Frontiers of Information Technology & Electronic Engineering
www.jzus.zju.edu.cn; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

Identity-based threshold proxy re-encryption scheme from lattices and its applications*

Liqiang WU[†1], Yiliang HAN[†‡1], Xiaoyuan YANG[1,2], Minqing ZHANG[1]

[1]*Key Laboratory of Network and Information Security, Engineering University of People's Armed Police, Xi'an 710086, China*

[2]*MOE Key Laboratory of Computer Network and Information Security, Xidian University, Xi'an 710071, China*

[†]E-mail: latticewj@163.com; hanyil@163.com

Received July 21, 2020; Revision accepted Oct. 8, 2020; Crosschecked Oct. 12, 2021

Abstract: Threshold proxy re-encryption (TPRE) can prevent collusion between a single proxy and a delegatee from converting arbitrary files against the wishes of the delegator through multiple proxies, and can also provide normal services even when certain proxy servers are paralyzed or damaged. A non-interactive identity-based TPRE (IB-TPRE) scheme over lattices is proposed which removes the public key certificates. To accomplish this scheme, Shamir's secret sharing is employed twice, which not only effectively hides the delegator's private key information, but also decentralizes the proxy power by splitting the re-encryption key. Robustness means that a combiner can detect a misbehaving proxy server that has sent an invalid transformed ciphertext share. This property is achieved by lattice-based fully homomorphic signatures. As a result, the whole scheme is thoroughly capable of resisting quantum attacks even when they are available. The security of the proposed scheme is based on the decisional learning with error hardness assumption in the standard model. Two typical application scenarios, including a file-sharing system based on a blockchain network and a robust key escrow system with threshold cryptography, are presented.

Key words: Post-quantum cryptography; Threshold proxy re-encryption; Lattices; Robustness; Decentralization
https://doi.org/10.1631/FITEE.2000366 **CLC number:** TP309

1 Introduction

Proxy re-encryption (PRE) (Blaze et al., 1998) is a cryptographic primitive allowing a semi-trusted proxy to convert a ciphertext under Alice's public key into the ciphertext of the same message under Bob's public key, without allowing access to sensitive data or full decryption. PRE is extended to the identity-based environment, named identity-based PRE (IB-PRE) (Green and Ateniese, 2007), where public keys of Alice and Bob can be expressed by any unique strings. IB-PRE can dramatically ease the workload for public key certificate management due to the "identity matching public key" property, so it has more abundant and practical scenarios, such as secure email forwarding with IBE and attribute-based delegations.

1.1 Related works

Lou and Cao (2010) combined PRE with a threshold technique for the first time, and proposed an identity-based threshold proxy re-encryption (IB-TPRE) scheme based on bilinear mapping. David (2018) proposed a TPRE scheme called UMBRAL using BBS-PRE and Shamir's secret sharing. UMBRAL is a central solution to the NuCypher key management system (Egorov et al., 2017), and provides encryption and cryptographic access control

‡ Corresponding author

* Project supported by the National Natural Science Foundation of China (Nos. U1636114, 61572521, and 61772550), the Innovative Research Team in Engineering University of People's Armed Police, China (No. KYTD201805), the Natural Science Foundation of Shaanxi Province, China (No. 2021JM-252), and the Basic Research Project of Engineering University of People's Armed Police, China (No. WJY201914)

ORCID: Liqiang WU, https://orcid.org/0000-0002-4314-3592; Yiliang HAN, https://orcid.org/0000-0002-2116-5408

that is performed by a decentralized network and PRE. However, these schemes are unable to resist quantum attacks because of the Shor algorithm (Shor, 1997).

To deal with the challenges of quantum computing, cryptosystems from lattices, especially from learning with error (LWE), are booming because of similar average/worst case equivalence under a quantum/classical reduction, and because of the simple algebraic structure and almost linear operations. The first bidirectional and interactive PRE scheme (Xagawa, 2010) was constructed based on the LWE hardness assumption (Regev, 2009). Subsequent research on PRE over lattices includes mainly three aspects: (1) optimization of PRE schemes over LWE, such as a key-private PRE scheme under the standard model (Aono et al., 2013) and a unidirectional, collusion-resistant, and CCA-secure PRE scheme (Kirshanova, 2014); (2) PRE schemes based on NTRU—Nuñez et al. (2015) proposed two unidirectional, multi-hop, non-interactive PRE schemes based on NTRU and Polyakov et al. (2017) constructed an NTRU-PRE scheme and estimated its performance with the Palisade Library (Polyakov et al., 2018); (3) enhancement of the PRE security model—An indistinguishable security model under honest re-encryption attack (HRA) (Cohen, 2019) was proposed, along with the concept of adaptive security (Fuchsbauer et al., 2019), which allows corrupting users at any stage of simulation.

To work under the identity-based framework, the first anonymous multi-hop IB-PRE scheme (Singh et al., 2014) was constructed under the random oracle model, which was bidirectional and interactive. Another IB-PRE scheme (Singh et al., 2014) was constructed, generating two private keys for each user, i.e., for decryption and re-encryption. The resulting scheme achieved indistinguishability of ciphertext under a selective-identity chosen-plaintext attack (IND-sID-CPA) and weak collusion resistance.

To enhance security and fault tolerance, a lattice-based universal threshold constructor was proposed (Boneh et al., 2017) using homomorphic encryptions and homomorphic signatures, which could extend any cryptographic scheme into a threshold cryptosystem. A re-splittable threshold PRE scheme (Li et al., 2017) was constructed using two basic encryption schemes from lattices and Shamir's secret sharing, in which any ciphertext share provided by the proxy server is publicly verifiable. However, due to the use of decisional Diffie–Hellman (DDH) over the discrete logarithm assumption, the resulting scheme cannot resist quantum attack, and many exponential operations with high complexity are involved in the verification, thus reducing the overall performance.

1.2 Motivation and our contributions

From the perspective of practice, distributed architectures, such as blockchain and Hadoop, which can provide strong reliability and scalability, have been rapidly developed and deployed recently. In an IB-PRE system, the semi-trusted proxy function is typically performed by a single server to carry out ciphertext conversion, so it almost cannot meet the current requirements under the new situation mainly due to three aspects: First, the centralized proxy server may not be constantly online due to a single point of failure or a denial-of-service attack; Second, a single proxy node in complete control of the whole re-encryption key may develop into an attractive attack target at the risk of permission abuse; Third, the single proxy server is the crucial node of the network connection, and its ciphertext conversion speed will become a bottleneck of the whole system.

A feasible solution to this problem is to decentralize the proxy conversion permission, for example, to divide and delegate it in multiple entities. Only when a certain number of entities meet the requirements of re-encryption, can the ciphertext conversion be completed successfully, thus forming an IB-TPRE scheme (Lou and Cao, 2010). IB-TPRE can effectively reduce or avoid some security risks, such as permission abuse, key exposure, or full control of the re-encryption key caused by a single proxy, and significantly improve fault tolerance and security. A comparison of IB-PRE and IB-TPRE is shown in Fig. 1. Here, if more than two of the three proxies perform ciphertext conversion, they can complete the conversion task successfully.

From a theoretical perspective, the development of quantum computers has posed serious threats to the security of public-key cryptography. So, the construction of cryptographic schemes against quantum attack is an important trend in cryptography research.

To address these issues, a non-interactive IB-TPRE scheme from LWE (Regev, 2009) is proposed,

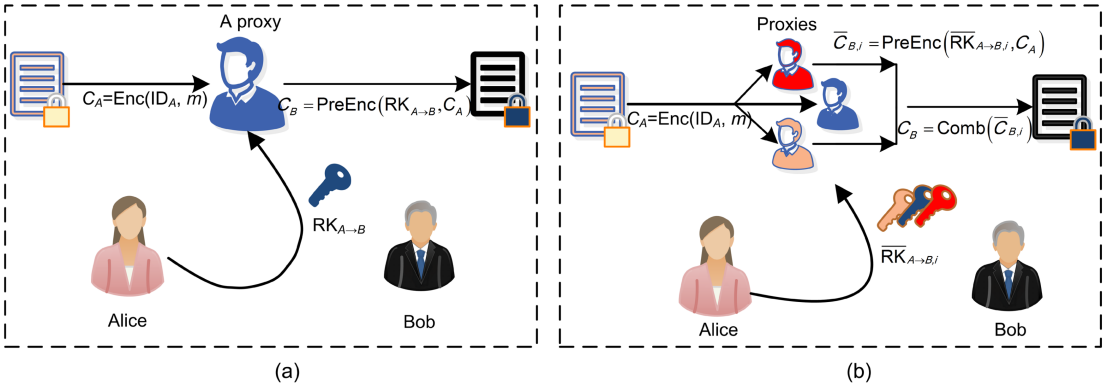

Fig. 1 Ciphertext conversion in IB-PRE (a) and IB-TPRE (b)

and the contributions of this paper are summarized as follows:

1. Shamir's secret sharing is employed twice in our construction. For the first time, it splits a public vector into multiple feature vectors attached to each proxy server. The user's private trapdoor is used to hide the delegator's private key information perfectly by sampling a short lattice basis with regard to its feature vector. For the second time, it splits the re-encryption key. The homomorphism between the re-encryption key shares and ciphertext shares makes the overall ciphertext conversion hold, which effectively decentralizes the proxy power.

2. A homomorphic signature from lattices is applied to make our scheme robust. Robustness means that a forged or wrong ciphertext share can be detected immediately. Each proxy server owns a share of the re-encryption key and the corresponding signature. Therefore, on verification, the evaluation circuit (defined by the original ciphertext), the new message (the re-encrypted ciphertext share), and the new signature (the signature corresponding to the re-encrypted ciphertext share) must be consistent. Our scheme realizes robustness using the unforgeability of homomorphic signatures.

3. Our scheme has many practical applications because of its high availability, low trust, and strong security. High availability means that ciphertext transformation can be completed successfully, even if one or more proxy servers are unavailable. Low trust means that dishonest or malicious transformation behaviors of proxy servers are detected by public verification, thus reducing the trust in them. Strong security means that quantum attacks are resisted by constructing schemes over lattices. Our scheme can be applied in access control in a decentralized environment, such as a file-sharing system based on a blockchain network and a robust key escrow system with threshold cryptography.

2 Preliminaries

2.1 Lattices and LWE

Denote the sets of real numbers and integers by $\mathbb{R}$ and $\mathbb{Z}$, respectively. Vectors are represented as lowercase bold letters, and matrices are represented as uppercase bold letters. The symbol $(\cdot|\cdot)$ represents the horizontal connection and $(\cdot;\cdot)$ represents the vertical connection. An n-dimensional vector is designed as $\boldsymbol{s}=(s_1, s_2, \ldots, s_n)$. For matrix $\boldsymbol{S}=(\boldsymbol{s}_1, \boldsymbol{s}_2, \ldots, \boldsymbol{s}_k)$, $\|\boldsymbol{S}\|$ denotes the longest vector in $\boldsymbol{S}$; that is, $\|\boldsymbol{S}\|=\max\|\boldsymbol{s}_i\|$ for $1\le i\le k$. $\tilde{\boldsymbol{S}}=(\tilde{\boldsymbol{s}}_1,\tilde{\boldsymbol{s}}_2,\cdots,\tilde{\boldsymbol{s}}_k)$ is the Gram-Schmidt orthogonalization of $\boldsymbol{S}=(\boldsymbol{s}_1, \boldsymbol{s}_2, \ldots, \boldsymbol{s}_k)$, so $\|\tilde{\boldsymbol{S}}\|$ is the Gram-Schmidt norm of $\boldsymbol{S}$. $[N]$ denotes the set $\{1, 2, \ldots, N\}$ for an integer N. A share of secret S in Shamir's secret sharing scheme is denoted by $\overline{S}$.

The lattice is defined as follows: Given n-dimensional linearly independent vectors $\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_m\in\mathbb{Z}^n$, the integer lattice generated by matrix $\boldsymbol{B}=[\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_m]\in\mathbb{Z}^{n\times m}$ is the set of vectors:

$$L(\boldsymbol{B})=\left\{\boldsymbol{y}\in\mathbb{Z}^n \mid \exists \boldsymbol{s}\in\mathbb{Z}^m, \boldsymbol{y}=\boldsymbol{B}\boldsymbol{s}=\sum\nolimits_{i=1}^{m} s_i\boldsymbol{b}_i\right\}. \quad (1)$$

Given a prime q, a vector $\boldsymbol{e}\in\mathbb{Z}^m$, and a matrix $\boldsymbol{A}\in\mathbb{Z}_q^{n\times m}$, q-ary lattices are expressed as

$$\begin{cases} \Lambda_q(\boldsymbol{A})=\left\{\boldsymbol{e}\in\mathbb{Z}^m \mid \exists \boldsymbol{s}\in\mathbb{Z}_q^n, \boldsymbol{A}^{\mathrm{T}}\boldsymbol{s}=\boldsymbol{e} \bmod q\right\}, \\ \Lambda_q^{\perp}(\boldsymbol{A})=\left\{\boldsymbol{e}\in\mathbb{Z}^m \mid \boldsymbol{A}\boldsymbol{e}=\boldsymbol{0} \bmod q\right\}, \\ \Lambda_q^{\boldsymbol{u}}(\boldsymbol{A})=\left\{\boldsymbol{e}\in\mathbb{Z}^m \mid \boldsymbol{A}\boldsymbol{e}=\boldsymbol{u} \bmod q\right\}. \end{cases} \tag{2}$$

Let L be a subset of $\mathbb{Z}^n$. For a variance σ, let a Gaussian function centered at $\boldsymbol{0}$ be $\rho_\sigma(\boldsymbol{x})=\mathrm{e}^{-\pi\|\boldsymbol{x}\|^2/\sigma^2}$. If $\rho_\sigma(L)=\sum_{\boldsymbol{x}}\rho_{\boldsymbol{x}\in L}(\boldsymbol{x})$ is finite and computable, the Gaussian distribution over lattices is denoted by

$$\forall \boldsymbol{x}\in L,\ D_{L,\sigma}(\boldsymbol{x})=\rho_\sigma(\boldsymbol{x})/\rho_\sigma(L). \tag{3}$$

Definition 1 (LWE) (Regev, 2009) On input of a prime q, an integer n, and a discrete Gaussian distribution denoted as $\psi_\sigma=D_{L,\sigma}$, a $(\mathbb{Z}_q;\ n;\ \psi_\sigma)$-LWE problem comes from an unspecified challenge oracle $\mathcal{O}$, being either $\mathcal{O}_S$ or $\mathcal{O}_\$$:

$\mathcal{O}_S$ outputs samples with $(\boldsymbol{A},\ \boldsymbol{A}^{\mathrm{T}}\boldsymbol{s}+\boldsymbol{e})\in \mathbb{Z}_q^{n\times m}\times\mathbb{Z}_q^m$, where matrix $\boldsymbol{A}\in\mathbb{Z}_q^{n\times m}$ is uniformly distributed, noise $\boldsymbol{e}\in\mathbb{Z}_q^m$ is selected according to ψ_σ^m, and $\boldsymbol{s}\in\mathbb{Z}_q^n$ is from a uniform distribution.

$\mathcal{O}_\$$ outputs true uniform samples from $U\left(\mathbb{Z}_q^{n\times m}\times\mathbb{Z}_q^m\right)$ at random.

The decisional $(\mathbb{Z}_q;\ n;\ \psi_\sigma)$-LWE problem refers to an algorithm $\mathcal{A}$ that can decide its distribution from $\mathcal{O}_S$ or $\mathcal{O}_\$$ after repeated queries to the challenge oracle $\mathcal{O}$, where $\boldsymbol{s}$ is chosen from the discrete Gaussian distribution or the uniform distribution (Lindner and Peikert, 2011).

The computational $(\mathbb{Z}_q;\ n;\ \psi_\sigma)$-LWE problem is to recover $\boldsymbol{s}\in\mathbb{Z}_q^n$ precisely given some samples with $(\boldsymbol{A},\boldsymbol{A}^{\mathrm{T}}\boldsymbol{s}+\boldsymbol{e})\in\mathbb{Z}_q^{n\times m}\times\mathbb{Z}_q^m$ from $\mathcal{O}_S$.

Regev (2009) proved that the LWE problem with certain choices of q and σ could be reduced to some classical worst-case lattice problems (Gap-SVP and SIVP).

The short integer solution (SIS) problem, seen as duals of LWE, has served as the foundation for some "minicrypt" primitives, such as one-way and collision-resistant hash functions. Our homomorphic signatures used for robustness are based on the SIS problem.

Definition 2 (SIS) For any $n\in\mathbb{Z}$ and any functions $m=m(n)$, $q=q(n)$, and $\beta=\beta(n)$, the SIS problem $\mathrm{SIS}_{q,m,\beta}$ is as follows: Given an integer q, a matrix $\boldsymbol{A}\in\mathbb{Z}_q^{n\times m}$ chosen uniformly at random, and a real number β, the target is to find a non-zero integer vector $\boldsymbol{z}\in\mathbb{Z}^m\setminus\{\boldsymbol{0}\}$ satisfying $\boldsymbol{A}\boldsymbol{z}=\boldsymbol{0}$ mod q and $\|\boldsymbol{z}\|\leq\beta$.

Micciancio and Regev (2007) showed that solving the average-case $\mathrm{SIS}_{q,m,\beta}$ for certain parameters is equivalent to solving worst-case instances on hard lattice problems.

2.2 Related functions and tools

1. Functions of Bits and Power2

Let $\boldsymbol{u}\in\mathbb{Z}_q^n$ and $l=\lceil\log_2 q\rceil$. There are bit vectors $\boldsymbol{u}_i\in\{0,1\}^n$ such that $\boldsymbol{u}=\sum_{i=0}^{l-1}\left(2^i\boldsymbol{u}_i\right)$ and

$$\mathrm{Bits}(\boldsymbol{u})=[\boldsymbol{u}_0\mid\boldsymbol{u}_1\mid\cdots\mid\boldsymbol{u}_{l-1}]\in\{0,1\}^{1\times(nl)}. \tag{4}$$

Let $\boldsymbol{X}=[\boldsymbol{x}_1\mid\boldsymbol{x}_2\mid\cdots\mid\boldsymbol{x}_m]\in\mathbb{Z}_q^{n\times m}$, where $\boldsymbol{x}_i$ are column vectors. Then

$$\mathrm{Power2}(\boldsymbol{X})=\begin{bmatrix} \boldsymbol{x}_1 & \boldsymbol{x}_2 & \cdots & \boldsymbol{x}_m \\ 2\boldsymbol{x}_1 & 2\boldsymbol{x}_2 & \cdots & 2\boldsymbol{x}_m \\ \vdots & \vdots & & \vdots \\ 2^{l-1}\boldsymbol{x}_1 & 2^{l-1}\boldsymbol{x}_2 & \cdots & 2^{l-1}\boldsymbol{x}_m \end{bmatrix}\in\mathbb{Z}_q^{(nl)\times m}. \tag{5}$$

It is easy to check that $\mathrm{Bits}(\boldsymbol{u})\cdot\mathrm{Power2}(\boldsymbol{X})=\boldsymbol{u}\boldsymbol{X}\in\mathbb{Z}_q^{1\times m}$.

2. Trapdoor generation algorithm and its properties

Theorem 1 (Trapdoor generation) (Agrawal et al., 2010) Let $q\geq 3$ be odd, $n\in\mathbb{Z}$, and $m=\lceil 6n\log_2 q\rceil$. The probabilistic algorithm $\mathrm{TrapGen}(q,n)$ produces a pair $\left(\boldsymbol{A}\in\mathbb{Z}_q^{n\times m},\ \boldsymbol{T}_{\boldsymbol{A}}\in\mathbb{Z}_q^{m\times m}\right)$ in polynomial time such that $\boldsymbol{A}$ is a matrix whose distribution is within $\mathrm{negl}(n)$ statistical distance and that $\boldsymbol{T}_{\boldsymbol{A}}$ is a short basis for lattice $\Lambda_q^{\perp}(\boldsymbol{A})$ satisfying $\left\|\tilde{\boldsymbol{T}}_{\boldsymbol{A}}\right\|\leq O(\sqrt{n\log_2 q})$ and $\left\|\boldsymbol{T}_{\boldsymbol{A}}\right\|\leq O(n\log_2 q)$.

The following lemma includes some standard properties of discrete Gaussian distributions and powerful tools designed for (H)IBE constructions:

Lemma 1 (Agrawal et al., 2010) Let $q\geq 2$ and $\boldsymbol{A}$ be a matrix in $\mathbb{Z}_q^{n\times m}$ with $m>n$. Let $\boldsymbol{T}_A$ be a basis for $\Lambda_q^{\perp}(\boldsymbol{A})$ and $\sigma \geq \|\tilde{\boldsymbol{T}}_A\|\omega\left(\sqrt{\log_2 m}\right)$. Then for $\boldsymbol{u}\in\mathbb{Z}_q^n$, we have

(1) $\Pr\left[\boldsymbol{x}\leftarrow D_{\Lambda_q^{\boldsymbol{u}}(\boldsymbol{A}),\sigma}:\|\boldsymbol{x}\|>\sqrt{m}\sigma\right]\leq \mathrm{negl}(n)$.

(2) There is a probabilistic polynomial time (PPT) algorithm SamplePre($\boldsymbol{A}$, $\boldsymbol{T}_A$, $\boldsymbol{u}$, σ), which returns $\boldsymbol{x}\in\Lambda_q^{\boldsymbol{u}}(\boldsymbol{A})$ sampled from a distribution statistically close to $D_{\Lambda_q^{\boldsymbol{u}}(\boldsymbol{A}),\sigma}$, no matter whether $\Lambda_q^{\boldsymbol{u}}(\boldsymbol{A})$ is empty or not.

(3) Let $\boldsymbol{A}\in\mathbb{Z}_q^{n\times m}$ be of rank n, $\boldsymbol{B}\in\mathbb{Z}_q^{n\times m}$, $\boldsymbol{R}\in\{-1,\ 1\}^{m\times m}$, and $\sigma\geq\|\tilde{\boldsymbol{T}}_A\|\omega\left(\sqrt{\log_2 m}\right)$. Set $\boldsymbol{F}=(\boldsymbol{A}\,|\,\boldsymbol{AR}+\boldsymbol{B})\in\mathbb{Z}_q^{n\times(2m)}$. There is a PPT algorithm SampleLeft($\boldsymbol{A}$, $\boldsymbol{AR}$+$\boldsymbol{B}$, $\boldsymbol{T}_A$, $\boldsymbol{u}$, σ), which takes a trapdoor $\boldsymbol{T}_A$ for $\Lambda_q^{\perp}(\boldsymbol{A})$ and outputs a short vector $\boldsymbol{e}\in D_{\Lambda_q^{\boldsymbol{u}}(\boldsymbol{F}),\sigma}^{2m}$.

(4) Let $\boldsymbol{B}\in\mathbb{Z}_q^{n\times m}$ be of rank n, $\boldsymbol{A}\in\mathbb{Z}_q^{n\times m}$, $\boldsymbol{R}\in\{-1,\ 1\}^{m\times m}$, and $\sigma\geq\|\tilde{\boldsymbol{T}}_B\|\cdot\|\boldsymbol{R}\|\omega\left(\sqrt{\log_2 m}\right)$. Set $\boldsymbol{F}=(\boldsymbol{A}\,|\,\boldsymbol{AR}+\boldsymbol{B})\in\mathbb{Z}_q^{n\times(2m)}$. There is a PPT algorithm SampleRight($\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{R}$, $\boldsymbol{T}_B$, $\boldsymbol{u}$, σ), which takes a trapdoor $\boldsymbol{T}_B$ for $\Lambda_q^{\perp}(\boldsymbol{B})$ and outputs a short vector $\boldsymbol{e}\in D_{\Lambda_q^{\boldsymbol{u}}(\boldsymbol{F}),\sigma}^{2m}$.

(5) There is an encoding with the full-rank difference (FRD) function $H:\mathbb{Z}_q^n\rightarrow\mathbb{Z}_q^{n\times n}$ that satisfies the following: (a) For distinct $\boldsymbol{u},\boldsymbol{v}\in\mathbb{Z}_q^n$, matrix $H(\boldsymbol{u})-H(\boldsymbol{v})\in\mathbb{Z}_q^{n\times n}$ is of full rank; (b) H is computable in polynomial time (in $n\log_2 q$).

Next, we describe how a private key generator (PKG) may extend a low-dimensional lattice into an arbitrary higher-dimensional one, without any loss of quality.

(6) Let $\boldsymbol{A}\in\mathbb{Z}_q^{n\times m}$ be of rank n, $\boldsymbol{B}\in\mathbb{Z}_q^{n\times m}$, $\boldsymbol{R}\in\{-1,\ 1\}^{m\times m}$, and $\sigma\geq\|\tilde{\boldsymbol{T}}_A\|\omega\left(\sqrt{\log_2 m}\right)$. Set $\boldsymbol{F}=(\boldsymbol{A}\,|\,\boldsymbol{AR}+\boldsymbol{B})\in\mathbb{Z}_q^{n\times(2m)}$. There is a PPT algorithm SampleBasisLeft($\boldsymbol{A}$, $\boldsymbol{AR}$+$\boldsymbol{B}$, $\boldsymbol{T}_A$, σ), which takes a trapdoor $\boldsymbol{T}_A$ for $\Lambda_q^{\perp}(\boldsymbol{A})$ and outputs a short basis $\boldsymbol{T}_F\in\Lambda_q^{\perp}(\boldsymbol{F})$ statistically distributed to $\psi_{\sigma}^{(2m)\times(2m)}$.

(7) Let $\boldsymbol{B}\in\mathbb{Z}_q^{n\times m}$ be of rank n, $\boldsymbol{A}\in\mathbb{Z}_q^{n\times m}$, $\boldsymbol{R}\in\{-1,\ 1\}^{m\times m}$, and $\sigma\geq\|\tilde{\boldsymbol{T}}_B\|\cdot\|\boldsymbol{R}\|\omega\left(\sqrt{\log_2 m}\right)$. Set $\boldsymbol{F}=(\boldsymbol{A}\,|\,\boldsymbol{AR}+\boldsymbol{B})\in\mathbb{Z}_q^{n\times(2m)}$. There is a PPT algorithm SampleBasisRight($\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{R}$, $\boldsymbol{T}_B$, σ), which takes a trapdoor $\boldsymbol{T}_B$ for $\Lambda_q^{\perp}(\boldsymbol{B})$ and outputs a short basis $\boldsymbol{T}_F\in\Lambda_q^{\perp}(\boldsymbol{F})$ statistically distributed to $\psi_{\sigma}^{(2m)\times(2m)}$.

2.3 IB-TPRE syntax and security

2.3.1 IB-TPRE definition and security model

Definition 3 (IB-TPRE) An IB-TPRE scheme includes eight algorithms, which can be classified into three categories according to their functionality. In terms of the delegation relationship, the parties involved in IB-TPRE include a PKG, a delegator, a delegatee, and N proxies. If k ($k\leq N$) or more proxies attend, then the functionality of PRE can be realized.

1. Key generation algorithms

(1) Setup(λ). On input of a security parameter λ, PKG runs and outputs the public parameter PP and the master key MSK.

(2) KeyGen(MSK, id). On input of the master key MSK and a user id, PKG outputs the private key $\mathrm{SK}_{\mathrm{id}}$.

(3) ReKeyGen(id_1, id_2, $\mathrm{SK}_{\mathrm{id}_1}$, N, k). On input of user id_1's private key, the total number of key shares N, and threshold k, the delegator id_1 outputs N re-encryption key shares $\{\mathrm{kFrag}_i\}$ ($1\leq i\leq N$) and the verification key $\mathrm{VK}=\{\mathrm{vk}_1, \mathrm{vk}_2, \ldots, \mathrm{vk}_N\}$. In general, the ReKeyGen algorithm can directly generate shares, or temporally generate a whole re-encryption key $\mathrm{RK}_{\mathrm{id}_1\rightarrow\mathrm{id}_2}$ and then divide it into N shares.

2. Encryption and decryption algorithms

(1) Enc(id, M). On input of the user id and a message M to be encrypted, a sender (anyone) outputs a ciphertext C.

(2) Dec(C, $\mathrm{SK_{id}}$). On input of the private key $\mathrm{SK_{id}}$ and a ciphertext C, the delegator or delegatee outputs the decrypted message M or $\perp$.

3. Ciphertext share-processing algorithms

(1) PreEnc(C_1, $\{\mathrm{kFrag}_i\}$). On input of an original ciphertext of user id_1 and re-encryption key shares $\{\mathrm{kFrag}_i\}$, a proxy outputs ciphertext shares $\{\mathrm{cFrag}_i\}$ corresponding to a user id_2. In truth, it requires only at least k ($k\le N$) proxies to perform this algorithm.

(2) Verify($\{\mathrm{vk}_i\}$, $\{\mathrm{cFrag}_i\}$). On input of a user's ciphertext share $\{\mathrm{cFrag}_i\}$ and verification key $\{\mathrm{vk}_i\}$, the delegatee outputs 1 if the ciphertext share is legal; otherwise, it outputs 0.

(3) Comb(C_1, $\{\mathrm{cFrag}_i\}_{i\in S}$). The input contains a ciphertext share set S of size k'. If $k'\ge k$, the delegatee outputs a complete ciphertext C_2 that can be decrypted by id_2; otherwise, it outputs $\perp$.

The correctness of an IB-TPRE scheme includes two requirements:

The first one is the correctness of decryption. $\forall M\in\{0, 1\}$, Setup(λ)$\rightarrow$(PP, MSK), KeyGen(MSK, id_1)$\rightarrow$ $\mathrm{SK}_{\mathrm{id}_1}$, KeyGen(MSK, id_2)$\rightarrow$ $\mathrm{SK}_{\mathrm{id}_2}$, ReKeyGen($\mathrm{id}_1$, id_2, $\mathrm{SK}_{\mathrm{id}_1}$, N, k)$\rightarrow\{\mathrm{kFrag}_i\}$ ($1\le i\le N$), Enc(id_1, M)$\rightarrow C_1$, and PreEnc(C_1, $\{\mathrm{kFrag}_i\}$)$\rightarrow$ $\{\mathrm{cFrag}_i\}$, the following expressions hold: (1) Dec(C_1, $\mathrm{SK}_{\mathrm{id}_1}$)$\rightarrow M$; (2) Comb($C_1$, $\{\mathrm{cFrag}_i\}_{i\in S}$)$\rightarrow C_2$ and Dec(C_2, $\mathrm{SK}_{\mathrm{id}_2}$)$\rightarrow M$, when k shares are collected.

The second requirement is the correctness of ciphertext share verification. For all $\{\mathrm{kFrag}_i\}$ ($1\le i\le N$), all ciphertext shares $\{\mathrm{cFrag}_i\}$ ($1\le i\le N$) obtained by PreEnc(C_1,$\{\mathrm{kFrag}_i\}$) satisfy Verify($\{\mathrm{vk}_i\}$, $\{\mathrm{cFrag}_i\}$)$\rightarrow$1.

IB-TPRE's selective security, called IND-sID-CPA, is described with a game played between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$:

1. Init

Adversary $\mathcal{A}$ outputs a target identity id^* intended to be attacked.

2. Setup

Challenger $\mathcal{C}$ performs the Setup algorithm to obtain PP and MSK, and then sends PP to adversary $\mathcal{A}$. Γ_{C} is a set of corrupted users, for which the adversary has made a private key query, and Γ_{H} is a set of honest users, for which the adversary has never made a private key query. Γ_{C} is empty at the beginning and Γ_{H} has only one element id^*.

3. Query phase 1

Adversary $\mathcal{A}$ chooses some queries in the following three ways:

(1) KeyGen query

When the adversary issues a private key query on a user id ($\mathrm{id}\notin\Gamma_{\mathrm{H}}$), challenger $\mathcal{C}$ runs the algorithm KeyGen(MSK, id), returns the private key to adversary $\mathcal{A}$, and finally adds id to set Γ_{C}.

(2) ReKeyGen query

When adversary $\mathcal{A}$ issues re-encryption key share queries between users id_1 and id_2 if $\mathrm{id}_1\in\Gamma_{\mathrm{H}}$, challenger $\mathcal{C}$ temporally generates a whole $\mathrm{RK}_{\mathrm{id}_1\rightarrow\mathrm{id}_2}$ (this is similar to the re-encryption key from id_1 to id_2 in the IB-PRE system, and all the N re-encryption key shares can be derived from it), and then divides it into N shares. Finally, challenger $\mathcal{C}$ returns all N shares to adversary $\mathcal{A}$ if $\mathrm{id}_2\in\Gamma_{\mathrm{H}}$; otherwise, if $\mathrm{id}_2\in\Gamma_{\mathrm{C}}$, it returns only the previous $k-1$ shares and keeps the key shares labeled k, $k+1$, …, N confidential. $\mathcal{A}$ is not allowed to obtain more than k re-encryption key shares when $\mathrm{id}_1\in\Gamma_{\mathrm{H}}$ and $\mathrm{id}_2\in\Gamma_{\mathrm{C}}$, which could allow $\mathcal{A}$ to decrypt the challenge ciphertext by a sequence of re-encryptions. Note that when $\mathrm{id}_1\in\Gamma_{\mathrm{C}}$, adversary $\mathcal{A}$ can generate all re-encryption key shares with $\mathrm{SK}_{\mathrm{id}_1}$ in a non-interactive setting, so we ignore the case for simplicity. In addition, when $\mathrm{id}_1\notin(\Gamma_{\mathrm{H}}\cup\Gamma_{\mathrm{C}})$ is queried, there are two cases. First, if $\mathrm{id}_2\in\Gamma_{\mathrm{C}}$, challenger $\mathcal{C}$ adds id_1 to Γ_{H}, and then returns the result as a ReKeyGen query for $\mathrm{id}_1\in\Gamma_{\mathrm{H}}$ and $\mathrm{id}_2\in\Gamma_{\mathrm{C}}$ as mentioned above. Second, if $\mathrm{id}_2\notin\Gamma_{\mathrm{C}}$ is queried, challenger $\mathcal{C}$ adds id_2 to Γ_{H} (if $\mathrm{id}_2\in\Gamma_{\mathrm{H}}$, ignore this step), and then returns the result as a ReKeyGen query for $\mathrm{id}_1\in\Gamma_{\mathrm{H}}$ and $\mathrm{id}_2\in\Gamma_{\mathrm{H}}$.

(3) PreEnc query

This query works only when $\mathrm{id}_1\in\Gamma_{\mathrm{H}}$ and $\mathrm{id}_2\in\Gamma_{\mathrm{H}}$, which is the integration of some algorithms in sequence: (a) asking our KeyGen(id_1) without returning the private key to id_1; (b) computing ReKeyGen(id_1, id_2, $\mathrm{SK}_{\mathrm{id}_1}$, N, k) and retaining only the first $k-1$ key shares $\{\mathrm{kFrag}_i\}$ ($1\le i\le k-1$) corresponding to the first $k-1$ proxy servers; (c) computing PreEnc(C_1, $\{\mathrm{kFrag}_i\}$)$\rightarrow\{\mathrm{cFrag}_i\}$ ($1\le i\le k-1$).

The adversary can repeat this polynomial for different queries.

4. Challenge phase

Make sure that identity id^* belongs to set Γ_{H}, and that query KeyGen(MSK, id^*) is not permitted. Challenger $\mathcal{C}$ picks a random bit $b\in\{0, 1\}$ and an arbitrary C from the ciphertext space. If b=1, set the challenge ciphertext to C^*=Encrypt(id^*, M). If b=0, set the challenge ciphertext to C^*=C.

5. Query phase 2

Adversary $\mathcal{A}$ could ask extra queries for the private key, re-encryption key, re-encryption on the identity id ($\mathrm{id}\neq\mathrm{id}^*$), and challenger responses, same as in query phase 1.

6. Guess

$\mathcal{A}$ makes a guess $b'\in\{0, 1\}$ and wins the game if b=b'.

The advantage of adversary $\mathcal{A}$ is defined as

$$\mathrm{Adv}_{\mathcal{A}}^{\text{IND-sID-CPA}}(\mathrm{id}^*)=\left|\Pr[b=b']-\frac{1}{2}\right|.$$

Definition 4 (IND-sID-CPA security) An IB-PRE scheme is IND-sID-CPA secure if $\mathrm{Adv}_{\mathcal{A}}^{\text{IND-sID-CPA}}(\mathrm{id}^*)$ is negligible for any adversary $\mathcal{A}$ in polynomial time.

Some comments are given as follows:

1. A strong definition allows an adversary to obtain a re-encryption key from an uncorrupted identity to a corrupted identity except for one from the challenged identity to a compromised identity. In this work, we do not pursue such a definition.

2. The adversary does not explicitly handle the definition of the re-encryption query for $\mathrm{id}_1\in\Gamma_{\mathrm{C}}$ and $\mathrm{id}_2\in(\Gamma_{\mathrm{H}}\cup\Gamma_{\mathrm{C}})$, because the re-encryption query performed by the adversary is the same as the re-encryption key query when $\mathrm{id}_1\in\Gamma_{\mathrm{C}}$. This omission is not a loss of generality.

2.3.2 Robustness of IB-TPRE

Robustness is an important property for the IB-TPRE scheme. The basic idea is that an adversary cannot forge a ciphertext share that is not obtained by the re-encryption key share honestly, but can be verified correctly. The robustness of the IB-TPRE scheme is described with the following game $\mathrm{Expt}_{\mathcal{A}}^{\mathrm{Rb}}$:

The interaction processes of Init, KeyGen queries, ReKeyGen queries, and PreEnc queries are the same as that described in Definition 4, but at the guessing stage, the adversary randomly selects $\mathrm{id}^*\in\Gamma_{\mathrm{H}}$, outputs a corresponding ciphertext share $\mathrm{cFrag}_{j^*,\mathrm{id}^*}$ ($1\leq j^*\leq N$), and satisfies

(1) $\left\{\mathrm{cFrag}_{j^*,\mathrm{id}^*}\right\}\in\mathrm{Pre\,Enc}\left(C_1,\left\{\mathrm{kFrag}_{j^*,\mathrm{id}^*}\right\}\right)$;

(2) $\mathrm{Verify}\left(\left\{vk_{j^*}\right\},\left\{\mathrm{cFrag}_{j^*,\mathrm{id}^*}\right\}\right)=1$.

Then the adversary's advantage is defined as $\mathrm{Adv}_{\mathcal{A}}^{\mathrm{Rb}}=\Pr\left[\mathrm{Expt}_{\mathcal{A}}^{\mathrm{Rb}}(\lambda)=1\right]$.

Definition 5 (Robustness) (Boneh et al., 2017) If the advantage $\mathrm{Adv}_{\mathcal{A}}^{\mathrm{Rb}}$ is negligible, the IB-TPRE scheme is robust.

2.3.3 IB-TPRE collusion resistance

Yin et al. (2018) introduced two definitions of collusion resistance, called strong anti-collusion and weak anti-collusion, of PRE. The difference between these two is whether the exact private key of the delegator is obtained when the proxy with a re-encryption key and the delegatee with its private key collude.

Our proposed scheme satisfies weak anti-collusion as defined in Definition 6:

Definition 6 (Weak anti-collusion) In a threshold PRE scheme with weak anti-collusion, if some proxy servers and the delegatee collude, they cannot obtain the exact private key of the delegator, but could obtain an approximate value of the delegator's private key.

2.4 Homomorphic signature and threshold secret sharing

2.4.1 Homomorphic signature

Our IB-TPRE provides robustness using an SIS-based homomorphic signature, which refers to a series of same operations performed on the original message and its corresponding signature without knowing the signing key. As a result, anyone with a verification key can verify the computation results in public. It is widely applied in outsourcing computing, retrievable proof, public verifiability, and other fields.

Definition 7 (Homomorphic signature) A homomorphic signature (HS) scheme (Boneh et al., 2017) usually includes four algorithms:

(1) HS.KeyGen(λ, d, N)

When inputting a security parameter λ, a circuit depth bound d, and a message set bound N, it outputs a signing key hssk and a verification key hsvk.

(2) HS.Sign(hssk, M)

When the signing key hssk and the message M to be signed are inputted, it outputs a signature σ.

(3) HS.SignEval(C, σ)

When inputting an evaluation circuit C: $\{0, 1\}^N \to \{0, 1\}$ and a signature σ, it outputs a homomorphically computed signature σ^*.

(4) HS.Verify(hsvk, y, C, σ^*)

On input of a verification key hsvk, a circuit C, an output value y, and a signature σ^*, the verification algorithm accepts the signature (and outputs 1) or rejects it (and outputs 0).

The correctness of HS is defined as follows: For any input λ, $d\in\mathbb{Z}$, HS.KeyGen(λ, d, N)$\to$(hsvk, hssk), $M\in\{0, 1\}^N$, HS.Sign(hssk, M)$\to\sigma$, and any circuit C: $\{0, 1\}^N\to\{0, 1\}$ with depth d and $C(M)\to y$, the following equation holds:

$$\Pr\left[\text{HS.Verify}\left(\text{hsvk}, y, C, \text{HS.SignEval}(C,\sigma)\right)=1\right]=1.$$

Two security properties for a homomorphic signature are required (Boneh et al., 2017). The first property is unforgeability; that is, given homomorphically signed data σ, the adversary cannot produce a circuit C or a valid signature σ_y, for which $C(\sigma)=\sigma_y$. The other property is context hiding; that is, given a homomorphically computed signature σ^*, the adversary does not obtain any useful information about the original message M that was signed, other than what was already implied by the output $C(M)=y$.

At present, the security of the HS scheme (Boneh and Freeman, 2011; Gorbunov et al., 2015) can be reduced to the SIS assumption. These two schemes are sufficient to completely meet the design requirements for our robustness.

2.4.2 Shamir's threshold secret sharing

Shamir's threshold secret sharing scheme includes secret segmentation and secret reconstruction.

1. Secret segmentation

To share secret S among N participants by Shamir's (k, N) threshold scheme where $k\le N$, it works as follows:

(1) Randomly choose $k-1$ coefficients a_1, a_2, …, $a_{k-1}\in\mathbb{Z}_q$.

(2) Construct a polynomial with degree $k-1$:

$$f(x)=S+a_1x+\cdots+a_{k-1}x^{k-1}.$$

(3) For each participant labeled i ($1\le i\le N$), calculate $\overline{S}_i=f(i)$ as its share and send $\overline{S}_i$ to the i^{th} participant.

2. Secret reconstruction

When any k in N participants are prepared to recover secret (without loss of generality, suppose that the valid shares are $\overline{S}_1,\overline{S}_2,\cdots,\overline{S}_k$), the polynomial $f(x)$ can be reconstructed as

$$f(x)=\sum_{j=1}^{k}\lambda_j\overline{S}_j,$$

where λ_j denotes Lagrange coefficients:

$$\lambda_j=\prod_{i=1,i\ne j}^{k}\frac{x-i}{j-i}\bmod q.$$

So, $S=f(0)$, and the secret S is restored.

3 Identity-based threshold proxy re-encryption scheme from lattices

Set security parameters $n\in\mathbb{Z}_q$, $m\ge\lceil 6n\log_2 q\rceil$, a prime $q\ge m^2\omega(\log_2 n)$, and $l=\lceil\log_2 q\rceil$. Define a uniform distribution U_q on $\mathbb{Z}_q$ and a discrete Gaussian distribution ψ_i on $\mathbb{Z}_q$ with noise parameter $i\in\{\gamma, e\}$. Choose a pseudo-random function $F_{\text{prfk}}:\mathbb{Z}_q^{2m+1}\to[-r, r]^{2m+1}$ for a bounded integer r. Let the number of the total proxy servers be N and the threshold be k; then define $\eta=(N!)^2$. Identities are assumed by the elements in $\mathbb{Z}_q^n$, and the function H refers to the FRD map $H:\mathbb{Z}_q^n\to\mathbb{Z}_q^{n\times n}$. In addition, to be more universal, denote Π_{HS}=(HS.KeyGen, HS.Sign, HS.SignEval,

HS.Verify) as an abstract representation of a homomorphic signature. The re-encryption key shares for proxy servers are generated and distributed by individual users without the involvement of PKG.

3.1 Key generation algorithms

1. Setup(1^λ)

Generate a uniformly distributed matrix $\boldsymbol{A}_0 \in \mathbb{Z}_q^{n\times m}$ with its trapdoor $\boldsymbol{T}_{\boldsymbol{A}_0} \in \mathbb{Z}_q^{m\times m}$ by algorithm TrapGen. Choose two uniformly distributed matrices $\boldsymbol{A}_1 \in \mathbb{Z}_q^{n\times m}$ and $\boldsymbol{B} \in \mathbb{Z}_q^{n\times m}$ and a vector $\boldsymbol{u} \in \mathbb{Z}_q^n$. The user dictionary UD and delegation dictionary DD are initialized by empty sets. Publish the public key PP=$\{\boldsymbol{A}_0, \boldsymbol{A}_1, \boldsymbol{B}, \boldsymbol{u}\}$ and keep the master private key $\mathrm{MSK} = \{\boldsymbol{T}_{\boldsymbol{A}_0}\} \in \mathbb{Z}_q^{m\times m}$.

2. KeyGen(MSK, id)

When a user id submits a request to PKG for a private key, check the user directory UD first. If there is already a private key record for the user id, retrieve and return it to the user. Otherwise, we perform the following:

(1) Convert the identity id into $H(\mathrm{id}) \in \mathbb{Z}_q^{n\times n}$, obtain the corresponding public key $\boldsymbol{F}_{\mathrm{id}}=(\boldsymbol{A}_0|\boldsymbol{A}_1+H(\mathrm{id})\boldsymbol{B})$, and calculate the private key $\boldsymbol{e}_{\mathrm{id}} \in \psi_\gamma^{2m}$ by SampleLeft($\boldsymbol{A}_0$, $\boldsymbol{A}_1+H(\mathrm{id})\boldsymbol{B}$, $\boldsymbol{T}_{\boldsymbol{A}_0}$, $\boldsymbol{u}$, ψ_γ) satisfying $\boldsymbol{F}_{\mathrm{id}}\boldsymbol{e}_{\mathrm{id}}=\boldsymbol{u}$.

(2) Obtain $\boldsymbol{T}_{\mathrm{id}}$=SampleBasisLeft($\boldsymbol{A}_0, \boldsymbol{A}_1+H(\mathrm{id})\boldsymbol{B}$, $\boldsymbol{T}_{\boldsymbol{A}_0}, s_\mathrm{t}$) satisfying $\boldsymbol{F}_{\mathrm{id}}\boldsymbol{T}_{\mathrm{id}} = \boldsymbol{0}$, where s_t is the Gaussian parameter used to generate the user trapdoor.

Return private key $\mathrm{SK}_{\mathrm{id}}=(\boldsymbol{e}_{\mathrm{id}}, \boldsymbol{T}_{\mathrm{id}})$ to id and add it to UD.

3. ReKeyGen(id_1, id_2, $\mathrm{SK}_{\mathrm{id}_1}$, N, k)

On generating the re-encryption between id_1 and id_2, check the delegation directory DD first. If re-encryption key shares are already contained in the directory, distribute these shares to the corresponding proxy server and then exit. Otherwise, compute all the re-encryption key shares $\{\mathrm{kFrag}_i\}$ ($1\le i\le N$) as follows:

(1) Run Shamir's secret sharing to split $\boldsymbol{u} \in \mathbb{Z}_q^n$ to $\bar{\boldsymbol{u}}_i \in \mathbb{Z}_q^n$ ($1\le i\le N$) as feature vectors for each proxy server indexed by i, and store them locally (this process needs to be performed only once). For $i\in\{1, 2, \ldots, N\}$, let the feature vector of the i^{th} proxy server be $\bar{\boldsymbol{u}}_i \in \mathbb{Z}_q^n$, and $\bar{\boldsymbol{e}}_{i,\mathrm{id}_1} \leftarrow \mathrm{SamplePre}\left(\boldsymbol{F}_{\mathrm{id}_1}, \boldsymbol{T}_{\mathrm{id}_1}, \bar{\boldsymbol{u}}_i, \psi_\gamma\right)$ satisfy $\boldsymbol{F}_{\mathrm{id}_1}\bar{\boldsymbol{e}}_{i,\mathrm{id}_1} = \bar{\boldsymbol{u}}_i$.

(2) Choose a matrix $\boldsymbol{R} \in \psi_e^{(2ml)\times n}$, a vector $\boldsymbol{e} \in \psi_e^{2ml}$, and a matrix $\boldsymbol{E} \in \psi_e^{(2ml)\times(2m)}$. Then compute $\boldsymbol{P} = \boldsymbol{R}\boldsymbol{F}_{\mathrm{id}_2} + \boldsymbol{E} \in \mathbb{Z}_q^{(2ml)\times(2m)}$ and $\boldsymbol{Q} = \boldsymbol{R}\boldsymbol{u} + \boldsymbol{e} \in \mathbb{Z}_q^{2ml}$, where $\boldsymbol{F}_{\mathrm{id}_2} = \left(\boldsymbol{A}_0 \mid \boldsymbol{A}_1 + H(\mathrm{id}_2)\boldsymbol{B}\right)$.

(3) Run Shamir's secret sharing on each coefficient from ($\boldsymbol{P}$, $\boldsymbol{Q}$) by the following:

For $i\in\{1, 2, \ldots, 2ml\}$ and $j\in\{1, 2, \ldots, 2m\}$, select a random polynomial $y_{i,j}(x) \in \mathbb{Z}_q[x]$ satisfying $y_{i,j}(0)=P_{i,j}$ with its degree equal to k–1. For $i\in\{1, 2, \ldots, 2ml\}$, choose a polynomial $w_i(x) \in \mathbb{Z}_q[x]$ satisfying $w_i(0)=Q_i$ with its degree equal to k–1.

For $1\le j\le N$, the partial decryption share for the j^{th} proxy server is

$$(\bar{\boldsymbol{P}}_j, \bar{\boldsymbol{Q}}_j) = \begin{bmatrix} y_{1,1}(j) & \cdots & y_{1,2m}(j) & w_1(j) \\ y_{2,1}(j) & \cdots & y_{2,2m}(j) & w_2(j) \\ \vdots & & \vdots & \vdots \\ y_{2ml,1}(j) & \cdots & y_{2ml,2m}(j) & w_{2ml}(j) \end{bmatrix} \in \mathbb{Z}_q^{(2ml)\times(2m+1)}.$$

(4) The re-encryption key share for the j^{th} proxy server is $\overline{\mathbf{RK}}_j = \left(\bar{\boldsymbol{P}}_j, \bar{\boldsymbol{Q}}_j - \mathrm{Power2}\left(\bar{\boldsymbol{e}}_{j,\mathrm{id}_1}\right)\right)$ ($1\le j\le N$).

(5) Generate verification and signing keys (hsvk, hssk) by HS.KeyGen. Select N keys prfk_1, prfk_2, …, prfk_N independently by the pseudo-random function F_{prfk}. For $i\in\{1, 2, \ldots, N\}$, set $\boldsymbol{X}_i = (\overline{\mathbf{RK}}_i, \mathrm{prfk}_i)$ and compute $\bar{\sigma}_i = \mathrm{HS.Sign}(\mathrm{hssk}, \boldsymbol{X}_i)$.

Publish hsvk to verify the signature in the subsequent process. Send shares $\{\mathrm{kFrag}_i\} = \left\{\overline{\mathbf{RK}}_i, \mathrm{prfk}_i, \bar{\sigma}_i\right\}$ ($1\le i\le N$) to the proxy server with index i over a secure channel, and add them to DD.

3.2 Encryption and decryption algorithms

1. Encrypt(id, M)

To encrypt a bit $M\in\{0, 1\}$, it works as follows:

(1) On input of the user $\mathrm{id} \in \mathbb{Z}_q^n$, encode it as

$\boldsymbol{F}_{\mathrm{id}}=\left(\boldsymbol{A}_0 \mid \boldsymbol{A}_1+H(\mathrm{id})\boldsymbol{B}\right)\in\mathbb{Z}_q^{n\times(2m)}$.

(2) Choose a uniform vector $\boldsymbol{s}\in\mathbb{Z}_q^{n\times 1}$ randomly.

(3) Choose a uniform integer matrix $\boldsymbol{R}_{\mathrm{id}}\in\{-1, 1\}^{m\times m}$ randomly.

(4) Choose $x\in\psi_e$ and $\boldsymbol{y}\in\psi_e^{1\times m}$, and then calculate $\boldsymbol{z}=\boldsymbol{y}\boldsymbol{R}_{\mathrm{id}}\in\psi_e^{1\times m}$.

(5) Calculate $\boldsymbol{c}_1=\boldsymbol{s}^{\mathrm{T}}\boldsymbol{F}_{\mathrm{id}}+\eta(\boldsymbol{y}\mid\boldsymbol{z})\in\mathbb{Z}_q^{1\times(2m)}$ and $c_2=\boldsymbol{s}^{\mathrm{T}}\boldsymbol{u}+\eta x+M\lfloor q/2\rfloor\in\mathbb{Z}_q$. Output ciphertext $\boldsymbol{C}=(\boldsymbol{c}_1,c_2)\in\mathbb{Z}_q^{2m+1}$.

2. Dec($\boldsymbol{C}_{\mathrm{id}}$, $\boldsymbol{e}_{\mathrm{id}}$)

On input of a ciphertext $\boldsymbol{C}_{\mathrm{id}}=(\boldsymbol{c}_1, c_2)$ and the corresponding private key $\boldsymbol{e}_{\mathrm{id}}$, compute $M'=c_2-\boldsymbol{c}_1\boldsymbol{e}_{\mathrm{id}}$. If $M'\in(\lfloor q/4\rfloor, \lfloor 3q/4\rfloor)$, set $M'=1$; otherwise, $M'=0$.

3.3 Ciphertext share processing algorithms

1. PreEnc$\left(\boldsymbol{C}_{\mathrm{id}_1},\{\mathrm{kFrag}_i\}\right)$

The input contains a user id_1's ciphertext $\boldsymbol{C}_{\mathrm{id}_1}=(\boldsymbol{c}_{1,\mathrm{id}_1}, c_{2,\mathrm{id}_1})$ and a PRE key share $\{\mathrm{kFrag}_i\}=\left\{\overline{\boldsymbol{RK}}_i=\left(\overline{\boldsymbol{P}}_i,\overline{\boldsymbol{Q}}_i-\mathrm{Power2}(\overline{\boldsymbol{e}}_{i,\mathrm{id}_1})\right),\mathrm{prfk}_i,\overline{\sigma}_i\right\}$.

(1) Compute $\boldsymbol{c}'_{1,\mathrm{id}_1}=\mathrm{Bits}\left(\boldsymbol{c}_{1,\mathrm{id}_1}\right)\cdot\overline{\boldsymbol{P}}_i\in\mathbb{Z}_q^{2m}$ and $c'_{2,\mathrm{id}_2}=c_{2,\mathrm{id}_2}+\mathrm{Bits}\left(\boldsymbol{c}_{1,\mathrm{id}_1}\right)\cdot\left(\overline{\boldsymbol{Q}}_i-\mathrm{Power2}(\overline{\boldsymbol{e}}_{i,\mathrm{id}_1})\right)\in\mathbb{Z}_q$.

(2) Compute $(\boldsymbol{e}'_1, e'_2)=F_{\mathrm{prfk}_i}(\boldsymbol{c}_{1,\mathrm{id}_1},c_{2,\mathrm{id}_1})\in\mathbb{Z}_r^{2m+1}$.

(3) Compute $\overline{\boldsymbol{c}}_{1,\mathrm{id}_2}=\boldsymbol{c}'_{1,\mathrm{id}_2}+\eta\boldsymbol{e}'_1$ and $\overline{c}_{2,\mathrm{id}_2}=c'_{2,\mathrm{id}_2}+\eta e'_2$. Set $\overline{\boldsymbol{C}}_{i,\mathrm{id}_2}=\left(\overline{\boldsymbol{c}}_{1,\mathrm{id}_2},\overline{c}_{2,\mathrm{id}_2}\right)$.

(4) Homomorphically evaluate the signature $\overline{\sigma}_{i,\mathrm{id}_2}=\mathrm{HS.SignEval}\left(g_{\boldsymbol{C}_{\mathrm{id}_1}},\overline{\sigma}_i\right)$, where circuit $g_{\boldsymbol{C}_{\mathrm{id}_1}}$ is defined as follows:

$$
\begin{aligned}
&g_{\boldsymbol{C}_{\mathrm{id}_1}}\left(\overline{\mathbf{RK}}_i=\left(\overline{\boldsymbol{P}}_i,\overline{\boldsymbol{Q}}_i-\mathrm{Power2}(\overline{\boldsymbol{e}}_{i,\mathrm{id}_1})\right),\mathrm{prfk}_i\right)\\
&=\Big(\mathrm{Bits}\left(\boldsymbol{c}_{1,\mathrm{id}_1}\right)\cdot\overline{\boldsymbol{P}}_i,c_{2,\mathrm{id}_1}+\mathrm{Bits}\left(\boldsymbol{c}_{1,\mathrm{id}_1}\right)\cdot\left(\overline{\boldsymbol{Q}}_i\right.\\
&\quad\left.-\mathrm{Power2}(\overline{\boldsymbol{e}}_{i,\mathrm{id}_1})\right)\Big)+\eta\cdot F_{\mathrm{prfk}_i}(\boldsymbol{c}_{1,\mathrm{id}_1},c_{2,\mathrm{id}_1})\in\mathbb{Z}_q^{2m+1}.
\end{aligned}
$$

Output the ciphertext share $\left\{\boldsymbol{C}_{\mathrm{cFrag},i}\right\}=\left\{\boldsymbol{C}_{i,\mathrm{id}_2},\overline{\sigma}_{i,\mathrm{id}_2}\right\}$.

2. Verify$\left(\mathrm{hsvk},\boldsymbol{C}_{\mathrm{cFrag},i}\right)$

On input of the verification key hsvk and a ciphertext share $\left\{\boldsymbol{C}_{\mathrm{cFrag},i}\right\}=\left\{\overline{\boldsymbol{C}}_{i,\mathrm{id}_2},\overline{\sigma}_{i,\mathrm{id}_2}\right\}$, the verification algorithm outputs HS.Verify$\left(\mathrm{hsvk},\overline{\boldsymbol{C}}_{i,\mathrm{id}_2},g_{\boldsymbol{C}_{\mathrm{id}_1}},\overline{\sigma}_{i,\mathrm{id}_2}\right)$.

3. Comb$\left(\boldsymbol{C}_{\mathrm{id}_1},\left\{\boldsymbol{C}_{\mathrm{cFrag},i}\right\}_{i\in S}\right)$

Assume that the participant set is S, and $k'=|S|$ represents the valid share size. If $k'<k$, output $\perp$; otherwise, calculate a whole ciphertext.

(1) For each ciphertext share $\{\boldsymbol{C}_{\mathrm{cFrag},i}\}$ ($i\in S$), check Verify(hsvk, $\{\boldsymbol{C}_{\mathrm{cFrag},i}\}$). If the verification fails, output $\perp$ and exit.

(2) Parse $\boldsymbol{C}_{\mathrm{cFrag},i}=\left\{\overline{\boldsymbol{C}}_{i,\mathrm{id}_2},\overline{\sigma}_{i,\mathrm{id}_2}\right\}$, and then take the proxy server's index i and $\overline{\boldsymbol{C}}_{i,\mathrm{id}_2}=\left(\overline{\boldsymbol{c}}_{1,\mathrm{id}_2},\overline{c}_{2,\mathrm{id}_2}\right)$. Calculate each Lagrange coefficient as

$$\lambda_i=\prod_{j\in S,i\neq j}\frac{-j}{i-j}.$$

At last, recover a whole ciphertext as

$$
\begin{aligned}
\boldsymbol{C}_{\mathrm{id}_2}&=\left(\boldsymbol{c}_{1,\mathrm{id}_2},c_{2,\mathrm{id}_2}\right)\\
&=\sum_{i\in S}\lambda_i\left(\overline{\boldsymbol{c}}_{1,\mathrm{id}_2},\overline{c}_{2,\mathrm{id}_2}\right)_i+\left(0,\left(1-\sum_{i\in S}\lambda_i\right)c_{2,\mathrm{id}_1}\right).
\end{aligned}
$$

The workflow of our proposed scheme is shown in Fig. 2. It works as follows: (1) The PKG initializes and generates PP and MSK; (2) Users submit their identity information to the PKG and receive their private keys; (3) A delegator named Alice receives a ciphertext C_{Alice} that is encrypted under her own identity from someone; (4) When Alice is unable to handle the ciphertext for some reason, she will forward the ciphertext to proxy servers; (5) At the same time, Alice generates N proxy key shares $\{\mathrm{kFrag}_i\}$ ($1\leq i\leq N$) and authorizes the decryption permission to a delegatee named Bob; (6) Several of the N proxy servers obtain some ciphertext shares $\{\mathrm{cFrag}_i\}$ by re-encryption and forward them to Bob; (7) Bob verifies the validity of each received ciphertext share $\{\mathrm{cFrag}_i\}$ and accepts it if legal; (8) If Bob collects at least k ($k\leq N$) ciphertext shares, he can combine one whole ciphertext C_{Bob} intended for him; (9) Bob uses his private key to decrypt C_{Bob} and receives the original message.

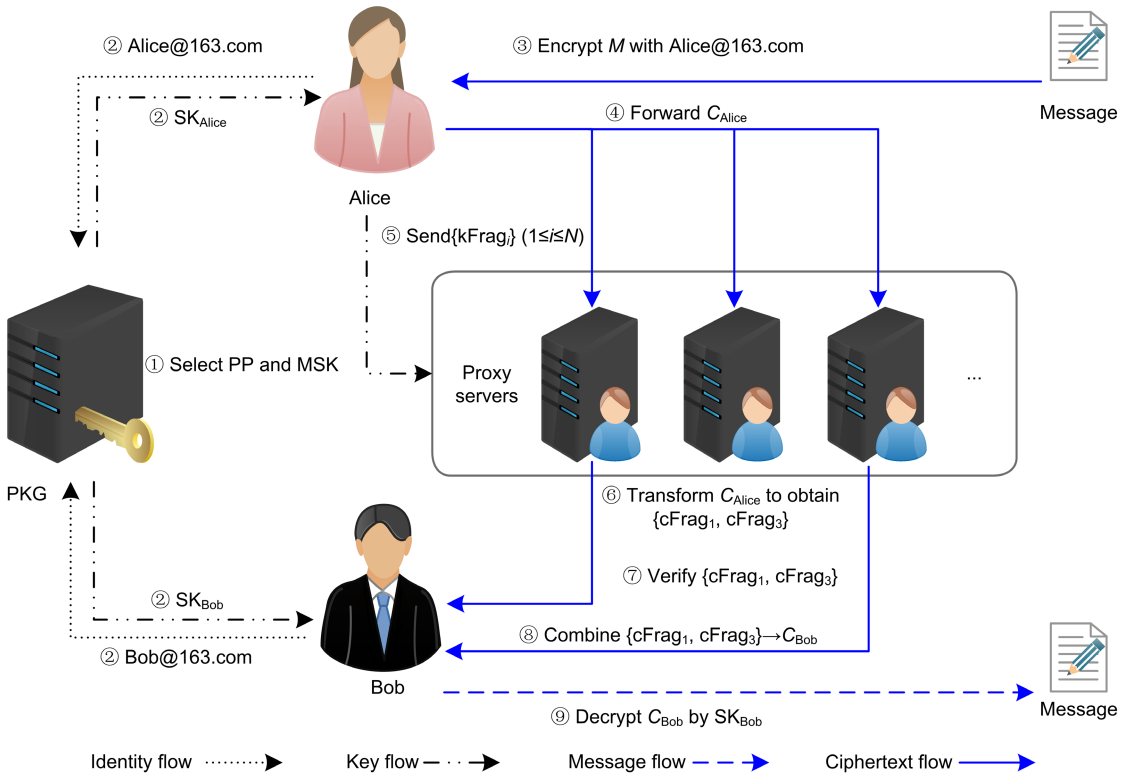

Fig. 2 Workflow of our proposed IB-TPRE scheme

4 Analysis of the IB-TRPE scheme

4.1 Correctness

Theorem 2 (Correctness) Assume $m=\lceil 6n\log_2 q\rceil$, $\sigma \geq \left\|\tilde{\boldsymbol{T}}_{A_0}\right\| \omega\left(\sqrt{\log_2 n}\right)$, and B_γ (B_e) the upper bound of the discrete Gaussian distribution ψ_γ (ψ_e). The output of the pseudo-random function F_{prfk} is a uniform distribution with its upper bound B_r, and set $q \geq 8\max\left\{4ml, \sqrt{2mk}(N!)^3\right\} B_e B_\gamma + 8\sqrt{2mk}(N!)^3 B_r B_\gamma$. The probability of successful decryption of our new IB-TPRE is close to 1 in a single hop.

Proof (1) For a normal ciphertext, decrypt it by $c_2 - \boldsymbol{c}_1 \boldsymbol{e}_{\mathrm{id}}$ as follows:

$$\begin{aligned} & c_2 - \boldsymbol{c}_1 \boldsymbol{e}_{\mathrm{id}} \\ &= \boldsymbol{s}^{\mathrm{T}} \boldsymbol{u} + \eta x + M\lfloor q/2 \rfloor + [\boldsymbol{s}^{\mathrm{T}} \boldsymbol{F}_{\mathrm{id}} + \eta(\boldsymbol{y} \mid z)] \boldsymbol{e}_{\mathrm{id}} \\ &= \underbrace{\eta[x - (\boldsymbol{y} \mid z) \boldsymbol{e}_{\mathrm{id}}]}_{\text{noise}} + M\lfloor q/2 \rfloor. \end{aligned} \quad (6)$$

Only when the noise $\eta[x-(\boldsymbol{y}|z)\boldsymbol{e}_{\mathrm{id}}]$ is no greater than $\lfloor q/4 \rfloor$, can the decryption algorithm recover M correctly. In truth, none of the parameters x, $(\boldsymbol{y}|z)$, and $\boldsymbol{e}_{\mathrm{id}}$ will exceed B_e or B_γ; that is, $\eta(x-(\boldsymbol{y}\mid z)\boldsymbol{e}_{\mathrm{id}}) \leq \left(1+\sqrt{2m}\right)\eta B_e B_\gamma$. Therefore, $q \geq 4\left(1+\sqrt{2m}\right)\eta B_e B_\gamma$ is sufficient.

(2) For a transformed ciphertext, decrypt it with id_2's private key. We can obtain Eq. (7) at the top of the next page.

The fifth equal sign in Eq. (7) at the top of the next page holds because $\sum_{i\in S}\lambda_i \overline{\boldsymbol{P}} = \boldsymbol{P}$, $\sum_{i\in S}\lambda_i \overline{\boldsymbol{Q}}_i = \boldsymbol{Q}$, and $\mathrm{Bits}(\boldsymbol{c}_{1,\mathrm{id}_1}) \cdot \mathrm{Power2}(\overline{\boldsymbol{e}}_{i,\mathrm{id}_1}) = \boldsymbol{c}_{1,\mathrm{id}_1}\overline{\boldsymbol{e}}_{i,\mathrm{id}_1}$.

The sixth equal sign in Eq. (7) holds because $\boldsymbol{c}_{1,\mathrm{id}_1} = \boldsymbol{s}^{\mathrm{T}}\boldsymbol{F}_{\mathrm{id}} + \eta(\boldsymbol{y}\mid z)$ and $\sum_{i\in S}\lambda_i \boldsymbol{F}_{\mathrm{id}}\overline{\boldsymbol{e}}_{i,\mathrm{id}_1} = \boldsymbol{u}$.

Then decrypt the ciphertext using Eq. (8) at the top of the next page.

The third equal sign holds because $\boldsymbol{P} = \boldsymbol{R}\boldsymbol{F}_{\mathrm{id}_2} + \boldsymbol{E}$, $\boldsymbol{Q} = \boldsymbol{R}\boldsymbol{u} + \boldsymbol{e}$, $\boldsymbol{Q} - \boldsymbol{P}\boldsymbol{e}_{\mathrm{id}_2} = \boldsymbol{e} - \boldsymbol{E}\boldsymbol{e}_{\mathrm{id}_2}$, and $c_{2,\mathrm{id}_1} = \boldsymbol{s}^{\mathrm{T}}\boldsymbol{u} + \eta x + M\lfloor q/2 \rfloor$.

$$
\begin{aligned}
&\boldsymbol{C}_{\mathrm{id}_2}=\left(\boldsymbol{c}_{1,\mathrm{id}_2},c_{2,\mathrm{id}_2}\right)\\
&=\sum_{i\in S}\lambda_i\left(\overline{\boldsymbol{c}}_{1,\mathrm{id}_2},\overline{c}_{2,\mathrm{id}_2}\right)_i+\left(0,\left(1-\sum_{i\in S}\lambda_i\right)c_{2,\mathrm{id}_1}\right)\\
&=\sum_{i\in S}\lambda_i\left[\left(\mathrm{Bits}(\boldsymbol{c}_{1,\mathrm{id}_1})\overline{\boldsymbol{P}_i}+\eta\left(\boldsymbol{e}_1'\right)_i,c_{2,\mathrm{id}_1}+\mathrm{Bits}(\boldsymbol{c}_{1,\mathrm{id}_1})\right)\left(\overline{\boldsymbol{Q}}_i-\mathrm{Power2}(\overline{\boldsymbol{e}}_{i,\mathrm{id}_1})\right)+\eta\left(e_2'\right)_i\right]+\left(0,\left(1-\sum_{i\in S}\lambda_i\right)c_{2,\mathrm{id}_1}\right)\\
&=\left(\mathrm{Bits}(\boldsymbol{c}_{1,\mathrm{id}_1})\sum_{i\in S}\lambda_i\overline{\boldsymbol{P}_i}+\sum_{i\in S}\lambda_i\eta\left(\boldsymbol{e}_1'\right)_i,c_{2,\mathrm{id}_1}+\mathrm{Bits}(\boldsymbol{c}_{1,\mathrm{id}_1})\sum_{i\in S}\lambda_i\overline{\boldsymbol{Q}}_i-\sum_{i\in S}\lambda_i\mathrm{Bits}(\boldsymbol{c}_{1,\mathrm{id}_1})\mathrm{Power2}(\overline{\boldsymbol{e}}_{i,\mathrm{id}_1})+\sum_{i\in S}\lambda_i\eta(e_2')_i\right)\\
&=\left(\mathrm{Bits}(\boldsymbol{c}_{1,\mathrm{id}_1})\boldsymbol{P}+\sum_{i\in S}\lambda_i\eta\left(\boldsymbol{e}_1'\right)_i,c_{2,\mathrm{id}_1}+\mathrm{Bits}(\boldsymbol{c}_{1,\mathrm{id}_1})\boldsymbol{Q}-\sum_{i\in S}\lambda_i\boldsymbol{c}_{1,\mathrm{id}_1}\overline{\boldsymbol{e}}_{i,\mathrm{id}_1}+\sum_{i\in S}\lambda_i\eta\left(e_2'\right)_i\right)\\
&=\left(\mathrm{Bits}(\boldsymbol{c}_{1,\mathrm{id}_1})\boldsymbol{P}+\sum_{i\in S}\lambda_i\eta\left(\boldsymbol{e}_1'\right)_i,c_{2,\mathrm{id}_1}+\mathrm{Bits}(\boldsymbol{c}_{1,\mathrm{id}_1})\boldsymbol{Q}-\boldsymbol{s}^{\mathrm{T}}\boldsymbol{u}-\sum_{i\in S}\lambda_i\eta\left(\boldsymbol{y}\,|\,z\right)\overline{\boldsymbol{e}}_{i,\mathrm{id}_1}+\sum_{i\in S}\lambda_i\eta\left(e_2'\right)_i\right).
\end{aligned}
\tag{7}
$$

$$
\begin{aligned}
&c_{2,\mathrm{id}_2}-\boldsymbol{c}_{1,\mathrm{id}_2}\boldsymbol{e}_{\mathrm{id}_2}\\
&=c_{2,\mathrm{id}_1}+\mathrm{Bits}(\boldsymbol{c}_{1,\mathrm{id}_1})\boldsymbol{Q}-\boldsymbol{s}^{\mathrm{T}}\boldsymbol{u}-\sum_{i\in S}\lambda_i\eta\left(\boldsymbol{y}\,|\,z\right)\overline{\boldsymbol{e}}_{i,\mathrm{id}_1}+\sum_{i\in S}\lambda_i\eta\left(e_2'\right)_i-\left(\mathrm{Bits}(\boldsymbol{c}_{1,\mathrm{id}_1})\boldsymbol{P}+\sum_{i\in S}\lambda_i\eta\left(\boldsymbol{e}_1'\right)_i\right)\boldsymbol{e}_{\mathrm{id}_2}\\
&=c_{2,\mathrm{id}_1}+\mathrm{Bits}(\boldsymbol{c}_{1,\mathrm{id}_1})\left(\boldsymbol{Q}-\boldsymbol{P}\boldsymbol{e}_{\mathrm{id}_2}\right)-\boldsymbol{s}^{\mathrm{T}}\boldsymbol{u}-\sum_{i\in S}\lambda_i\eta\left(\boldsymbol{y}\,|\,z\right)\overline{\boldsymbol{e}}_{i,\mathrm{id}_1}+\sum_{i\in S}\lambda_i\eta\left(e_2'\right)_i-\sum_{i\in S}\left[\lambda_i\eta\left(\boldsymbol{e}_1'\right)_i\right]\boldsymbol{e}_{\mathrm{id}_2}\\
&=\underbrace{\eta x+\mathrm{Bits}(\boldsymbol{c}_{1,\mathrm{id}_1})\left(\boldsymbol{e}-\boldsymbol{E}\boldsymbol{e}_{\mathrm{id}_2}\right)-\sum_{i\in S}\lambda_i\eta\left(\boldsymbol{y}\,|\,z\right)\overline{\boldsymbol{e}}_{i,\mathrm{id}_1}+\sum_{i\in S}\lambda_i\eta\left(e_2'\right)_i-\sum_{i\in S}\left[\lambda_i\eta\left(\boldsymbol{e}_1'\right)_i\right]\boldsymbol{e}_{\mathrm{id}_2}}_{\text{noise}}+M\lfloor q/2\rfloor.
\end{aligned}
\tag{8}
$$

It can be verified that the transformed ciphertext is decrypted correctly despite there is more converted noise than the initial noise. The following lemma follows a simple combinatorial argument:

Lemma 2 (Boneh et al., 2017) For any set $S\in[N]\cup\{0\}$ of size k and $1\le i\le N$, the product $(N!)^2\lambda_i^S$ is an integer and $(N!)^2\lambda_i^S\le(N!)^3$, where λ_i^S are efficiently computable Lagrange coefficients.

From Lemma 2, we have

$$
\begin{cases}
\sum_{i\in S}\lambda_i\eta\left(\boldsymbol{y}\,|\,z\right)\overline{\boldsymbol{e}}_i\le\sqrt{2m}k(N!)^3B_eB_\gamma,\\
\sum_{i\in S}\lambda_i\eta\left(e_2'\right)_i\le k(N!)^3B_r,\\
\sum_{i\in S}\lambda_i\eta\left(\boldsymbol{e}_1'\right)_i\boldsymbol{e}_{\mathrm{id}_2}\le\sqrt{2m}k(N!)^3B_rB_\gamma,\\
\mathrm{Bits}(\boldsymbol{c}_{1,\mathrm{id}_1})(\boldsymbol{e}-\boldsymbol{E}\boldsymbol{e}_{\mathrm{id}_2})\le 4mlB_eB_\gamma.
\end{cases}
$$

Therefore, the new noise can be expressed as a uniformly distributed noise and a linear combination of the product of uniformly distributed noise and Gaussian noise; i.e., the upper bound of all the kinds of noise is

$$
\begin{aligned}
&\|\,\text{noise}\,\|\\
&\le\eta B_e+4mlB_eB_\gamma+\sqrt{2m}k(N!)^3B_eB_\gamma\\
&\quad+\sqrt{2m}k(N!)^3B_r+\sqrt{2m}k(N!)^3B_rB_\gamma\\
&\le 2\max\left\{4ml,\sqrt{2m}k(N!)^3\right\}B_eB_\gamma+2\sqrt{2m}k(N!)^3B_rB_\gamma.
\end{aligned}
$$

When $q\ge 8\max\left\{4ml,\sqrt{2m}k(N!)^3\right\}B_eB_\gamma+8\sqrt{2m}k\cdot(N!)^3B_rB_\gamma$, the decryption algorithm can recover plaintext correctly after one transformation.

In fact, our scheme can realize limited multi-hop, which is a limited version of the multi-hop property because the re-encryption function introduces some noise in the ciphertext after transformation. For this reason, depending on the parameters used, the accumulated noise causes the decryption to fail after a certain number of re-encryptions. So, correctness verification for the limited multi-hop of our proposed scheme is left for further research.

Correctness of the verification for a ciphertext share is guaranteed by HS.Verify from the HS scheme. The evaluation circuit $g_{C_{\mathrm{id}_1}}$ in PreEnc, exactly defined by the original ciphertext $\boldsymbol{C}_{\mathrm{id}_1}$, takes the signed re-encryption key share as its input. On one hand, the processes for $\overline{\boldsymbol{C}}_{\mathrm{id}_2}=(\overline{\boldsymbol{c}}_{1,\mathrm{id}_2},\overline{\boldsymbol{c}}_{2,\mathrm{id}_2})$ (steps (1)–(3) in PreEnc) can be viewed as the calculation at the message level. On the other hand, the same operations will be performed on the signature by $\mathrm{HS.SignEVal}(g_{C_{\mathrm{id}_1}},\overline{\sigma}_i)=\overline{\sigma}_{i,\mathrm{id}_2}$ (step (4) in PreEnc). According to the correctness of the HS scheme, if $\overline{\sigma}_{i,\mathrm{id}_2}$ is the correct result honestly obtained by the proxy server with HS.SignEval, it can surely pass the verification algorithm, so the verification for a ciphertext share is correct.

4.2 Security

Theorem 3 (Security) The IB-TPRE system is IND-sID-CPA secure if the LWE assumption holds.

Proof The security proof is similar to that in Agrawal et al. (2010), except that we take a PRE interaction into consideration.

Assuming that there exists an adversary $\mathcal{A}$ that can break our scheme's IND-sID-CPA security, then a challenger $\mathcal{C}$ can be constructed to solve the decisional LWE hardness assumption.

1. Init

Adversary $\mathcal{A}$ outputs a target id^* intended to be attacked.

2. Setup

Challenger $\mathcal{C}$ generates system parameters as follows: It selects $\boldsymbol{A}_0$ randomly over $\mathbb{Z}_q^{n\times m}$ without any trapdoor; however, it generates $\boldsymbol{B}$ with a trapdoor $\boldsymbol{T_B}$ by the TrapGen algorithm. In addition, the public matrix $\boldsymbol{A}_1$ is not obtained at random; however, it is computed by $\boldsymbol{A}_1=\boldsymbol{A}_0\boldsymbol{R}^*-H(\mathrm{id}^*)\boldsymbol{B}$, where matrix $\boldsymbol{R}^*$ is randomly generated for the creation of challenge ciphertext. The corrupted user set Γ_{C} is empty at the beginning, and the honest user set Γ_{H} has only one element, which is id^*. The other settings are identical as described in the real scheme.

3. Query phase 1

Adversary $\mathcal{A}$ chooses some queries.

(1) KeyGen query

Adversary $\mathcal{A}$ can make a private key query for user id. If $\mathrm{id}\notin\Gamma_{\mathrm{H}}$, challenger $\mathcal{C}$ runs the algorithm KeyGen(MSK, id), where the public key is constructed as $\boldsymbol{F}_{\mathrm{id}}=(\boldsymbol{A}_0|\boldsymbol{A}_0\boldsymbol{R}^*+(H(\mathrm{id})-H(\mathrm{id}^*))\boldsymbol{B})$ and the private key is computed by SampleRight and SampleBasisRight with $\boldsymbol{T_B}$. At last, $\mathcal{C}$ returns the private key to $\mathcal{A}$, and then adds id to Γ_{C}. Otherwise, return $\perp$.

(2) ReKeyGen query

When adversary $\mathcal{A}$ issues re-encryption key shares between users id_1 and id_2, challenger $\mathcal{C}$ provides different solutions according to different cases, as shown in Table 1.

(3) PreEnc query

When $\mathrm{id}_1\in\Gamma_{\mathrm{H}}$ and $\mathrm{id}_2\in\Gamma_{\mathrm{H}}$, return a transformed ciphertext share by PreEnc after a ReKeyGen query between id_1 and id_2.

4. Challenge phase

Challenger $\mathcal{C}$ queries the LWE oracle $\mathcal{O}$ for $m+1$ instances $(\boldsymbol{v}_i,w_i)\in\mathbb{Z}_q^n\times\mathbb{Z}_q$ ($1\leq i\leq m+1$), which may be either a truly random distribution from $\mathcal{O}_{\$}$ or a noisy pseudo-random distribution from $\mathcal{O}_S$.

(1) Set $\boldsymbol{c}_1'=(w_1,w_2,\cdots,w_m)$, and compute $\boldsymbol{c}_1''=\boldsymbol{c}_1'\boldsymbol{R}^*$ and $\boldsymbol{c}_1^*=(\boldsymbol{c}_1'\,|\,\boldsymbol{c}_1'')$.

(2) Compute $c_2^*=w_{m+1}+M\lfloor q/2\rfloor$.

(3) Output the challenge ciphertext $\boldsymbol{C}^*=(\boldsymbol{c}_1^*,c_2^*)\in\mathbb{Z}_q^{2m}\times\mathbb{Z}_q$.

5. Query phase 2

$\mathcal{A}$ continues to make queries as does in query phase 1. Queries KeyGen for id^* and ReKeyGen from id^* to $\mathrm{id}_2\in\Gamma_{\mathrm{C}}$ are not permitted.

6. Guess

$\mathcal{A}$ guesses $b'\in\{0,1\}$ to distinguish the real ciphertext from the random ciphertext, and challenger $\mathcal{C}$ picks a bit $r'=b'$ accordingly, which will distinguish whether the obtained samples are subject to LWE distributions from $\mathcal{O}_S$ or random distributions from $\mathcal{O}_{\$}$.

Table 1 Strategies to answer ReKeyGen queries

Delegator	Delegatee	User processing	Returned content
$\mathrm{id}_1\notin(\Gamma_\mathrm{H}\cup\Gamma_\mathrm{C})$	$\mathrm{id}_2\in\Gamma_\mathrm{H}$	No action	Obtain $\mathbf{RK}_{\mathrm{id}_1}\leftarrow$ KeyGen with $\boldsymbol{T_B}$, compute re-encryption key shares $\{\mathrm{kFrag}_i\}$ $(1\le i\le N)$ by ReKeyGen, and return all the shares
	$\mathrm{id}_2\notin(\Gamma_\mathrm{H}\cup\Gamma_\mathrm{C})$	Add id_2 to Γ_H	
	$\mathrm{id}_2\in\Gamma_\mathrm{C}$	Add id_1 to Γ_H	Obtain $\mathbf{RK}_{\mathrm{id}_1}\leftarrow$ KeyGen with $\boldsymbol{T_B}$, compute re-encryption key shares $\{\mathrm{kFrag}_i\}$ $(1\le i\le N)$ by ReKeyGen, and return only the first $k-1$ shares
$\mathrm{id}_1\in\Gamma_\mathrm{H}$	$\mathrm{id}_2\in\Gamma_\mathrm{H}$	No action	If $\mathrm{id}_1\neq\mathrm{id}^*$, obtain $\mathbf{RK}_{\mathrm{id}_1}\leftarrow$ KeyGen with $\boldsymbol{T_B}$ and compute re-encryption key shares $\{\mathrm{kFrag}_i\}$ $(1\le i\le N)$ by ReKeyGen. Otherwise, randomly generate a temporary $\mathbf{RK}_{\mathrm{id}_1\to\mathrm{id}_2}$ and divide it into N shares. At last, return all of them to N proxy servers
	$\mathrm{id}_2\in\Gamma_\mathrm{C}$	No action	If $\mathrm{id}_1\neq\mathrm{id}^*$, obtain $\mathbf{RK}_{\mathrm{id}_1}\leftarrow$ KeyGen with $\boldsymbol{T_B}$ and compute re-encryption key shares $\{\mathrm{kFrag}_i\}$ $(1\le i\le N)$ by ReKeyGen. Otherwise, randomly generate a temporary $\mathbf{RK}_{\mathrm{id}_1\to\mathrm{id}_2}$ and divide it into N shares. At last, return only the first $k-1$ shares
	$\mathrm{id}_2\notin(\Gamma_\mathrm{H}\cup\Gamma_\mathrm{C})$	No action	
$\mathrm{id}_1\in\Gamma_\mathrm{C}$		No action	$\perp$

Next, we analyze the indistinguishability of the simulated behaviors from challenger $\mathcal{C}$.

1. In the real scheme, matrix $\boldsymbol{A}_1$ is taken from the uniform distribution, while challenger $\mathcal{C}$ generates matrix $\boldsymbol{A}_1=\boldsymbol{A}_0\boldsymbol{R}^*-H(\mathrm{id}^*)\boldsymbol{B}$. It is identical by the hash leftover lemma (Agrawal et al., 2010), which states that the uniform distribution $\boldsymbol{A}_1$ is statistically indistinguishable from the distribution $\boldsymbol{A}_0\boldsymbol{R}^*$, where $\boldsymbol{R}^*$ is a square randomly chosen from $\{-1, 1\}^{m\times m}$.

2. When answering the private key queries or re-encryption key queries, challenger $\mathcal{C}$ constructs the public key as $\boldsymbol{F}_\mathrm{id}=(\boldsymbol{A}_0|\boldsymbol{A}_0\boldsymbol{R}^*+(H(\mathrm{id})-H(\mathrm{id}^*))\boldsymbol{B})$. If $\mathrm{id}_1\neq\mathrm{id}^*$, $\boldsymbol{H}(\mathrm{id})-\boldsymbol{H}(\mathrm{id}^*)$ is a full-rank matrix by FRD definition, and $\boldsymbol{T_B}$ can be regarded as a trapdoor for $\Lambda_q^\perp(\boldsymbol{B})$. Challenger $\mathcal{C}$ can now generate a private key $\boldsymbol{e}_\mathrm{id}$ satisfying $\boldsymbol{F}_\mathrm{id}\boldsymbol{e}_\mathrm{id}=\boldsymbol{u}$ using algorithm $\mathrm{SampleRight}(\boldsymbol{A}_0, (H(\mathrm{id})-H(\mathrm{id}^*))\boldsymbol{B}, \boldsymbol{R}^*, \boldsymbol{T_B}, \psi_\gamma)$. Similarly, $\mathcal{C}$ first generates a user trapdoor $\boldsymbol{T}_\mathrm{id}=\mathrm{SampleBasisRight}(\boldsymbol{A}_0, (H(\mathrm{id})-H(\mathrm{id}^*))\boldsymbol{B}, \boldsymbol{R}^*, \boldsymbol{T_B}, s_\mathrm{t})$, and then computes $\overline{\boldsymbol{e}}_{i,\mathrm{id}}$ $(1\le i\le N)$ that satisfies $\boldsymbol{F}_\mathrm{id}\overline{\boldsymbol{e}}_{i,\mathrm{id}}=\overline{\boldsymbol{u}}$ by $\overline{\boldsymbol{e}}_{i,\mathrm{id}}\leftarrow\mathrm{SamplePre}(\boldsymbol{F}_\mathrm{id}, \boldsymbol{T}_\mathrm{id}, \overline{\boldsymbol{u}}_i, \psi_\gamma)$. Finally, $\mathcal{C}$ obtains a re-encryption key share for each proxy server labeled i. If $\mathrm{id}=\mathrm{id}^*$, $\boldsymbol{F}_\mathrm{id}=(\boldsymbol{A}_0|\boldsymbol{A}_0\boldsymbol{R}^*)$ has nothing to do with $\boldsymbol{B}$ and the challenger's trapdoor $\boldsymbol{T_B}$ disappears. As a result, the challenger is unable to generate a private key for id^* or re-encryption key shares from id^* to others, which prevents the adversary from obtaining the private key for id^* or translating id^* to user id instead, for which adversary $\mathcal{A}$ holds a private key.

3. If asked for the re-encryption key share query on $\mathrm{id}_1=\mathrm{id}^*\in\Gamma_\mathrm{H}$ and $\mathrm{id}_2\in\Gamma_\mathrm{H}$, challenger $\mathcal{C}$ selects $\mathbf{RK}_{1\to2}=(\boldsymbol{P}, \boldsymbol{Q})$, consisting of freshly random matrices $\boldsymbol{P}\in\mathbb{Z}_q^{(2ml)\times(2m)}$ and $\boldsymbol{Q}\in\mathbb{Z}_q^{2ml}$. We stress that it is unable to distinguish the selected pair $(\boldsymbol{P}, \boldsymbol{Q})$ mentioned above from $\boldsymbol{P}=\boldsymbol{R}\boldsymbol{F}_{\mathrm{id}_2}+\boldsymbol{E}\in\mathbb{Z}_q^{(2ml)\times(2m)}$ and $\boldsymbol{Q}=\boldsymbol{R}\boldsymbol{u}+\boldsymbol{e}\in\mathbb{Z}_q^{2ml}$ computed in the real scheme, because the decisional LWE problem implies that it is indistinguishable between the LWE distribution and the true random uniform distribution.

In the following, we will show that $k-1$ re-encryption key share and re-encryption share queries will not leak the private key information of id^*.

The challenger randomly chooses the uniformly distributed $\mathbf{RK}_{1\to2}=(\boldsymbol{P}, \boldsymbol{Q})$ as the re-encryption key and sends the first $k-1$ key shares $(\overline{\boldsymbol{P}}_i, \overline{\boldsymbol{Q}}_i)$ $(1\le i\le k-1)$ to the first $k-1$ corrupted servers that are owned by adversary $\mathcal{A}$, thus forming a set S^* $(|S^*|=k-1)$. After that, if asked for re-encryption, challenger $\mathcal{C}$ does the following:

(1) Calculate Lagrange coefficients $\lambda_i^{S^*}$, where $i \in \tilde{S}^*$ and $\tilde{S}^* = S^* \cup \{0\}$.

(2) Select an integer vector $\boldsymbol{e}_i \xleftarrow{\$} [-r, r]^{2m+1}$ randomly for $i \in \tilde{S}^*$.

(3) Calculate

$$(\boldsymbol{c}_1', c_2') = \lambda_0^{\tilde{S}^*}\left(\mathrm{Bits}(\boldsymbol{c}_{1,\mathrm{id}_1})\boldsymbol{P}, c_{2,\mathrm{id}_1} + \mathrm{Bits}(\boldsymbol{c}_{1,\mathrm{id}_1})\boldsymbol{Q}\right) + \sum_{i\in S^*}\lambda_i^{\tilde{S}^*}\left(\mathrm{Bits}(\boldsymbol{c}_{1,\mathrm{id}_1})\overline{\boldsymbol{P}}_i, c_{2,\mathrm{id}_1} + \mathrm{Bits}(\boldsymbol{c}_{1,\mathrm{id}_1})\overline{\boldsymbol{Q}}_i\right) + \left[0, \left(1 - \sum_{i\in\tilde{S}^*}\lambda_i^{\tilde{S}^*}\right)c_{2,\mathrm{id}_1}\right] + \sum_{i\in\tilde{S}^*}\lambda_i^{\tilde{S}^*}\eta\boldsymbol{e}_i.$$

The first part $\lambda_0^{\tilde{S}^*}\left(\mathrm{Bits}(\boldsymbol{c}_{1,\mathrm{id}_1})\boldsymbol{P}, c_{2,\mathrm{id}_1} + \mathrm{Bits}(\boldsymbol{c}_{1,\mathrm{id}_1})\boldsymbol{Q}\right)$ can be regarded as one ciphertext share provided by the participant whose number is 0 and the share is $\{\boldsymbol{P}, \boldsymbol{Q}\}$.

The second part $\sum_{i\in\tilde{S}^*}\lambda_i^{\tilde{S}^*}\left(\mathrm{Bits}(\boldsymbol{c}_{1,\mathrm{id}_1})\overline{\boldsymbol{P}}_i, c_{2,\mathrm{id}_1} + \mathrm{Bits}(\boldsymbol{c}_{1,\mathrm{id}_1})\overline{\boldsymbol{Q}}_i\right)$ can be regarded as $k-1$ ciphertext shares provided by the participants whose numbers are 1, 2, …, $k-1$ from set S^*. The sum of these two parts exactly reaches the threshold k. Therefore, we have

$$(\boldsymbol{c}_1', c_2') = \left(\mathrm{Bits}(\boldsymbol{c}_{1,\mathrm{id}_1})\boldsymbol{P}, c_{2,\mathrm{id}_1} + \mathrm{Bits}(\boldsymbol{c}_{1,\mathrm{id}_1})\boldsymbol{Q}\right) + \sum_{i\in\tilde{S}^*}\lambda_i^{\tilde{S}^*}\eta\boldsymbol{e}_i.$$

The calculation process and the final expression are the same as their counter parts of the Comb algorithm, and the resulting distribution is statistically indistinguishable. In truth, challenger $\mathcal{C}$ does not know the real re-encryption key. In other words, when $\mathrm{id}_1 = \mathrm{id}^* \in \Gamma_{\mathrm{H}}$ and $\mathrm{id}_2 \in \Gamma_{\mathrm{H}}$, providing $k-1$ re-encryption key shares and answering the re-encryption share query will not reveal any information about the private key for id_1 and the re-encryption key between id_1 and id_2.

(4) As to the challenge ciphertext, if the instances are from the LWE distribution, that is, assuming $\boldsymbol{c}_1^* = (w_1, w_2, ..., w_m) = \boldsymbol{s}^{\mathrm{T}}\boldsymbol{A}_0 + \boldsymbol{x}$, then $\boldsymbol{C}^* = (\boldsymbol{c}_1^*, c_2^*)$ is distributed precisely as in the real scheme. First, since $\boldsymbol{F}_{\mathrm{id}^*} = (\boldsymbol{A}_0 \mid \boldsymbol{A}_0\boldsymbol{R}^*)$, then

$$\boldsymbol{c}_1^* = \begin{bmatrix} \boldsymbol{s}^{\mathrm{T}}\boldsymbol{A}_0 + \eta\boldsymbol{y} \\ \boldsymbol{s}^{\mathrm{T}}(\boldsymbol{A}_0\boldsymbol{R}^*) + \eta\boldsymbol{y}\boldsymbol{R}^* \end{bmatrix} = \boldsymbol{s}^{\mathrm{T}}\begin{bmatrix} \boldsymbol{A}_0 \\ \boldsymbol{A}_0\boldsymbol{R}^* \end{bmatrix} + \eta\begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{y}\boldsymbol{R}^* \end{bmatrix} = \boldsymbol{s}^{\mathrm{T}}\boldsymbol{F}_{\mathrm{id}^*} + \eta\begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{y}\boldsymbol{R}^* \end{bmatrix},$$

which is exactly identical to the first part of a valid ciphertext created by our IB-TPRE scheme. Second, $c_2^* = w_{m+1} + M\lfloor q/2 \rfloor = \boldsymbol{s}^{\mathrm{T}}\boldsymbol{u} + \eta x + M\lfloor q/2 \rfloor$, which is exactly identical to the second part of a valid ciphertext created by our scheme. If adversary $\mathcal{A}$ can distinguish challenge ciphertext from a random distribution, challenger $\mathcal{C}$ can solve the decisional LWE hardness assumption. Hence, our scheme is semantically secure.

Theorem 4 (Robustness) If a homomorphic signature scheme Π_{HS}=(HS.KeyGen, HS.Sign, HS.SignEval, HS.Verify) satisfies unforgeability, the proposed IB-TPRE scheme is robust.

Proof Intuitively speaking, a dishonest proxy server can successfully obtain an invalid re-encrypted ciphertext share and the corresponding signature using an arbitrarily chosen evaluation circuit. However, as described in HS.Verify(hsvk, y, C, σ^*), the correct evaluation circuit C is essentially defined by the original ciphertext, and the verification fails if the forged evaluation circuit is different from the correct one; therefore, it forces the proxy servers to perform the transformation honestly.

In a formal proof, the robustness of the new IB-TPRE scheme can be reduced to the unforgeability of a homomorphic signature. If an adversary $\mathcal{A}$ can break through the robustness game described in Definition 5, then a simulator $\mathcal{S}$ can be constructed to break the unforgeability of the HS by interacting with adversary $\mathcal{A}$. The process is as follows:

$\mathcal{S}$ first obtains the verification key hsvk, and adversary $\mathcal{A}$ chooses the re-encryption key between id_1 and id_2 intended to be attacked. When adversary $\mathcal{A}$ submits a forged re-encrypted ciphertext share $\boldsymbol{C}_{\mathrm{cFrag},i}^* = \left\{\overline{\boldsymbol{C}}_{i,\mathrm{id}_2}^*, \overline{\sigma}_{i,\mathrm{id}_2}^*\right\}$ to the simulator $\mathcal{S}$, $\mathcal{S}$ reconstructs $\left(\mathrm{hsvk}, g_{\boldsymbol{C}_{\mathrm{id}_1}}, \overline{\boldsymbol{C}}_{i,\mathrm{id}_2}^*, \overline{\sigma}_{i,\mathrm{id}_2}^*\right)$ and submits it to the homomorphic signature oracle.

If adversary $\mathcal{A}$ can win the robustness game, that is, $\left\{\mathrm{cFrag}_{j^*,\mathrm{id}^*}\right\} \notin \mathrm{PreEnc}\left(C_1, \left\{\mathrm{kFrag}_{j^*,\mathrm{id}^*}\right\}\right)$, where $1 \le j^* \le N$, but $\mathrm{Verify}\left(\left\{\mathrm{hsvk}_{j^*}\right\}, \left\{\mathrm{cFrag}_{j^*,\mathrm{id}^*}\right\}\right) = 1$, then the simulator $\mathcal{S}$ can successfully forge an illegal signature satisfying $\mathrm{HS.Verify}\left(\left\{\mathrm{hsvk}_{j^*}\right\}, g_{\boldsymbol{C}_{\mathrm{id}_1}}, \overline{\boldsymbol{C}}_{i,\mathrm{id}_2}{}^*, \overline{\sigma}_{i,\mathrm{id}_2}{}^*\right) = 1$, which will be submitted to the homomorphic signature oracle later. This means that the unforgeability of homomorphic signature scheme Π_{HS} is breakable. However, this is contrary to our assumption.

Therefore, our new IB-TPRE is robust, provided that the homomorphic signature algorithm Π_{HS} is unforgeable.

Theorem 5 (Weak collusion resistance) Our IB-TRPE scheme achieves weak collusion resistance.

Proof Collusion resistance means that it is impossible to derive $\mathbf{SK}_A$ for the delegator, even if the dishonest proxy with $\mathbf{RK}_{A\to B}$ and a malicious delegatee Bob with $\mathbf{SK}_B$ collude. Collusion resistance becomes important only when the same key pair is used for other purposes (e.g., signing and key derivation). Its goal is to allow Alice to delegate decryption privileges while preserving the signature privileges under the same public key.

Weak collusion resistance in our IB-TPRE is guaranteed by two steps. The first step is to introduce multiple proxies to prevent collusion between a proxy and the delegatee. Conversely, to obtain a whole PRE key, an attacker must first collect k shares out of N re-encryption key shares by corrupting k proxy servers to obtain $\mathbf{RK}_{A\to B} = \left(\boldsymbol{P} = \boldsymbol{RF} + \boldsymbol{E}, \boldsymbol{Q} = \boldsymbol{Ru} + \boldsymbol{e} - \sum_{i\in S}\lambda_i \mathrm{Power2}\left(\overline{\boldsymbol{e}}_i\right)\right)$, where λ_i is a Lagrange coefficient for $i \in S$. Even if an attacker succeeds in step one, he/she can compute $\mathbf{SK}_{\mathrm{equal}} = \boldsymbol{Q} - \boldsymbol{Pe}_B = \boldsymbol{e} - \boldsymbol{Ee}_B - \sum_{i\in S}\lambda_i \mathrm{Power2}\left(\overline{\boldsymbol{e}}_i\right)$, where $\boldsymbol{e}_B$ is the private key for user B. The second step claims that $\mathbf{SK}_{\mathrm{equal}}$ allows recovering a similar secret key that can decrypt any ciphertext only for Alice. We explain it as follows:

$$
\begin{aligned}
& c_2 - \mathrm{Bits}(\boldsymbol{c}_1)\mathbf{SK}_{\mathrm{equal}} \\
&= \boldsymbol{s}^{\mathrm{T}}\boldsymbol{u} + \eta x + M\lfloor q/2 \rfloor - \mathrm{Bits}(\boldsymbol{c}_1)\boldsymbol{e} \\
&\quad + \sum_{i\in S}\lambda_i \left[\boldsymbol{s}^{\mathrm{T}}\boldsymbol{F}_{\mathrm{id}} + \eta(\boldsymbol{y} \mid \boldsymbol{z})\overline{\boldsymbol{e}}_i\right] \\
&= \underbrace{\eta\left[x + (\boldsymbol{y} \mid \boldsymbol{z})\sum_{i\in S}(\lambda_i \overline{\boldsymbol{e}}_i)\right] - \mathrm{Bits}(\boldsymbol{c}_1)(\boldsymbol{e} - \boldsymbol{Ee}_B)}_{\text{noise}} + M\lfloor q/2 \rfloor.
\end{aligned}
\tag{9}
$$

The message will be recovered correctly because the noise above is quite small, and it seems that Alice's private key is leaked completely if collusion occurred. However, this is not true because we might have recovered an equivalent private key with the ability to perform delegated decryption rights, but not the same key as Alice's original private key. On one hand, if an attacker has both $\mathbf{RK}_{A\to B}$ and $\mathbf{SK}_B$, he/she is quite capable of re-encrypting and reading any data that, originally, is decryptable by $\mathbf{SK}_A$. In addition, $\mathbf{SK}_{\mathrm{equal}}$ works only if it meets $\mathrm{Bits}(\boldsymbol{c}_1)$ when decrypted. On the other hand, another private user trapdoor $\boldsymbol{S}_A$ might be used for signing, which will never be recovered due to an irreversible lattice basis extension from the SampleBasisLeft algorithm. Therefore, our scheme is weak collusion resistant.

4.3 Comparison

There are two typical existing (IB-)TPRE constructions. The first one is UMBRAL (David, 2018), which is the central scheme of the NuCypher key management system. It is constructed on the decisional bilinear Diffie-Hellman (DBDH) assumption, which is vulnerable to quantum attack. The other one is the re-splittable threshold PRE scheme over lattices; it realizes robustness using a decisional discrete logarithm assumption that sacrifices efficiency, and it will also suffer from a quantum attack on re-encryption verification.

In our IB-TPRE scheme, the validity of a ciphertext share is verified by a homomorphic signature, which can be instantiated by the current efficient SIS-based fully homomorphic signature scheme (Boneh and Freeman, 2011; Gorbunov et al., 2015), and thus the final system can completely resist quantum attack. Also, when the original signature is evaluated by the Boolean circuit, that is, $\mathrm{Bits}(\boldsymbol{c}_{1,\mathrm{id}_1})$, the homomorphic signature calculation is a Boolean operation and is not highly complex, so it is more efficient. These three schemes are compared in

Table 2. The combined size of the ciphertext share and the key share does not include the payload for verification, and the ciphertext conversion does not include the computation complexity of verification for robustness.

5 Applications

The new IB-TPRE scheme has the advantages of high availability, low trust, and strong security. Therefore, it can become a core technique for private information secure sharing in outsourcing data security, secure multi-party computing, and decentralized encrypted computing. Two examples are given for two application scenarios.

5.1 File-sharing system based on a blockchain network

In a file-sharing system constructed by the traditional PRE, the proxy function is typically performed by an independent server. However, once the centralized proxy no longer remains neutral and refuses to provide re-encryption services, or offers wrong re-encryption services deliberately for some reason (such as war and commercial competition), it will directly lead to the failure of file sharing. To resolve this problem, a distributed file-sharing system with our IB-TRPE is constructed, where ciphertext transformation is carried out by a decentralized blockchain network, as shown in Fig. 3. Compared with the P2P technique, blockchain not only is decentralized, but also considers a trust model for each node. The verification of ciphertext shares in our IB-TPRE scheme provides a checking method for this trust relationship.

Step 1: Users can select the storage location at will, such as Amazon S3 and IPFS. A sender, such as Alice, encrypts files with her public key or with a hybrid encryption mode before uploading them.

Step 2: When Alice wants to share an encrypted file File_A with her friend Bob, each node from the blockchain network will act as a proxy server to receive a re-encryption key share between Alice and Bob. After that, Alice publishes a re-encryption task to the whole network, providing information such as the address of the file to be shared and the identities of recipients.

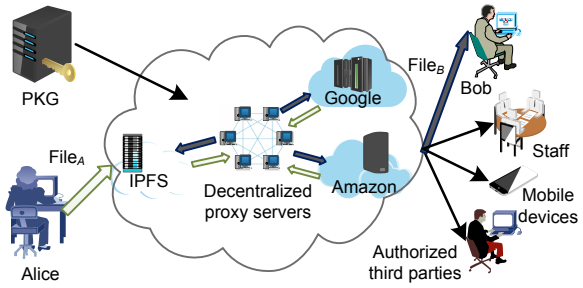

Fig. 3 A file-sharing system based on a blockchain network

Table 2 Property and complexity comparison of related works

Scheme	Underlying scheme	Construction assumption	Method to be robust	Tool for robustness	Resisting quantum attack	Identity-based
David (2018)'s	BBS98 (Blaze et al., 1998)	DBDH	Non-interactive, zero-knowledge	Decisional discrete logarithm	Neither encryption nor share verification	No
Li et al. (2017)'s	Singh (Singh et al., 2014)	Lattice	Non-interactive, zero-knowledge	Decisional discrete logarithm	Encryption but not share verification	No
Ours	BV-PRE (Polyakov et al., 2017)	Lattice	Homomorphic signature	SIS	Both encryption and share verification	Yes

Scheme	Message	Key share size	Ciphertext share size	Re-encryption complexity	Verification complexity	Mathematical structure
David (2018)'s	$\{0,1\}^l$	$6\lvert\mathbb{Z}_q\rvert$	$4\lvert\mathbb{Z}_q\rvert$	2Exp	2Exp+Hash	Group
Li et al. (2017)'s	$\{0,1\}$	$(nl+1)(n+1)\lvert\mathbb{Z}_q\rvert$	$(n+1)\lvert\mathbb{Z}_q\rvert$	$(nl+1)(n+1)$zMult	$(nl+1)(n+1)$(Exp+zMult)	Group+lattice
Ours	$\{0,1\}$	$2ml(2m+1)\lvert\mathbb{Z}_q\rvert$	$(2m+1)\lvert\mathbb{Z}_q\rvert$	$2ml(2m+1)$zMult	HS.Verify	Lattice

$\lvert\mathbb{Z}_q\rvert$ represents an integer on modulo q. n is the dimension of the polynomial, m is the sample number of LWE, and l is the length of the message encrypted each time. Exp represents an exponential operation on the group. Hash represents a hash operation. zMult represents a multiplication operation for two integers. HS.Verify represents a verification algorithm from a homomorphic signature, and the addition operation is ignored

Step 3: When receiving a re-encryption request, each network node allows the re-encryption service to obtain some tokens independently. For example, the node that offers the first k correct ciphertext shares for Bob will obtain some rewards from the system, and the robustness offers a method to identify whether the conversion result is correct.

Step 4: The storage server working for Bob will put the legal ciphertext shares together to form a new encrypted file $File_B$. Finally, Bob downloads and decrypts it.

The system resists collusion between proxy servers and the receiver by nature. It can also control the initiative of ciphertext conversion even if the attacker grasps a complete re-encryption key, but the plaintext will not be exposed online because it appears in the encrypted state. Therefore, the system realizes end-to-end encryption and privacy protection.

5.2 Robust key escrow system with threshold cryptography

There may be two conflicting requirements in current telecommunication systems. On one hand, users want to communicate with others secretly. On the other hand, to fight against crime and protect national security, the government requires the interception of user traffic. The key escrow system (Wang et al., 2019) is a solution that balances privacy protection and authorized monitoring.

A new robust key escrow system with threshold cryptography, evolving from our new IB-TRPE scheme, includes mainly the following four components: (1) The key generation center (KGC), which is responsible for initialization and maintenance of the key cryptosystem, is performed by PKG in our scheme; (2) Users such as Alice and Bob communicate with each other; (3) Key escrow agents (KEAs), which are authorized to escrow user keys, are played by proxy servers; (4) The legal enforcement agency (LEA) of the government, which is responsible for monitoring all user communication, has a unique identifier like a delegatee. The architecture of the proposed system is shown in Fig. 4.

Step 1: Alice and Bob apply their private keys to KGC. After that, they can communicate with each other securely. To be more efficient, the architecture can use hybrid encryption, in which the session key is randomly generated to encrypt actual messages by symmetric encryption, and the session key itself is also encrypted by the receiver's identity. LEA obtains its private key from the KGC under its own identity, and then publishes its identity to all users.

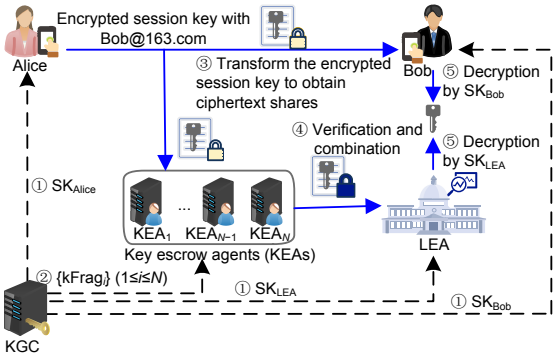

Fig. 4 A robust key escrow system with threshold cryptography

Step 2: KEAs, represented by N proxy servers in our IB-TRPE scheme, register in the KGC to obtain re-encryption key shares from Alice to the LEA, as well as from Bob to the LEA. Note that generating re-encryption key shares by KGC, not individual users, is feasible because KGC also has the user private key.

Step 3: When it is necessary to monitor the communications between Alice and Bob, the LEA applies the authorization to the government or regulatory affairs. After authorization, LEA forwards the authorization to KEAs for verifying correctness and timeliness. The KEAs convert Alice's ciphertext into some Bob's ciphertext shares by their own re-encryption key shares, and then send the converted ciphertext shares to the LEA for monitoring.

Step 4: After receiving the converted ciphertext shares, the LEA checks their validity. When k valid shares are collected, they are combined into a complete ciphertext. The LEA decrypts it using its private key to obtain a session key, and obtains further communication content.

The excellent properties of the new IB-TPRE scheme enhance the key escrow system to be more secure and versatile: (1) The encrypted shares have been kept secret in an encrypted state, so there is no need for a secure channel between KEAs and the LEA; (2) Our IB-TPRE can detect some wrong or dishonest conversions of KEAs in time, which makes the system more reliable and secure; (3) It needs only

k out of N KEAs online to complete the transformation; (4) The system avoids the "once monitor, monitor forever" approach in each session and achieves fine-grained control (Cheng et al., 2013), allowing only reconstruction of the current session key without leaking the previous and future communication content.

6 Conclusions

We proposed an IB-TPRE scheme over lattices by combining the threshold PRE, identity-based encryption, and homomorphic signature. After the IB-TPRE scheme was instantiated with an SIS-based fully homomorphic signature scheme, the final system can completely resist quantum attack. It provides encryption and cryptographic access control performed by a decentralized network, and some new compelling applications can be realized with the performance advantages under distributed systems.

Contributors

Liqiang WU designed the research. Xiaoyuan YANG processed the data. Yiliang HAN performed the security proof. Liqiang WU drafted the paper. Minqing ZHANG helped organize the paper. Yiliang HAN and Xiaoyuan YANG revised and finalized the paper.

Compliance with ethics guidelines

Liqiang WU, Yiliang HAN, Xiaoyuan YANG, and Minqing ZHANG declare that they have no conflict of interest.

References

Agrawal S, Boneh D, Boyen X, 2010. Efficient lattice (H)IBE in the standard model. Int Conf on the Theory and Applications of Cryptographic Techniques, p.553-572. https://doi.org/10.1007/978-3-642-13190-5_28

Aono Y, Boyen X, Phong LT, et al., 2013. Key-private proxy re-encryption under LWE. Int Conf on Cryptology in India, p.1-18. https://doi.org/10.1007/978-3-319-03515-4_1

Blaze M, Bleumer G, Strauss M, 1998. Divertible protocols and atomic proxy cryptography. Int Conf on the Theory and Applications of Cryptographic Techniques, p.127-144. https://doi.org/10.1007/BFb0054122

Boneh D, Freeman DM, 2011. Homomorphic signatures for polynomial functions. Annual Int Conf on the Theory and Applications of Cryptographic Techniques, p.149-168. https://doi.org/10.1007/978-3-642-20465-4_10

Boneh D, Gennaro R, Goldfeder S, et al., 2017. A lattice-based universal thresholdizer for cryptographic systems. IACR Cryptology ePrint Archive. https://eprint.iacr.org/2017/251

Cheng Y, Wang ZY, Ma J, et al., 2013. Efficient revocation in ciphertext-policy attribute-based encryption based cryptographic cloud storage. *J Zhejiang Univ-Sci C (Comput & Electron)*, 14(2):85-97. https://doi.org/10.1631/jzus.C1200240

Cohen A, 2019. What about Bob? The inadequacy of CPA security for proxy reencryption. IACR Int Workshop on Public Key Cryptography, p.287-316. https://doi.org/10.1007/978-3-030-17259-6_10

David N, 2018. UMBRAL: a Threshold Proxy Re-encryption Scheme. https://github.com/nucypher/umbral-doc/blob/master/umbral-doc.pdf

Egorov M, Wilkison M, Nuñez D, 2017. NuCypher KMS: Decentralized Key Management System. https://arxiv.org/abs/1707.06140

Fuchsbauer G, Kamath C, Klein K, et al., 2019. Adaptively secure proxy re-encryption. IACR Int Workshop on Public Key Cryptography, p.317-346. https://doi.org/10.1007/978-3-030-17259-6_11

Gorbunov S, Vaikuntanathan V, Wichs D, 2015. Leveled fully homomorphic signatures from standard lattices. Proc 47th Annual ACM Symp on Theory of Computing, p.469-477. https://doi.org/10.1145/2746539.2746576

Green M, Ateniese G, 2007. Identity-based proxy re-encryption. Int Conf on Applied Cryptography and Network Security, p.288-306. https://doi.org/10.1007/978-3-540-72738-5_19

Kirshanova E, 2014. Proxy re-encryption from lattices. Int Workshop on Public Key Cryptography, p.77-94. https://doi.org/10.1007/978-3-642-54631-0_5

Li JY, Ma CG, Zhao Q, 2017. Resplittable threshold multi-broker proxy re-encryption scheme from lattices. *J Commun*, 38(5):157-164 (in Chinese). https://doi.org/10.11959/j.issn.1000-436x.2017109

Lindner R, Peikert C, 2011. Better key sizes (and attacks) for LWE-based encryption. Cryptographers' Track at the RSA Conf, p.319-339. https://doi.org/10.1007/978-3-642-19074-2_21

Lou SM, Cao ZF, 2010. Identity-based proxy re-encryption with threshold multi-proxy. *J Nat Sci Heilongjiang Univ*, 27(2):151-156 (in Chinese).

Micciancio D, Regev O, 2007. Worst-case to average-case reductions based on Gaussian measures. *SIAM J Comput*, 37(1):267-302. https://doi.org/10.1137/S0097539705447360

Nuñez D, Agudo I, Lopez J, 2015. NTRUReEncrypt: an efficient proxy re-encryption scheme based on NTRU. Proc 10th ACM Symp on Information, Computer and Communications Security, p.179-189. https://doi.org/10.1145/2714576.2714585

Polyakov Y, Rohloff K, Sahu G, et al., 2017. Fast proxy re-encryption for publish/subscribe systems. *ACM Trans Priv Secur*, 20(4):14.

https://doi.org/10.1145/3128607

Polyakov Y, Rohloff K, Ryan GW, 2018. PALISADE Lattice Cryptography Library User Manual v1.2.0.

Regev O, 2009. On lattices, learning with errors, random linear codes, and cryptography. *J ACM*, 56(6):34. https://doi.org/10.1145/1568318.1568324

Shor PW, 1997. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J Comput*, 26(5):1484-1509. https://doi.org/10.1137/S0097539795293172

Singh K, Rangan CP, Banerjee AK, 2014. Lattice based identity based unidirectional proxy re-encryption scheme. Int Conf on Security, Privacy, and Applied Cryptography Engineering, p.76-91. https://doi.org/10.1007/978-3-319-12060-7_6

Wang Z, Ma ZF, Luo SS, et al., 2019. Key escrow protocol based on a tripartite authenticated key agreement and threshold cryptography. *IEEE Access*, 7:149080-149096. https://doi.org/10.1109/ACCESS.2019.2946874

Xagawa K, 2010. Cryptography with Lattices. MS Thesis, Tokyo Institute of Technology, Tokyo, Japan.

Yin W, Wen QY, Li WM, et al., 2018. A new insight-proxy re-encryption under LWE with strong anti-collusion. Int Conf on Information Security Practice and Experience, p.559-577. https://doi.org/10.1007/978-3-319-99807-7_36