SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

October 2023, Vol. 66 202101:1–202101:12
https://doi.org/10.1007/s11432-022-3612-6

Lattice-based autonomous path proxy re-encryption in the standard model

Wenli XIE[1], Jian WENG[1*], Yao TONG[2], Xiaojian LIANG[1], Lisha YAO[1] & Feixiang ZHAO[1]

[1]*College of Cyber Security, Jinan University, Guangzhou* 510632*, China;*
[2]*Guangzhou Fongwell Data Limited Company, Guangzhou* 510632*, China*

Received 7 May 2022/Revised 27 July 2022/Accepted 1 November 2022/Published online 23 May 2023

Abstract Autonomous path proxy re-encryption (AP-PRE) is a type of PRE that implements control on the delegation path in a multi-hop PRE. AP-PRE forces the proxy to perform the transformation along a predefined path without revealing the underlying plaintext. There are several applications of AP-PRE, including electronic medical systems, data sharing, and email systems. However, as far as we know, the existing AP-PRE scheme is provably secure in the random oracle model under the classical number-theoretic assumption, which might be vulnerable to quantum computers. Therefore, it raises the intriguing question of how to construct a quantum-resistant AP-PRE scheme. In this study, we proposed an AP-PRE scheme based on the widely accepted quantum-resistant learning with errors (LWE) assumptions. Our scheme supports the polynomial length of the delegation path. Furthermore, our scheme is proved to be selective-path CPA (sCPA) secure in the standard model under LWE assumptions.

Keywords autonomous path proxy re-encryption, proxy re-encryption, lattice-based cryptography, learning with errors, controlled delegation

Citation Xie W L, Weng J, Tong Y, et al. Lattice-based autonomous path proxy re-encryption in the standard model. Sci China Inf Sci, 2023, 66(10): 202101, https://doi.org/10.1007/s11432-022-3612-6

1 Introduction

Proxy re-encryption (PRE) was first proposed by Blaze et al. [1], which is a public-key encryption scheme that allows a semi-trusted proxy to convert a ciphertext under Alice's (delegator's) public key to the ciphertext under Bob's (delegatee's) public key without revealing the underlying plaintext by using Alice's re-encryption key. PRE is classified into two types based on the direction of delegation: unidirectional and bidirectional. A unidirectional PRE means that the re-encryption key can only transform the ciphertext under Alice's public key to Bob's public key, whereas a bidirectional PRE means that the re-encryption key can transform the ciphertext from Alice to Bob and vice versa. PRE can be categorized into single-hop PRE and multi-hop PRE based on the number of transformations.

A multihop PRE allows the ciphertext to be transformed several times, which is useful in various applications, such as email systems [2] and data sharing [3]. However, the conventional multi-hop PRE has a drawback: it does not support controlled delegation. Suppose that Alice has a confidential file that only Bob has access to. Therefore, Alice may delegate the decryption right to Bob. But what if the proxy already had the re-encryption key from Bob to Carol? In such a case, the proxy might delegate the decryption right to Carol against Alice's will.

Some modified PREs have been proposed to implement delegation control, such as type-based PRE (TB-PRE) [4] and conditional PRE (CPRE) [5]. In a TB-PRE scheme, the delegator categorizes his ciphertexts into different subsets identified by different types of strings and assigns different subsets to different delegatees. Only the re-encryption key associated with a type string s_1 can be used to re-encrypt the ciphertext associated with s_1. Informally, CPREs are similar to TB-PREs in spirit, except that CPRE

* Corresponding author (email: cryptjweng@gmail.com)

schemes improve on TB-PREs by hiding the type strings (called conditions in CPRE) embedded in the ciphertext and re-encryption key. However, in the aforementioned cases, the delegator only has control over the selection of delegatees in the first hop. The delegator is unaware of the subsequent delegatees when the re-encrypted ciphertext is further transformed.

To make sure that the delegation is always done among those delegatees that the delegator trusts, Cao et al. [6] advanced a concept of autonomous path PRE (AP-PRE), which ensures the delegator's control over the entire delegation path. For example, Alice may designate a sequence of users (such as Alice, Bob, and Carol) and send the re-encryption keys to the proxy, and then, the proxy will convert Alice's ciphertext to Bob's ciphertext. If Bob is too busy to deal with this message, the proxy will transform Bob's ciphertext into Carol's ciphertext. Note that all re-encryption keys are generated by Alice. Therefore, the proxy can only perform the re-encryption of Alice's ciphertext on the specific path. In an AP-PRE, the delegatee obtains the ciphertext from the proxy without interacting with the delegator, and the delegation is terminated if the delegatee accepts the decryption right. As far as we know, the AP-PRE scheme proposed in [6] is proved to be CPA (chosen plaintext attack) secure under decisional bilinear Diffie-Hellman (DBDH) assumption in the random oracle model.

The preceding facts motivate us to develop an AP-PRE scheme that satisfies quantum resistance and selective-path CPA (sCPA) security in the standard model.

Related work. Blaze et al. [1] proposed the first PRE scheme. This is a multi-hop bidirectional PRE scheme with CPA security. Canetti et al. [7] defined the chosen ciphertext attack (CCA) security model for PRE and presented two multi-hop bidirectional PRE schemes with CCA security: one is built in the random oracle model, and the other studies in the standard model. PRE is also being researched in identity-based scenarios. Green et al. [8] defined the notion of identity-based PRE (IB-PRE) and proposed the first multi-hop and unidirectional IB-PRE scheme. Then Wang et al. [9] proposed the first multi-hop CCA-secure IB-PRE, which addressed the open problem mentioned in [8]. To combine PRE with attribute-based encryption (ABE), Liang et al. [10] introduced the concept of attribute-based PRE (AB-PRE). Li et al. [11] proposed a ciphertext-policy AB-PRE scheme that uses a key-homomorphic constrained pseudorandom function to achieve fine-grained access control. All the aforementioned concepts and constructions do not provide fine-grained control over the proxy.

To overcome the limitations of conventional PRE, Weng et al. [5] proposed the concept of CPRE. They provided a single-hop unidirectional CPRE scheme with CCA security and left two open problems on how to construct CCA-secure CPRE schemes with anonymous conditions or that support more expressive predicates. Fang et al. [12] presented an efficient construction of a fuzzy CPRE scheme and proved its CCA-security under the DBDH assumption in the random oracle model. Zhao et al. [13] formalized definitions and security concepts for attribute-based CPRE (AB-CPRE) and proposed the first CCA-secure AB-PRE scheme. TB-PRE is a concept proposed by Tang [4]. He also proposed two TB-PREs: one is CPA secure with ciphertext privacy, and the other is CCA secure without ciphertext privacy. Though these two PREs implement fine-grained control on delegation, the delegator can only control the selection of the first delegatee. Therefore, Cao et al. [6] improved multi-hop PRE by enabling the delegator to designate a delegation path. Meanwhile, the delegation of a ciphertext originating from user i must follow two rules: (1) the re-encrypted ciphertext on path_i (which means the delegation path created by user i) cannot branch off path_i with meaningful decryption and (2) the original ciphertext under pk_j (public key for user j) cannot be inserted into path_i, where $i \neq j$, with meaningful decryption.

In terms of lattice-based PRE schemes, the first scheme is proposed in [14], which has multi-hop and bidirection properties. In 2014, Kirshanova [15] proposed a CCA lattice-based PRE that uses the public key encryption scheme in [16] as a foundation. However, this scheme is a single-hop PRE. Jiang et al. [17] proposed the first lattice-based multi-hop unidirectional PRE scheme. Recently, Liang et al. [18] proposed the first lattice-based AB-CPRE scheme, and Susilo et al. [19] proposed an HRA-secure AB-PRE scheme. Both schemes used the ABE scheme proposed in [20]. The purpose of this paper is to construct a lattice-based AP-PRE.

Our contribution. To the best of our knowledge, no lattice-based AP-PRE scheme exists at the moment. In this study, we construct a lattice-based autonomous path PRE scheme that achieves selective-path CPA security without the use of a random oracle. The main challenge is that the delegator must generate all re-encryption keys on his delegation path without knowing the secret key of intermediate delegatees, but in the meantime, intermediate delegatees cannot re-encrypt the ciphertext on the delegation path to a ciphertext with meaningful decryption. This means we should separate the delegation and decryption rights, which is quite different from the conventional multi-hop PRE concepts. The security

Table 1 Comparison between our scheme and multi-hop PRE schemes

Schemes	Type	Assumption	Security	Quantum -resistant	Delegation control	Autonomou -path	Standard model
Cao et al. [6]	AP-PRE	DBDH	CPA	×	✓	✓	×
Wang et al. [21]	PRE	DBDH	CCA2	×	×	×	✓
Wang et al. [9]	PRE	DBDH	CCA2	×	×	×	×
Xagawa et al. [14]	PRE	LWE	CPA	✓	×	×	✓
Aono et al. [22]	PRE	LWE	CPA,KP	✓	×	×	✓
Jiang et al. [17]	IB-PRE,PRE	LWE	CPA	✓	×	×	✓
Luo et al. [23]	AB-PRE	LWE	sCPA	✓	×	×	✓
Our scheme	AP-PRE	LWE	sCPA	✓	✓	✓	✓

Table 2 Efficiency comparison[a)]

Size	Our scheme	Xagawa et al. [14]	Aono et al. [22]	Jiang et al. [17]	Luo et al. [23]
ct	$3m\lceil\log q\rceil$	$(n+m)\lceil\log q\rceil$	$(n+m)\lceil\log q\rceil$	$m\lceil\log q\rceil$	$(t+2)m\lceil\log q\rceil$
rk	$m^2\lceil\log(s_1\sqrt{m})\rceil$	$nm\lceil\log q\rceil$	$(n\lceil\log q\rceil+m)(n+m)\lceil\log q\rceil$	$4m^2\lceil\log(s_2\sqrt{m})\rceil$	$4m^2\lceil\log(s_3\sqrt{m})\rceil$
pk	$2mn\lceil\log q\rceil$	$(n+m)^2\lceil\log q\rceil^2$	$nm\lceil\log q\rceil$	$n\lceil\log q\rceil$	t

a) Let m denotes the bit length of message which satisfies $m=6n\lceil\log q\rceil$. n and m denote the number of rows and columns of $\boldsymbol{A}\in\mathbb{Z}_q^{n\times m}$, which is used in encryption. ct, rk, and pk represent ciphertext, re-encryption key, and public key, respectively. t is the bit length of attributes. Let $s_1=m^{2.5}\omega(\sqrt{\log m})$, $s_2=m\omega(\log n)$ and $s_3=\omega((m+1)^{d+1.5})\cdot\omega(\sqrt{m\log m})$ where d is the bound on the depth of the circuit representation of some functions.

proof would also be challenging because when the challenge public key is a uniform matrix without a trapdoor, it is difficult to answer the re-encryption key generation query on the challenge path without a trapdoor in the game. Our AP-PRE construction has the following features.

• Our construction is the first lattice-based AP-PRE scheme in which the delegator has complete control over the transformation of the ciphertext originating from him throughout the delegation path. It is built based on some useful tools from lattice-based cryptography, such as Gaussian sampling and the lattice trapdoor.

• We prove that our AP-PRE scheme is selective-path CPA secure under the learning with error (LWE) assumption without using a random oracle.

The only existing AP-PRE scheme, as shown in Table 1 [6,9,14,17,21–23], is neither quantum resistant nor proven to be secure in the standard model. The multi-hop PRE schemes in the list do not support delegation control in contrast to our scheme. Furthermore, some PREs with controlled delegation might not support control of the delegation path. In a nutshell, our scheme satisfies the properties that the preceding schemes do not. We also compare the efficiency of our scheme with different latticed-based multi-hop PRE, IB-PRE, and AB-PRE schemes in Table 2 [14, 17, 22, 23]. We compare the size of the re-encryption key used for one hop and the size of the ciphertext and public key in the table. IB-PRE [17] and AB-PRE [23] have a small public key size because identities and attributes are public keys in these schemes. The size of the ciphertext, re-encryption key, and public key of our scheme is relatively small.

2 Preliminaries

In this section, we first introduce some notations used in our paper and then recall some necessary knowledge.

We use a lower-case bold letter $\boldsymbol{a}$ to denote a vector and an upper-case bold letter $\boldsymbol{A}$ to denote a matrix. Let $D_{\mathbb{Z}^n,s}$ denote the discrete Gaussian distribution over $\mathbb{Z}^n$ with parameter s. And let $\|\boldsymbol{v}\|$ denote the ℓ_2 norm of a vector $\boldsymbol{v}$. The norm of any matrix $\boldsymbol{A}$ represented by $\|\boldsymbol{A}\|$ denotes the ℓ_2 norm of the longest column vector. $\|\boldsymbol{R}\|_{\mathrm{GS}}:=\|\widetilde{\boldsymbol{R}}\|$ where $\widetilde{\boldsymbol{R}}$ is the Gram-Schmidt (GS) orthogonalization of $\boldsymbol{R}$. We define $\|\boldsymbol{A}\|_2:=\sup_{\|\boldsymbol{e}\|=1}\|\boldsymbol{A}\boldsymbol{e}\|$, then we have $\|\boldsymbol{A}\|_{\mathrm{GS}}\leqslant\|\boldsymbol{A}\|\leqslant\|\boldsymbol{A}\|_2\leqslant\sqrt{m}\|\boldsymbol{A}\|$ and $\|\boldsymbol{A}\boldsymbol{B}\|_2\leqslant\|\boldsymbol{A}\|_2\|\boldsymbol{B}\|_2$.

2.1 Lattice

Definition 1 (Lattice)**.** Given n linearly independent vectors $\boldsymbol{B}=\{\boldsymbol{b}_1,\boldsymbol{b}_2,\ldots,\boldsymbol{b}_n\}\subset\mathbb{R}^m$, the lattice generated by these vectors is

$$\Lambda=\mathcal{L}(\boldsymbol{B})=\{\boldsymbol{B}\boldsymbol{x}\mid\boldsymbol{x}\in\mathbb{Z}^n\},$$

where $\boldsymbol{B}$ is a basis of the lattice.

Definition 2 (Successive minima)**.** Let Λ be a lattice of rank n. For $i \in \{1, \ldots, n\}$ we define the i-th successive minima as

$$\lambda_i(\Lambda) = \inf\{r \mid \dim(\mathrm{span}(\Lambda \cap \overline{B}(0, r))) \geqslant i\},$$

where $\overline{B}(0, r) = \{\boldsymbol{x} \in \mathbb{R}^m \mid \|\boldsymbol{x}\| \leqslant r\}$ is the closed ball of radius r around 0.

Definition 3 (Dual lattice)**.** For a full-rank lattice Λ, we define its dual lattice as

$$\Lambda^* = \{\boldsymbol{y} \in \mathbb{R}^m \mid \forall \boldsymbol{x} \in \Lambda,\ \langle \boldsymbol{x}, \boldsymbol{y} \rangle \in \mathbb{Z}\}.$$

Definition 4 (q-ary lattice)**.** Given a matrix $\boldsymbol{A} \in \mathbb{Z}_q^{n \times m}$, where m, n, q are integers, and a vector $\boldsymbol{u} \in \mathbb{Z}_q^n$. A q-ary integer lattice and coset of this lattice can be defined as

$$\Lambda_q^{\perp}(\boldsymbol{A}) = \{\boldsymbol{x} \in \mathbb{Z}^m : \boldsymbol{A}\boldsymbol{x} \equiv \boldsymbol{0} \bmod q\},$$

$$\Lambda_q^{\boldsymbol{u}}(\boldsymbol{A}) = \{\boldsymbol{x} \in \mathbb{Z}^m : \boldsymbol{A}\boldsymbol{x} \equiv \boldsymbol{u} \bmod q\}.$$

Definition 5 (B-bounded noise distribution)**.** A noise distribution χ over $\mathbb{Z}$ is B-bounded, if $\Pr_{x \leftarrow \chi}[|x| \geqslant B] \leqslant 2^{-\widetilde{\Omega}(n)}$.

Definition 6 (Decisional LWE (DLWE))**.** Given $n, m \geqslant O(n \log q)$ and a B-bounded noise distribution χ, the DLWE problem is defined to distinguish between the following two distributions:

$$(\boldsymbol{A}, \boldsymbol{A}^{\mathrm{T}}\boldsymbol{s} + \boldsymbol{e}) \text{ and } (\boldsymbol{A}, \boldsymbol{u}),$$

where $\boldsymbol{A} \leftarrow \mathbb{Z}_q^{n \times m}$, $\boldsymbol{s} \leftarrow \mathbb{Z}_q^n$, $\boldsymbol{e} \leftarrow \chi^m$, and $\boldsymbol{u} \leftarrow \mathbb{Z}_q^m$ are sampled independently.

Theorem 1 (Hardness of LWE [24, 25])**.** Given $q = q(n) \leqslant 2^n$, $m = \mathrm{poly}(n)$, and a B-bounded noise distribution χ where $B = B(n)$ and $q/B \geqslant 2^{n^{\varepsilon}}$, for all $\varepsilon > 0$, solving $\mathrm{LWE}_{n,m,q,\chi}$ problem is as hard as quantumly solving GapSVP_{γ} and classically solving SIVP_{γ}, where $\gamma = \widetilde{O}(n/\alpha)$.

2.2 Gaussians distribution over lattice

We briefly recall Gaussian distributions over lattice. For any positive integer $n \geqslant 1$ and real $s > 0$, the Gaussian function $\rho_s : \mathbb{R}^n \to (0, 1]$ is defined as

$$\rho_s(\boldsymbol{x}) = \exp(-\pi \|\boldsymbol{x}\| / s^2).$$

For a lattice coset $\boldsymbol{c} + \Lambda \subset \mathbb{R}^n$ and $s > 0$, the discrete Gaussian probability distribution $D_{\boldsymbol{c}+\Lambda,s}$ simply assigns probability proportional to $\rho_s(\boldsymbol{x})$ to each $\boldsymbol{x} \in \boldsymbol{c} + \Lambda$, and probability zero elsewhere.

$$D_{\boldsymbol{c}+\Lambda,s} \propto \begin{cases} \rho_s(\boldsymbol{x}), & \text{if } \boldsymbol{x} \in \boldsymbol{c} + \Lambda, \\ 0, & \text{otherwise.} \end{cases}$$

We also recall a very important quantity called the smoothing parameter, which is introduced by Micciancio and Regev [26], of a lattice Λ. Intuitively, this parameter provides a width beyond which the discrete Gaussian measure on a lattice behaves like a continuous one.

Definition 7 (The smoothing parameter)**.** For any n-dimensional lattice Λ and positive real $\varepsilon > 0$, the smoothing parameter $\eta_{\varepsilon}(\Lambda)$ can be defined as the smallest s such that $\rho_{1/s}(\Lambda^* \ \{0\}) \leqslant \varepsilon$.

The following are two upper bounds on the smoothing parameter.

Lemma 1 ([26])**.** For a full-rank lattice $\Lambda \subseteq \mathbb{R}^n$, we have $\eta_{\varepsilon}(\Lambda) \leqslant \sqrt{n}/\lambda_1(\Lambda^*)$, where $\varepsilon = 2^{-n}$.

Lemma 2 ([26, 27])**.** For any full-rank lattice $\Lambda \subseteq \mathbb{R}^n$ and $\varepsilon \in \{0, 1/2\}$,

$$\eta_{\varepsilon}(\Lambda) \leqslant \lambda_n(\Lambda) \cdot \sqrt{\frac{\ln(2n(1 + 1/\varepsilon))}{\pi}}.$$

In particular, for any superlogarithmic function $\omega(\log n)$, there exists a negligible function $\varepsilon(n)$ such that $\eta_{\varepsilon}(\Lambda) \leqslant \lambda_n(\Lambda) \cdot \sqrt{\omega(\log n)}$.

2.3 Trapdoors and sampling

Here we show some useful tools that are used in our scheme and security proof.

Theorem 2 ([28]). Let n, m, q be integers, $q \geqslant 3$ be odd and $m = \lceil 6n \log q \rceil$. There is a probabilistic polynomial-time (PPT) algorithm $\mathbf{TrapGen}(1^n, m, q)$ that outputs a pair $(\boldsymbol{A} \in \mathbb{Z}_q^{n\times m}, \boldsymbol{S} \in \mathbb{Z}^{m\times m})$ such that $\boldsymbol{A}$ is statistically close to a uniform matrix in $\mathbb{Z}_q^{n\times m}$ and $\boldsymbol{S}$ is the basis for $\Lambda_q^{\perp}(\boldsymbol{A})$ satisfying

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

with overwhelming probability in n.

Lemma 3 ([26, 27]). Given positive integers $n, q > 2$ and $m > n$. Given $\boldsymbol{A} \in \mathbb{Z}_q^{n\times m}$ with a trapdoor $\boldsymbol{T_A}$, where $\boldsymbol{T_A}$ is a short basis for lattice $\Lambda_q^{\perp}(\boldsymbol{A})$ and $\sigma \geqslant \|\boldsymbol{T_A}\|_{\mathrm{GS}} \cdot \omega(\sqrt{\log m})$. We have

– For a vector $\boldsymbol{u} \in \mathbb{Z}_q^n$, it holds that $\Pr[\boldsymbol{x} \leftarrow D_{\Lambda_q^{\boldsymbol{u}}(\boldsymbol{A}),\sigma} \mid \|\boldsymbol{x}\| > \sqrt{m} \cdot \sigma] \leqslant \mathrm{negl}(m)$.

– For a matrix $\boldsymbol{B} \in \mathbb{Z}_q^{n\times k}$, there exits a PPT algorithm $\mathbf{SamplePre}(\boldsymbol{A}, \boldsymbol{T_A}, \boldsymbol{B}, \sigma)$ which outputs a matrix $\boldsymbol{X} \in \Lambda_q^{\boldsymbol{B}}(\boldsymbol{A})$ such that $\boldsymbol{AX} = \boldsymbol{B}$ distributed statistically close to $D_{\Lambda_q^{\boldsymbol{B}},\sigma}$.

– $\Pr[\boldsymbol{R} \leftarrow D_{\Lambda_q^{\boldsymbol{B}}(\boldsymbol{A}),\sigma} \mid \|\boldsymbol{R}\| > m \cdot \sigma] \leqslant \mathrm{negl}(m)$.

Lemma 4 ([27]). Let n be a positive integer and $m \geqslant 2n \lg q$, and let q be a positive prime integer. Then for all but a $2q^{-n}$ fraction of all $\boldsymbol{A} \in \mathbb{Z}_q^{n\times m}$ and for any $s \geqslant \omega(\sqrt{\log m})$, the distribution of syndrome $\boldsymbol{u} = \boldsymbol{Ae} \bmod q$ is statistically close to uniform over $\mathbb{Z}_q^n$, where $\boldsymbol{e} \leftarrow D_{\mathbb{Z}^m,s}$.

3 AP-PRE

3.1 Definition

In this subsection, we introduce the definition of AP-PRE and its security model. But before that, we give the explanation of some notations here. path_i is the path that originating from user i. l_i is the number of delegatees in path_i, and thus the length of path_i is $l_i + 1$. User i_j is the j-th delegatee in path_i. User i, which is also represented as user i_0, is the delegator.

Definition 8 (AP-PRE). An AP-PRE scheme consists of seven PPT algorithms as follows.

– $\mathbf{Setup}(1^\lambda) \to \mathrm{pp}$. Take as input a system's security parameter 1^λ, this algorithm outputs the public parameter pp of the system.

– $\mathbf{KeyGen}(\mathrm{pp}, i) \to (\mathrm{pk}_i, \mathrm{sk}_i)$. Take as input the public parameter pp and a user's identity i, this algorithm outputs a key pair $(\mathrm{pk}_i, \mathrm{sk}_i)$.

– $\mathbf{CreatePath}(\mathrm{pp}, \mathrm{pk}_i) \to (\mathrm{path}_i, l_i)$. Take as input the public parameter pp and a user's public key pk_i, this algorithm outputs the delegation path and the number of delegatees. The autonomous delegation path designed by user i can be represented as $\mathrm{path}_i = \{\mathrm{pk}_{i_j}\}_{j\in\{0,1,\ldots,l_i\}}$. Note that $\mathrm{pk}_{i_0} = \mathrm{pk}_i$. It outputs an autonomous delegation path path_i with length $l_i + 1$ if $i_m \neq i_n$ for any $m, n \in \{0, \ldots, l_i\}$. Otherwise, it outputs $\perp$.

– $\mathbf{Enc}(\mathrm{pp}, \mu, \mathrm{pk}_i) \to c_0^i$. Take as input the public parameter pp, a message μ and the delegator's public key pk_i, this algorithm outputs the corresponding ciphertext c_0^i.

– $\mathbf{ReKeyGen}(\mathrm{pp}, \mathrm{sk}_i, \mathrm{path}_i, l_i) \to \mathrm{rk}_i$. This algorithm takes as input the public parameter pp, the delegator's secret key sk_i, the delegation path path_i and the number of delegatees l_i. And it outputs the re-encryption key chain as $\mathrm{rk}_i = \{\mathrm{rk}_{j\to j+1}^i\}_{j\in\{0,\ldots,l_i-1\}}$ where $\mathrm{rk}_{j\to j+1}^i$ is the re-encryption key from user i_j to user i_{j+1} in path_i.

– $\mathbf{ReEnc}(\mathrm{pp}, \mathrm{path}_i, c_j^i, \mathrm{rk}_{j\to j+1}^i) \to c_{j+1}^i$. For any $j \in \{0, \ldots, l_i - 1\}$, this algorithm takes as input the public parameter pp, the ciphertext c_j^i under pk_{i_j} in path_i and the re-encryption key from user i_j to user i_{j+1} in path_i. Finally, it outputs the re-encrypted ciphertext c_{j+1}^i under the public key $\mathrm{pk}_{i_{j+1}}$.

– $\mathbf{Dec}(\mathrm{pp}, c_j^i, \mathrm{sk}_{i_j}) \to \mu/\perp$. For any $j \in \{0, \ldots, l_i\}$, this algorithm takes as input the public parameter pp, the ciphertext under pk_{i_j} and the user's secret key sk_{i_j}. Finally, it outputs the message μ or $\perp$.

Correctness. The correctness of the above AP-PRE holds if

(1) For any security parameter λ, user's identity i and any message μ from the message space, it holds that

$$\Pr[\mu = \mathbf{Dec}(\mathrm{pp}, \mathbf{Enc}(\mathrm{pp}, \mu, \mathrm{pk}_i), \mathrm{sk}_i)] = 1 - \mathrm{negl}(\lambda),$$

where $\mathrm{pp} \leftarrow \mathbf{Setup}(1^\lambda)$ and $(\mathrm{pk}_i, \mathrm{sk}_i) \leftarrow \mathbf{KeyGen}(\mathrm{pp}, i)$.

(2) For any security parameter λ, user's identity i and $j \in \{1, \ldots, l_i\}$, it holds that

$$\Pr[\mu = \mathbf{Dec}(\mathrm{pp}, c_j^i, \mathrm{sk}_{i_j})] = 1 - \mathrm{negl}(\lambda),$$

where $\mathrm{pp} \leftarrow \mathrm{Setup}(1^\lambda)$, $(\mathrm{pk}_{i_j}, \mathrm{sk}_{i_j}) \leftarrow \mathbf{KeyGen}(\mathrm{pp}, i_j)$ and $c_k^i \leftarrow \mathbf{ReEnc}(\mathrm{pp}, \mathrm{path}_i, c_{k-1}^i, \mathrm{rk}_{k-1\to k}^i)$ for any $k \in \{1, \ldots, j\}$. Note that $\mathrm{rk}_{k-1\to k}^i$ is a re-encryption key in $\mathrm{rk}_i \leftarrow \mathbf{ReKeyGen}(\mathrm{pp}, \mathrm{sk}_i, \mathrm{path}_i, l_i)$.

3.2 Security model

Our definition of sCPA security for AP-PRE is weaker than the CPA security model proposed in [6]. In our sCPA security model, the challenge path should be given at the beginning of the game. We define the sCPA game between the challenger $\mathcal{C}$ and a PPT adversary $\mathcal{A}$ as follows.

Init. To simplify the symbolic representation, we let $d = l_{i^*}$. $\mathcal{A}$ announces a sequence $\mathcal{S} = (i^*, i_1^*, \ldots, i_d^*)$ originating from user i^*. $\mathcal{C}$ runs $\mathbf{Setup}(1^\lambda)$ to get the public parameter pp and creates three tables: T_{kp} records the key pairs that have been queried, T_{rk} records the re-encryption keys on each path and T_{path} records the delegation path that have been queried. $\mathcal{C}$ generates the key pairs for all users in $\mathcal{S}$ and then generates the challenge path $\mathrm{path}_{i^*} = (\mathrm{pk}_{i^*}, \mathrm{pk}_{i_1^*}, \ldots, \mathrm{pk}_{i_d^*})$ as well as the corresponding re-encryption keys. Finally, $\mathcal{C}$ records the key pairs, the re-encryption keys and path_{i^*} in three tables and sends them as well as pp to $\mathcal{A}$.

Query phase 1. $\mathcal{A}$ makes the following types of queries.

– **Uncorrupted key generation query** $\boldsymbol{\mathcal{O}}_{\mathrm{uncorrupted}}(i)$. On input the identity of a user by $\mathcal{A}$, $\mathcal{C}$ runs the algorithm $\mathbf{KeyGen}(\mathrm{pp}, i)$ to generate the key pair $(\mathrm{pk}_i, \mathrm{sk}_i)$, records $(\mathrm{pk}_i, \mathrm{sk}_i)$ in the table T_{kp} and outputs pk_i if there is no key pair for user i in T_{kp}. Otherwise, $\mathcal{C}$ searches for $(\mathrm{pk}_i, \mathrm{sk}_i)$ in T_{kp} and outputs pk_i.

– **Corrupted key generation query** $\boldsymbol{\mathcal{O}}_{\mathrm{corrupted}}(i \notin \mathcal{S})$. On input the identity of a user by $\mathcal{A}$, $\mathcal{C}$ runs the algorithm $\mathbf{KeyGen}(\mathrm{pp}, i)$ to generate the key pair $(\mathrm{pk}_i, \mathrm{sk}_i)$, records $(\mathrm{pk}_i, \mathrm{sk}_i)$ in T_{kp} and outputs $(\mathrm{pk}_i, \mathrm{sk}_i)$ if $\mathcal{A}$ has not made an uncorrupted key generation on i before. Otherwise, $\mathcal{C}$ searches T_{kp} for $(\mathrm{pk}_i, \mathrm{sk}_i)$ and outputs the key pair.

– **Path creation query** $\boldsymbol{\mathcal{O}}_{\mathrm{path}}(i \neq i^*, \mathrm{path}_i)$. On input the identity of a user and the autonomous path path_i originating from user i, $\mathcal{C}$ outputs $\perp$ if $\mathcal{A}$ has made a path creation query on (i, path_i) before or $\mathbf{CreatPath}(\mathrm{pp}, i)$ returns $\perp$. Otherwise, $\mathcal{C}$ generates the re-encryption keys $\mathrm{rk}_i = (\mathrm{rk}_{0\to 1}^i, \mathrm{rk}_{1\to 2}^i, \ldots, \mathrm{rk}_{l_i-1\to l_i}^i)$ by running $\mathbf{ReKeyGen}(\mathrm{pp}, \mathrm{sk}_i, \mathrm{path}_i, l_i)$. Then $\mathcal{C}$ records path_i in T_{path} and rk_i in T_{rk}. Finally, $\mathcal{C}$ returns "path_i is created".

– **Re-encryption key generation query** $\boldsymbol{\mathcal{O}}_{\mathrm{rk}}(i, \mathrm{pk}_{i_j}, \mathrm{pk}_{i_{j+1}})$. On input the index of the path and two public keys, $\mathcal{C}$ outputs $\perp$ if T_{path} does not contain $\mathrm{path}_i = (\ldots, \mathrm{pk}_{i_j}, \mathrm{pk}_{i_{j+1}}, \ldots)$ for user i. Otherwise, $\mathcal{C}$ searches for rk_i in T_{rk} and return $\mathrm{rk}_{j\to j+1}^i$ to $\mathcal{A}$.

– **Re-encryption query** $\boldsymbol{\mathcal{O}}_{\mathrm{reen}}(i, c_j^i, \mathrm{pk}_{i_j}, \mathrm{pk}_{i_{j+1}})$. On input the identity of a user, the ciphertext and two public keys by $\mathcal{A}$, $\mathcal{C}$ checks if T_{path} contains a path $\mathrm{path}_i = (\ldots, \mathrm{pk}_{i_j}, \mathrm{pk}_{i_{j+1}}, \ldots)$ for user i. If not, $\mathcal{C}$ returns $\perp$ to $\mathcal{A}$. Otherwise, $\mathcal{C}$ retrieves rk_i from T_{rk}. Then $\mathcal{C}$ generates the re-encrypted ciphertext c_{j+1}^i by running the algorithm $\mathbf{ReEnc}(\mathrm{pp}, \mathrm{path}_i, c_j^i, \mathrm{rk}_{j\to j+1}^i)$ and finally returns c_{j+1}^i to $\mathcal{A}$.

Challenge phase. $\mathcal{A}$ submits a message pair (μ_0, μ_1), and $\mathcal{C}$ chooses a uniformly random bit b and returns $c^* \leftarrow \mathbf{Enc}(\mathrm{pp}, \mu_b, \mathrm{pk}_{i^*})$.

Query phase 2. The second query phase is the same as the first phase.

Guess. $\mathcal{A}$ returns a bit $b' \in \{0, 1\}$. $\mathcal{C}$ outputs 1 if $b' = b$. Otherwise, $\mathcal{C}$ outputs 0.

Definition 9 (sCPA-AP-PRE)**.** An AP-PRE scheme is sCPA secure, if any PPT adversary wins the sCPA game above only with negligible advantages.

Remark 1. In sCPA security definition for AP-PRE, $\forall j \in \{1, \ldots, d\}$, $\mathcal{A}$ can make re-encryption queries to re-encrypt c^* to $c_j^{i^*}$. Also, $\mathcal{A}$ can make a query on $\boldsymbol{\mathcal{O}}_{\mathrm{path}}(i_j^*, \mathrm{path}_{i_j^*})$ where $\mathrm{path}_{i_j^*}$ contains some corrupted users. Then, the sCPA security of AP-PRE implies that the re-encrypted ciphertext in $\mathrm{path}_{i_j^*}$ from $c_j^{i^*}$ cannot be correctly decrypted by any user in $\mathrm{path}_{i_j^*}$, thus it does nothing for $\mathcal{A}$ to distinguish μ_b. Besides, in the above sCPA security definition, any user can only designate one path which is constant.

4 Our AP-PRE scheme

In this section, we first give an intuitive construction of the AP-PRE scheme and point out the weakness of this simple scheme. Next, we introduce the complete construction of our lattice-based AP-PRE schemes and analyse its sCPA security.

4.1 Intuitive construction

The delegation of the ciphertext originating from user i should meet three requirements. (1) All re-encryption keys on the delegation path must be generated by the delegator. (2) $\forall j \in \{1, \ldots, l_i\}$, user i_j on path_i cannot generate an available re-encryption key to re-encrypt c_j^i to the ciphertext under other public keys with meaningful decryption. (3) The original ciphertext cannot be inserted into a path with meaningful decryption. Suppose that Alice designates a path (such as Alice, Bob, and Carol) and re-encrypts her ciphertext c_{alice} to the ciphertext $c_{\text{bob}}^{\text{alice}}$, Bob cannot generate a $\text{rk}_{\text{bob}\to\text{carol}}^{\text{alice}}$ to transform $c_{\text{bob}}^{\text{alice}}$ to the ciphertext that can be decrypted by Carol or other users. Furthermore, the proxy cannot use $\text{rk}_{\text{bob}\to\text{carol}}^{\text{alice}}$ to re-encrypt Bob's original ciphertext to Carol's ciphertext with meaningful decryption. Next, we introduce the concrete construction of our first attempt and highlight that it does not meet the second requirement.

– $\mathbf{Setup}(1^\lambda) \to \text{pp}$. The public parameters are $\text{pp} = (n, m, q, \chi^m, s, \boldsymbol{D} \in \mathbb{Z}_q^{n\times m})$.

– $\mathbf{KeyGen}(\text{pp}, i) \to (\text{pk}_i, \text{sk}_i)$. This algorithm generates $(\boldsymbol{A}_i, \boldsymbol{T}_{\boldsymbol{A}_i})$ by running $\mathbf{TrapGen}(1^n, m, q)$. Then, it generates $\boldsymbol{R}_{\boldsymbol{A}_i} \in \mathbb{Z}^{m\times m}$ such that $\boldsymbol{A}_i\boldsymbol{R}_{\boldsymbol{A}_i} = \boldsymbol{D}$, by running $\boldsymbol{R}_{\boldsymbol{A}_i} \leftarrow \mathbf{SamplePre}(\boldsymbol{A}_i, \boldsymbol{T}_{\boldsymbol{A}_i}, \boldsymbol{D}, s)$. Finally, it outputs the public key $\text{pk}_i = \boldsymbol{A}_i$ and the secret key $\text{sk}_i = (\boldsymbol{T}_{\boldsymbol{A}_i}, \boldsymbol{R}_{\boldsymbol{A}_i})$.

– $\mathbf{Enc}(\text{pp}, \text{pk}_i, \boldsymbol{\mu}) \to c_0^i$. To encrypt $\boldsymbol{\mu} \in \{0,1\}^m$, this algorithm randomly chooses $\boldsymbol{s} \leftarrow \mathbb{Z}_q^n$ and takes Gaussian noise vectors $\boldsymbol{e}_{0,1}^i, \boldsymbol{e}_{0,2}^i \in \chi^m$. Let $\text{pk}_i = \boldsymbol{A}_i$ and set

$$\begin{aligned}\boldsymbol{c}_{0,1}^i &= \boldsymbol{A}_i^{\mathrm{T}}\boldsymbol{s} + \boldsymbol{e}_{0,1}^i \in \mathbb{Z}_q^m,\\ \boldsymbol{c}_{0,2}^i &= \boldsymbol{D}^{\mathrm{T}}\boldsymbol{s} + \boldsymbol{e}_{0,2}^i + \lfloor q/2 \rfloor \boldsymbol{\mu} \in \mathbb{Z}_q^m.\end{aligned}$$

Output the ciphertext $\boldsymbol{c}_0^i = (\boldsymbol{c}_{0,1}^i, \boldsymbol{c}_{0,2}^i) \in \mathbb{Z}_q^{2m}$.

– $\mathbf{Dec}(\text{pp}, \text{sk}_{i_j}, \boldsymbol{c}_j^i) \to \boldsymbol{\mu}/\perp$. Let the secret key $\text{sk}_{i_j} = (\boldsymbol{T}_{\boldsymbol{A}_{i_j}}, \boldsymbol{R}_{\boldsymbol{A}_{i_j}})$. This algorithm proceeds the decryption as follows:

$$\hat{\boldsymbol{\mu}} = \boldsymbol{c}_{j,2}^i - \boldsymbol{R}_{\boldsymbol{A}_{i_j}}^{\mathrm{T}} \boldsymbol{c}_{j,1}^i.$$

Let $\hat{\boldsymbol{\mu}} = (\hat{\mu}_1, \ldots, \hat{\mu}_m)$. For any $i \in \{1, \ldots, m\}$, it sets $\mu_i = 0$ if $\hat{\mu}_i \in [-\lfloor q/4 \rfloor, \lfloor q/4 \rfloor])$. Otherwise sets $\mu_i = 1$. Finally, it outputs $\boldsymbol{\mu} = \{\mu_k\}_{k\in\{1,\ldots,m\}}$.

– $\mathbf{CreatPath}(\text{pp}, \text{pk}_i) \to (\text{path}_i, l_i)$. The autonomous delegation path can be represented as $\text{path}_i = (\text{pk}_i = \text{pk}_{i_0}, \ldots, \text{pk}_{i_j}, \ldots, \text{pk}_{i_{l_i}})$ where $\text{pk}_{i_j} \leftarrow \mathbf{KeyGen}(\text{pp}, i_j)$. This algorithm outputs path_i if $\text{pk}_{i_v} \neq \text{pk}_{i_w}$ holds for any $v, w \in \{0, \ldots, l_i\}$. Otherwise, it outputs $\perp$.

– $\mathbf{ReKeyGen}(\text{pp}, \text{sk}_i, \text{path}_i, l_i) \to \text{rk}_i = \{\text{rk}_{j\to j+1}^i\}_{j\in\{0,1,\ldots,l_i-1\}}$. To generate these re-encryption keys, the delegator with sk_i computes the re-encryption key between user i_j and user i_{j+1} in path_i, where $j \in \{0, 1, \ldots, l_i - 1\}$, as follows:

$$\boldsymbol{Q}_{j\to j+1}^i \leftarrow \mathbf{SamplePre}(\boldsymbol{A}_i, \boldsymbol{T}_{\boldsymbol{A}_i}, \boldsymbol{A}_{i_{j+1}} - \boldsymbol{A}_{i_j}, s).$$

Such that $\boldsymbol{A}_i\boldsymbol{Q}_{j\to j+1}^i = \boldsymbol{A}_{i_{j+1}} - \boldsymbol{A}_{i_j}$. Let $\text{rk}_{j\to j+1}^i = \boldsymbol{Q}_{j\to j+1}^i$ and output $\text{rk}_i = \{\text{rk}_{j\to j+1}^i\}_{j\in\{0,1,\ldots,l_i-1\}}$.

– $\mathbf{ReEnc}(\text{pp}, \boldsymbol{c}_j^i, \text{path}_i, \text{rk}_{j\to j+1}^i) \to \boldsymbol{c}_{j+1}^i$, where $j \in \{0, \ldots, l_i - 1\}$. This algorithm computes

$$\begin{aligned}\boldsymbol{c}_{j+1,2}^i &= \boldsymbol{c}_{j,2}^i = \boldsymbol{c}_{0,2}^i,\\ \boldsymbol{c}_{j+1,1}^i &= \boldsymbol{Q}_{j\to j+1}^{i\mathrm{T}} \boldsymbol{c}_{0,1}^i + \boldsymbol{c}_{j,1}^i.\end{aligned}$$

Note that $\boldsymbol{c}_{0,1}^i$ is the first part of the original ciphertext. Finally, this algorithm outputs $\boldsymbol{c}_{j+1}^i = (\boldsymbol{c}_{0,1}^i, \boldsymbol{c}_{j+1,1}^i, \boldsymbol{c}_{j+1,2}^i)$.

In the construction above, for $\boldsymbol{c}_j^i = (\boldsymbol{c}_{0,1}^i, \boldsymbol{c}_{j,1}^i, \boldsymbol{c}_{j,2}^i)$ and $\mathrm{rk}_{j\to j+1}^i = \boldsymbol{Q}_{j\to j+1}^i$, the re-encryption is correct because

$$\begin{aligned}\boldsymbol{c}_{j+1,1}^i &= \boldsymbol{Q}_{j\to j+1}^{i\mathrm{T}} \boldsymbol{c}_{0,1}^i + \boldsymbol{c}_{j,1}^i \\ &= (\boldsymbol{A}_{i_{j+1}}^{\mathrm{T}} - \boldsymbol{A}_{i_j}^{\mathrm{T}})\boldsymbol{s} + \boldsymbol{A}_{i_j}^{\mathrm{T}}\boldsymbol{s} + \boldsymbol{e}' \\ &= \boldsymbol{A}_{i_{j+1}}^{\mathrm{T}}\boldsymbol{s} + \boldsymbol{e}',\end{aligned}$$

where $\boldsymbol{e}' = \boldsymbol{Q}_{j\to j+1}^{i\mathrm{T}} \boldsymbol{e}_{0,1}^i + \boldsymbol{e}_{j,1}^i$.

We illustrate that this simple scheme obeys the above rules except for the second one. For the first rule, the delegator can generate all re-encryption keys on the path with the help of lattice trapdoor $\boldsymbol{T}_{\boldsymbol{A}_i}$. For the third rule, the original ciphertext cannot be easily inserted into the path due to the difference of form between the original and re-encrypted ciphertext. Next, we show that the re-encrypted ciphertext can branch off the path. Suppose that the form of $\boldsymbol{c}_j^i = (\boldsymbol{c}_{0,1}^i, \boldsymbol{c}_{j,1}^i, \boldsymbol{c}_{j,2}^i)$ is

$$\boldsymbol{c}_{0,1}^i = \boldsymbol{A}_i^{\mathrm{T}}\boldsymbol{s} + \boldsymbol{e}_{0,1}^i \ , \ \boldsymbol{c}_{j,1}^i = \boldsymbol{A}_{i_j}^{\mathrm{T}}\boldsymbol{s} + \boldsymbol{e}_{j,1}^i \ , \ \boldsymbol{c}_{j,2}^i = \boldsymbol{D}^{\mathrm{T}}\boldsymbol{s} + \boldsymbol{e}_{0,2}^i + \lfloor q/2 \rfloor \boldsymbol{\mu}.$$

We say that user i_j could generate an available $\mathrm{rk}_{i_j\to k}$, where user k could be any user in the system. User i_j runs $\boldsymbol{Q}_{i_j\to k} \leftarrow \mathbf{SamplePre}(\boldsymbol{A}_{i_j}, \boldsymbol{T}_{\boldsymbol{A}_{i_j}}, \boldsymbol{A}_k - \boldsymbol{A}_i)$ and sends $\mathrm{rk}_{i_j\to k} = \boldsymbol{Q}_{i_j\to k}$ and $\boldsymbol{c}_j^i = (\boldsymbol{c}_{j,1}^i, \boldsymbol{c}_{0,1}^i, \boldsymbol{c}_{j,2}^i)$ to the proxy. The proxy proceeds the calculation as follows:

$$\boldsymbol{c}_{k,1} = \boldsymbol{Q}_{i_j\to k}^{\mathrm{T}} \boldsymbol{c}_{j,1}^i + \boldsymbol{c}_{0,1}^i.$$

The resulted ciphertext under the public key of user k is $\boldsymbol{c}_k = (\boldsymbol{c}_{j,1}^i, \boldsymbol{c}_{k,1}, \boldsymbol{c}_{j,2}^i)$. Therefore, the ciphertext $\boldsymbol{c}_j^i$ on path_i is re-encrypted to the ciphertext of other users without the permission of user i. Obviously, it is because any delegatee on the path still has the power to create available re-encryption keys. To settle down the aforementioned issues, we introduce an extra matrix $\boldsymbol{H}_i$ and its trapdoor $\boldsymbol{T}_{\boldsymbol{H}_i}$ to generate the re-encryption key. Then, $\boldsymbol{A}_i$ and $\boldsymbol{R}_{\boldsymbol{A}_i}$ are used to decrypt only. Our formal construction is shown in the following content.

4.2 Formal construction

Before showing our AP-PRE scheme, we first list the parameters that we used in the scheme.

– λ-security parameter.
– (n, m, q, χ^m)-lattice parameters, where $m = \lceil 6n \log q \rceil$ and $\chi^m = D_{\mathbb{Z}^m, \alpha q}$ is the noise distribution.
– ℓ-the max length of the autonomous delegation path.
– s-Gaussian parameter, where $s = m^{2.5}\omega(\sqrt{\log m})$.
– α-the LWE error rate, where $\alpha \leqslant \frac{1}{4(\mathrm{O}(\ell)+3)m^{7.5}\cdot\omega(\sqrt{\log m})^2}$.

Our scheme works for $\ell, q = \mathrm{poly}(n)$.

– $\mathbf{Setup}(1^\lambda) \to \mathrm{pp}$. The global setup algorithm sets the lattice parameters as (n, m, q, χ^m, s) as mentioned above, chooses a uniformly random matrix $\boldsymbol{D} \leftarrow \mathbb{Z}_q^{n\times m}$ and sets $\mathrm{pp} = (n, m, q, \chi^m, s, \boldsymbol{D})$.

– $\mathbf{KeyGen}(\mathrm{pp}, i) \to (\mathrm{pk}_i, \mathrm{sk}_i)$. To generate a key pair for user i, this algorithm generates two pairs $(\boldsymbol{H}_i, \boldsymbol{T}_{\boldsymbol{H}_i})$ and $(\boldsymbol{A}_i, \boldsymbol{T}_{\boldsymbol{A}_i})$ by running

$$(\boldsymbol{H}_i, \boldsymbol{T}_{\boldsymbol{H}_i}) \leftarrow \mathbf{TrapGen}(1^n, m, q),$$

$$(\boldsymbol{A}_i, \boldsymbol{T}_{\boldsymbol{A}_i}) \leftarrow \mathbf{TrapGen}(1^n, m, q).$$

It also generates $\boldsymbol{R}_{\boldsymbol{A}_i} \in \mathbb{Z}^{m\times m}$ such that $\boldsymbol{A}_i\boldsymbol{R}_{\boldsymbol{A}_i} = \boldsymbol{D}$ by running $\boldsymbol{R}_{\boldsymbol{A}_i} \leftarrow \mathbf{SamplePre}(\boldsymbol{A}_i, \boldsymbol{T}_{\boldsymbol{A}_i}, \boldsymbol{D}, s)$ Finally, it outputs the public key $\mathrm{pk}_i = (\boldsymbol{H}_i, \boldsymbol{A}_i)$ and the secret key $\mathrm{sk}_i = (\boldsymbol{T}_{\boldsymbol{H}_i}, \boldsymbol{R}_{\boldsymbol{A}_i})$.

– $\mathbf{Enc}(\mathrm{pp}, \boldsymbol{\mu}, \mathrm{pk}_i) \to c_0^i$. To encrypt $\boldsymbol{\mu} \in \{0,1\}^m$, this algorithm randomly chooses $\boldsymbol{s} \leftarrow \mathbb{Z}_q^n$ and takes Gaussian noise vectors $\boldsymbol{e}_{0,1}^i, \boldsymbol{e}_{0,2}^i, \boldsymbol{e}_{0,3}^i \in \chi^m$. Let $\mathrm{pk}_i = (\boldsymbol{H}_i, \boldsymbol{A}_i)$ and set

$$\begin{aligned}\boldsymbol{c}_{0,1}^i &= \boldsymbol{H}_i^{\mathrm{T}}\boldsymbol{s} + \boldsymbol{e}_{0,1}^i \in \mathbb{Z}_q^m, \\ \boldsymbol{c}_{0,2}^i &= \boldsymbol{A}_i^{\mathrm{T}}\boldsymbol{s} + \boldsymbol{e}_{0,2}^i \in \mathbb{Z}_q^m, \\ \boldsymbol{c}_{0,3}^i &= \boldsymbol{D}^{\mathrm{T}}\boldsymbol{s} + \boldsymbol{e}_{0,3}^i + \lfloor q/2 \rfloor \boldsymbol{\mu} \in \mathbb{Z}_q^m.\end{aligned}$$

Output the ciphertext $\boldsymbol{c}_0^i = (\boldsymbol{c}_{0,1}^i, \boldsymbol{c}_{0,2}^i, \boldsymbol{c}_{0,3}^i) \in \mathbb{Z}_q^{3m}$.

– $\mathbf{Dec}(\mathrm{pp}, \mathrm{sk}_{i_j}, \boldsymbol{c}_j^i) \to \boldsymbol{\mu}/\perp$. To decrypt the ciphertext $\boldsymbol{c}_j^i = (\boldsymbol{c}_{j,1}^i, \boldsymbol{c}_{j,2}^i, \boldsymbol{c}_{j,3}^i) \in \mathbb{Z}_q^{3m}$, this algorithm uses the secret key $\mathrm{sk}_{i_j} = (\boldsymbol{T}_{\boldsymbol{H}_{i_j}}, \boldsymbol{R}_{\boldsymbol{A}_{i_j}})$ to do the following computation:

$$\hat{\boldsymbol{\mu}} = \left[-\boldsymbol{R}_{\boldsymbol{A}_{i_j}}^{\mathrm{T}} \ \ \boldsymbol{I}_{m\times m}\right] \begin{bmatrix} \boldsymbol{c}_{j,2}^i \\ \boldsymbol{c}_{j,3}^i \end{bmatrix}.$$

Let $\hat{\boldsymbol{\mu}} = (\hat{\mu}_1, \ldots, \hat{\mu}_m)$. For any $i \in \{1, \ldots, m\}$, it sets $\mu_i = 0$ if $\hat{\mu}_i \in [-\lfloor q/4 \rfloor, \lfloor q/4 \rfloor)$. Otherwise sets $\mu_i = 1$. Finally, it outputs $\boldsymbol{\mu} = \{\mu_k\}_{k\in\{1,\ldots,m\}}$.

– $\mathbf{CreatPath}(\mathrm{pp}, \mathrm{pk}_i) \to (\mathrm{path}_i, l_i \leqslant \ell)$. On input the public parameters and the public key pk_i of user i, this algorithm outputs the autonomous delegation path_i with l_i delegatees in the path. The autonomous delegation path can be represented as $\mathrm{path}_i = (\mathrm{pk}_i = \mathrm{pk}_{i_0}, \ldots, \mathrm{pk}_{i_j}, \ldots, \mathrm{pk}_{i_{l_i}})$ where $\mathrm{pk}_{i_j} \leftarrow \mathbf{KeyGen}(\mathrm{pp}, i_j)$. This algorithm outputs path_i if $\mathrm{pk}_{i_v} \neq \mathrm{pk}_{i_w}$ holds for any $v, w \in \{0, \ldots, l_i\}$. Otherwise, it outputs $\perp$.

– $\mathbf{ReKeyGen}(\mathrm{pp}, \mathrm{sk}_i, \mathrm{path}_i, l_i) \to \mathrm{rk}_i = \{\mathrm{rk}_{j\to j+1}^i\}_{j\in\{0,1,\ldots,l_i-1\}}$. On input the secret key sk_i, the delegation path and its length, this algorithm outputs rk_i or $\perp$. To generate these re-encryption keys, the delegator with sk_i computes the re-encryption key between user i_j and user i_{j+1} in path_i, where $j \in \{0, 1, \ldots, l_i - 1\}$, as follows:

$$\boldsymbol{Q}_{j\to j+1}^i \leftarrow \mathbf{SamplePre}(\boldsymbol{H}_i, \boldsymbol{T}_{\boldsymbol{H}_i}, \boldsymbol{A}_{i_{j+1}} - \boldsymbol{A}_{i_j}, s).$$

Such that $\boldsymbol{H}_i \boldsymbol{Q}_{j\to j+1}^i = \boldsymbol{A}_{i_{j+1}} - \boldsymbol{A}_{i_j}$. Let $\mathrm{rk}_{j\to j+1}^i = \boldsymbol{Q}_{j\to j+1}^i$. Finally, the algorithm outputs $\mathrm{rk}_i = \{\mathrm{rk}_{j\to j+1}^i\}_{j\in\{0,1,\ldots,l_i-1\}}$.

– $\mathbf{ReEnc}(\mathrm{pp}, \mathrm{path}_i, \boldsymbol{c}_j^i, \mathrm{rk}_{j\to j+1}^i) \to \boldsymbol{c}_{j+1}^i$, where $j \in \{0, \ldots, l_i - 1\}$. To re-encrypt the ciphertext under pk_{i_j} to the ciphetext under $\mathrm{pk}_{i_{j+1}}$, this algorithm output $\perp$ if $(\mathrm{pk}_{i_j}, \mathrm{pk}_{i_{j+1}}) \notin \mathrm{path}_i$. Otherwise, it makes $\boldsymbol{c}_j^i = (\boldsymbol{c}_{j,1}^i, \boldsymbol{c}_{j,2}^i, \boldsymbol{c}_{j,3}^i)$ and computes

$$\boldsymbol{c}_{j+1,1}^i = \boldsymbol{c}_{j,1}^i \ , \ \boldsymbol{c}_{j+1,3}^i = \boldsymbol{c}_{j,3}^i,$$

$$\boldsymbol{c}_{j+1,2}^i = \boldsymbol{Q}_{j\to j+1}^{i\mathrm{T}} \boldsymbol{c}_{j,1}^i + \boldsymbol{c}_{j,2}^i.$$

Finally, this algorithm outputs $\boldsymbol{c}_{j+1}^i = (\boldsymbol{c}_{j+1,1}^i, \boldsymbol{c}_{j+1,2}^i, \boldsymbol{c}_{j+1,3}^i)$.

In our formal construction, the secret key $\mathrm{sk}_i = (\boldsymbol{T}_{\boldsymbol{H}_i}, \boldsymbol{R}_{\boldsymbol{A}_i})$ has two parts: $\boldsymbol{T}_{\boldsymbol{H}_i}$ is used to generate re-encryption key and $\boldsymbol{R}_{\boldsymbol{A}_i}$ is used for decryption. For a ciphertext $\boldsymbol{c}_j^i$, user i can use $\boldsymbol{T}_{\boldsymbol{H}_i}$ to sample a matrix $\boldsymbol{Q}_{j\to j+1}^i$ for re-encryption, while user i_j can use $\boldsymbol{c}_{j,2}^i$, $\boldsymbol{c}_{j,3}^i$ and $\boldsymbol{R}_{\boldsymbol{A}_{i_j}}$ to recover the plaintext. Besides, the equation $\boldsymbol{H}_i \boldsymbol{Q}_{j\to j+1}^i = \boldsymbol{A}_{i_{j+1}} - \boldsymbol{A}_{i_j}$ ensures that the proxy must first generate $\boldsymbol{c}_j^i$ before it can generate $\boldsymbol{c}_{j+1}^i$. Therefore, the proxy must generate the re-encrypted ciphertext along path_i. The way to generate the re-encryption key in our scheme also affects our security proof. Suppose that i^* is the target user and path_{i^*} is the delegation path originating from i^*. To generate the key pairs and the re-encryption keys related to path_{i^*}, we should first randomly choose $\boldsymbol{Q}_{j\to j+1}^{i^*}$ and generate $\boldsymbol{A}_{i_j^*}$ before we generate $\boldsymbol{A}_{i_{j+1}^*}$. So the challenge path should be determined at the beginning of the game.

Correctness. Based on the above parameters, the correctness holds as follows.

(1) For any user i who has $\mathrm{sk}_i = (\boldsymbol{T}_{\boldsymbol{H}_i}, \boldsymbol{R}_{\boldsymbol{A}_i})$, the decryption of the original ciphertext $\boldsymbol{c}_0^i = (\boldsymbol{c}_{0,1}^i, \boldsymbol{c}_{0,2}^i, \boldsymbol{c}_{0,3}^i)$ under pk_i is

$$\begin{aligned} \left[-\boldsymbol{R}_{\boldsymbol{A}_i}^{\mathrm{T}} \ \ \boldsymbol{I}_{m\times m}\right] \begin{bmatrix} \boldsymbol{c}_{0,2}^i \\ \boldsymbol{c}_{0,3}^i \end{bmatrix} &= -\boldsymbol{R}_{\boldsymbol{A}_i}^{\mathrm{T}} \boldsymbol{c}_{0,2}^i + \boldsymbol{c}_{0,3}^i \\ &= -(\boldsymbol{R}_{\boldsymbol{A}_i}^{\mathrm{T}} \boldsymbol{A}_i^{\mathrm{T}} \boldsymbol{s} + \boldsymbol{R}_{\boldsymbol{A}_i}^{\mathrm{T}} \boldsymbol{e}_{0,2}^i) + \boldsymbol{D}_i^{\mathrm{T}} \boldsymbol{s} + \boldsymbol{e}_{0,3}^i + \lfloor q/2 \rfloor \boldsymbol{\mu} \\ &= \boldsymbol{e}_{0,3}^i - \boldsymbol{R}_{\boldsymbol{A}_i}^{\mathrm{T}} \boldsymbol{e}_{0,2}^i + \lfloor q/2 \rfloor \boldsymbol{\mu}, \end{aligned}$$

where $\|\boldsymbol{e}_{0,3}^i - \boldsymbol{R}_{\boldsymbol{A}_i}^{\mathrm{T}} \boldsymbol{e}_{0,2}^i\| \leqslant \|\boldsymbol{e}_{0,3}^i\| + \|\boldsymbol{R}_{\boldsymbol{A}_i}^{\mathrm{T}} \boldsymbol{e}_{0,2}^i\| \leqslant \sqrt{m}\alpha q + m\sqrt{m}\alpha q s \leqslant 2m\sqrt{m}\alpha q s \leqslant q/(2(\mathrm{O}(\ell)+3)ms) \leqslant \lfloor q/4 \rfloor$, which ensure the correct decryption to $\boldsymbol{\mu} \in \{0,1\}^m$.

(2) For any $j \in \{0, \ldots, l_i - 1\}$, given the re-encryption key $\mathrm{rk}_{j\to j+1}^i = \boldsymbol{Q}_{j\to j+1}^i$ satisfying $\boldsymbol{H}_i \boldsymbol{Q}_{j\to j+1}^i = \boldsymbol{A}_{i_{j+1}} - \boldsymbol{A}_{i_j}$, the re-encrypted ciphertext represented as $\boldsymbol{c}_{j+1}^i = (\boldsymbol{c}_{j+1,1}^i, \boldsymbol{c}_{j+1,2}^i, \boldsymbol{c}_{j+1,3}^i)$ can be calculated

as

$$\begin{aligned}
\boldsymbol{c}^i_{j+1,1} &= \boldsymbol{c}^i_{j,1},\\
\boldsymbol{c}^i_{j+1,3} &= \boldsymbol{c}^i_{j,3},\\
\boldsymbol{c}^i_{j+1,2} &= \boldsymbol{Q}^{i\mathrm{T}}_{j\to j+1}\boldsymbol{c}^i_{j,1} + \boldsymbol{c}^i_{j,2}\\
&= \boldsymbol{Q}^{i\mathrm{T}}_{j\to j+1}(\boldsymbol{H}_i^{\mathrm{T}}\boldsymbol{s} + \boldsymbol{e}^i_{0,1}) + (\boldsymbol{A}^{\mathrm{T}}_{i_j}\boldsymbol{s} + \boldsymbol{e}^i_{j,2})\\
&= \boldsymbol{A}^{\mathrm{T}}_{i_{j+1}}\boldsymbol{s} + \boldsymbol{Q}^{i\mathrm{T}}_{j\to j+1}\boldsymbol{e}^i_{0,1} + \boldsymbol{e}^i_{j,2},
\end{aligned}$$

where $\boldsymbol{e}^i_{j,2} = \boldsymbol{e}^i_{0,2} + \sum_{k=1}^{j}\boldsymbol{Q}^{i\mathrm{T}}_{k-1\to k}\boldsymbol{e}^i_{0,1}$ and $\|\boldsymbol{e}^i_{j,2}\| \leqslant \|\boldsymbol{e}^i_{0,2}\| + \sum_{k=1}^{j}(\|\boldsymbol{Q}^{i\mathrm{T}}_{k-1\to k}\boldsymbol{e}^i_{0,1}\|) \leqslant (j+1)q/(4(\mathrm{O}(\ell)+3)ms) \leqslant \lfloor q/4\rfloor$. So the decryption of c^i_{j+1} is

$$\begin{aligned}
\begin{bmatrix}-\boldsymbol{R}^{\mathrm{T}}_{\boldsymbol{A}_{i_{j+1}}} & \boldsymbol{I}_{m\times m}\end{bmatrix}\begin{bmatrix}\boldsymbol{c}^i_{j+1,2}\\ \boldsymbol{c}^i_{j+1,3}\end{bmatrix} &= -\boldsymbol{R}^{\mathrm{T}}_{\boldsymbol{A}_{i_{j+1}}}\boldsymbol{c}^i_{j+1,2} + \boldsymbol{c}^i_{j+1,3}\\
&= -\boldsymbol{R}^{\mathrm{T}}_{\boldsymbol{A}_{i_{j+1}}}\boldsymbol{e}^i_{j+1,2} + \boldsymbol{e}^i_{0,3} + \lfloor q/2\rfloor\boldsymbol{\mu},
\end{aligned}$$

where $\|\boldsymbol{R}^{\mathrm{T}}_{\boldsymbol{A}_{i_{j+1}}}\boldsymbol{e}^i_{j+1,2} + \boldsymbol{e}^i_{0,3}\| \leqslant ms\cdot(j+2)q/(4(\mathrm{O}(\ell)+3)ms) + \sqrt{m}\alpha q \leqslant (j+3)q/(4(\mathrm{O}(\ell)+3)) \leqslant \lfloor q/4\rfloor$. Therefore, the re-encrypted ciphertext can be decrypted correctly.

4.3 Security proof

We show that our AP-PRE scheme is CPA secure in the selective-path model.

Theorem 3. The AP-PRE scheme above is sCPA secure under the hardness of LWE.

Proof sketch. We build an algorithm $\mathcal{B}$, which solves LWE problem by invoking a PPT adversary $\mathcal{A}$.

At the beginning, $\mathcal{B}$ is given a random matrix $[\boldsymbol{H}|\boldsymbol{A}|\boldsymbol{D}]$ and a vector $\boldsymbol{b}$ which might be uniformly random or an LWE instance. For the simplicity of symbolic expression, we set $d = l_{i^*}$ where i^* is the target delegator and l_{i^*} is the number of delegatees in the challenge path. Then $\mathcal{A}$ announces a sequence of user identities $\mathcal{S} = \{i^*_0 = i^*, i^*_1, \ldots, i^*_d\}$ where $d \leqslant \ell$. After receiving $\mathcal{S}$, $\mathcal{B}$ sets $\boldsymbol{D}$ to be public parameter and sets $\mathrm{pk}_{i^*} = (\boldsymbol{H}, \boldsymbol{A})$. For any $j \in \{1, \ldots, d\}$, $\mathcal{B}$ chooses $\boldsymbol{Q}^{i^*}_{j-1\to j} \leftarrow D_{\mathbb{Z}^{m\times m},s}$, computes $\boldsymbol{A}_{i^*_j} = \boldsymbol{H}\boldsymbol{Q}^{i^*}_{j-1\to j} + \boldsymbol{A}_{i^*_{j-1}}$ and runs $\mathbf{TrapGen}(1^n, m, q)$ to generate $(\boldsymbol{H}_{i^*_j}, \boldsymbol{T}_{\boldsymbol{H}_{i^*_j}})$. And $\mathcal{B}$ records $(\mathrm{pk}_{i^*_j} = (\boldsymbol{H}_{i^*_j}, \boldsymbol{A}_{i^*_j}), \mathrm{sk}_{i^*_j} = \boldsymbol{T}_{\boldsymbol{H}_{i^*_j}})$ in T_{kp}. During the game, $\mathcal{A}$ can make a corrupted key generation query on any i for which $i \notin \mathcal{S}$. When $\mathcal{A}$ makes a re-encryption key generation on $\mathcal{O}_{\mathrm{rk}}(i^*, \mathrm{pk}_{i^*_j}, \mathrm{pk}_{i^*_{j+1}})$, $\mathcal{B}$ returns $\mathrm{rk}^{i^*}_{j\to j+1} = \boldsymbol{Q}^{i^*}_{j\to j+1}$. In the **Challenge** phase, $\mathcal{B}$ returns $\boldsymbol{b} + [\boldsymbol{0}, \boldsymbol{0}, \lfloor q/2\rfloor\boldsymbol{\mu}_b]$ to $\mathcal{A}$. Finally, $\mathcal{B}$ outputs the result that $\mathcal{A}$ returns.

Proof. Our proof can be described as a sequence of games. The first game of the sequence is identical to the game as defined in Definition 9. In the last game, $\mathcal{A}$'s advantage is zero. The LWE problem is used in the proof of indistinguishability between the last two games. Note that a user sequence $\mathcal{S} = \{i^*_0 = i^*, i^*_1, \ldots, i^*_d\}$ is announced by the adversary in **Init** phase.

Game 0. This is the game identical to the game in Definition 9.

Game j**.** Recall that d is the number of delegatees in the challenge path and j denotes the j-th delegatee in the challenge path. For any $j \in \{1, \ldots, d\}$, the difference between **Game** $j-1$ and **Game** j can be described as follows:

In **Init** phase, the challenger in **Game** $j-1$ generates $\boldsymbol{A}_{i^*_j}$ by running $(\boldsymbol{A}_{i^*_j}, \boldsymbol{T}_{\boldsymbol{A}_{i^*_j}}) \leftarrow \mathbf{TrapGen}(1^n, m, q)$. **Game** j is identical to **Game** $j-1$ except that $\mathcal{C}$ generates $\boldsymbol{A}_{i^*_j}$ for user i^*_j in different ways. Let $(\boldsymbol{H}_{i^*_j}, \boldsymbol{A}_{i^*_j})$ be the public key for user i^*_j, where $\boldsymbol{H}_{i^*_j}$ is generated in the same way as the original scheme. $\boldsymbol{A}_{i^*_j} \in \mathbb{Z}_q^{n\times m}$ is constructed as

$$\boldsymbol{A}_{i^*_j} = \boldsymbol{H}_{i^*}\boldsymbol{Q}^{i^*}_{j-1\to j} + \boldsymbol{A}_{i^*_{j-1}}, \text{ where } \boldsymbol{Q}^{i^*}_{j-1\to j} \leftarrow D_{\mathbb{Z}^{m\times m},s}.$$

Then the key pairs that generated in **Init** phase are recorded in T_{kp}, the challenge path $\mathrm{path}_{i^*} = \{(\boldsymbol{H}_{i^*_j}, \boldsymbol{A}_{i^*_j})\}_{j\in\{0,1,\ldots,d\}}$ is recorded in T_{path} and the re-encryption key $\mathrm{rk}_{i^*} = \{\boldsymbol{Q}^{i^*}_{j-1\to j}\}_{j\in\{1,\ldots,d\}}$ is recorded in T_{rk}. Because all key pairs and re-encryption keys that related to path_{i^*} are recorded in three tables in advance, all oracles are queried as defined in Subsection 3.2.

We show that **Game** j is statistically indistinguishable from **Game** $j-1$ by Lemma 4. Let $\boldsymbol{B}$ be uniform over $\mathbb{Z}_q^{n\times m}$. By Lemma 4 the distribution of $(\boldsymbol{H}_{i^*}, \boldsymbol{H}_{i^*}\boldsymbol{Q}^{i^*}_{j-1\to j} \bmod q)$ is statistically close to $(\boldsymbol{H}, \boldsymbol{B})$. Therefore, in **Game** j, $\boldsymbol{A}_{i^*_j}$ is statistically close to uniform over $\mathbb{Z}_q^{n\times m}$. It follows that in the

adversary's view, $\boldsymbol{H}_{i^*}\boldsymbol{Q}^{i^*}_{j-1\to j}+\boldsymbol{A}_{i^*_{j-1}}$ is statistically indistinguishable from a uniformly random matrix in $\mathbb{Z}_q^{n\times m}$. Hence, **Game** j and **Game** $j-1$ are statistically indistinguishable.

Game $d+1$**.** **Game** $d+1$ is identical to **Game** d except that now $\boldsymbol{A}_{i^*}$ is a uniformly random matrix but not generated by running $\mathbf{TrapGen}(1^n,m,q)$. By Theorem 2, the distribution of $\boldsymbol{A}_{i^*}$ in **Game** d is statistically close to a uniform matrix in $\mathbb{Z}_q^{n\times m}$. So $\mathcal{A}$ has negligible advantages to distinguish between **Game** d and **Game** $d+1$.

Game $d+2$**.** Recall that $\boldsymbol{A}_{i^*}$ and $\boldsymbol{D}$ are uniformly random matrices in **Game** $d+1$. In this game, $\mathcal{C}$ randomly chooses a matrix $\boldsymbol{H}_{i^*}$ in $\mathbb{Z}_q^{n\times m}$ and sets $(\boldsymbol{H}_{i^*},\boldsymbol{A}_{i^*})$ to be the public key for user i^*. The challenge ciphertext is $\boldsymbol{b}+[\boldsymbol{0},\boldsymbol{0},\lfloor q/2\rfloor\boldsymbol{\mu}_b]$ where $\boldsymbol{b}=(\boldsymbol{b}_1,\boldsymbol{b}_2,\boldsymbol{b}_3)\in\mathbb{Z}_q^{3m}$ can be represented as

$$\begin{aligned}\boldsymbol{b}_1&=\boldsymbol{H}_{i^*}^{\mathrm{T}}\boldsymbol{s}+\boldsymbol{e}_{0,1}^{i^*},\\ \boldsymbol{b}_2&=\boldsymbol{A}_{i^*}^{\mathrm{T}}\boldsymbol{s}+\boldsymbol{e}_{0,2}^{i^*},\\ \boldsymbol{b}_3&=\boldsymbol{D}^{\mathrm{T}}\boldsymbol{s}+\boldsymbol{e}_{0,3}^{i^*},\end{aligned}$$

for some $\boldsymbol{s}\in\mathbb{Z}_q^n$ and $\boldsymbol{e}_{0,1}^{i^*},\boldsymbol{e}_{0,2}^{i^*},\boldsymbol{e}_{0,3}^{i^*}\in\chi^m$. By Theorem 2, $\boldsymbol{H}_{i^*}$ in **Game** $d+1$ is statistically indistinguishable from a uniform matrix in $\mathbb{Z}_q^{n\times m}$. Therefore, $\boldsymbol{H}_{i^*}$ is indistinguishable between **Game** $d+1$ and **Game** $d+2$.

Game $d+3$**.** **Game** $d+3$ is identical to **Game** $d+2$ except that the challenge ciphertext $\boldsymbol{c}_0^{i^*}$ is always chosen as a random element from $\mathbb{Z}_q^{3m}$. Since the challenge ciphertext is always a fresh random element in the ciphertext space, $\mathcal{A}$ has no advantage in winning this game.

At this point, all remains to prove is that **Game** $d+3$ and **Game** $d+2$ are computationally indistinguishable. To prove this, we suppose that $\mathcal{A}$ has non-negligible advantages in distinguishing **Game** $d+3$ and **Game** $d+2$, while $\mathcal{B}$ is the algorithm to solve LWE problem.

Instance. $\mathcal{B}$ obtains an LWE instance: $(\boldsymbol{H},\boldsymbol{A},\boldsymbol{D})\in\mathbb{Z}_q^{n\times m}\times\mathbb{Z}_q^{n\times m}\times\mathbb{Z}_q^{n\times m}$ and $(\boldsymbol{b}_1,\boldsymbol{b}_2,\boldsymbol{b}_3)\in\mathbb{Z}_q^m\times\mathbb{Z}_q^m\times\mathbb{Z}_q^m$. We have $(\boldsymbol{b}_1,\boldsymbol{b}_2,\boldsymbol{b}_3)$ are either random or

$$\begin{aligned}\boldsymbol{b}_1&=\boldsymbol{H}^{\mathrm{T}}\boldsymbol{s}+\boldsymbol{e}_1,\\ \boldsymbol{b}_2&=\boldsymbol{A}^{\mathrm{T}}\boldsymbol{s}+\boldsymbol{e}_2,\\ \boldsymbol{b}_3&=\boldsymbol{D}^{\mathrm{T}}\boldsymbol{s}+\boldsymbol{e}_3,\end{aligned}$$

for some $\boldsymbol{s}\in\mathbb{Z}_q^n$ and $\boldsymbol{e}_1,\boldsymbol{e}_2,\boldsymbol{e}_3\in\chi^m$.

Targeting. $\mathcal{A}$ announces to $\mathcal{B}$ the target delegator i^* and a set $\mathcal{S}=(i^*,i_1^*,\ldots,i_d^*)$.

Setup. $\mathcal{B}$ sets lattice parameters as in **Game** $d+2$. Let $\mathrm{pp}=(n,m,q,\chi^m,s,\boldsymbol{D})$ where $\boldsymbol{D}$ is an LWE instance given in **Instance** and $(\boldsymbol{A},\boldsymbol{H})$ be the public key for user i^*. For any $j\in\{1,\ldots,d\}$, $\mathcal{B}$ generates $\mathrm{rk}^{i^*}_{j-1\to j}$ and $\mathbf{A}_{i^*_j}$ as in **Game** $d+2$.

Queries. $\mathcal{B}$ answers $\mathcal{A}$'s all queries as in **Game** $d+2$.

Challenge ciphertext. After receiving $(\boldsymbol{\mu}_0,\boldsymbol{\mu}_1)$, $\mathcal{B}$ samples a random bit $b\leftarrow\{0,1\}$ and constructs the challenge ciphertext $\boldsymbol{c}_0^{i^*}=(\boldsymbol{c}_{0,1}^{i^*},\boldsymbol{c}_{0,2}^{i^*},\boldsymbol{c}_{0,3}^{i^*})$ as follows:

$$\boldsymbol{c}_{0,1}^{i^*}=\boldsymbol{b}_1,\ \boldsymbol{c}_{0,2}^{i^*}=\boldsymbol{b}_2,\ \boldsymbol{c}_{0,3}^{i^*}=\boldsymbol{b}_3+\lfloor q/2\rfloor\boldsymbol{\mu}_b.$$

Obviously, if the LWE challenge is pseudorandom, our challenge ciphertext $\boldsymbol{c}_0^{i^*}$ is distributed as in **Game** $d+2$. However, if the LWE instance is random, our challenge ciphertext $\boldsymbol{c}_0^{i^*}$ is distributed as in **Game** $d+3$.

Guess. After making the second phase queries, $\mathcal{A}$ guesses if he is interacting with **Game** $d+2$ or **Game** $d+3$ challenger. $\mathcal{B}$ will output $\mathcal{A}$'s guess as an answer to the LWE challenge.

As stated above, if $\mathcal{A}$ has non-negligible advantages in distinguishing **Game** $d+2$ and **Game** $d+3$, then $\mathcal{B}$ has non-negligible advantages in solving LWE problem. Our scheme can be proved to be adaptive-path secure through general conversion. But the reduction loss is relatively large. Therefore, we are working on a more interesting way to construct an adaptive-path secure AP-PRE scheme.

5 Conclusion

In this work, we propose our LWE-based AP-PRE along with the analysis of its selective-path CPA security. We present an intuitive construction and emphasize that the re-encrypted ciphertext in this

scheme can deviate from the path. To solve this problem, we divide the ciphertext into delegation ciphertext and encrypted messages, which realizes the separation of delegation and decryption rights. Finally, we have two unresolved problems. The first one is to construct an adaptive-path CPA secure AP-PRE based on LWE. The second one is to construct an HRA [19, 29] secure lattice-based AP-PRE.

Acknowledgements Jian WENG was supported by Major Program of Guangdong Basic and Applied Research Project (Grant No. 2019B030302008), National Natural Science Foundation of China (Grant Nos. 61825203, U22B2028), National Key Research and Development Plan of China (Grant No. 2020YFB1005600), Guangdong Provincial Science and Technology Project (Grant No. 2021A0505030033), Science and Technology Major Project of Tibetan Autonomous Region of China (Grant No. XZ202201ZD0006G), National Joint Engineering Research Center of Network Security Detection and Protection Technology, Guangdong Key Laboratory of Data Security and Privacy Preserving and Guangdong Hong Kong Joint Laboratory for Data Security and Privacy Protection. This work was also supported by Special Funds for the Cultivation of Guangdong College Students' Scientific and Technological Innovation ("Climbing Program" Special Funds) (Grant No. pdjh2021a0050). We all thank the anonymous reviewers for their valuable comments and suggestions which improve the content and presentation of this work a lot.

References

1 Blaze M, Bleumer G, Strauss M. Divertible protocols and atomic proxy cryptography. In: Proceedings of International Conference on the Theory and Application of Cryptographic Techniques, Espoo, 1998. 127–144
2 Shao J, Cao Z. Multi-use unidirectional identity-based proxy re-encryption from hierarchical identity-based encryption. Inf Sci, 2012, 206: 83–95
3 Deng H, Qin Z, Wu Q, et al. Flexible attribute-based proxy re-encryption for efficient data sharing. Inf Sci, 2020, 511: 94–113
4 Tang Q. Type-based proxy re-encryption and its construction. In: Proceedings of International Conference on Cryptology in India, Kharagpur, 2008. 130–144
5 Weng J, Deng R H, Ding X H, et al. Conditional proxy re-encryption secure against chosen-ciphertext attack. In: Proceedings of the 4th International Symposium on Information, Computer, and Communications Security, Sydney, 2009. 322–332
6 Cao Z, Wang H, Zhao Y. AP-PRE: autonomous path proxy re-encryption and its applications. IEEE Trans Depend Secure Comput, 2017, 16: 833–842
7 Canetti R, Hohenberger S. Chosen-ciphertext secure proxy re-encryption. In: Proceedings of the 14th ACM Conference on Computer and Communications Security, New York, 2007. 185–194
8 Green M, Ateniese G. Identity-based proxy re-encryption. In: Proceedings of International Conference on Applied Cryptography and Network Security, Zhuhai, 2007. 288–306
9 Wang H, Cao Z, Wang L. Multi-use and unidirectional identity-based proxy re-encryption schemes. Inf Sci, 2010, 180: 4042–4059
10 Liang X H, Cao Z F, Lin H, et al. Attribute based proxy re-encryption with delegating capabilities. In: Proceedings of the 4th International Symposium on Information, Computer, and Communications Security, Sydney, 2009. 276–286
11 Li Z, Sharma V, Ma C, et al. Ciphertext-policy attribute-based proxy re-encryption via constrained PRFs. Sci China Inf Sci, 2021, 64: 169301
12 Fang L M, Wang J D, Ge C P, et al. Fuzzy conditional proxy re-encryption. Sci China Inf Sci, 2013, 56: 052116
13 Zhao J, Feng D G, Zhang Z F. Attribute-based conditional proxy re-encryption with chosen-ciphertext security. In: Proceedings of IEEE Global Telecommunications Conference, Miami, 2010. 1–6
14 Xagawa D K. Cryptography with lattices. Dissertation for Ph.D. Degree. Tokyo: Tokyo Institute of Technology, 2005
15 Kirshanova E. Proxy re-encryption from lattices. In: Proceedings of International Workshop on Public Key Cryptography, Buenos Aires, 2014. 77–94
16 Micciancio D, Peikert C. Trapdoors for lattices: simpler, tighter, faster, smaller. In: Proceedings of Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, 2012. 700–718
17 Jiang M M, Hu Y P, Wang B C, et al. Lattice-based multi-use unidirectional proxy re-encryption. Secur Comm Netw, 2015, 8: 3796–3803
18 Liang X J, Weng J, Yang A J, et al. Attribute-based conditional proxy re-encryption in the standard model under LWE. In: Proceedings of European Symposium on Research in Computer Security, Darmstadt, 2021. 147–168
19 Susilo W, Dutta P, Duong D H, et al. Conditional proxy re-encryption secure against chosen-ciphertext attackLattice-based HRA-secure attribute-based proxy re-encryption in standard model. In: Proceedings of European Symposium on Research in Computer Security, Darmstadt, 2021. 169–191
20 Boneh D, Gentry C, Gorbunov S, et al. Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In: Proceedings of Annual International Conference on the Theory and Applications of Cryptographic Techniques, Berlin, 2014. 533–556
21 Wang H B, Cao Z F. A fully secure unidirectional and multi-use proxy re-encryption scheme. ACM CCS, Poster Session, 2009. https://www.sigsac.org/ccs/CCS2009/pd/abstract_16.pdf
22 Aono Y, Boyen X, Phong L T, et al. Key-private proxy re-encryption under LWE. In: Proceedings of the 14th International Conference on Cryptology in India, Mumbai, 2013. 1–18
23 Luo F, Al-Kuwari S, Wang F, et al. Attribute-based proxy re-encryption from standard lattices. Theor Comput Sci, 2021, 865: 52–62
24 Regev O. On lattices, learning with errors, random linear codes, and cryptography. J ACM, 2009, 56: 1–40
25 Peikert C. Public-key cryptosystems from the worst-case shortest vector problem. In: Proceedings of the 41st Annual ACM Symposium on Theory of Computing, New York, 2009. 333–342
26 Micciancio D, Regev O. Worst-case to average-case reductions based on Gaussian measures. SIAM J Comput, 2007, 37: 267–302
27 Gentry C, Peikert C, Vaikuntanathan V. Trapdoors for hard lattices and new cryptographic constructions. In: Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, 2008. 197–206
28 Agrawal S, Boneh D, Boyen X. Efficient lattice (H) IBE in the standard model. In: Proceedings of Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, 2010. 553–572
29 Cohen A. What about bob? The inadequacy of CPA security for proxy reencryption. In: Proceedings of IACR International Workshop on Public Key Cryptography, Beijing, 2019. 287–316