# Delegation of Decryption Rights with Revocability from Learning with Errors

**Abstract**

The notion of decryption rights delegation was initially introduced by Blaze et al. in EUROCRYPT 1998. It, defined as *proxy re-encryption*, allows a semi-trusted proxy to convert a ciphertext intended for a party to another ciphertext of the same plaintext, without knowledge of the underlying plaintext and decryption key. It has been explored to many real-world applications, e.g., encrypted email forwarding. However, the intrinsic all-or-nothing share feature of proxy re-encryption yields a limitation that the share cannot be revoked. This may hinder the scalability of its applications in practice. In this paper, for the first time, we define the concept of revocability in terms of decryption rights delegation. The novel concept enables data owner to revoke the shared decryption rights when needed. Inspired by the seminal lattice-based proxy re-encryption proposed in PKC 2014, we design a concrete lattice-based construction which satisfies the notion. In our construction, we make use of binary-tree structure to implement the revocation of decryption rights, so that the update of re-encryption key is reduced to $O(logN)$ (instead of $O(N)$), where $N$ is the maximum number of delegatee. Furthermore, the security of our scheme is based on the standard learning with errors problem, which could be reduced to the worst-case hard problems (such as GapSVP and SIVP) in the context of lattices. The scheme is chosen ciphertext secure in the standard model. As of independent interest, our scheme achieves both backward and forward security, which means that once a user is revoked after a time period $\mathbf{t}$, it cannot gain access to all encrypted files before and after $\mathbf{t}$.

**Keywords:** Revocability, proxy re-encryption, lattice, learning with errors.

## 1 Introduction

The concept of proxy re-encryption (PRE) was first introduced by Blaze, Bleumer and Strauss[7] in EUROCRYPT 1998 to enable an intermediate proxy to convert a ciphertext of Alice to that of Bob without compromising the information of the underlying plaintext. Alice here is known as the delegator while Bob is the delegatee. The semi-trusted proxy can fulfil the conversion with help of a re-encryption key given by the delegator. PRE has been employed into many real-world applications, e.g., encrypted email forwarding, and domain interoperability manager (DIM) module in digital rights management (DRM) systems.

As for the first example, while Alice is on vacation, the email proxy may convert Alice's incoming encrypted emails to those which can be decrypted by secretary Bob via the re-encryption key given by Alice, so that Bob can handle the emails on behalf of Alice. The PRE mechanism provides scalable and convenient features over data sharing: (i) Alice does not need always to be online; (ii) Alice does get rid of download-decrypt-and-re-encrypt mode to relieve computation and communication complexity in data sharing; (iii) Alice does not have to share her secret/decryption key with Bob for encrypted data sharing.

PRE can also be employed to DRM systems. Digital content providers may leverage DRM mechanism to protect the ownership and access rights of digital content from being infringed by malicious Internet users. A DRM system is able to bind digital content with ownership license, meanwhile, only an authenticated user can access the content. DRM, nevertheless, suffers from a domain limitation that a digital content in the domain $\mathcal{A}$ can only be accessed by the devices within that domain. That makes DRM non-scalable in practical use. DIM intermediate module, one of the typical applications of PRE technology, may come to help solve the domain problem. Specifically, the module can be used to convert a ciphertext (license) of the domain $\mathcal{A}$ to the ciphertext (license) belonging to another domain, say $\mathcal{B}$. Furthermore, the module cannot extract the underlying plaintext from the ciphertexts (licenses), so that the confidentiality of data but also effective cross-domain conversion are guaranteed (please refer to Figure 1). But the DIM fails to support the revocability of decryption rights delegation.

Traditional PRE, being as a type of all-or-nothing decryption rights delegation, cannot allow a delegator to revoke its shared decryption rights from delegatee (after a re-encryption key is issued to the proxy). In the above encrypted

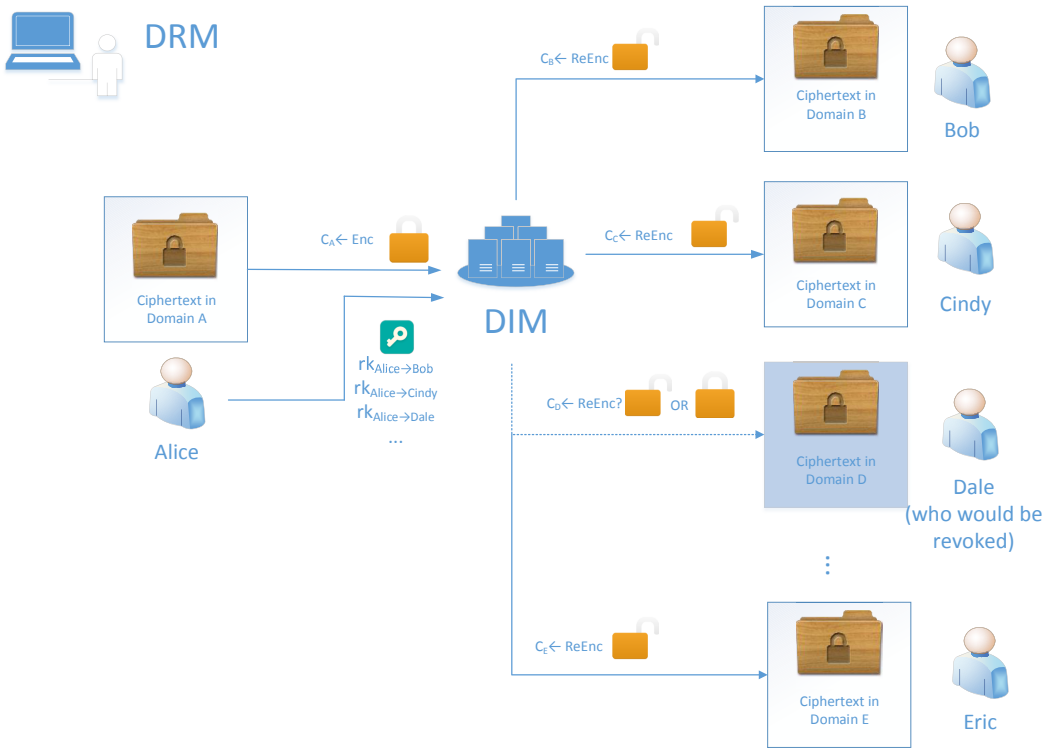



Figure 1. Effective DIM Interface in DRM

email forwarding setting, Bob can keep gaining access to all encrypted emails intended for Alice (even if Alice is back to work). This may not scale well in practice because one may prefer to only share decryption rights with others within some fixed time slots, for example, after returning to work, Alice may choose to handle the emails on her own without any interference of Bob. To the best of our knowledge, there is no PRE scheme (in the context of public key encryption) dealing with the issue of revocability.

One may think that revoking the delegation of decryption rights is trivial in the sense that a delegator may just request a proxy to delete the corresponding re-encryption key (so that the re-encryption may be terminated). This naive solution, we state, may work in the context where the sharing is not time-related/updated. In practice, it may be difficult to isolate data sharing from time period. In payTV application (e.g. Netflix), an encrypted movie may be watched by subscribers based on payment status. A re-encryption key here may relate with a time period, say a month, so that subscribers can decrypt and watch the movie in the month they've paid the subscription fee. Simply deleting re-encryption key to revoke a subscriber's rights may not scale well, for instance, the subscriber may join back in the next month. Another concern here for the revocation is that if the revoked subscriber is still able to gain access to the movies which are encrypted before the revocation point. In addition to maintain the confidentiality of data, the efficiency of revocation must be taken into account. How to guarantee data confidentiality in the revocation of time-related decryption rights delegation without linear complexity that motivates our work.

In recent years, lattice-based cryptography has attracted numerous attention from cryptographic researchers. The lattice as an alternative underlying primitive is more and more applicable to cryptographic schemes. Compared to the traditional cryptography based on number theory hard problems (e.g., integer factorization and discrete logarithm), the promising features of lattice-based cryptography are as follows: (i) it is conjectured to be secure against quantum computer attacks; (ii) it is with algorithmic simplicity and high parallelism; (iii) it has an average-case/worst-case reduction for commonly used hard assumptions. This paper also seeks a way to construct PRE with revocability in the lattice-based setting to present secure system which can hold against the attacks of quantum computers.

## 1.1 Our Contributions

1. To the best of our knowledge, we propose the notion of revocable PRE, for the first time, in which the revocability is linked with time period. The definition and security model of revocable PRE are defined in this paper. In the security model, an adversary should follow the restrictions defined as in traditional PRE setting

and furthermore, it is allowed to update re-encryption key and revoke system users. However, it is restricted that the adversary cannot decrypt the ciphertexts which are encrypted before/after the time period of the revocation.

2. At a first glance, a trivial revocation system (as mentioned in the previous section) may incur that the revocation complexity is linear in the number of users. Our scheme relieves the workload of delegator from linear to logarithmic complexity by using binary tree structure to organize re-encryption keys. We also provide a non-interactive re-encryption key update technique which is used to shorten the time of key update (w.r.t. user revocation).

3. The forward security is considered in our construction. To achieve the goal, we need to update the ciphertext of delegator that is stored in server whilst the corresponding delegatee is revoked, say at time period $\mathbf{t}$. After confirming to revoke the decryption rights of Bob at $\mathbf{t}$, Alice will update the re-encryption key from herself to Bob so that a proxy cannot convert the ciphertext of Alice to Bob by using the updated re-encryption key. But the proxy can still use the re-encryption keys that are generated at $\mathbf{t}' < \mathbf{t}$ to convert the ciphertexts under $\mathbf{t}'$ to Bob. The decryption capability of Bob is not revoked in terms of the ciphertexts generated before $\mathbf{t}$. To tackle this problem (to guarantee forward security), after the update of re-encryption key, we make an update for the ciphertext of Alice as well by executing algorithm **UpCipher** (after **ReKeyRev**) to update the ciphertexts of Alice under $\mathbf{t}' < \mathbf{t}$.

4. A concrete lattice-based revocable PRE construction is presented in this paper. The CCA security of our construction is proved based on the hardness of the learning with errors (LWE) in the standard model, which is as hard as several worst-case lattice problems, such as GapSVP for some factors $\tilde{O}(n/\alpha)$ or $\mathrm{SIVP}_\gamma$ for some polynomial factor $\gamma = poly(n)$.

The main technical roadmap of our construction is to design an interface to combine the technique of revocable IBE [11] with PRE technique[18] to achieve efficient, non-interactive and collusion-free PRE with revocability. Specifically, we embed time information into public key of user and split the public key into three parts and further, merge re-encryption key into a binary tree structure. The time information is compatible with [18] in terms of re-encryption key generation, we therefore inject randomness to guarantee that the time term on both sides of the equation will not be eliminated, which is immune to a means of attack in [14] (note more details are given in Section 5). Furthermore, we prove that our scheme can hold against chosen ciphertext attacks (CCA). The tricky part of security proof is on how to answer re-encryption key queries issued by adversary. In the real system, denoted by $\mathbf{A_R}$ and $\mathbf{A_L}$ the matrices should satisfy $[\mathbf{A}'_\mathbf{0}|\mathbf{A}'_\mathbf{1} + \mathbf{H}'\mathbf{G}] = \mathbf{A_R} + \mathbf{A_L}$, where $\mathbf{A_R}$ and $\mathbf{A_L}$ are generated randomly. The method we adopt is to use an invertible unimodular matrix $\mathbf{U}$ to constitute the public key of user in the simulation. Let $[\mathbf{A}'_\mathbf{0}|-\mathbf{A}'_\mathbf{0}\mathbf{R}'_\mathbf{1}] = [\mathbf{U}\mathbf{A}_{\mathbf{R_1}} + (\mathbf{I}-\mathbf{U})\mathbf{A}_{\mathbf{L_1}}|\mathbf{U}\mathbf{A}_{\mathbf{R_2}} + (\mathbf{I}-\mathbf{U})\mathbf{A}_{\mathbf{L_2}}]$, we choose $\mathbf{X_{00}}$, $\mathbf{X'_{00}}$, $\mathbf{X_{10}}$ and $\mathbf{X'_{10}}$ from a Gaussian distribution with parameter $s$, so that

$$\begin{pmatrix}\mathbf{X'_{00}}\\ \mathbf{X'_{10}}\end{pmatrix} + \begin{pmatrix}\mathbf{X_{00}}\\ \mathbf{X_{10}}\end{pmatrix} = \begin{pmatrix}\overline{\mathbf{X_{00}}}\\ \overline{\mathbf{X_{10}}}\end{pmatrix}, \begin{pmatrix}\mathbf{X'_{00}}\\ \mathbf{X'_{10}}\end{pmatrix}\mathbf{R'_1} + \begin{pmatrix}\mathbf{X_{00}}\\ \mathbf{X_{10}}\end{pmatrix}\mathbf{R'_1} = \begin{pmatrix}\overline{\mathbf{X_{01}}}\\ \overline{\mathbf{X_{11}}}\end{pmatrix}, \begin{pmatrix}\mathbf{X'_{00}}\\ \mathbf{X'_{10}}\end{pmatrix}\mathbf{R'_2} = \begin{pmatrix}\overline{\mathbf{X_{02}}}\\ \overline{\mathbf{X_{12}}}\end{pmatrix},$$

and further the re-encryption key from user $pk^*$ to $pk'$ is constructed as

$$rk_{pk^*\rightarrow pk'} = \begin{pmatrix}\mathbf{X_{00}} + \mathbf{X'_{00}} & \mathbf{X_{01}} + \mathbf{X'_{01}} & \mathbf{X'_{02}}\\ \mathbf{X_{10}} + \mathbf{X'_{10}} & \mathbf{X_{11}} + \mathbf{X'_{11}} & \mathbf{X'_{12}}\\ \mathbf{0} & \mathbf{0} & \mathbf{I}\end{pmatrix}.$$

## 1.2 Related Work

There have been many research works on user revocation to date. Boneh and Franklin [9] design a key revocation mechanism that allows legitimate users to periodically update the secret key corresponding to time slot. However, their key update algorithm requires a trusted key issuer which consumes computation and communication complexity linearly in the number of non-revoked users. Besides, the key issuer requires a secure channel to transmit the updated keys to system users. Following the seminal work [9], fuzzy identity-based encryption (IBE) and binary tree data structure are creatively combined together to yield an identity revocation scheme [8]. The scheme reduces the key update complexity (on key issuer side) from linear to logarithmic level. Chen et al. [11] later propose an IBE scheme from lattices with efficient key revocation, and prove the scheme to be selective secure in the standard model and

Table 1: Comparison with related PRE schemes

| PRE | Based on | Revocability | security | standard model |
|---|---|---|---|---|
| XT10 [17] | LWE | × | CPA | √ |
| ABPW13 [4] | LWE | × | CPA | √ |
| Kir14 [18] | LWE | × | ?* | √ |
| NAL15 [25] | NTRU | × | CPA | √ |
| Fan16 [14] | LWE | × | CCA | √ |
| Ours | LWE | √ | CCA | √ |

* [14] points out that [18] is not secure.

under the LWE assumption. Seo et al. [27] design a concrete construction based on pairings, in which the construction is able to hold against the attacks, called decryption key exposure attack. Lee et al. [19] propose an adaptive-identity security revocable IBE (RIBE) scheme with pairings by using the subset difference method. Ling et al. [21] deliver the first construction of lattice-based revocable predicate encryption, satisfying the notion of full-hiding security in the standard model.

Since its introduction, PRE has been well studied for the past decades. Ateniese and Fu et al. [6] design a first unidirectional PRE scheme, which is used as a mechanism of access control over encrypted file system. Their scheme is based on bilinear pairings and achieves the security of chosen plaintext attack (CPA) in the standard model. Green and Ateniese [16] introduce the notion of identity-based PRE (IB-PRE) and propose a concrete construction satisfying the notion. The construction is unidirectional, multi-hop, and proved to be CPA secure in random oracle model. Canetti and Hohenberger et al. [10] present the first CCA secure bidirectional PRE with high efficiency in re-encryption. The security is based on the decisional bilinear Diffie-Hellman (DBDH) assumption in the standard model. In order to achieve CCA security, they leverage one-time signature to guarantee integrity of ciphertext. A new notion, called key privacy, is introduced in [5] in the sense that a proxy cannot obtain the identities of delegator and delegatee from a given re-encryption key (i.e. achieving anonymity). Their construction is with CPA security in the standard model. A few replayable CCA (RCCA) secure unidirectional PRE schemes in the standard model have been proposed by Libert and Vergnaud et al. [20]. Aono et al. [4] introduce the first lattice-based PRE based on the LWE problem, which is CPA secure in the standard model. Kirshanova et al. [18] present a CCA1 secure PRE on top of [23].

The aforementioned schemes, however, cannot provide the revocability of decryption rights delegation. We compare our scheme with other related PRE schemes in Table 1 in terms of functionality and security. We state that our scheme is the first of its type achieving revocability and CCA security with LWE in the standard model.

### 1.3 Paper Outline

The rest of the paper is organized as follows. In section 2, some basic definitions, hard problems, and some conclusions in lattices are given. In section 3, we present the definition of revocable PRE (RPRE) and further formalize the security model. In section 4, we give a concrete RPRE scheme and prove its security in the standard model. In section 5, we conclude our work.

## 2 Preliminaries

### 2.1 Notation

Throughout the paper we say that a function in $n$ is *negligible*, denoted by $negl(n)$, if it vanishes faster than the inverse of any polynomial in $n$. We say that a probability $p(n)$ is *overwhelming* if $1 - p(n)$ is negligible. The statistical distance between two distribution $\mathcal{X}$ and $\mathcal{Y}$(or two random variables having those distributions), viewed as functions over a countable domain $\mathcal{D}$, is defined as $\Delta(\mathcal{X};\mathcal{Y}) = \frac{1}{2}\sum_{s\in\mathcal{D}} |Pr[\mathcal{X}=s] - Pr[\mathcal{Y}=s]|$. We say that $\mathcal{X}$ and $\mathcal{Y}$ are statistically close if $d(\lambda) = \Delta(\mathcal{X}(\lambda);\mathcal{Y}(\lambda))$ is a negligible function of $\lambda$, where $\mathcal{X}(\lambda)$ and $\mathcal{Y}(\lambda)$ be ensembles of random variables.

We denote column vectors by lower-case bold letters(e.g., $\mathbf{x}$) and matrices by upper-case bold letters(e.g., $\mathbf{X}$). We identity a matrix $\mathbf{X}$ with the ordered set $\{\mathbf{x}_j\}$ of its column vectors, and let $\mathbf{X}||\mathbf{X}'$ denote the concatenation of the matrices $\mathbf{X}, \mathbf{X}'$. And we define $\|\mathbf{X}\| = max_j\|\mathbf{x}_j\|$, where $\|\cdot\|$ denotes the Euclidean norm, $\langle\cdot\rangle$ denotes inner product.

## 2.2 Lattice Definiton

**Definition 1** (Integer Lattice [15, 22]). *Let* $\mathbf{B} = [\mathbf{b_1}|\dots|\mathbf{b_m}] \in \mathbb{R}^{m\times m}$ *be an* $m \times m$ *matrix whose columns are linearly independent vectors* $\mathbf{b_1}, \dots, \mathbf{b_m} \in \mathbb{R}^m$. *The m-dimensional full-rank lattice* $\Lambda$ *generated by* $\mathbf{B}$ *is the set,*

$$\Lambda = \mathcal{L}(\mathbf{B}) = \{\mathbf{y} \in \mathbb{R}^m \quad s.t. \quad \exists \mathbf{s} \in \mathbb{Z}^m, \mathbf{y} = \mathbf{Bs} = \sum_{i=1}^{m} \mathbf{s_i}\mathbf{b_i}\}$$

*Here, we are interested in integer lattices, i.e., when* $\mathcal{L}$ *is contained in* $\mathbb{Z}^m$. *We let* $det(\Lambda)$ *denote the determinant of* $\Lambda$. *The dual lattice of* $\Lambda$, *denoted* $\Lambda^*$, *is defined to be* $\Lambda^* = \{\mathbf{x} \in \mathbb{R}^n : \forall \mathbf{v} \in \Lambda, \langle \mathbf{x}, \mathbf{v}\rangle \in \mathbb{Z}\}$.

**Definition 2** (q-ary lattice). *For prime* $q$, $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ *and* $\mathbf{u} \in \mathbb{Z}_q^n$, *define:*

$$\Lambda_q(\mathbf{A}) := \{\mathbf{e} \in \mathbb{Z}^m \quad s.t. \quad \exists \mathbf{s} \in \mathbb{Z}_q^n \text{ where } \mathbf{A}^\top \mathbf{s} = \mathbf{e} \bmod q\}$$

$$\Lambda_q^\perp(\mathbf{A}) := \{\mathbf{e} \in \mathbb{Z}^m \quad s.t. \quad \mathbf{Ae} = \mathbf{0} \bmod q\}$$

$$\Lambda_q^{\mathbf{u}}(\mathbf{A}) := \{\mathbf{e} \in \mathbb{Z}^m \quad s.t. \quad \mathbf{Ae} = \mathbf{u} \bmod q\}$$

*We can observe that if* $\mathbf{t} \in \Lambda_q^{\mathbf{u}}(\mathbf{A})$ *then* $\Lambda_q^{\mathbf{u}}(\mathbf{A}) = \Lambda_q^\perp(\mathbf{A}) + \mathbf{t}$ *and hence* $\Lambda_q^{\mathbf{u}}(\mathbf{A})$ *is a shift of* $\Lambda_q^\perp(\mathbf{A})$.

## 2.3 The Gram-Schmidt Norm

**Definition 3** (Gram-Schmidt norm). *Let* $\mathbf{S}$ *be a set of vectors* $\mathbf{S} = \{\mathbf{s_1}, \dots, \mathbf{s_k}\}$ *in* $\mathbb{R}^m$. *We use the following standard notations:*

- $\|\mathbf{S}\|$ denotes the $L_2$ length of the longest in $\mathbf{S}$, i.e., $\max_{1\le i\le k}\|\mathbf{s_i}\|$.
- $\tilde{\mathbf{S}} := \{\tilde{\mathbf{s}}_1, \dots, \tilde{\mathbf{s}}_k\} \subset \mathbb{R}^m$ denotes the Gram-Schmidt orthogonalization of the vectors $\mathbf{s_1}, \dots, \mathbf{s_k}$ taken in that order.

We refer to $\|\tilde{\mathbf{S}}\|$ as the Gram-Schmidt norm of $\mathbf{S}$.

**Lemma 1** ([13], Lemma 7.1). *There is a deterministic poly-time algorithm* $ToBasis(\mathbf{S}, \mathbf{B})$ *that, given a full rank set* $S$ *of lattice vectors in* $\Lambda = \mathcal{L}(\mathbf{B})$, *outputs a basis* $T$ *of* $\Lambda$ *such that* $\|\tilde{\mathbf{t}}_\mathbf{i}\| \le \|\tilde{\mathbf{s}}_\mathbf{i}\|$ *for all i.*

In 1996, Ajtai [3] showed how to sample an essentially uniform matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ with an associated basis $\mathbf{S_A}$ of $\Lambda_q^\perp(\mathbf{A})$ with low Gram-Schmidt norm. Here we use an improved algorithm from [1]. The following Theorems 3.2 are derived from [1] taking $\sigma := \frac{1}{3}$.

**Theorem 1.** *Let* $q \geqslant 3$ *be odd and* $m := \lceil 6n \log q \rceil$. *There is a probabilistic polynomial-time algorithm* TrapGen(q,n) *that outputs a pair* $(\mathbf{A} \in \mathbb{Z}_q^{n\times m}, \mathbf{S} \in \mathbb{Z}^{n\times m})$ *such that* $\mathbf{A}$ *is statistically close to a uniform matrix in* $\mathbb{Z}_q^{n\times m}$ *and* $\mathbf{S}$ *is a basis for* $\Lambda_q^\perp(\mathbf{A})$ *satisfying*

$$\|\widetilde{\mathbf{S}}\| \leqslant O(\sqrt{n \log q}) \quad and \quad \|\mathbf{S}\| \leqslant O(n \log q)$$

*with all but negligible probability in* $n$.

## 2.4 The LWE Problems

In this paper, the security of our construction is reduced to the *learning with errors* problem, which may be seen as average case problem related to the family of lattices described above.

**Definition 4** (Learning with Errors [26]). *For a prime* $q$, *a positive integer* $n$, *and a distribution* $\chi$ *over* $\mathbb{Z}_q$, *the* $LWE_{q,\chi}$ *problem is to distinguish, given oracle access to any desired* $m = poly(n)$ *samples, between the distribution* $A_{\mathbf{s},\chi}$ *(for uniformly random and secret* $\mathbf{s} \in \mathbb{Z}_q^n$*)and the uniform distribution over* $\mathbb{Z}_q^n \times \mathbb{Z}_q$.

We give an outline of Gaussian distributions over lattice. For any $s > 0$ and dimension $m \ge 1$, the Gaussian function $\rho_s : \mathbb{R}^m \to (0, 1]$ is defined as $\rho_s(\mathbf{x}) = exp(-\pi\|\mathbf{x}\|^2/s^2)$. For any coset $\Lambda_\mathbf{y}^\perp(A)$, and probability zero elsewhere. We summarize several facts from the literature about discrete Gaussian over lattices, again specialized to our family of interest.

**Lemma 2** ([24], Lemma 4.4). *For any n-dimensional lattice* $\Lambda$, *vector* $\mathbf{c} \in \mathbb{R}^n$, *and reals* $0 < \epsilon < 1$, $s \geqslant \eta_\epsilon(\Lambda)$*(the smoothing parameter* $\eta_\epsilon(\Lambda)$ *is the smallest real* $s > 0$ *such that* $\rho_{1/s}(\Lambda^*\backslash\{0\}) \leqslant \epsilon$*), we have*

$$\Pr_{\mathbf{x}\sim\mathcal{D}_{\Lambda,s,\mathbf{c}}}\{\|\mathbf{x} - \mathbf{c}\| > s\sqrt{n}\} \leqslant \frac{1+\epsilon}{1-\epsilon} \cdot 2^{-n}$$

**Lemma 3** ([15]). *There are two PPT algorithms SampeGaussia*$(\mathbf{A}, \mathbf{T_A}, \sigma, \mathbf{c})$ *and a PPT algorithm SampePre*$(\mathbf{A}, \mathbf{T_A}, \sigma, \mathbf{u})$*, the former returns* $\mathbf{x} \in \Lambda_q^{\perp}(\mathbf{A})$ *drawn from a distribution statistically close to* $\mathcal{D}_{\Lambda,s,\mathbf{c}}$*, and the latter returns* $x \in \Lambda_q^{\mathbf{u}}(\mathbf{A})$ *sampled from a distribution statistically close to* $\mathcal{D}_{\Lambda_q^{\mathbf{u}}(\mathbf{A}),\sigma}$*, whenever* $\Lambda_q^{\mathbf{u}}(\mathbf{A})$ *is not empty, where* $\mathbf{T_A}$ *be a basis for* $\Lambda_q^{\perp}(\mathbf{A})$ *and* $\sigma \geqslant \|\widetilde{\mathbf{T_A}}\|\omega(\sqrt{logm})$*, for* $\mathbf{c} \in \mathbb{R}^m$ *and* $\mathbf{u} \in \mathbb{Z}_q^n$*.*

## 2.5 Encoding Vectors as Matrices

Our construction needs a function $\mathbf{H_f} : \mathbb{Z}_q^n \to \mathbb{Z}_q^{n\times n}$ which is able to map vectors (in $\mathbb{Z}_q^n$) to matrices (in $\mathbb{Z}_q^{n\times n}$), and the security proof of our scheme requires the function satisfying *strong injectivity*, i.e., for two distinct vectors $\mathbf{u}, \mathbf{v}$, $det(\mathbf{H_f}(\mathbf{u}) - \mathbf{H_f}(\mathbf{v})) \neq 0$.

**Definition 5.** *Let a prime* $q$ *and a positive integer* $n$*. We say that a function* $\mathbf{H_f} : \mathbb{Z}_q^n \leftarrow \mathbb{Z}_q^{n\times n}$ *is an encoding with full rank differences (FRD) if:*

1. *For all distinct* $\mathbf{u}, \mathbf{v} \in \mathbb{Z}_q^n$*, the matrix* $\mathbf{H_f}(\mathbf{u}) - \mathbf{H_f}(\mathbf{v}) \in \mathbb{Z}_q^{n\times n}$ *is full rank.*
2. $\mathbf{H_f}$ *is computable in polynomial time.*

We use an injective FRD encoding function in [2], and a short instruction is as follows. We have the finite field $\mathbb{Z}_q$, a polynomial $g \in \mathbb{F}[X]$ of degree less than n, and let coeffs$(g) \in \mathbb{F}^n$ be defined n-vector which element is coefficients of $g$. Let $f$ be some polynomial of degree n in $\mathbb{F}[X]$ that is irreducible. For input $\mathbf{u} = (u_0, u_1, \ldots, u_{n-1})$ define the polynomial $g(x) = \sum_{i=0}^{n-1} u_i x^i$.

Define $\mathbf{H_f}(\mathbf{u})$ as

$$\mathbf{H_f}(\mathbf{u}) := \begin{pmatrix} \text{coeffs}(g) \\ \text{coeffs}(x \cdot g \bmod f) \\ \text{coeffs}(x^2 \cdot g \bmod f) \\ \vdots \\ \text{coeffs}(x^{n-1} \cdot g \bmod f) \end{pmatrix} \in \mathbb{F}^{n\times n} \tag{1}$$

**Theorem 2** ([12]). *Let* $\mathbb{F}$ *be a field and* $f$ *a polynomial in* $\mathbb{F}[X]$*. If* $f$ *is irreducible in* $\mathbb{F}(X)$ *then the function* $\mathbf{H_f}(\mathbf{u})$ *defined in (1) is an encoding with full rank differences.*

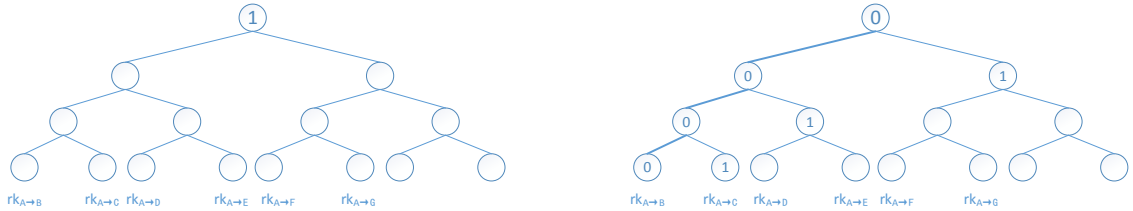
## 2.6 The Binary-tree Data Structure

In order to reduce the number of re-encryption key update (on the side of delegator), we use a binary tree [8] and further assign re-encryption key to leaf node of the tree. Each user has keys computed on of all nodes on the path from the leaf node corresponding to that user to the root node for the decryption of ciphertext encrypted under the time period $\mathbf{t}$. When no user is revoked, the delegator just needs to submit the key update computed on the node of binary tree to the proxy. When a certain number of users are revoked, delegator first locates the minimal set of nodes in the tree which contains a common ancestor among all the leaf nodes for non-revoked users.

We use the following notations. $\mathsf{BT}$ denotes the binary tree. If $\mathsf{root}$ denotes root node and $\nu$ denotes a leaf node, the $\mathsf{Path}(\nu)$ denotes the set of node on the path from $\nu$ to the $\mathsf{root}$(including $\nu$ and the $\mathsf{root}$). If $\theta$ is a non-leaf node, $\theta_l$ and $\theta_r$ denote the left and right child of $\theta$. We assume that the mark of each re-encryption key, such as $rk_{A\to B}, rk_{A\to C}, rk_{A\to D}, \ldots$, is assigned to each leaf node $\nu$. Upon system registration, the delegator provides the proxy with a set of distinct re-encryption keys for each node in $\mathsf{Path}(\nu)$.

We here present an $\mathsf{KUNodes}$ algorithm which takes as input a binary tree $\mathsf{BT}$, a time $\mathbf{t}$, and a revocation list $\mathsf{RL}$. The delegator is able to determine a minimal set $\mathsf{Y}$ which includes none ancestor of node in $\mathsf{RL}$ with corresponding time on or before $\mathbf{t}$ (revoked re-encryption key), and all other leaf nodes (non-revoked re-encryption key) have exactly one ancestor in the set $\mathsf{Y}$. In other words, the algorithm $\mathsf{KUNodes}$ finds a minimal set containing ancestors of non-revoked re-encryption key. Its output is all non-revoked children of the revoked nodes. The delegator further publishes a re-encryption key update for all nodes in $\mathsf{Y}$.

A mark of re-encryption key is assigned to every leaf node $\nu$, and then to form a valid re-encryption key corresponding to the time $t$ if the set $\mathsf{Y}$ and $\mathsf{Path}(\nu)$ have a common node. Through this operation, every revocation list $\mathsf{RL}$ only needs the delegator to carry out the logarithmic work of the maximal number of re-encryption keys and linear number of revoked re-encryption keys. Figure 1a shows an example where there is no revoked user, while Bob is revoked and those nodes flagged by "1" are included in set $\mathsf{Y}$ in Figure 1b.




(a) No user is revoked (b) User B is revoked

Figure 1: Examples of binary tree revocation structure

**Algorithm 1** KUNodes

**Require:** BT, RL, $\mathbf{t}$
1: $\mathsf{X}, \mathsf{Y} \leftarrow \emptyset$
2: **for** $(\nu_i, \mathbf{t_i}) \in \mathsf{RL}$ **do**
3: **if** $\mathbf{t_i} \leqslant \mathbf{t}$ **then** add $\mathsf{Path}(\nu_i)$ to $\mathsf{X}$ **end if**
4: **end for**
5: **for** $\theta \in \mathsf{X}$ **do**
6: **if** $\theta_l \notin \mathsf{X}$ **then** add $\theta_l$ to $\mathsf{Y}$ **end if**
7: **if** $\theta_r \notin \mathsf{X}$ **then** add $\theta_r$ to $\mathsf{Y}$ **end if**
8: **end for**
9: **if** $\mathsf{Y} = \emptyset$ **then** add root to $\mathsf{Y}$ **end if**
10: **return** $\mathsf{Y}$

# 3 Syntax of RPRE

We start with defining the general syntax of a revocable proxy re-encryption scheme.

## 3.1 System Definition

**Definition 6** (Revocable Proxy Re-encryption)**.** *A proxy re-encryption with revocability includes nine probabilistic polynomial time (PPT) algorithms, namely* **Setup**, **KeyGen**, **Encrypt**, **Decrypt**, **ReKeyGen**, **UpKey**, **ReEnc**, **ReKeyRev**, *and* **UpCip** *with associated message space $M$, time space $\mathcal{T}$. We assume that the size of $\mathcal{T}$ is polynomial in the security parameter $1^n$. Each algorithm is run by either one of parties, proxy, delegator, and delegatee. The proxy maintains a revocation list RL and state ST. In what follows, an algorithm is called stateful if it updates RL or ST. Here we regard time space as discrete rather than continuous.*

- $(param, RL, ST) \leftarrow$ **Setup** $(1^n)$: Intake a security parameter $n$, the setup algorithm outputs a global public parameter $param$, a revocation list RL(initially empty), and a state ST, where $n \in \mathbb{N}$.
- $(pk, sk) \leftarrow$ **KeyGen** $(param)$: Intake $param$, the key generation algorithm outputs a public key $pk$ and a secret key $sk$. We let $param$ include into the following algorithms as an implicit input.
- $C \leftarrow$ **Encrypt**$(pk, M, \mathbf{t})$: Intake a public key $pk$, a message $M \in \mathcal{M}$, and a time $\mathbf{t} \in \mathcal{T}$, the encryption algorithm outputs a ciphertext $C \in \mathcal{C}$.
- $rk_{pk\to pk'|\mathbf{t}} \leftarrow$ **ReKeyGen**$(pk, sk, pk', \mathbf{t}, RL, ST)$: Intake two public keys $pk, pk'$, a private key $sk$, a time $\mathbf{t} \in \mathcal{T}$, the revocation list RL, and the state ST, the re-encryption key generation algorithm outputs a re-encryption key $rk_{pk\to pk'|\mathbf{t}}$ or a special symbol $\perp$ indicating that $pk'$ has been revoked.
- $upkey_{\mathbf{t}\to\mathbf{t}'} \leftarrow$ **UpKey**$(pk, sk, \mathbf{t}, \mathbf{t}', ST)$: Intake a public key $pk$, a private key $sk$, two time periods $\mathbf{t} < \mathbf{t}' \in \mathcal{T}$ and the state ST, the update key generation algorithm outputs a update key $upkey_{\mathbf{t}\to\mathbf{t}'}$.
- $C_{\mathbf{t}'} \leftarrow$ **UpCip**$(pk, C, upkey_{\mathbf{t}\to\mathbf{t}'}, ST)$: Intake the public key $pk$, the state $ST$, a cipherext $C$ of $pk$ at time $\mathbf{t}$ and a update key $upkey_{\mathbf{t}\to\mathbf{t}'}$, the update ciphertext algorithm outputs the updated ciphertext of $pk$ at time $\mathbf{t}'$, where $\mathbf{t} < \mathbf{t}'$.

- $C_R \leftarrow \mathbf{ReEnc}(rk, C)$: Intake a proxy re-encryption key $rk$ and a ciphertext $C$, the re-encryption algorithm outputs a ciphertext $C_R$.
- $RL \leftarrow \mathbf{ReKeyRev}(pk, pk', \mathbf{t}, RL, ST)$: Intake public keys $pk, pk'$, a revocation time $\mathbf{t} \in \mathcal{T}$, the revocation list $RL$, and the state $ST$, the re-encryption key revocation algorithm outputs an updated revocation list $RL$.
- $M \leftarrow \mathbf{Decrypt}(sk, C)$: Intake a secret key $sk$ and a ciphertext $C$, the decryption algorithm outputs a message $M \in \mathcal{M}$.

**Correctness**. The correctness requires that for all $n \in \mathbb{N}$, $\mathbf{t} \in \mathcal{T}$, $M \in \mathcal{M}$, all $(pk, sk) \leftarrow \mathbf{KeyGen}(param)$, and all possible valid states $ST$ and revocation lists $RL$, if a user with the public key $pk$ was not revoked before or, at time $\mathbf{t}$, the followings hold:

- We have $\mathbf{Decrypt}(sk, \mathbf{Encryption}(pk, M, \mathbf{t_0})) = M$, where $\mathbf{t_0} \leq \mathbf{t}$;
- We have $\mathbf{Decrypt}(sk, \mathbf{UpCip}(pk, \mathbf{Encryption}(pk, M, \mathbf{t_0}), \mathbf{UpKey}(pk, sk, \mathbf{t_0}, \mathbf{t_1}, ST), \mathrm{ST})) = M$, where $\mathbf{t_0} < \mathbf{t_1} \leq \mathbf{t}$;
- Given a re-encryption key $rk_{pk \to pk'|\mathbf{t_1}} \leftarrow \mathbf{ReKeyGen}(pk, sk, pk', \mathbf{t_1}, RL, ST)$ and for any $C \leftarrow \mathbf{Encrypt}(pk, M, \mathbf{t_1})$, we have $\mathbf{Decrypt}(sk', ReEnc(rk_{pk \to pk'|\mathbf{t_1}}, C)) = M$, where $\mathbf{t_1} \leq \mathbf{t}$.

## 3.2 System Workflow

We here give a concise flow chart to illustrate our system. We assume there are four system users, in which Alice is delegator, and the rest of them are delegatees. There are two blocks in Figure 2, indicating the workflow before and after revocation, respectively. In general, our system works as follows. A data owner, Alice, encrypts her data and further uploads the ciphertext to a semi-trusted cloud server, who acts as a proxy. To fulfil secure data sharing, with the re-encryption keys (given by Alice), $rk_{Alice \to Bob}$, $rk_{Alice \to Cindy}$, $rk_{Alice \to Dale}$, the proxy can convert Alice's encryption to the ciphertexts intended for Bob, Cindy and Dale, respectively. If Alice decides to revoke the decryption rights delegation of Bob, she may send a request with necessary information to the proxy. By the necessary information, we mean the new re-encryption keys for Cindy and Dale along with a ciphertext update key. The proxy further updates the ciphertext of Alice by using the update key (without compromising the underlying plaintext). We here note that the new re-encryption keys are corresponding to the updated ciphertext of Alice, so that the proxy is allowed to convert the updated ciphertexts for Cindy and Dale. In our concrete construction (which is introduced in Section 4), we relate time period with encryption such that delegation of decryption rights (i.e. re-encryption key) is also limited to a time slot. A valid re-encryption requires that the time slot embedded into the re-encryption key must match the one associated with the ciphertext. The ciphertext update stage lifts the ciphertext from an old time slot say $\mathbf{t}$ to a new one $\mathbf{t}'$, so that the re-encryption key (from Alice to Bob) under $\mathbf{t}$ is effective no more in re-encryption. In this way, re-encryption is only valid for the non-revoked users, Cindy and Dale. It is worth of mentioning that we make use of binary tree structure to reduce the re-encryption key update complexity to $O(logN)$ in this paper, where $N$ is the number of delegatee.

## 3.3 Security Notion

We formalize the RPRE-CCA security below. Our security model considers not only the standard notion of PRE security but also the re-encryption key revocability.

**RPRE-CCA Game.** Let $1^n$ be the security parameter, $\mathcal{A}$ be any PPT adversary. Consider the following experiment for a RPRE scheme $\Pi$ with a plaintext space $\mathcal{M}$, a key space $\mathcal{K}$, a ciphertext space $\mathcal{C}$ and the revocation list $RL$ and the state $ST$:

Before proceeding to security game, we divide all users into two categories: honest user (HU) and corrupted user (CU). HU is a set of honest users only allowing $\mathcal{A}$ to know the corresponding public keys, while CU is a set of corrupted users manipulated by $\mathcal{A}$.

**Setup**. Output the public parameters $param$, a revocation list RL (initially empty), and a state ST, where $param$ is sent to $\mathcal{A}$. $\mathcal{A}$ is given the target $pk^*$ and time $\mathbf{t}^*$, labeling it as honest.

**Phase1**. $\mathcal{A}$ can adaptively make a polynomial number of queries of the following oracles $\mathbf{O} = (\mathbf{O_{KeyGen}}, \mathbf{O_{ReKeyGen}}, \mathbf{O_{UpKey}}, \mathbf{O_{ReEnc}}, \mathbf{O_{Dec}}, \mathbf{O_{ReKeyRev}}, \mathbf{O_{UpCip}})$:

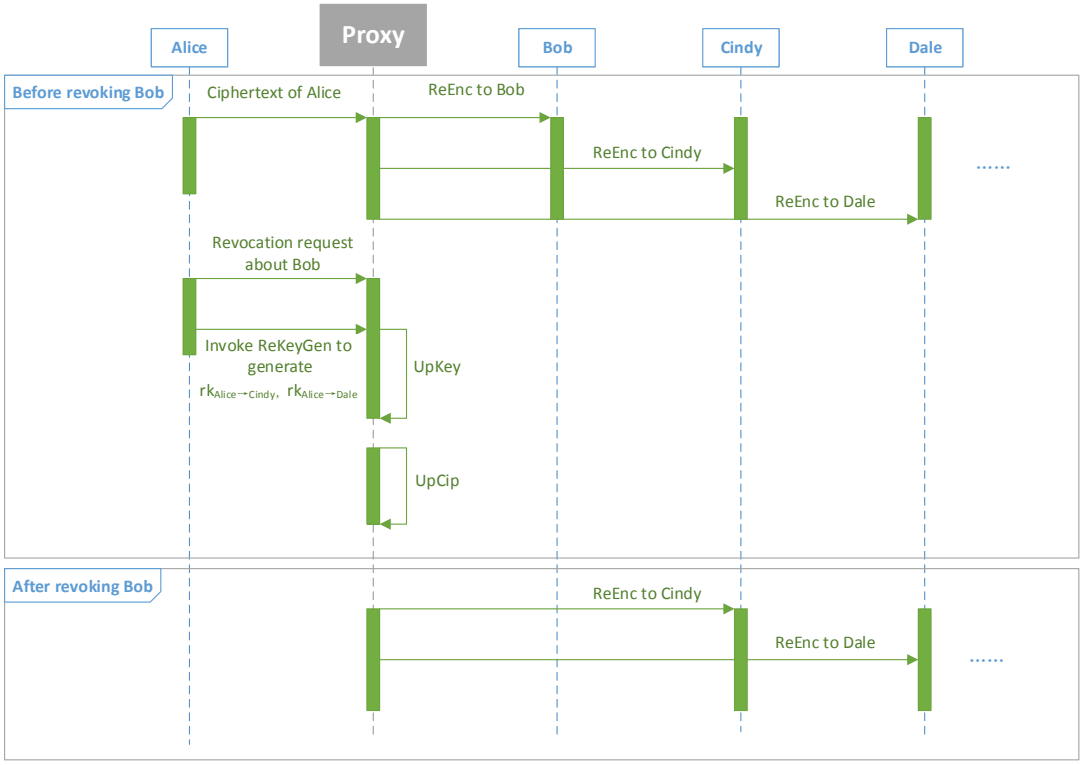



Figure 2: System Workflow

- $\mathbf{O_{KeyGen}}$: if $\mathcal{A}$ request a key of user $i$ with a tag *honest*, the challenger returns $pk_i$ and records $i \in HU$; otherwise, the challenger returns $(pk_i, sk_i)$ and records $i \in CU$, where $(pk_i, sk_i) \leftarrow KeyGen(param)$.
- $\mathbf{O_{ReKeyGen}}$: $\mathcal{A}$ is allowed to ask a re-encryption key query $rk_{pk \to pk'|\mathbf{t}}$ from $pk$ to $pk'$ under a time $\mathbf{t}$, the challenger responds by running the **ReKeyGen** algorithm to generate a re-encryption key $rk_{pk \to pk'|\mathbf{t}}$ for the adversary. If a query indicates that $pk = pk'$ or $pk \in HU, pk' \in CU$, it will be ignored. The adversary can repeat polynomial times for different couple of identities.
- $\mathbf{O_{ReEnc}}$: $\mathcal{A}$ is allowed to query re-encryption tuple $(pk, pk', \mathbf{t}, C_{pk})$, the challenger responds by running **ReKeyGen** algorithm to generate a re-encryption key $rk_{pk \to pk'|\mathbf{t}}$ and further computing ciphertext $C_{pk'}$ by running **ReEnc** algorithm. If $pk = pk'$ or $pk \in HU, pk' \in CU$, the query will be ignored.
- $\mathbf{O_{Dec}}$: $\mathcal{A}$ is allowed to ask a decryption query on $C$ of user $pk$ (where $pk \neq pk^*$), the challenger runs **Decrypt** to return $m$.
- $\mathbf{O_{ReKeyRev}}$: $\mathcal{A}$ outputs a tuple ($pk$, $\mathbf{t}$), the challenger updates RL by running **ReKeyRev**.
- $\mathbf{O_{UpKey}}$: $\mathcal{A}$ sends a tuple $(pk, \mathbf{t}, \mathbf{t}')$ to the challenger. The challenger runs the algorithm **UpKey** to generate $upkey_{\mathbf{t} \to \mathbf{t}'}$.
- $\mathbf{O_{UpCip}}$: $\mathcal{A}$ is allowed to query $(pk, \mathbf{t}, \mathbf{t}', C)$. The challenger runs the algorithm **UpCip** to convert ciphertext $C$ under $\mathbf{t}$ to the one under $\mathbf{t}'$, where $\mathbf{t} < \mathbf{t}'$.

**Challenge.** $\mathcal{A}$ outputs two equal length plaintext $m_0, m_1 \in \mathcal{M}$. The challenger picks a random bit $b \in \{0, 1\}$ and sets the challenge ciphertext to $C^* = \mathbf{Encrypt}(pk^*, m_b, \mathbf{t}^*)$.
**Phase 2.** Same as Phase 1.
**Guess.** $\mathcal{A}$ outputs a guess $b' \in \{0, 1\}$ and wins if $b = b'$.
The following restrictions must be hold:

- $\mathbf{O_{ReKeyGen}}$ and $\mathbf{O_{ReKeyRev}}$ must be queried in non-decreasing order, i.e., the time of the current queries must be later than or equal the time of previous queries.

- $\mathbf{O_{ReKeyRev}}$ cannot be queried at time $\mathbf{t}$, if it has been queried at $\mathbf{t}$ once.
- When $\mathbf{O_{ReKeyGen}}$ is queried at time $\mathbf{t}^*$, user $pk^*$ must be in RL.
- $\mathcal{A}$ is not allowed to query the decryption oracle for the challenge ciphertext $C^*$ of $pk^*$ at time $\mathbf{t}^*$. If $C^*$ at time $\mathbf{t}^*$ is converted to $C^*$ at time $\mathbf{t}$ for $\mathbf{t}^* < \mathbf{t}$ by using $\mathbf{O_{UpCip}}$ or $\mathbf{O_{UpKey}}$, $\mathcal{A}$ still cannot access to the decryption oracle for the query of $C^*$ at time $\mathbf{t}$.

We refer to $\mathcal{A}$ in the above game as an RPRE-CCA adversary. We define the advantage of $\mathcal{A}$ in attacking an RPRE scheme $\epsilon$ as

$$Adv_{\epsilon,\mathcal{A}} = \left| Pr\left[ b = b' : \begin{array}{c} param \leftarrow \mathbf{Setup}(1^n, N) \\ (m_0, m_1) \leftarrow \mathcal{A}(param, pk^*, t^*)^{\mathbf{O}} \\ C^* \leftarrow \mathbf{Encrypt}(param, pk^*, m_b) \\ b' \leftarrow \mathcal{A}(C^*)^{\mathbf{O}} \end{array} \right] - \frac{1}{2} \right|$$

**Definition 7.** *We say that an RPRE scheme is CCA if for all probabilistic polynomial time algorithm $\mathcal{A}$ and negligible function $\varepsilon$, we always have that $Adv_{\epsilon,\mathcal{A}}$ is a negligible function, that is, $Adv_{\epsilon,\mathcal{A}} \leqslant \varepsilon$*

# 4 Construction

## 4.1 Intuition of Our Construction

We first consider how to establish a connection between public key and time period for a user. Recall that a public key in a PRE scheme consists of three parts: the first two parts $[\mathbf{A_0}|\mathbf{A_1}]$ (of the $pk = ([\mathbf{A_0}|\mathbf{A_1}|\mathbf{A_2}])$ are regarded as an entirety, and the last part $[\mathbf{A_2}]$ is regarded as the other entirety in our construction. The re-encryption key is generated in the following two steps. First, we choose two matrices randomly, namely, $\mathbf{A_R}$ and $\mathbf{A_L}$, such that the sum of the two equals the first two parts of $pk$. We extract $\mathbf{SRK_R}$ and $\mathbf{SRK_L}$ through a Gaussian sampler from the $\mathbf{A_R}$, $\mathbf{A_L}$, and further store them in the nodes $\theta \in Path(v)$ and $\theta \in \mathsf{KUNodes}(v)$, respectively. Second, we add a random vector $\mathbf{s}$ to protect the time period from being cancelled out during some computation intaking the part $\mathbf{A_2}$. $\mathbf{X}_{\theta,\mathbf{t}}$ is also extracted through a Gaussian sampler and stored in the node $\theta \in \mathsf{KUNodes}(v)$. The revocation of re-encryption key relies on if $\mathbf{SRK_R}$, $\mathbf{SRK_L}$, and $\mathbf{URK_t}$ are in the same node of the tree. In order to answer the re-encryption key query issued by adversary in the simulation, we divided the first two parts of $pk_A$ into $\mathbf{U}[\mathbf{A_{R_1}}|\mathbf{A_{R_2}}]$ and $[\mathbf{I}-\mathbf{U}][\mathbf{A_{L_1}}|\mathbf{A_{L_2}}]$ in accordance with a certain proportion of $\mathbf{U} : [\mathbf{I}-\mathbf{U}]$, where $\mathbf{U}$ is a unimodular matrix. Since the challenger of the security game will not have the private keys of honest users, we exploit Gaussian random sampling method to extract $R_1^*$ and $R_2^*$ from the Gauss distribution to construct honest user's public key $pk = ([\mathbf{A_0}| - \mathbf{A_0R_1}| - \mathbf{A_0R_2} - \mathbf{H_f}(\mathbf{t}^*)\mathbf{G}], \mathbf{H})$, and we add $-\mathbf{H_f}(\mathbf{t}^*)\mathbf{G}$ to each honest user's key. In this way, we enable the challenger to answer the queries of re-encryption key and decryption.

## 4.2 Our RPRE Scheme

Below we present our RPRE scheme from lattices.

- **Setup**$(1^n)$: On input a security parameter $1^n$, it chooses $r$ that is a fixed function $w(\sqrt{\log n})$. Set the modulus $q = p^e = poly(n)$ and $k = \mathcal{O}(\log n)$. The dimension of the public key is $m = l + 2nk$, where $l = \mathcal{O}(nk)$. We adopt the standard trapdoor generation function in [23] to build the gadget matrix $\mathbf{G}$ and the two functions associated with it, namely, the invert function $g_{\mathbf{G}}$ and the sampler $f_{\mathbf{G}}$. The trapdoors $\mathbf{R}_1, \mathbf{R}_2 \sim \mathcal{D}^{l\times nk}$ so that $(\mathbf{A_0}, \mathbf{A_0R_1}, \mathbf{A_0R_2})$ is $\mathsf{negl}(n)-$far from uniformly distribution $\mathbb{Z}_q^{n\times l} \times \mathbb{Z}_q^{n\times nk} \times \mathbb{Z}_q^{n\times nk}$. All $\mathbf{H} \in \mathbb{Z}_q^{n\times n}$ that are used in out scheme are invertible, and the difference between two such matrices, $\mathbf{H}' - \mathbf{H}''$, is also invertible. This occurs with a non-negligible probability of $(1-1/p)^e$ when $p$ is prime. Therefore we can always find those matrices with rejection samplings.

  The LWE error rate $1/\alpha = \mathcal{O}((nk)^3)\cdot r^3$. We define a encoding function as $\mathsf{enc}(\mathbf{m}) = \mathbf{Bm} \in \mathbb{Z}^{nk}$, which encodes the message space $\{0,1\}^{nk}$ to the cosets of $\Lambda/2\Lambda$ for the lattice $\Lambda = \Lambda(\mathbf{G}^\top)$ using any basis $\mathbf{B} \in \mathbb{Z}^{nk}$ of $\Lambda$, and this encoding can be efficiently inverted.

- **KeyGen**$(param)$: On input $\mathbf{A_0} \xleftarrow{\$} \mathbb{Z}_q^{n\times l}$, $\mathbf{R_1}, \mathbf{R_2} \xleftarrow{\$} \mathcal{D}^{l\times nk}$, an invertible matrix $\mathbf{H} \xleftarrow{\$} \mathbb{Z}_q^{n\times n}$. The public key is $pk = (\mathbf{A}, \mathbf{H})$ where $\mathbf{A} = [\mathbf{A_0}|\mathbf{A_1}|\mathbf{A_2}] = [\mathbf{A_0}|-\mathbf{A_0R_1}|-\mathbf{A_0R_2}] \in \mathbb{Z}_q^{n\times(l+2nk)}$. The private key is $sk = [\mathbf{R_1}|\mathbf{R_2}] \in \mathbb{Z}^{l\times 2nk}$.

- **Encryption**$(pk, m, \mathbf{t})$: On input $m \in \{0,1\}^{nk}$, a vector $\mathbf{r} \xleftarrow{\$} \mathbb{Z}_q^n$, a time vector $\mathbf{t} \leftarrow \mathbb{Z}_q^n$. Compose $\mathbf{A_u} = [\mathbf{A_0}|\mathbf{A_1} + \mathbf{HG}|\mathbf{A_2} + \mathbf{H_f(t)G}]$, and sample three error vectors $\mathbf{e_0} \xleftarrow{\$} \mathcal{D}_{\mathbb{Z},\alpha q}^l$, $\mathbf{e_1}, \mathbf{e_2} \xleftarrow{\$} \mathcal{D}_{\mathbb{Z},s}^{nk}$, where $s^2 = (\|\mathbf{e_0}\|^2 + l\alpha q^2) \cdot r^2$. The composed error vector is concatenated by the three vectors $\mathbf{e} = (\mathbf{e_0}|\mathbf{e_1}|\mathbf{e_2}) \in \mathbb{Z}^m$. Let

$$\mathbf{b}^\top = 2(\mathbf{r}^\top[\mathbf{A_0}|\mathbf{A_1} + \mathbf{HG}|\mathbf{A_2} + \mathbf{H_f(t)G}] \bmod q) + \mathbf{e}^\top + (\mathbf{0}, \mathbf{0}, enc(\mathbf{m}))^\top \bmod 2q$$

  where the dimension of first zero vector is $l$, and that of the second is $nk$. Return the ciphertext $c = \mathbf{b} \in \mathbb{Z}_{2q}^m$.

- **Decryption**$(pk, sk, c)$: Recall that $pk = [\mathbf{A}, \mathbf{H}]$, $sk = [\mathbf{R_1}|\mathbf{R_2}]$, and $c = \mathbf{b}$. Compute $\mathbf{A_u} = [\mathbf{A_0}|\mathbf{A_1} + \mathbf{HG}|\mathbf{A_2} + \mathbf{H_f(t)G}]$ through matrix $\mathbf{H_u}$, and perform the following steps:

  1. Output $\perp$ if the form of $c$ is invalid or $\mathbf{H_f(t)} = 0$. Otherwise, call the algorithm $\mathsf{Invert}^{\mathcal{O}}([\mathbf{R_1}|\mathbf{R_2}], \mathbf{A_u}, \mathbf{b}, \mathbf{H_f(t)})$ to get values $\mathbf{z} \in \mathbb{Z}_q^n$ and $\mathbf{e} = (\mathbf{e_0}, \mathbf{e_1}, \mathbf{e_2}) \in \mathbb{Z}_q^l \times \mathbb{Z}_q^{nk} \times \mathbb{Z}_q^{nk}$ so that $\mathbf{b}^\top = \mathbf{z}^\top\mathbf{A_u} + \mathbf{e}^\top \bmod q$. If the call to $\mathsf{Invert}$ fails for any reason, output $\perp$.
  2. If $\|\mathbf{e_0}\| \geqslant \alpha q\sqrt{l}$ or $\mathbf{e_1}, \mathbf{e_2} \geqslant \alpha q\sqrt{2lnk} \cdot w(\sqrt{\log n})$, output $\perp$.
  3. Let $\mathbf{v} = \mathbf{b} - \mathbf{e} \bmod 2q$, parsed as $\mathbf{v} = (\mathbf{v_0}, \mathbf{v_1}, \mathbf{v_2}) \in \mathbb{Z}_{2q}^l \times \mathbb{Z}_{2q}^{nk} \times \mathbb{Z}_{2q}^{nk}$. If $\mathbf{v_0} \notin 2\Lambda(\mathbf{A_0}^\top)$, output $\perp$. Otherwise, compute

$$\mathbf{v}^\top \begin{pmatrix} \mathbf{R_1} & \mathbf{R_2} \\ \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{pmatrix} \bmod 2q \in \mathbb{Z}_{2q}^{nk}$$

     and apply $encode^{-1}$ to the last $nk$ coordinates if it exists, otherwise output $\perp$.

- **ReKeyGen**$(pk, sk, pk', \mathbf{t}, \mathsf{RL}, \mathsf{ST})$:

  Recall that $pk = ([\mathbf{A_0}|\mathbf{A_1}|\mathbf{A_2}], \mathbf{H})$, $sk = [\mathbf{R_1}|\mathbf{R_2}]$, and $pk' = ([\mathbf{A'_0}|\mathbf{A'_1}|\mathbf{A'_2}], \mathbf{H'})$. In this step, the algorithm's input is the public key of the delegator and the delegatee, and a part of re-encryption key needed in our construction is generated by the private key of the delegator. The specific process is as follows:

  1. For each $\theta \in \mathsf{Path}(\upsilon)$, if $\mathbf{A_R}$ and $\mathbf{A_L}$ have not to been defined, the first two items of the delegatee's public key $[\mathbf{A'_0}|\mathbf{A'_1} + \mathbf{H'G}]$ would be divided into two parts: denoted by $\mathbf{A_R}$ and $\mathbf{A_L}$ satisfy $[\mathbf{A'_0}|\mathbf{A'_1} + \mathbf{H'G}] = \mathbf{A_R} + \mathbf{A_L}$.
     We make $\mathbf{A}_R$ parse as two matrices $\mathbf{A_{R_1}} \in \mathbb{Z}_q^{n\times l}$ and $\mathbf{A_{R_2}} \in \mathbb{Z}_q^{n\times nk}$. Making use of the first part of the secret key $\mathbf{R_1}$(the Gaussian matrix) and the invertible matrix $\mathbf{H} \in \mathbb{Z}_q^{n\times n}$ from the public key. More concretely, we sample column wise so that for each column of $\mathbf{A_{R_1}}$ and obtain an $l + nk$ dimensional column of the part of re-encryption key by executing $\mathsf{Sample}^{\mathcal{O}}$. And we derive an $(l + nk) \times l$ matrix after sampling $l$ times and parse it as two matrices $\mathbf{X_{00}} \in \mathbb{Z}^{l\times l}$ and $\mathbf{X_{10}} \in \mathbb{Z}^{nk\times l}$ with Gaussian entries of parameter $s$.

$$[\mathbf{A_0}| - \mathbf{A_0R_1} + \mathbf{HG}] \begin{pmatrix} \mathbf{X_{00}} \\ \mathbf{X_{10}} \end{pmatrix} = [\mathbf{A_{R_1}}]$$

     And continue sampling for the cosets obtained from the columns of $\mathbf{A_{R_2}}$, we derive an $(l+nk)\times nk$ matrix after sampling $nk$ times and parse it as two mtrices $\mathbf{X_{01}} \in \mathbb{Z}^{l\times nk}$ and $\mathbf{X_{11}} \in \mathbb{Z}^{nk\times nk}$ with Gaussian entries of parameter $s\sqrt{\frac{l}{2}}$:

$$[\mathbf{A_0}| - \mathbf{A_0R_1} + \mathbf{HG}] \begin{pmatrix} \mathbf{X_{01}} \\ \mathbf{X_{11}} \end{pmatrix} = [\mathbf{A_{R_2}}]$$

     Combined with the above two steps, we could get

$$[\mathbf{A_0}| - \mathbf{A_0R_1} + \mathbf{HG}] \begin{pmatrix} \mathbf{X_{00}} & \mathbf{X_{01}} \\ \mathbf{X_{10}} & \mathbf{X_{11}} \end{pmatrix} = [\mathbf{A_{R_1}}|\mathbf{A_{R_2}}] = [\mathbf{A_R}]$$

     We denote $\begin{pmatrix} \mathbf{X_{00}} & \mathbf{X_{01}} \\ \mathbf{X_{10}} & \mathbf{X_{11}} \end{pmatrix}$ by $\mathbf{X}_{\theta,\mathbf{R}}$, then store $\mathbf{A_R}\mathbf{X}_{\theta,\mathbf{R}}$ in node $\theta \in \mathsf{Path}(\upsilon)$, and output $\mathbf{SRK_R} = (\theta, \mathbf{X}_{\theta,\mathbf{R}})_{\theta\in\mathsf{Path}(\upsilon)}$.

2. For each $\theta \in \mathsf{KUNodes}(\upsilon)$, if $\mathbf{A_R}$ and $\mathbf{A_L}$ are not defined, by the above definition we definite $\mathbf{A_R}$ and $\mathbf{A_L}$ to satisfy $[\mathbf{A'_0}|\mathbf{A'_1}+\mathbf{H'G}] = \mathbf{A_R}+\mathbf{A_L}$.

   The same calculation method as the first step is adopted to generate $\mathbf{X}_{\theta,\mathbf{L}} = \begin{pmatrix} \mathbf{X'_{00}} & \mathbf{X'_{01}} \\ \mathbf{X'_{10}} & \mathbf{X'_{11}} \end{pmatrix}$ correspond to $\mathbf{A_L}$, and store $\mathbf{A_L}$ and $\mathbf{X}_{\theta,\mathbf{L}}$ in the node $\theta \in \mathsf{KUNodes}$, that is $[\mathbf{A_0}|-\mathbf{A_0R_1}+\mathbf{HG}]\begin{pmatrix} \mathbf{X'_{00}} & \mathbf{X'_{01}} \\ \mathbf{X'_{10}} & \mathbf{X'_{11}} \end{pmatrix} = [\mathbf{A_L}]$. And output $\mathbf{SRK_L} = (\theta, \mathbf{X}_{\theta,\mathbf{L}})_{\theta\in\mathsf{KUNodes}(\upsilon)}$.

3. In this step, our algorithm whose input is the public key of the delegator and the delegatee and the private key of the delegator is used to generate the time-control part of the proxy re-encryption key needed in the scheme. The specific process is as follows:

   Select matrix function $\mathbf{H_f}(\mathbf{x}) : \mathbf{x} \to \mathbb{Z}_q^{n\times n}, \mathbf{x} \in \mathbb{Z}_q^n$ whose input is the time vector $\mathbf{t} \in \mathbb{Z}_q^n$ and the random vector $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^n$. In order to hold the containing time $\mathbf{t}$ item in the final term, we have to make the containing $\mathbf{t}$ item on both sides of the equation inequality, to avoid it being eliminated(under the assumption that the third column and the third row of proxy re-encryption key are identity matrix $\mathbf{I}$). So we introduce a random vector $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^n$ in $\mathbf{A_2}+\mathbf{H_f}(\mathbf{t})\mathbf{G}$, make the right-hand side of the equation $\mathbf{A'_2}+\mathbf{H_f}(\mathbf{t}+\mathbf{s})\mathbf{G}$:

$$[\mathbf{A_0}|\mathbf{A_1}+\mathbf{HG}|\mathbf{A_2}+\mathbf{H_f}(\mathbf{t})\mathbf{G}]\begin{pmatrix} \mathbf{X_{02}} \\ \mathbf{X_{12}} \\ \mathbf{I} \end{pmatrix} = [\mathbf{A'_2}+\mathbf{H_f}(\mathbf{t}+\mathbf{s})\mathbf{G}]$$

   The method used here is the same as **SRKeyGen** in the previous step. Sampling for the cosets is obtained from the columns of the matrix $\mathbf{A'_2}-\mathbf{A_2}+(\mathbf{H_f}(\mathbf{t}+\mathbf{s})-\mathbf{H_f}(\mathbf{t}))\mathbf{G}$ by executing $\mathsf{Sample}^{\mathcal{O}}$. The outputs, namely $\mathbf{X_{02}} \in \mathbb{Z}^{l\times nk}, \mathbf{X_{12}} \in \mathbb{Z}^{nk\times nk}$, have Gaussian distributed entries with parameter $s\sqrt{l}$:

$$[\mathbf{A_0}|\mathbf{A_1}+\mathbf{HG}]\begin{pmatrix} \mathbf{X_{02}} \\ \mathbf{X_{12}} \end{pmatrix} = \mathbf{A'_2}-\mathbf{A_2}+(\mathbf{H_f}(\mathbf{t}+\mathbf{s})-\mathbf{H_f}(\mathbf{t}))\mathbf{G}$$

   We denote $\begin{pmatrix} \mathbf{X_{02}} \\ \mathbf{X_{12}} \end{pmatrix}$ by $\mathbf{X}_{\theta,\mathbf{t}}$, and then store $\theta$ and $\mathbf{X}_{\theta,\mathbf{t}}$ in node $\theta \in \mathsf{KUNodes}(\upsilon)$, and output $\mathbf{URK_t} = (\theta, X_{\theta,\mathbf{t}})_{\theta\in\mathsf{KUNodes}(\upsilon)}$.

4. $\forall(\alpha, X_{\alpha,R}) \in \mathbf{SRK}_R, (\beta, X_{\beta,L}) \in \mathbf{SRK}_L, (\gamma, X_{\gamma,t}) \in \mathbf{URK}_t$, if $\exists(\alpha,\beta,\gamma)$ satisfies $\alpha=\beta=\gamma$ then the re-encryption key is a matrix with Gaussian entries:

$$rk = \begin{pmatrix} \mathbf{X}_{\theta,R}+\mathbf{X}_{\theta,L} & \mathbf{X}_{\theta,\mathbf{t}} \\ 0 & \mathbf{I} \end{pmatrix} = \begin{pmatrix} \mathbf{X_{00}} & \mathbf{X_{01}} & \mathbf{X_{02}} \\ \mathbf{X_{10}} & \mathbf{X_{11}} & \mathbf{X_{12}} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} \end{pmatrix};$$

   on the otherwise, if any two sets of $\mathbf{SRK}_R, \mathbf{SRK}_L$ and $\mathbf{URK_t}$ do not have a common node, $rk \leftarrow \perp$.

5. Output re-encryption key $rk$ satisfying:

$$[\mathbf{A_0}|\mathbf{A_1}+\mathbf{HG}|\mathbf{A_2}+\mathbf{H_f}(\mathbf{t})\mathbf{G}]\begin{pmatrix} \mathbf{X}_{\theta,R}+\mathbf{X}_{\theta,L} & \mathbf{X}_{\theta,\mathbf{t}} \\ 0 & \mathbf{I} \end{pmatrix} = [\mathbf{A'_0}|\mathbf{A'_1}+\mathbf{H'G}|\mathbf{A'_2}+\mathbf{H_f}(\mathbf{t}+\mathbf{s})\mathbf{G}]$$

- **UpKey**$(pk, \mathbf{t}', \mathbf{t}, ST, sk)$: Recall that $sk = [\mathbf{R_1}|\mathbf{R_2}]$. The delegator would generate a proxy re-encryption key $rk_{\mathbf{t}'\to\mathbf{t}}$, and the ciphertext of delegator would be converted at time $\mathbf{t}'$ to the time $\mathbf{t}$ by the proxy. Using the first part of private key $sk$ : $\mathbf{R_1}$ and the invertible matrix $\mathbf{H}$ of the public key, and executing $Sample^{\mathcal{O}}$ algorithm(similar to the operations in algorithm **ReKeyGen**), output $\mathbf{X_1} \in \mathbb{Z}^{l\times nk}, \mathbf{X_2} \in \mathbb{Z}^{nk\times nk}$, where

$$[\mathbf{A_0}|\mathbf{A_1}+\mathbf{HG}]\begin{pmatrix} \mathbf{X_1} \\ \mathbf{X_2} \end{pmatrix} = \mathbf{H_f}(\mathbf{t})-\mathbf{H_f}(\mathbf{t}')\mathbf{G}$$

  The re-encryption key is the matrix:

$$rk_{\mathbf{t}'\to\mathbf{t}} = \begin{pmatrix} \mathbf{I} & \mathbf{0} & \mathbf{X_1} \\ \mathbf{0} & \mathbf{I} & \mathbf{X_2} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} \end{pmatrix}$$

- **ReEnc**$(rk, c)$: Recall that $c = (\mathbf{H_u}, \mathbf{b})$. Convert the ciphertext of delegator to the ciphertext of delegatee by using the proxy re-encryption key. The specific process is as follows:

$$\begin{aligned}
&\mathbf{b}^\top \cdot rk \\
&= 2(\mathbf{r}^\top[\mathbf{A_0}|\mathbf{A_1}+\mathbf{HG}|\mathbf{A_2}+\mathbf{H_f(t)G}] \bmod q) + \mathbf{e}^\top + (\mathbf{0},\mathbf{0},enc(\mathbf{m}))^\top \cdot \begin{pmatrix} \mathbf{X_{00}} & \mathbf{X_{01}} & \mathbf{X_{02}} \\ \mathbf{X_{10}} & \mathbf{X_{11}} & \mathbf{X_{12}} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} \end{pmatrix} \bmod 2q \\
&= 2(\mathbf{r}^\top[\mathbf{A_0}'|\mathbf{A_1}'+\mathbf{H'G}|\mathbf{A_2}'+\mathbf{H_f(t+s)G}] \bmod q) + \overline{\mathbf{e}}^\top + (\mathbf{0},\mathbf{0},enc(\mathbf{m}))^\top \bmod 2q, \qquad (2)
\end{aligned}$$

  where $\overline{\mathbf{e}} = (\overline{\mathbf{e_0}}, \overline{\mathbf{e_1}}, \overline{\mathbf{e_2}}) = (\mathbf{e_0X_{00}} + \mathbf{e_1X_{01}}, \mathbf{e_0X_{01}} + \mathbf{e_1X_{11}}, \mathbf{e_0X_{02}} + \mathbf{e_1X_{12}} + \mathbf{e_2})$.

  Finally, we output the ciphertext $c = (\mathbf{H_f(t+s)G}, \mathbf{b})$

- **ReKeyRev**$(pk, pk', \mathbf{t}, RL, ST)$: On input a public key $pk' = ([\mathbf{A_0}'|\mathbf{A_1}'|\mathbf{A_2}', \mathbf{H}'])$, a time $\mathbf{t}$, the revocation list RL, and the state ST, the algorithm adds $(pk \to pk', \mathbf{t})$ to $RL$ for all nodes $v$ associated with $pk \to pk'$ and return $RL$.

- **UpCipher**$(pk, \mathbf{b}^\top, rk_{\mathbf{t}\to\mathbf{t}'}, ST, )$: When user $pk$ is revoked and the proxy re-encryption keys are updated, the proxy could re-encrypt the ciphertext of delegator at time $\mathbf{t}$ to time $\mathbf{t}'$ as follows:

$$\begin{aligned}
&\mathbf{b}^\top \cdot rk_{\mathbf{t}\to\mathbf{t}'} \\
&= 2(\mathbf{r}^\top[\mathbf{A_0}|\mathbf{A_1}+\mathbf{HG}|\mathbf{A_2}+\mathbf{H_f(t)G}] \bmod q) + \mathbf{e}^\top + (\mathbf{0},\mathbf{0},enc(\mathbf{m}))^\top \cdot \begin{pmatrix} \mathbf{I} & \mathbf{0} & \mathbf{X_1} \\ \mathbf{0} & \mathbf{I} & \mathbf{X_2} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} \end{pmatrix} \bmod 2q \\
&= 2(\mathbf{r}^\top[\mathbf{A_0}|\mathbf{A_1}+\mathbf{HG}|\mathbf{A_2}+\mathbf{H_f(t')G}] \bmod q) + \widetilde{\mathbf{e}}^\top + (\mathbf{0},\mathbf{0},enc(\mathbf{m}))^\top \bmod 2q, \qquad (3)
\end{aligned}$$

  where $\widetilde{\mathbf{e}} = (\widetilde{\mathbf{e_0}}, \widetilde{\mathbf{e_1}}, \widetilde{\mathbf{e_2}}) = (\mathbf{e_0}, \mathbf{e_1}, \mathbf{e_0X_1} + \mathbf{e_1X_2} + \mathbf{e_2})$.

**Correctness.** We present the correctness of our scheme by showing both the original ciphertext and the re-encrypted ciphertext can be decrypted correctly. We verify that the process of re-encryption. The proxy can convert ciphertext of $pk$ to $pk'$ through the corresponding re-encryption key where the $pk'$ is not a revoked user: $\forall(\alpha, X_{\alpha,R}) \in \mathbf{SRK}_R, (\beta, X_{\beta,L}) \in \mathbf{URK}_t, (\gamma, X_{\gamma,t}) \in \mathbf{URK}_t$, $\exists(\alpha,\beta,\gamma)$ satisfies $\alpha = \beta = \gamma$. We can chalk up $rk = \begin{pmatrix} \mathbf{X}_{\theta,R} + \mathbf{X}_{\theta,L} & \mathbf{X}_{\theta,\mathbf{t}} \\ 0 & \mathbf{I} \end{pmatrix}$ by running $\mathbf{ReKeyGen}(\mathbf{SRK}_R, \mathbf{SRK}_L, \mathbf{URK_t})$. We further call the re-encryption algorithm **ReEnc** to get ciphertext $2(\mathbf{r}^\top[\mathbf{A_0}'|\mathbf{A_1}'+\mathbf{H'G}|\mathbf{A_2}'+\mathbf{H_f(t+s)G}] \bmod q) + \overline{\mathbf{e}}^\top + (\mathbf{0},\mathbf{0},enc(\mathbf{m}))^\top \bmod 2q$.

We here explore that how to set appropriate parameters include size of noise so that the re-encrypted ciphertext can be decrypted correctly. Our scheme has the same parameter setting of the original ciphertext as [MP12]. And the parameter setting of the re-encrypted ciphertext is as same as [EK14]'s, by taking $1/\alpha = \mathcal{O}(nk)^3 \cdot r^3$ we have the desired property for both error terms: $\overline{e_0}R_1' + \overline{e_1}, \overline{e_0}R_2' + \overline{e_2} \in \mathcal{P}_{1/2}(q \cdot B^{(-\top)})$. It can be proved that the **Decryption** algorithm in our revocable proxy re-encryption scheme is correct.

# 5 Security Analysis

We first elaborate that our scheme is immune to a means of attack in [14]. The attack method adopted in that article is that the adversary can tell the real system from the simulation system by distinguishing an equation with public key and proxy re-encryption key. The basic reason is that the second part of the third item — $\mathbf{H}_u\mathbf{G}$ is equal in two sides of the following equation:

$$[\mathbf{A_0}|\mathbf{A_1}+\mathbf{HG}|\mathbf{A_2}+\mathbf{H_uG}] \begin{pmatrix} \mathbf{X_{02}} \\ \mathbf{X_{12}} \\ \mathbf{I} \end{pmatrix} = [\mathbf{A_2'}+\mathbf{H_uG}].$$

It is easy to check and compare with equation:

$$[\mathbf{A_0}|\mathbf{A_1}+\mathbf{HG}]\begin{pmatrix}\mathbf{X_{02}}\\ \mathbf{X_{12}}\\ \mathbf{I}\end{pmatrix}=[\mathbf{A_0^*}|-\mathbf{A}^*\mathbf{R_1^*}]\begin{pmatrix}\mathbf{X_{00}}\\ \mathbf{X_{10}}\\ \mathbf{I}\end{pmatrix}\cdot\mathbf{R_2'}\neq\mathbf{A_2'}-\mathbf{A_2}.$$

The adversary could easily tell if the public key $pk_1, pk_2$, proxy re-encryption key $rk$ are from the real system or the simulation system after using the above comparison method.

But our scheme is immune to this attack. We add a random vector $\mathbf{s}\leftarrow\mathbb{Z}_q^n$ into the third part of matrix so that the time term on both sides of the equation would not be eliminated, and the adversary cannot distinguish the real system and the simulation system effectively.

$$[\mathbf{A_0}|\mathbf{A_1}+\mathbf{HG}|\mathbf{A_2}+\mathbf{H_f(t)G}]\begin{pmatrix}\mathbf{X_{02}}\\ \mathbf{X_{12}}\\ \mathbf{I}\end{pmatrix}=[\mathbf{A_2'}+\mathbf{H_f(t+s)G}]$$

Next we consider the security proof of our scheme. The challenger has to possess $\mathbf{R}$ and an invertible $\mathbf{H}$ so that he can could the LWE problem successfully.

The simulator must guarantee that he is able to answer the query of adversary, that is $\mathbf{H}$ is an invertible matrix. However, once the adversary askes decryption query about challenge ciphertext, then $\mathbf{H}$ is the zero matrix, the simulator cannot transform it to a $\mathbf{G}-$ *trapdoor* matrix and decrypt it to recover the corresponding plaintext. Therefore, there is no invertible $\mathbf{H}$ involved, we embed LWE instance into the challenge ciphertext, and the output of the adversary will help us tackle the decision-LWE problem.

**Theorem 3.** *Our scheme is CCA-secure under conditions of decision-LWE where* $\alpha'=\alpha/3\geqslant 2\sqrt{n}/q$.

*Proof.* First, we transform the LWE distribution $\mathcal{A}_{s,\alpha'}=(\mathbf{a},b=<\mathbf{s},\mathbf{a}>/q+e \bmod 1)$ to $(a, 2(<s,a> \bmod q)+e' \bmod 2q)$, where $e'\rightarrow D_{\mathbb{Z}_{\alpha,q}}$, $b\rightarrow 2qb+D_{\mathbb{Z}_{2qb},s}$, $s^2=(\alpha q)^2-(2\alpha' q)^2\geqslant 4n\geqslant\eta_\epsilon(\mathbb{Z})^2$. This converts a uniform distribution on $\mathbb{Z}_q^n\times T$ to a discretized uniform distribution on $\mathbb{Z}_q^n\times\mathbb{Z}_{2q}$. Once the LWE samples are the required style, we construct a column-wise matrix $\mathbf{A_0}^*$ and $\mathbf{b}^*$, the public key of the target user is generated as follows: select a invertible matrix $\mathbf{H_1}^*$, a time vector $\mathbf{t}^*$ and a matrix function $\mathbf{H_f(x)}$, private key $\mathbf{R_1}^*, \mathbf{R_2}^*\in\mathcal{D}$, output the public key $\mathbf{pk_{A^*}}=([\mathbf{A_0}^*|-\mathbf{A_0}^*\mathbf{R_1}^*-\mathbf{H_1}^*\mathbf{G}|-\mathbf{A_0}^*\mathbf{R_1}^*-\mathbf{H_f}(\mathbf{t}^*)\mathbf{G}],\mathbf{H_1}^*)$, where $t^*$ is statistically hidden from the adversary.

We choose $\mathbf{X_{00}}$ and $\mathbf{X_{10}}$ from a Gaussian distribution with parameter $s$ so as to generate public key of valid user. First, we set a unimodular matrix $\mathbf{U}\in\mathbb{Z}_q^{l\times l}$, and two matrices $\mathbf{A_{R_1}}, \mathbf{A_{R_2}}$ are generated respectively: Let

$$\mathbf{U}^{-1}[\mathbf{A_0}^*|-\mathbf{A_0}^*\mathbf{R_1}^*]\begin{pmatrix}\mathbf{X_{00}}\\ \mathbf{X_{10}}\end{pmatrix}=\mathbf{A_{R_1}},$$

$$[\mathbf{A_0}^*|-\mathbf{A_0}^*\mathbf{R_1}^*]\begin{pmatrix}\mathbf{X_{00}}\\ \mathbf{X_{10}}\end{pmatrix}=\mathbf{U}\mathbf{A_{R_1}},$$

and

$$\mathbf{U}^{-1}[\mathbf{A_0}^*|-\mathbf{A_0}^*\mathbf{R_1}^*]\begin{pmatrix}\mathbf{X_{00}}\\ \mathbf{X_{10}}\end{pmatrix}\mathbf{R_1'}=\mathbf{A_0'}\mathbf{R_1'},$$

$$[\mathbf{A_0}^*|-\mathbf{A_0}^*\mathbf{R_1}^*]\begin{pmatrix}\mathbf{X_{00}}\\ \mathbf{X_{10}}\end{pmatrix}\mathbf{R_1'}=\mathbf{U}\mathbf{A_{R_2}}.$$

Let $\mathbf{X'_{\theta,R}}$ be denoted by $[\begin{pmatrix}\mathbf{X_{00}}\\ \mathbf{X_{10}}\end{pmatrix}|\begin{pmatrix}\mathbf{X_{00}}\\ \mathbf{X_{10}}\end{pmatrix}\mathbf{R_1'}]$. And put $\mathbf{X'_{\theta,R}}$ into node $\theta\in\mathsf{Path}(\upsilon)$, and continue to produce $\mathbf{X'_{\theta,L}}$.

We choose $\mathbf{X'_{00}}$ and $\mathbf{X'_{01}}$ from the Gaussian distribution with parameter $s$, make

$$[\mathbf{I}-\mathbf{U}]^{-1}[\mathbf{A_0^*}|-\mathbf{A_0^*}\mathbf{R_1^*}]\begin{pmatrix}\mathbf{X'_{00}}\\ \mathbf{X'_{10}}\end{pmatrix}=\mathbf{A'_{L_1}},$$

$$[\mathbf{A_0^*}|-\mathbf{A_0^*}\mathbf{R_1^*}]\begin{pmatrix}\mathbf{X'_{00}}\\ \mathbf{X'_{10}}\end{pmatrix}=[\mathbf{I}-\mathbf{U}]\mathbf{A_{L_1}},$$

and

$$[\mathbf{I}-\mathbf{U}]^{-1}[\mathbf{A_0^*}|-\mathbf{A_0^*R_1^*}]\begin{pmatrix}\mathbf{X'_{00}}\\ \mathbf{X'_{10}}\end{pmatrix}\mathbf{R'_1}=\mathbf{A'_0L'_2},$$

$$[\mathbf{A_0^*}|-\mathbf{A_0^*R_1^*}]\begin{pmatrix}\mathbf{X'_{00}}\\ \mathbf{X'_{10}}\end{pmatrix}\mathbf{R'_1}=[\mathbf{I}-\mathbf{U}]\mathbf{A_{L_2}}.$$

Let $\mathbf{X'_{\theta,L}}$ be denoted by $[\begin{pmatrix}\mathbf{X'_{00}}\\ \mathbf{X'_{10}}\end{pmatrix}|\begin{pmatrix}\mathbf{X'_{00}}\\ \mathbf{X'_{10}}\end{pmatrix}\mathbf{R'_1}]$, And put $\mathbf{X'_{\theta,L}}$ into node $\theta\in\mathsf{KUNodes}(v)$, noticing that $[\mathbf{A_0^*}|-\mathbf{A_0^*R_1^*}][\mathbf{X'_{\theta,R}}+\mathbf{X'_{\theta,L}}]=[\mathbf{UA_{R_1}}+[\mathbf{I}-\mathbf{U}]\mathbf{A_{L_1}}|\mathbf{UA_{R_2}}+[\mathbf{I}-\mathbf{U}]\mathbf{A_{L_2}}]=[\mathbf{A'_0}|-\mathbf{A'_0R'_1}]$.

We choose $\mathbf{R'_2}\in\mathbb{Z}_q^{l\times nk}$ from a distribution $\mathcal{B}$ defined over $\mathbb{Z}$ that outputs 0 with probability 1/2 and $\pm1$ with probability 1/4 each: $[\mathbf{A_0^*}|-\mathbf{A_0^*R_1^*}]\begin{pmatrix}\mathbf{X'_{00}}\\ \mathbf{X'_{10}}\end{pmatrix}\mathbf{R'_2}=\mathbf{A'_0R'_2}$. So the whole public key of a honest user is $pk'=([\mathbf{A'_0}|-\mathbf{A'_0R'_1}|-\mathbf{A'_0R'_2}-\mathbf{H_f}(\mathbf{t}^*)\mathbf{G}])$. We add $-\mathbf{H_f}(\mathbf{t}^*)\mathbf{G}$ to each honest key. $\mathbf{A'_0R'_1}$ is negl(n)-far from uniform and $-\mathbf{H_f}(\mathbf{t}^*)$ is concealed from adversary.

Let

$$\begin{pmatrix}\mathbf{X'_{00}}\\ \mathbf{X'_{10}}\end{pmatrix}+\begin{pmatrix}\mathbf{X_{00}}\\ \mathbf{X_{10}}\end{pmatrix}=\begin{pmatrix}\overline{\mathbf{X_{00}}}\\ \overline{\mathbf{X_{10}}}\end{pmatrix},$$

$$\begin{pmatrix}\mathbf{X'_{00}}\\ \mathbf{X'_{10}}\end{pmatrix}R'_1+\begin{pmatrix}\mathbf{X_{00}}\\ \mathbf{X_{10}}\end{pmatrix}R'_1=\begin{pmatrix}\overline{\mathbf{X_{01}}}\\ \overline{\mathbf{X_{11}}}\end{pmatrix},$$

$$\begin{pmatrix}\mathbf{X'_{00}}\\ \mathbf{X'_{10}}\end{pmatrix}R'_2=\begin{pmatrix}\overline{\mathbf{X_{02}}}\\ \overline{\mathbf{X_{12}}}\end{pmatrix},$$

and

$$\begin{pmatrix}\mathbf{X_{00}}+\mathbf{X'_{00}} & \mathbf{X_{01}}+\mathbf{X'_{01}} & \mathbf{X'_{02}}\\ \mathbf{X_{10}}+\mathbf{X'_{10}} & \mathbf{X_{11}}+\mathbf{X'_{11}} & \mathbf{X'_{12}}\end{pmatrix}=\begin{pmatrix}\overline{\mathbf{X_{00}}} & \overline{\mathbf{X_{01}}} & \overline{\mathbf{X_{02}}}\\ \overline{\mathbf{X_{10}}} & \overline{\mathbf{X_{11}}} & \overline{\mathbf{X_{02}}}\end{pmatrix},$$

each entry of the resulting matrices $\mathbf{X_{01}},\mathbf{X_{11}},\mathbf{X'_{01}},\mathbf{X'_{11}},\mathbf{X'_{02}},\mathbf{X'_{12}}$ is the inner of product of a Gaussian $l$-dimensional row-vector and a $\{0,-1,1\}$-vector with half of the coordinates equal zero, which is equivalent to $l/2$ additions of Gaussians with parameter $s$. Since in the scheme we obtain $\begin{pmatrix}\overline{\mathbf{X_{01}}}\\ \overline{\mathbf{X_{11}}}\end{pmatrix},\begin{pmatrix}\overline{\mathbf{X_{02}}}\\ \overline{\mathbf{X_{12}}}\end{pmatrix}$ with parameter $s\sqrt{l/2}$, then the simulated re-encryption key is

$$rk_{pk^*\to pk'}=\begin{pmatrix}\overline{\mathbf{X_{00}}} & \overline{\mathbf{X_{01}}} & \overline{\mathbf{X_{02}}}\\ \overline{\mathbf{X_{10}}} & \overline{\mathbf{X_{11}}} & \overline{\mathbf{X_{12}}}\\ \mathbf{0} & \mathbf{0} & \mathbf{I}\end{pmatrix}.$$

1. If no valid user is revoked, $KUNodes(BT,RL,\mathbf{t}^*)\bigcap Path(v^*)=\emptyset$ the challenger replies the query about private key of $id^*$ through $\{(\theta,e=\begin{pmatrix}\mathbf{X_{00}} & \mathbf{X_{01}}\\ \mathbf{X_{10}} & \mathbf{X_{11}}\end{pmatrix})\}_{\theta\in path(v^*)}$ and a update key query for $\mathbf{t}^*$ and $\{(\theta,e_1=\begin{pmatrix}\mathbf{X_{00}}' & \mathbf{X_{01}}'\\ \mathbf{X_{10}}' & \mathbf{X_{11}}'\end{pmatrix},e_2=\begin{pmatrix}\overline{\mathbf{X_{02}}}\\ \overline{\mathbf{X_{12}}}\end{pmatrix})\}_{\theta\in KUNodes(BT,RL,\mathbf{t}^*)}$.

2. If a user is revoked, that is $rev=1$, the produce way of keys are as above. But the challenger can only reply an updated key query for $\mathbf{t}^*$ through $\begin{pmatrix}\mathbf{X'_{00}}\\ \mathbf{X'_{10}}\end{pmatrix}R'_2$ for $\theta,\theta\in KUNodes(BT,RL,\mathbf{t}^*)$

Next we consider the decryption query about ciphertext $c=(\mathbf{H_f}(\mathbf{t}'),\mathbf{b})$ for $pk'=([\mathbf{A'_0}|-\mathbf{A_0R'_1}|-\mathbf{A'_0R'_2}-\mathbf{H_f}(\mathbf{t}^*)G],\mathbf{H}')$, where

$$\mathbf{b}^\top=2(\mathbf{r}^\top[\mathbf{A_0}'|\mathbf{A_1}'+\mathbf{HG}|\mathbf{A_2}'+(\mathbf{H_f}(\mathbf{t}^*)-\mathbf{H_f}(\mathbf{t}'))\mathbf{G}]\bmod q)+\mathbf{e}^\top+(\mathbf{0},\mathbf{0},enc(\mathbf{m}))^\top\bmod 2q$$

We first check that $\mathbf{H_f}(\mathbf{t}')$ is invertible or not, and call $Invert^{\mathcal{O}}$ algorithm when $\mathbf{H_f}(\mathbf{t}^*)-\mathbf{H_f}(\mathbf{t}')$ is a invertible matrix, whose input is $(sk=[\mathbf{R'_1}|\mathbf{R'_2}],\mathbf{A_u}=[\mathbf{A_0}'|\mathbf{A_1}'+\mathbf{HG}|\mathbf{A_2}'+(\mathbf{H_f}(\mathbf{t}^*)-\mathbf{H_f}(\mathbf{t}'))\mathbf{G},\mathbf{b},(\mathbf{H_f}(\mathbf{t}^*)-\mathbf{H_f}(\mathbf{t}'))])$, output is $(\mathbf{z},\mathbf{e})\in\mathbb{Z}_q^n$ to satisfy $\mathbf{b}=\mathbf{zA_u}+\mathbf{e}\bmod q$. If norm of $\mathbf{e}$ is small enough, $\mathbf{v}=\mathbf{b}-\mathbf{e}=(\mathbf{v_0},\mathbf{v_1},\mathbf{v_2})$, maintaining $\mathbf{v_0}\in\mathbb{Z}_q^l$ and $\mathbf{v_0}\in2\Lambda(\mathbf{A_0}^\top)$, we could obtain that $\mathbf{v}=2(\mathbf{rA_u}\bmod q)+(\mathbf{0},\mathbf{0},enc(\mathbf{m}))\bmod 2q$.

We multiply the matrix to perform the decryption operation $\mathbf{v}\begin{pmatrix}\mathbf{R_1} & \mathbf{R_2}\\ \mathbf{I} & \mathbf{0}\\ \mathbf{0} & \mathbf{I}\end{pmatrix} = 2(\mathbf{r}^\top[\mathbf{H'G}|\mathbf{H_f}(\mathbf{t}^*) - \mathbf{H_f}(\mathbf{t'})]) + (\mathbf{0},\mathbf{0},enc(\mathbf{m}))$. And the message $\mathbf{m}$ could be recovered by running the $enc^{-1}$ algorithm at the last $nk$ coordinates. So the simulator can answer the decryption query of $c = (\mathbf{b}^\top, \mathbf{H_f}(\mathbf{t'}))$ for valid user with an overwhelming probability if $\mathbf{t'} \neq \mathbf{t}^*$, Otherwise, return $\perp$.

In order to answer the re-encryption query from $pk' = ([\mathbf{A_0'}| - \mathbf{A_0}\mathbf{R_1'}| - \mathbf{A_0'}\mathbf{R_2'} - \mathbf{H_f}(\mathbf{t}^*)\mathbf{G}]$ with $\mathbf{H'} \in \mathbb{Z}_q^{n\times n}$ to $pk'' = ([\mathbf{A_0''}| - \mathbf{A_0''}\mathbf{R_1''}| - \mathbf{A_0''}\mathbf{R_2''} - \mathbf{H_f}(\mathbf{t}^*)\mathbf{G}]$ with $\mathbf{H''} \in \mathbb{Z}_q^{n\times n}$,

we convert

$$\mathbf{b}^\top = 2(\mathbf{r}^\top[\mathbf{A_0}'|\mathbf{A_1}' + \mathbf{HG}|\mathbf{A_2}' + (\mathbf{H_f}(\mathbf{t}^*) - \mathbf{H_f}(\mathbf{t'}))\mathbf{G}] \bmod q) + \mathbf{e}^\top + (\mathbf{0},\mathbf{0},enc(\mathbf{m}))^\top \bmod 2q$$

to

$$\mathbf{b'}^\top = 2(\mathbf{r}^\top[\mathbf{A_0}''|\mathbf{A_1}'' + \mathbf{HG}|\mathbf{A_2}'' + (\mathbf{H_f}(\mathbf{t}^*) - \mathbf{H_f}(\mathbf{t'}))\mathbf{G}] \bmod q) + \mathbf{e'}^\top + (\mathbf{0},\mathbf{0},enc(\mathbf{m}))^\top \bmod 2q$$

by the re-encryption key $rk_{pk'\to pk''}$, which is decrypted by $sk'' = [\mathbf{R_1''}|\mathbf{R_2''}]$.

To answer the **UpKey** query of $pk^*$ from $\mathbf{t'}$ to $\mathbf{t}$, the challenger uses the first part of private key $\mathbf{R_1^*}$ and the algorithm $Sample^{\mathcal{O}}$, output $\mathbf{X_1^*}, \mathbf{X_2^*}$. The re-encryption key returned is $rk_{\mathbf{t'}\to\mathbf{t}} = \begin{pmatrix}\mathbf{I} & \mathbf{0} & \mathbf{X_1^*}\\ \mathbf{0} & \mathbf{I} & \mathbf{X_2^*}\\ \mathbf{0} & \mathbf{0} & \mathbf{I}\end{pmatrix}$. The query about **UpCipher** could be answered by the ciphertext at time $\mathbf{t'}$ multiply by the re-encryption key $rk_{\mathbf{t'}\to\mathbf{t}}$.

Finally, let us consider the challenge ciphertext of $m \in \{0,1\}^{nk}$, which is generated by the public key $pk^*$. The challenge ciphertext is $\mathbf{b'}^\top = 2(\mathbf{r}^\top[\mathbf{A_0}^*|\mathbf{A_1}^*|\mathbf{A_2}^*] \bmod q) + \mathbf{e}^\top + (\mathbf{0},\mathbf{0},enc(\mathbf{m}))^\top \bmod 2q$ when $\mathbf{t'} = \mathbf{t}^*$, and $\mathbf{s} \in \mathbb{Z}_q^n$ and $\mathbf{e}$ are fairly small.

We use $\mathbf{b}^*$ at the beginning of game rather than calculated $\mathbf{b}$. The $\mathbf{b}^* = 2(\mathbf{r}^\top\mathbf{A_0^*} \bmod q) + \widetilde{\mathbf{e_0}} \bmod q$ is LWE distribution where $\mathbf{s} \leftarrow \mathbb{Z}_q^n, \widetilde{\mathbf{e_0}} \leftarrow D_{\mathbb{Z},\alpha q}$. We set the first $nk$ item of $\mathbf{b}^{*t}$ to be ${\mathbf{b_0}}^\top$, the later $2nk$ item of $\mathbf{b}^{*t}$ to be

$${\mathbf{b_1}}^\top = {\mathbf{b_0}}^\top{\mathbf{R_1}}^* + \widetilde{\mathbf{e_1}}^\top \bmod 2q \in \mathbb{Z}_2 q^{nk}$$

$${\mathbf{b_2}}^\top = {\mathbf{b_0}}^\top{\mathbf{R_2}}^* + \widetilde{\mathbf{e_2}}^\top + enc(\mathbf{m}) \bmod 2q \in \mathbb{Z}_2 q^{nk},$$

where $\mathbf{e_1}, \mathbf{e_2} \leftarrow D^{nk}_{\alpha q\sqrt{m},r}$. So the final challenge ciphertext is $\mathbf{b} = ({\mathbf{b_0}}^\top, {\mathbf{b_1}}^\top, {\mathbf{b_2}}^\top), \mathbf{t}^*$ which has the same distribution as the ciphertext in the real system. The noise term in ${\mathbf{b_1}}^\top$ is $\widetilde{\mathbf{e_0}}\mathbf{R_1} + \widetilde{\mathbf{e_1}}^\top$, which has negl(n)-distance with $\mathcal{D}_{\mathbb{Z},s}$ where $s^2 = (\|\widetilde{e_0}^2 + l(\alpha q)^2 \cdot r^2\|)$, the same to $\mathbf{b_2}^\top$. Notice that $\mathbf{R_1^*}, \mathbf{R_2^*} \leftarrow D$, $(\mathbf{A_0^*R_1^*}, \mathbf{A_0^*R_2^*}, -\mathbf{b}^*\mathbf{R_1^*}, -\mathbf{b}^*\mathbf{R_2^*})$ is negl(n)-uniform distribution according to LHL(leftover hash lemma) and $\mathbf{A_0^*}, \mathbf{b}^*$ is uniform distribution. Therefore, the challenge ciphertext in the view of adversary has the same distribution as what ciphertext in the real system is, therefore the adversary cannot distinguish them. □

# 6 Conclusion

In this paper, we have introduced the notion of revocable PRE and further designed a concrete construction satisfying the notion. In the construction, the revocability is reflected on the update of re-encryption key. We have leveraged binary tree structure to reduce the complexity of re-encryption key update to $O(logN)$, where $N$ is the number of delegatee. We also have considered the update of ciphertext so that a revoked user (at time period $t$) cannot gain access to all the ciphertexts encrypted before $t$. That allows us to maintain forward security. Besides, our scheme enjoys some distinct features, for example, the generation of re-encryption key is non-interactive, ciphertext update and re-encryption are off-loaded to proxy. Our construction is lattice based and meanwhile proved CCA secure under the LWE assumption in the standard model. We here leave the efficiency simulation and implementation of the scheme as parts of our future work. This paper also leaves some interesting open problems. First of all, one may consider how to convert our construction with LWE to the version based on RLWE to reduce storage/communication cost. Second, the revocability currently is limited to $O(logN)$. There may be a way to reduce the complexity to constant.

# References


[1] S. Agrawal and X. Boyen. Identity-based encryption from lattices in the standard model. Manuscript.

[2] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient lattice (H)IBE in the standard model. In Henri Gilbert, editor, *Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, May 30 - June 3, 2010. Proceedings*, volume 6110 of *Lecture Notes in Computer Science*, pages 553–572. Springer, 2010.

[3] Miklós Ajtai. Generating hard instances of lattice problems (extended abstract). In Gary L. Miller, editor, *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996*, pages 99–108. ACM, 1996.

[4] Yoshinori Aono, Xavier Boyen, Le Trieu Phong, and Lihua Wang. Key-private proxy re-encryption under LWE. In Goutam Paul and Serge Vaudenay, editors, *Progress in Cryptology - INDOCRYPT 2013 - 14th International Conference on Cryptology in India, Mumbai, India, December 7-10, 2013. Proceedings*, volume 8250 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2013.

[5] Giuseppe Ateniese, Karyn Benson, and Susan Hohenberger. Key-private proxy re-encryption. In Marc Fischlin, editor, *Topics in Cryptology - CT-RSA 2009, The Cryptographers' Track at the RSA Conference 2009, San Francisco, CA, USA, April 20-24, 2009. Proceedings*, volume 5473 of *Lecture Notes in Computer Science*, pages 279–294. Springer, 2009.

[6] Giuseppe Ateniese, Kevin Fu, Matthew Green, and Susan Hohenberger. Improved proxy re-encryption schemes with applications to secure distributed storage. In *Proceedings of the Network and Distributed System Security Symposium, NDSS 2005, San Diego, California, USA*. The Internet Society, 2005.

[7] Matt Blaze, Gerrit Bleumer, and Martin Strauss. Divertible protocols and atomic proxy cryptography. In Kaisa Nyberg, editor, *Advances in Cryptology - EUROCRYPT '98, International Conference on the Theory and Application of Cryptographic Techniques, Espoo, Finland, May 31 - June 4, 1998, Proceeding*, volume 1403 of *Lecture Notes in Computer Science*, pages 127–144. Springer, 1998.

[8] Alexandra Boldyreva, Vipul Goyal, and Virendra Kumar. Identity-based encryption with efficient revocation. In Peng Ning, Paul F. Syverson, and Somesh Jha, editors, *Proceedings of the 2008 ACM Conference on Computer and Communications Security, CCS 2008, Alexandria, Virginia, USA, October 27-31, 2008*, pages 417–426. ACM, 2008.

[9] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the weil pairing. *SIAM J. Comput.*, 32(3):586–615, 2003.

[10] Ran Canetti and Susan Hohenberger. Chosen-ciphertext secure proxy re-encryption. In Peng Ning, Sabrina De Capitani di Vimercati, and Paul F. Syverson, editors, *Proceedings of the 2007 ACM Conference on Computer and Communications Security, CCS 2007, Alexandria, Virginia, USA, October 28-31, 2007*, pages 185–194. ACM, 2007.

[11] Jie Chen, Hoon Wei Lim, San Ling, Huaxiong Wang, and Khoa Nguyen. Revocable identity-based encryption from lattices. In Willy Susilo, Yi Mu, and Jennifer Seberry, editors, *Information Security and Privacy - 17th Australasian Conference, ACISP 2012, Wollongong, NSW, Australia, July 9-11, 2012. Proceedings*, volume 7372 of *Lecture Notes in Computer Science*, pages 390–403. Springer, 2012.

[12] Ronald Cramer, Ivan Damgård, and Marcel Keller. On the amortized complexity of zero-knowledge protocols. *J. Cryptology*, 27(2):284–316, 2014.

[13] Shafi Goldwasser Daniele Micciancio. *Complexity of Lattice Problems: A Cryptographic Perspective*, volume 671. Springer US, 1 edition, 2002.

[14] Xiong Fan and Feng-Hao Liu. Various proxy re-encryption schemes from lattices. *IACR Cryptology ePrint Archive*, 2016:278, 2016.

[15] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Cynthia Dwork, editor, *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*, pages 197–206. ACM, 2008.

[16] Matthew Green and Giuseppe Ateniese. Identity-based proxy re-encryption. In Jonathan Katz and Moti Yung, editors, *Applied Cryptography and Network Security, 5th International Conference, ACNS 2007, Zhuhai, China, June 5-8, 2007, Proceedings*, volume 4521 of *Lecture Notes in Computer Science*, pages 288–306. Springer, 2007.

[17] Xagawa Keita and Keisuke Tanaka. Proxy re-encryption based on learning with errors. *Technical report*, 1691:29–35, 2010.

[18] Elena Kirshanova. Proxy re-encryption from lattices. In Hugo Krawczyk, editor, *Public-Key Cryptography - PKC 2014 - 17th International Conference on Practice and Theory in Public-Key Cryptography, Buenos Aires, Argentina, March 26-28, 2014. Proceedings*, volume 8383 of *Lecture Notes in Computer Science*, pages 77–94. Springer, 2014.

[19] Kwangsu Lee, Dong Hoon Lee, and Jong Hwan Park. Efficient revocable identity-based encryption via subset difference methods. *Des. Codes Cryptography*, 85(1):39–76, 2017.

[20] Benoît Libert and Damien Vergnaud. Unidirectional chosen-ciphertext secure proxy re-encryption. *IEEE Trans. Information Theory*, 57(3):1786–1802, 2011.

[21] San Ling, Khoa Nguyen, Huaxiong Wang, and Juanyang Zhang. Revocable predicate encryption from lattices. In Tatsuaki Okamoto, Yong Yu, Man Ho Au, and Yannan Li, editors, *Provable Security - 11th International Conference, ProvSec 2017, Xi'an, China, October 23-25, 2017, Proceedings*, volume 10592 of *Lecture Notes in Computer Science*, pages 305–326. Springer, 2017.

[22] Daniele Micciancio. Lattice-based cryptography. In Henk C. A. van Tilborg and Sushil Jajodia, editors, *Encyclopedia of Cryptography and Security, 2nd Ed.*, pages 713–715. Springer, 2011.

[23] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings*, volume 7237 of *Lecture Notes in Computer Science*, pages 700–718. Springer, 2012.

[24] Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on gaussian measures. *SIAM J. Comput.*, 37(1):267–302, 2007.

[25] David Nuñez, Isaac Agudo, and Javier Lopez. Ntrureencrypt: An efficient proxy re-encryption scheme based on NTRU. In Feng Bao, Steven Miller, Jianying Zhou, and Gail-Joon Ahn, editors, *Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security, ASIA CCS '15, Singapore, April 14-17, 2015*, pages 179–189. ACM, 2015.

[26] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM*, 56(6):34:1–34:40, 2009.

[27] Jae Hong Seo and Keita Emura. Revocable identity-based encryption revisited: Security model and construction. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *Public-Key Cryptography - PKC 2013 - 16th International Conference on Practice and Theory in Public-Key Cryptography, Nara, Japan, February 26 - March 1, 2013. Proceedings*, volume 7778 of *Lecture Notes in Computer Science*, pages 216–234. Springer, 2013.