiki, M. H. Nguyen, and K. Tanaka. Proxy re-encryption in a stronger security model extended from CT-RSA2012. In Dawson [15], pages 277–292.
22. R. Kannan. Improved algorithms for integer programming and related lattice problems. In D. S. Johnson, R. Fagin, M. L. Fredman, D. Harel, R. M. Karp, N. A. Lynch, C. H. Papadimitriou, R. L. Rivest, W. L. Ruzzo, and J. I. Seiferas, editors, *STOC*, pages 193–206. ACM, 1983.
23. B. Libert and D. Vergnaud. Unidirectional chosen-ciphertext secure proxy re-encryption. *IEEE Transactions on Information Theory*, 57(3):1786–1802, 2011.

24. R. Lindner and C. Peikert. Better key sizes (and attacks) for LWE-based encryption. In A. Kiayias, editor, *CT-RSA*, volume 6558 of *Lecture Notes in Computer Science*, pages 319–339. Springer, 2011.
25. M. Liu and P. Q. Nguyen. Solving BDD by enumeration: An update. In Dawson [15], pages 293–309.
26. D. Micciancio and O. Regev. Lattice-based cryptography. In *Post-Quantum Cryptography*, pages 147–191. Springer, 2009.
27. National Institute of Standards and Technology (NIST). Recommendation for Key Management: Part 1: General (Revision 3). `http://csrc.nist.gov/publications/nistpubs/800-57/sp800-57_part1_rev3_general.pdf`. Accessed: 2014, January 16.
28. R. Nishimaki and K. Xagawa. Key-Private Proxy Re-Encryption from Lattices, Revisited. IEICE Transactions, 98-A(1):100–116, 2015.
29. M. Ruckert and M. Schneider. Estimating the security of lattice-based cryptosystems. Cryptology ePrint Archive, Report 2010/137, 2010. `http://eprint.iacr.org/`.
30. O. Regev. On lattices, learning with errors, random linear codes, and cryptography. In H. N. Gabow and R. Fagin, editors, *STOC*, pages 84–93. ACM, 2005.
31. J. W. Seo, D. H. Yum, and P. J. Lee. Comments on "unidirectional chosen-ciphertext secure proxy re-encryption". *IEEE Transactions on Information Theory*, 59(5):3256, 2013.
32. J. Shao. Anonymous id-based proxy re-encryption. In W. Susilo, Y. Mu, and J. Seberry, editors, *ACISP*, volume 7372 of *Lecture Notes in Computer Science*, pages 364–375. Springer, 2012.
33. J. Shao, Z. Cao, and P. Liu. SCCR: a generic approach to simultaneously achieve cca security and collusion-resistance in proxy re-encryption. *Security and Communication Networks*, 4(2):122–135, 2011.
34. J. Shao, P. Liu, Z. Cao, and G. Wei. Multi-use unidirectional proxy re-encryption. In *ICC*, pages 1–5. IEEE, 2011.
35. J. Shao, P. Liu, G. Wei, and Y. Ling. Anonymous proxy re-encryption. *Security and Communication Networks*, 5(5):439–449, 2012.
36. J. Shao, P. Liu, and Y. Zhou. Achieving key privacy without losing cca security in proxy re-encryption. *Journal of Systems and Software*, 85(3):655–665, 2012.
37. L. Wang, L. Wang, M. Mambo, and E. Okamoto. New identity-based proxy re-encryption schemes to prevent collusion attacks. In M. Joye, A. Miyaji, and A. Otsuka, editors, *Pairing*, volume 6487 of *Lecture Notes in Computer Science*, pages 327–346. Springer, 2010.
38. J. Weng, M.-R. Chen, Y. Yang, R. H. Deng, K. Chen, and F. Bao. CCA-secure unidirectional proxy re-encryption in the adaptive corruption model without random oracles. *SCIENCE CHINA Information Sciences*, 53(3):593–606, 2010.
39. D. Watanabe and M. Yoshino. Key Update Mechanism Using All-or-Nothing Transform for Network Storage of Encrypted Data. IEICE Transactions 98-A(1):162–170, 2015.

A A derived CCA-secure PKE scheme

Removing the re-encryption feature, the scheme in Section 4.2 is CCA-secure under the LWE assumption as a public key encryption (PKE) scheme, whose description is as follows.

Parameters generation $\mathsf{ParamsGen}^{cca}_{pke}(\lambda)$**:** Choose positive integers q, n, l, and take matrix $A \in \mathbb{Z}_q^{n\times n}$ randomly. Return $pp = (q, n, l, A)$, which is the input to all algorithms below.

Key generation $\mathsf{KeyGen}^{cca}_{pke}(pp, \lambda)$**:** Fix deviation $s \in \mathbb{R}$. Take Gaussian noise matrices $R, S \in \psi_s^{n\times l}$. The public key is $pk = P$ for $P = R - AS \in \mathbb{Z}_q^{n\times l}$, and the secret key is $sk = S$. Return (pk, sk).

Let $(\mathsf{SE}, \mathsf{SD})$ be a symmetric encryption scheme which is one-time secure, such as the one-time-pad.

Encryption $\mathsf{Enc}^{cca}_{pke}(m)$**:** symmetrically encrypt message $m \in \{0,1\}^*$ by letting $c_s = \mathsf{SE}_{G(\sigma)}(m)$. Choose random $\sigma \in \{0,1\}^l$. Let $h = H(\sigma, c_s)$.

Encrypt σ: use randomness h, take Gaussian noise vectors $e_1, e_2 \in \psi_s^{1\times n}$, and $e_3 \in \psi_s^{1\times l}$, and return ciphertext $c = (c_1, c_2) = \mathsf{Enc}^{cpa}_{pk}(\sigma; h)$, namely compute

$$c_1 = e_1 A + e_2 \in \mathbb{Z}_q^{1\times n}, c_2 = e_1 P + e_3 + \sigma \cdot \left\lfloor \frac{q}{2} \right\rfloor \in \mathbb{Z}_q^{1\times l}.$$

Return $ct = (c_1, c_2, c_s)$.

Decryption $\mathsf{Dec}^{cca}_{pke}(sk, ct)$**:** To decrypt $c = (c_1, c_2, c_s)$ by secret key $sk = S$, execute following steps.

1. (Reconstruction) Compute $\overline{\sigma} = c_1 S + c_2 \in \mathbb{Z}_q^l$. Let $\overline{\sigma} = (\overline{\sigma}_1, \ldots, \overline{\sigma}_l)$. If $\overline{\sigma}_i \in [-\lfloor \frac{q}{4} \rfloor, \lfloor \frac{q}{4} \rfloor) \subset \mathbb{Z}_q$, let $\sigma'_i = 0$; otherwise $\sigma'_i = 1$. Let $\sigma' = \sigma'_1 \cdots \sigma'_l$ and $h' = H(\sigma', c_s)$.
2. (Integrity check and output) Using σ', h', check

$$(c_1, c_2) = \mathsf{Enc}_{pk}^{cpa}(\sigma'; h'),$$

   namely compute $(c'_1, c'_2) = \mathsf{Enc}_{pk}^{cpa}(\sigma'; h')$ and if $(c'_1, c'_2) \neq (c_1, c_2)$, return $\perp$; otherwise return $\mathsf{SD}_{G(\sigma')}(c_s)$ as the message.

The CCA security of the PKE scheme can be derived from [18, 19] and the CPA security of the base scheme [24]. A direct proof is similar to the proof of Theorem 4.