Achieving Multi-Hop PRE via Branching Program

Zengpeng Li, *Student Member, IEEE*, Chunguang Ma, and Ding Wang

Abstract—Proxy re-encryption (PRE) is a fundamental cryptographic primitive in secure data sharing and e-mail forwarding, etc. To our knowledge, most existing efficient lattice-based PRE schemes focus on the construction of single-hop, key-private, multi-bit and chosen-ciphertext attack (CCA), etc. Few works of literature discussed the detailed multi-hop construction over lattices. Very recently, Chandran et al. (PKC'14) proposed a lattice-based PRE scheme that builds upon the key switching mechanism of Brakerski (CRYPTO'12) and pointed out that their scheme can achieve multi-hop PRE scheme by the ideal circuit family for a directed graph G. In this paper, we are still working along this line and achieving multi-hop PRE via the branching program (BP) which is one type of NC1 circuit and can be used to compute encrypted data. To our knowledge, we proposed the first multi-hop PRE scheme via BP which supports homomorphic evaluation. We also analyze the security of our scheme under decisional learning with errors (LWE) assumption.

Index Terms—Multi-hop, proxy re-encryption, key switching mechanism, homomorphic evaluation, learning with errors

1 Introduction

The adventure of cloud computing has greatly changed the IT landscape, while the potential security risks have been long cited as obstacles to wider adoption of cloud computing. Hence, utilizing various cryptography approaches to protect the data security and privacy were proposed [1], etc. As an important cryptographic primitive, proxy re-encryption (PRE) was used to design various cryptosystems (e.g., secure file systems, secure e-mail forwarding, and encrypted spam filtering, etc) to protect users' privacy, e.g., [2], [3], [4], etc. In this case, a scenario should be considered. For instance, the proxy server wants to convert the ciphertext of the delegator to the ciphertext of the delegatee and hopes that the transformed ciphertext can be decrypted by the secret of the delegatee. We call this scenario as the the single-hop PRE (hereafter 1-hop PRE) scheme. We note that, Blaze et al. [5] first proposed the concept of PRE at EUROCRYPT'98. The authors used the ElGamal scheme [6] as the building block to obtain the chosen plaintext attacks (or CPA) against a secure PRE scheme under the decisional Diffie-Hellman (or DDH) assumption. Since then the cryptographers proposed various subsequent PRE schemes, improvements, and optimizations from DDH assumption (e.g., [7], [8], [9] etc.). However, it is a matter of common sense that the cryptosystems under DDH assumption cannot stay away from the quantum computer attacks. Obliviously, the post-quantum cryptography has been on the agenda for a while. Lattice-based cryptography has popularized rapidly in post-quantum cryptography as a sort of important tool for preventing quantum attacks. Hence, many cryptographers are working to construct the lattice-based cryptographic primitives. Obviously, it makes sense to construct the lattice-based PRE scheme with new properties. To our knowledge, the emphasis of lattice-based cryptography tends to be on learning with errors (LWE) assumption. Xagawa applied the LWE to show the possibility of PRE over lattices in his thesis [10], but the drawback is that we cannot find the complete security proof. Subsequently, the first key-private PRE scheme over lattices was constructed by Aono et al. [11] at INDOCRYPT'13, but the authors only achieved weakly key-private. The major breakthroughs in PRE over lattices arrived with the contributions of Chandran et al. [12] and Kirshanova [13]. In a short, no trusted authority is needed to generate the re-encryption key simultaneously in the PRE scheme of Kirshanova [13]. At one time, an obfuscator that used for designing various re-encryption schemes (i.e., re-encryption, functional re-encryption, and multi-hop re-encryption) without going through fully homomorphic encryption (FHE) was proposed by Chandran et al. [12] under the decisional LWE assumption. Very recently, Nishimaki and Xagawa [14] used the re-encryption techniques of FHE [15] to design two types of PRE schemes with key-private property over lattices, where one PRE scheme is based on Regev scheme [16] and the other one is based on Linder-Peikert scheme [17]. Notably, we easily observe that most existing lattice-based PRE schemes focus on single-hop re-encryption construction while multi-hop PRE (hereafter L-hop PRE) scheme from lattice is more practical. Fortunately, Chandran et al. [12] have pointed out that their scheme can be transformed from 1-hop PRE to L-hop PRE via the ideal circuit family for a directed graph G, however, we cannot find a detailed construction. Hence, we are still working along this line in

- *Z. Li and C. Ma are with the College of Computer Sciences and Technology, Harbin Engineering University, Harbin 150001, P.R. China, and with the State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100864, China. E-mail: {lizengpeng, machungaung}@hrbeu.edu.cn.*
- *D. Wang is with the School of Electronics Engineering and Computer Science, Peking University, Beijing 100871, P.R. China. E-mail: wangdingg@pku.edu.cn.*

Manuscript received 4 Apr. 2017; revised 10 Aug. 2017; accepted 27 Aug. 2017. Date of publication 0 . 0000; date of current version 0 . 0000.
(Corresponding author: Ding Wang.)
Recommended for acceptance by Y. Cui.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TCC.2017.2764082

this paper to try to construct the multi-hop PRE scheme. As we know, by Barrington's theorem, the polynomial-size branching program (BP) is one type of NC1 circuit. In a word, the encrypted data [18] can be computed by BP or the decryption circuit can be represent by BP. Moreover, the Barrington's theorem implies that the majority function can be computed by a family of BP of constant width and polynomial size. However, the other types of NC1 circuit without those features cannot be adopted to compute the encrypted data. That's the advantage of BP to construct multi-hop PRE scheme. Hence, this situation raises a question:

Can we achieve multi-hop PRE *scheme via* BP*?*

As we know, one can convert any NC1 circuit into a special BP which contains polynomial-length 4^d for circuit depth d and width 5. Hence, to solve this issue, we can attain a multi-hop PRE scheme by using the BP. Concretely, armed with the homomorphic PRE scheme of Ma et al. [19], we first optimize the dimension of ciphertexts by using the method of Brakerski et al. [20] and get a short ciphertext. Then the short ciphertext and the re-encryption key (that obtained from the optimized key switching procedure) were implanted into NAND circuit. Lastly, we are going to transform the NAND circuit into BP (on the fly) [21], then evaluate BP homomorphically and output the L-hop re-encryption ciphertexts. Below, we sketch our techniques and provide a more detailed construction in the following Section 3.1.

1.1 Technical Overview

PRE is an important cryptographic primitive which can be used in various information security systems. In order to capture the actual condition, the multi-hop PRE came into being. We observe that 1-hop PRE can be transformed to L-hop PRE by the ideal circuit, which has been pointed out by Chandran et al. [12]. BP is one type of NC1 circuit, which can be adopted to deal with the encrypted data [20], [22], etc. Thus, in this paper, we construct a BP-based multi-hop PRE scheme, that's our main contribution.

We note that, NAND gate is the core of BP. In lattice setting, one uses the form of $\mathbf{X} \cdot \mathbf{G}^{-1}(\mathbf{Y})$ to express the NAND gate for two assigned matrix $\mathbf{X}$ and $\mathbf{Y}$. Importantly, with this structure, the re-encryption algorithm $\mathbf{C}_j := \mathbf{M} \cdot \mathbf{G}^{-1}(\mathbf{C}_i)$ of PRE can be achieved for the input-side ciphertext $\mathbf{C}_i$ and the re-encryption key $\mathbf{M}$. In the following, we just make an informal explanation for our technical ideas and omit the concrete dimension of matrix and vector.

- In this paper, we first give a much simpler key-switching procedure description via gadget matrix in Section 3.1, then utilize the optimized key switching procedure to restate the scheme of Chandran et al. [12] in Section 3.2.
- Then, in order to make the PRE scheme support homomorphic operation, we transform the encryption algorithm of optimized PRE scheme into fully homomorphic algorithm, namely that the form of the ciphertext $\mathbf{c} = \mathbf{A}^T\mathbf{r} + \lfloor \frac{q}{2} \rfloor \mathbf{u} \pmod q \in \mathbb{Z}_q^{(n+1)\times 1}$ was transformed into the form of $\mathbf{c} = \mathbf{A}^T \cdot \mathbf{r} + 2^{\ell_q - 1}\mu\mathbf{u}_1$ for $\mathbf{u}_1 := (1, 0, \ldots, 0)$. We stress that, if we adopt the original Gentry-Sahai-Vaikuntanathan (GSW [23]) style ciphertext $\mathbf{C} := \mathbf{A}^T \cdot \mathbf{R} + \mu \cdot \mathbf{G} \pmod q \in \mathbb{Z}_q^{(n+1)\times(n+1)\ell_q}$ for gadget matrix $\mathbf{G} \in \mathbb{Z}_q^{(n+1)\times(n+1)\ell_q}$ and random matrix $\mathbf{R} \in \{0,1\}^{m\times(n+1)\ell_q}$, we easily find that the encryption of GSW style lets the dimension of ciphertext scale up, and leads to the performance degrading. Hence, in this paper, inspired by the FHE with short ciphertext of Brakerski and Perlman [20], we adopt their method to scale down the ciphertext dimension, i.e., we develop the homomorphic PRE with short ciphertext $\mathbf{c} = \mathbf{A}^T \cdot \mathbf{r} + 2^{\ell_q - 1}\mu\mathbf{u}_1$ for $\mathbf{u}_1 := (1, 0, \ldots, 0)$.
- Lastly, in order to realize L-hop PRE, in lattice setting, the re-encryption key and the ciphertext are assembled into NAND gate circuit by following the same method with [11], [12], [14].[1] Next, we give a way how to assemble the NAND gates into BP. In particular, the BP starts to work at node $(1, 1)$ and ends at node (i, L), namely that $(1,1) :\xrightarrow{\mathbf{x}} (i, L)$, where a boolean permutation was denoted by $\mathbf{x} \in \{0,1\}^{L-1}$. We note that, BP contains a total of $(L + 1) \cdot w$ nodes for length L and width w. Each node (i, t) has two edges (or fan-outs) $p_{0,t}$ and $p_{1,t}$. Moreover, we associate each edge with a node at next level.[2] Thus, in this setting, as for the initial ciphertext $\mathbf{c}$, we associate it with the BP's node $(1, 1)$, and as for the re-encryption key $\mathbf{X}_{var(t)}$, we associate $\mathbf{X}_{var(t)}$ and $\mathbf{G} - \mathbf{X}_{var(t)}$ with the edges, then one can compute the BP on a permutation $rk = \{\mathbf{X}_{var(t)}, \mathbf{G} - \mathbf{X}_{var(t)}\}^L$ (a.k.a., a series of re-encryption keys) as input, i.e., $(1, 1) :\xrightarrow{rk} (1, L)$. In a nutshell, we first decompose the secret key into bits by the bit-decompose function Bit, we then encrypted each bit of secret key and introduced the Extend algorithm to assemble all of these encrypted bits into rk.
- Furthermore, we extend our multi-hop PRE via branching program to multi-bit and multi-hop PRE via branching program. In order to encrypt t-bit at one-time, hence, we use t branching programs BP to compute each bit simultaneously, i.e., $\mathsf{BP} = (\mathsf{BP}^{(1)}, \ldots, \mathsf{BP}^{(t)})$.

1.2 Related Work and Application

The construction of Blaze et al. [5] is the first bidirectional PRE scheme by adopting ElGamal scheme [6]. More concretely, for a generator g of group $\mathbb{G}$ and a prime order p of the group, the delegator and delegatee sample their own key pair (a, g^a) and (b, g^b) respectively, $a, b \leftarrow \mathbb{Z}_p$. The delegator encrypts a message m and generates the ciphertext which is the form of $c = (c_1, c_2) = (m \cdot g^r, (g^a)^r)$ for a randomly chosen $r \leftarrow Z_p$. The proxy converts the ciphertext c to the ciphertext c' of delegatee by computing $c' = (c_1, c_2^{b/a}) = (mg^r, (g^b)^r)$, where the re-encryption key from the delegator to the delegatee is $rk_{A\to B} = b/a$. We easily observe the special structure of rk so that the proxy can easily get a/b and computes the ciphertexts from the delegatee to the delegator. In reality, a re-encryption key works

1. I.e., the re-encryption ciphertext is $\mathbf{X} \cdot \mathbf{G}^{-1}(\mathbf{c})$ along with the input-side ciphertext $\mathbf{c}$ and the re-encryption key $\mathbf{X}$.

2. In simple terms, at level $i + 1$, the edge $p_{0,t}$ is associated with $(i + 1, p_{0,t}(j))$ for $t \in [L]$ and $i \in [w]$. Similarly, $p_{1,t}$ is associated with $(i + 1, p_{1,t}(j))$.

TABLE 1
Comparison of LWE-Based Uni-Direction PRE and Our Schemes

Scheme	Homomorphism	Security	Hop	Msg
Kirshanova [13]	×	CCA	Single	Multi
Fan et al. [26]	×	CCA	Multi	Multi
Xagawa [10]	×	NONE	Single	Multi
Aono et al. [11]	×	CPA	Single	Multi
Nishimaki et al. [14]	×	CPA	Single	Multi
Ma et al. [19]	✓	CPA	Single	Multi
Chandran et al. [12]	×	CPA	Multi	Multi
Our scheme	✓	CPA	Multi	Single/Multi

only in one direction that is more desirable in practice. Working along this line, many follow-up works to optimize and improve the performance were proposed [7], [8], [9], [24], [25]. However, most previous PRE schemes were designed by DDH assumption. In order to prevent quantum attacks, lattice-based PRE schemes were proposed, a comparison results of lattice-based PRE schemes was provided in the following Table 1.

As is shown by the Table 1, most of these schemes are only support single-hop except [12], [26]. Moreover, all of them must encrypt multi-bit by encrypting vector directly and they didn't explain how to achieve single-bit PRE. In this case, our scheme supports single-bit and single-hop PRE, single-bit and multi-hop PRE, and multi-bit and multi-hop PRE. Importantly, our scheme supports homomorphic evaluation.

FHE is an important tool for computing the encrypted data [15] homomorphically and has many potential application prospects in various aspects of cloud computing. Concretely, two types of PRE schemes were proposed by Chandran et al. [12], the authors used Regev scheme [16] and Gentry-Peikert-Vaikuntanathan (GPV, also called dual Regev) scheme [27] respectively, which corresponds to Regev style key switching and GPV style key switching separately. However, Nishimaki et al. [14] used Regev style key switching to construct two types of key-private PRE which are based on Regev [16] scheme and Lindner and Peikert (LP) [17] scheme respectively.

Chandran et al. [12] pointed out that we can obtain a multi-hop PRE scheme by the ideal circuit, but the scheme only supports limited hops. Another multi-hop PRE scheme was proposed by Fan et al. [26], but they adopted the complicated Micciancio-Peikert scheme [28] as the building block. The other schemes focus on the single-hop schemes and obtain multi-hop by iterating the single-hop scheme.

1.3 Paper Organization

In Section 2, we present some related notations. In Section 3, we describe and optimize the building blocks. Finally, our multi-hop homomorphic PRE construction was presented in Section 4.

2 Preliminaries

Vector $\mathbf{x}$ is denoted by the bold lower-case letter and matrix $\mathbf{A}$ is denoted by the bold upper-case letter. The symbol ":=" implies deterministic assignment. Considering the truncated discrete Gaussian distribution $\mathcal{D}_{\mathbb{Z}_q^m,\sigma}$, we denote $\mathcal{D}_{\mathbb{Z}_q^m,\sigma}$ over $\mathbb{Z}^m$ with parameter σ. Note that $\mathcal{D}_{\mathbb{Z}^m,\sigma}$ is $\sqrt{m}\cdot\sigma$-bounded.

Definition 2.1 ([29]). *If the inequality* $\Pr_{x\xleftarrow{\$}\chi}[|x|\geq B]\leq 2^{-\tilde{\Omega}(n)}$ *holds, then the B-bounded distribution ensemble over integers is $\chi=\chi(\lambda)$ (i.e., $|\chi|\leq B$).*

2.1 Learning with Errors and Lattice Background

Lemma 2.2 ([30] Lemma 2.1). *If the parameters $\lambda\in\mathbb{Z}$, n, $q\in\mathbb{N}$, $m\geq n\log q+2\lambda$ and vectors $\mathbf{r}\xleftarrow{R}\{0,1\}^m$ and $\mathbf{y}\xleftarrow{R}\mathbb{Z}_q^n$, $\mathbf{A}\xleftarrow{R}\mathbb{Z}_q^{m\times n}$ are hold, then the statistical distance is*

$$\Delta\big((\mathbf{A},\mathbf{A}^T\cdot\mathbf{r}),(\mathbf{A},\mathbf{y})\big)\leq 2^{-\lambda}, \tag{2.1}$$

for the distributions $(\mathbf{A},\mathbf{A}^T\mathbf{r})$ and $(\mathbf{A},\mathbf{y})$.

Lemma 2.3 ([31] Lemma 4.4).

1) *For $\forall\ k>0, \Pr[|e|>k\cdot\sigma, e\leftarrow\mathcal{D}_\sigma^1]\leq 2\cdot\exp(-\frac{k^2}{2})$;*
2) *For $\forall\ k>0$, $\Pr[\|\mathbf{e}\|>k\cdot\sigma\cdot\sqrt{m},\mathbf{e}\leftarrow\mathcal{D}_\sigma^m]\leq k^m\cdot\exp(\frac{m}{2}\cdot(1-k^2))$.*

Remark 2.4. We suppose $\sigma\geq 2\sqrt{n}$ throughout the paper. Thus, if $\mathbf{e}\leftarrow\mathcal{D}_\sigma^m$ then we have, on average, that $\|\mathbf{e}\|\approx\sqrt{m}\cdot\sigma$. Lemma 2.3 (2) implies $\|\mathbf{e}\|\leq 2\sigma\sqrt{m}$ with overwhelming probability.

Definition 2.5 (Decision-LWE$_{n,q,\chi,m}$). *An independent sample $(\mathbf{A},\mathbf{b})\in\mathbb{Z}_q^{m\times n}\times\mathbb{Z}_q^{m\times 1}$ is distributed according to either: (1) the uniform distribution (i.e., $\{(\mathbf{A},\mathbf{b}):\mathbf{A}\leftarrow\mathbb{Z}_q^{m\times n},\mathbf{b}\leftarrow\mathbb{Z}_q^{m\times 1}\}$), or (2) $\mathcal{A}_{\mathbf{s},\chi}$ for a uniform random $\mathbf{s}\in\mathbb{Z}_q^n$ (i.e., $\{(\mathbf{A},\mathbf{b}):\mathbf{A}\leftarrow\mathbb{Z}_q^{m\times n},\mathbf{s}\leftarrow\mathbb{Z}_q^{n\times 1},\mathbf{e}\leftarrow\chi^{m\times 1},\mathbf{b}=\mathbf{A}\cdot\mathbf{s}+\mathbf{e}\pmod q\}$). Then, the above two distributions are computationally indistinguishable.*

2.2 Basic Tools

The following basic tools were proposed by Brakerski et al. [29], [32]. First we fix the following parameters $q,m\in\mathbb{N}$ and let $\ell_q=\lfloor\log q\rfloor+1$ and $N=m\cdot\ell_q$.

Definition 2.6. *The algorithm* PowerOf2 *takes as input an m-dimension random vector $\mathbf{v}\in\mathbb{Z}_q^m$, then outputs an N-dimension vector*

$$\big(v_1,2v_1,\ldots,2^{\ell_q-1}v_1,\ldots,v_m,2v_m,\ldots,2^{\ell_q-1}v_m\big)\in\mathbb{Z}_q^N.$$

Definition 2.7. *The (bit-decomposition) algorithm* Bit *takes a vector $\mathbf{v}\in\mathbb{Z}_q^m$ as input, then outputs an N-dimension vector*

$$(v_{1,0},\ldots,v_{1,\ell_q-1},\ldots,v_{m,0},\ldots,v_{m,\ell_q-1})\in\mathbb{Z}_q^N,$$

where $v_{i,j}$ is the jth bit in v_i's binary representation. In other words, $v_i=\sum_{j=0}^{\ell_q-1}2^j v_{i,j}$.

Definition 2.8. *The algorithm* Bit^{-1} *takes as input a vector (needs not be binary)*

$$\mathbf{v}=(v_{1,0},\ldots,v_{1,\ell-1},\ldots,v_{m,0},\ldots,v_{m,\ell-1})^T\in\mathbb{Z}_q^N,$$

and outputs the vector

$$\left(\sum_{j=0}^{\ell-1}2^j\cdot v_{1,j},\ldots,\sum_{j=0}^{\ell-1}2^j\cdot v_{m,j}\right)^T\in\mathbb{Z}_q^m.$$

Lemma 2.9 ([28]). *If there exists a computable gadget matrix* $\mathbf{G}$ *that proposed by Micciancio and Peikert [28], then there exists a computable deterministic inverse function (a.k.a., "short pre-image")* $\mathbf{G}^{-1}(\cdot)$ *along with* $\mathbf{G}$*, where* $\mathbf{G} = \mathbf{I}_m \otimes \mathbf{g} \in \mathbb{Z}_q^{m \times N}$ *for any* $N \geq m\lceil \log q \rceil$ *and* $\mathbf{g} = (1, 2, 4, \ldots, 2^{\ell-1})^T$*. Meanwhile, the inverse function* $\mathbf{G}^{-1}(\mathbf{M})$ *takes as input a matrix* $\mathbf{M} \in \mathbb{Z}_q^{m \times m'}$ *for any* m' *and outputs* $\mathbf{G}^{-1}(\mathbf{M}) \in \{0,1\}^{N \times m'}$ *such that there exists* $\mathbf{G}\mathbf{G}^{-1}(\mathbf{M}) = \mathbf{M}$*.*

Remark 2.10. Here we remark that, considering the vector $\mathbf{v} \in \mathbb{Z}_q^m$ there exists $\mathsf{PowerOf2}(\mathbf{v}) = \mathbf{v}^T\mathbf{G}$. For the vector $\mathbf{v} \in \mathbb{Z}_q^N$ we consider $\mathsf{Bit}^{-1}(\mathbf{v}) = \mathbf{G}\mathbf{v}$. As for the vector $\mathbf{a} \in \mathbb{Z}_q^m$, we can rename the algorithm $\mathsf{Bit}(\mathbf{a})$ as $\mathbf{G}^{-1}(\mathbf{a})$.

2.3 Branching Program

In this section, we give a brief description of the computational model of permutation BP. The definition of BP is similar to [20], [22].

Definition 2.11. *The instruction* $(p_{0,t}, p_{1,t})_{t\in[L]}$ *is denoted by a permutation* BP $\prod$ *along with* ℓ *variables, width* k*, and length* L*. Considering a function* $var : [L] \to [\ell]$*, each tuple* $(p_{0,t}, p_{1,t})_{t\in[L]}$ *is composed of a pair of permutations* $p_{0,t}$*,* $p_{1,t}$*:* $[k] \to [k]$*. The* BP *takes a binary vector* $\mathbf{x} = (x_1, \ldots, x_\ell) \in (0,1)^\ell$ *as input, and outputs a random bit* $b \in \{0,1\}$*. The execution of* $\prod$ *is as follows:*

- *Keeps a state integer* $s \in \{0, \ldots, k\}$ *and initializes* $s_0 = 1$*;*
- *The* tth *instruction* $s_t := p_{x_{var(t)}}(s_{t-1})$ *recursively determines the next state for every step* $t = 1, \ldots, L$*.*

In brief, if $x_{var(t)} = 0$ *then* $s_t := p_{0,t}(s_{t-1})$*, and otherwise* $s_t := p_{1,t}(s_{t-1})$*. Finally, after the* Lth *iteration, the results of* BP *is 1 if and only if* $s_L = 1$*.*

Remark 2.12. We remark that, if ones wants to homomorphically evaluate a BP then an integer state $s \in \{1,2,3,4,5\}$ should be represented by a 0-1 state vector $\mathbf{v} \in \{0,1\}^5$. More concretely, the key point is as follows identical equation $\mathbf{v}_t[i] = 1 \Leftrightarrow s_t = i$. Thus, we first initialize $\mathbf{v}_0[1] = 1$ and $\mathbf{v}_0[i] = 0$ for $i \in \{2,3,4,5\}$ and evaluate BP according to the following recursive formula: $\mathbf{v}_t[i] = 1 \Leftrightarrow p_{t,x_{var(t)}}(s_{t-1}) = i$. Therefore, for every $1 \leq i \leq 5$, there exists an identical equation $\mathbf{v}_t[i] = 1$ if and only if we have

- either $\mathbf{v}_{t-1}[p_{t,0}^{-1}(i)] = 1$ and $x_{var(t)} = 0$
- or $\mathbf{v}_{t-1}[p_{t,1}^{-1}(i)] = 1$ and $x_{var(t)} = 1$.

Hence, the following formula holds equivalently

$$\begin{aligned}\mathbf{v}_t[i] &= \mathbf{v}_{t-1}[p_{t,0}^{-1}(i)](1 - x_{vat(t)}) + \mathbf{v}_{t-1}[p_{t,1}^{-1}(i)]x_{vat(t)} \\ &= \mathbf{v}_{t-1}[\gamma_{t,i,0}](1 - x_{vat(t)}) + \mathbf{v}_{t-1}[\gamma_{t,i,1}]x_{vat(t)},\end{aligned} \tag{2.2}$$

where $\gamma_{t,i,0} \stackrel{\Delta}{=} p_{t,0}^{-1}(i)$ and $\gamma_{t,i,1} \stackrel{\Delta}{=} p_{t,1}^{-1}(i)$ are constant, public, and computable according to the above BP's description. After the L times iteration, if and only if $s_L = 1$ can be accepted by us, namely that we obtain $\mathbf{v}_L[1]$. We stress that, the following our homomorphic evaluations are inherently the above form of BP.

Below, we give the theorem of Barrington from [21].

Theorem 2.13 (Barrington's Theorem [21]). *If one obtains the explanation of the circuit* C *and* BP*, then he can compute* $\prod$ *in* $\mathrm{poly}(\ell, 4^d)$*. Meanwhile, every boolean* NAND *circuit* C *has* ℓ *inputs and depth* d*, which can be computed by a permutation* BP *with length-*4^d *and width-5.*

2.4 Predecessor Function for Circuit

In this section, we adopt the definition and corollary from Brakerski et al. [20].

Definition 2.14 (Predecessor Function for Circuit, $\mathsf{Pred}_\Psi(i)$). *The predecessor function of* Ψ *is defined as* $\mathsf{Pred}_\Psi(i)$ *where the index of length is* i *and a circuit is* Ψ *as in Barington Theorem. Then the output gate of* $\mathsf{Pred}_\Psi(i)$ *is labelled by 0 and the input gate of* $\mathsf{Pred}_\Psi(i)$ *is labelled by the index of the variable.*

Corollary 2.15 (Barrington's Theorem On-The-Fly, BPOTF). *If given access to the program* $\mathsf{Pred}_\Psi(i)$*, then there exists a uniform machine* BPOTF *outputing the layers* $(p_{0,t}, p_{1,t})$ *of* BP *by using the Theorem 2.13 for* $t \in [L]$ *width of* BPOTF*. We remark that, each layer needs time* $O(d)$*, and* BPOTF *uses total* $O(d)$ *space.*

2.5 Homomorphic Proxy Re-Encryption (PRE)

We remark that, in order to make reading easier, we start by recalling the definition of FHE from [23], [30].

Definition 2.16. $L = L(\lambda)$ *is consider as a level function. An* L*-*FHE *scheme for a class of circuits* $\{\mathcal{C}_\lambda\}_{\lambda\in N}$ *contains four probabilistic polynomial time (*PPT*) algorithms* {KeyGen, Enc, Dec, Eval} *such that:*

- KeyGen *algorithm takes the security parameter* 1^λ *as input and generates a public key* pk *and a secret key* sk*;*
- Enc *algorithm takes as input a message* $m \in \{0,1\}^*$ *along with* pk *and generates a ciphertext* c*;*
- Dec *algorithm takes as input a* c*, and outputs* $m \in \{0,1\}^*$ *under the* sk*;*
- Eval *algorithm takes a* pk*, a sequence of* $c_1, \ldots, c_\ell$*, and a circuit* $C \in \mathcal{C}_\lambda$ *as input, then it generates a evaluated ciphertext* c^**. Notably,* ℓ *is a polynomial over* λ*.*

The two important properties are required:

- *Correctness.* The correctness holds if

$$m = \mathsf{Dec}\Big(sk, \big(\mathsf{Enc}(pk, m)\big)\Big);$$

for any λ, $m \in \{0,1\}^*$, and (pk, sk) is generated by $\mathsf{KeyGen}(1^\lambda)$.

- *Homomorphisms.* The following equation holds

$$\mathcal{C}(m_1, \ldots, m_\ell) = \mathsf{Dec}\Big(sk, \big(\mathsf{Eval}\big(pk, (C, \mathsf{Enc}(pk, m_1), \ldots, \mathsf{Enc}(pk, m_\ell))\big)\big)\Big).$$

for any λ, any $m_1, \ldots, m_\ell \in \{0,1\}^*$, and $C \in \mathcal{C}_\lambda$.

Definition 2.17 (Security). *If the following equation is negligible in* λ *for any* PPT *adversary* $\mathcal{A}$

$$|\Pr[\mathcal{A}(pk, \mathsf{Enc}(pk, m_0)) = 1] - \Pr[\mathcal{A}(pk, \mathsf{Enc}(pk, m_1)) = 1]|,$$

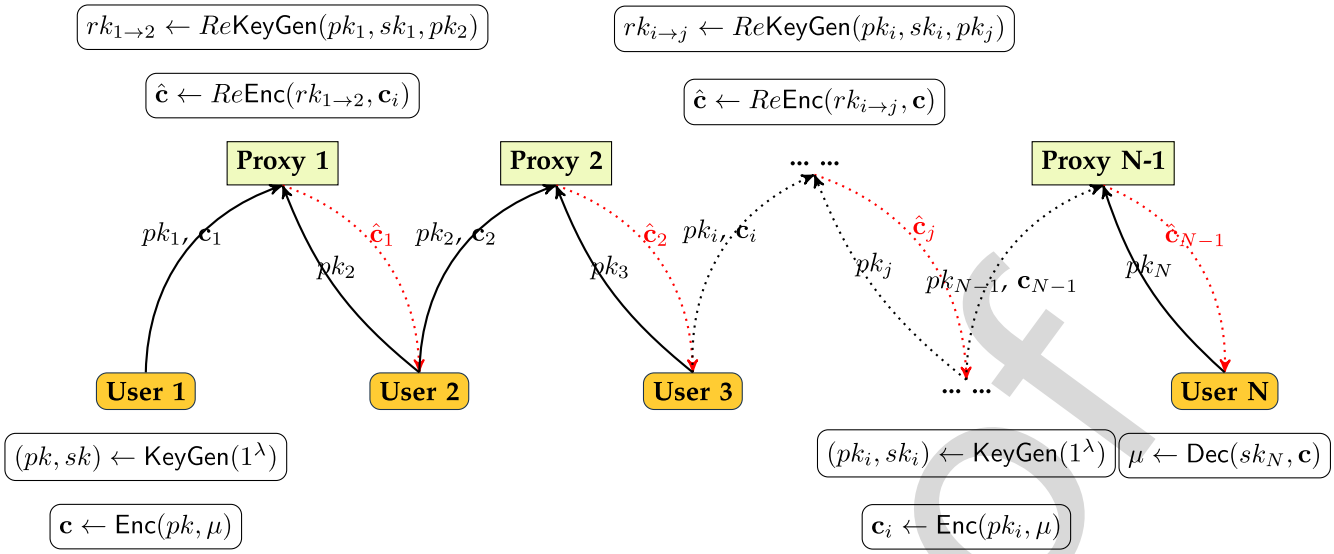


Fig. 1. Multi-hop proxy re-encryption scheme.

then the FHE *scheme is indistinguishability against* CPA *(a.k.a., IND-CPA-secure), where m_0, m_1 are chosen randomly from the plaintext space by $\mathcal{A}$ and $(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$.*

Definition 2.18 (Compactness). *If the evaluated ciphertext is generated by the* Eval *algorithm which has length at most $\alpha = \alpha(\lambda)$, then the L-*FHE *scheme is* compact *for a class of circuits $\{\mathbb{C}_k\}_{k\in\mathbb{N}}$.*

Below, we use the same syntax and security definition of PRE as Canetti et al. [7] while the uni-directional PRE can be regarded as an extension of bi-directional PRE [7].

Syntax. The uni-directional PRE contains a tuple of PPT algorithms

$$\mathsf{PRE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{ReKeyGen}, \mathsf{ReEnc}, \mathsf{Dec}),$$

as follows:

- $\mathsf{params} \leftarrow \mathsf{Setup}(1^\lambda)$: Takes λ as input and generates the public parameter params;
- $(pk, sk) \leftarrow \mathsf{KeyGen}(\mathsf{params})$: Takes as input params and generates a pk and a sk;
- $\mathbf{c} \leftarrow \mathsf{Enc}(pk, \mu)$: Takes a message $\mu \in \{0,1\}$ and pk as input, then outputs the ciphertexts $\mathbf{c}$;
- $rk \leftarrow \mathsf{ReKeyGen}(pk_i, sk_i, pk_j)$: Takes a key pair (pk_i, sk_i) from the input side player i and a public key pk_j from the output side player j, then outputs the re-encryption key $rk^{(i\to j)}$ from player i to player j, when the context clear, we omit the superscript and write it as rk;
- $\mathbf{c}_j \leftarrow \mathsf{ReEnc}(rk, \mathbf{c}_i)$: Takes rk and $\mathbf{c}_i$ from player i as input, and outputs re-encryption ciphertexts $\mathbf{c}_j$ for player j;
- $\mu \leftarrow \mathsf{Dec}(\mathbf{c}, sk)$: Takes $\mathbf{c}$ and sk as input and outputs a message μ or an error symbol $\perp$.

Proposition 2.19 (Single/Multi-Hop PRE**).** *The number of the* PRE *hops from the input-side to the output-side is defined as $L := j - i$, which means the proxy server of* PRE *is only allowed to re-encrypt the ciphertexts L times. Apparently, the single-hop* PRE *scheme is denoted by $L = 1$. If $L > 1$, then the scheme is L-hop* PRE *(a.k.a., multi-hop* PRE*).*

Remark 2.20. We remark that, we use the Fig. 1 to explain the multi-hop PRE scheme.

Definition 2.21 (Correctness). *There exist two cases.*

- *The input-side decryption, the delegator computes the results of $\mathsf{Dec}(sk, \mathsf{Enc}(pk, \mu))$ with overwhelming probability.*
- *The output-side decryption, i.e., after L-hop re-encryption from player $i = 1$ to player $j := i + L$, the delegator computes and obtains the outputs of $\mathsf{Dec}(sk_L, \mathsf{ReEnc}(rk^{(L-1\to L)}, \ldots, \mathsf{ReEnc}(rk^{(1\to 2)}, \mathbf{c}_1)))$ with overwhelming probability for the re-encryption key $rk^{(i\to j)}$ from i to j.*

Definition 2.22 (Security). *We denote the following security game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$. Notably, $\mathcal{A}$ is eligible for choosing all sort of queries, such as the public key queries, re-encryption key queries, and ciphertext queries.*

1) *At first, $\mathcal{C}$ runs $\mathsf{Setup}(1^\lambda) \to (pk^*, sk^*)$ and adds (pk^*, sk^*) to the honest set Γ_H, then sends pk^* to $\mathcal{A}$;*
2) *Upon receiving pk^*, $\mathcal{A}$ makes the public key queries, there exist two cases: 1). if $\mathcal{A}$ un-corrupts key-generation oracle, then sends pk' to $\mathcal{A}$ and adds index i to Γ_H; 2). if $\mathcal{A}$ corrupts key-generation oracle, then sends (pk', sk') to $\mathcal{A}$ and adds index i to the corrupted set Γ_C;*
3) *The re-encryption key queries are issued by $\mathcal{A}$ for pk_i from player i and pk_j from player j where $i \neq j$, then $rk_{(i\to j)}$ is obtained by $\mathcal{C}$ by $\mathsf{ReKeyGen}(mpk, (pk_i, sk_i), pk_j)$ and $\mathcal{C}$ sends back $rk_{(i\to j)}$ to $\mathcal{A}$; (Notably, the re-encryption key generation queries were not permitted to occur between an uncorrupted party and a corrupted, i.e., they are corrupted parties, or alternatively both are honest parties.)*
4) *The query $(i, j, \mathbf{c}_i)$ is submitted by $\mathcal{A}$, then $\mathcal{C}$ in turn generates $\mathbf{c}_j \leftarrow \mathsf{ReEnc}(pk_i, pk_j, \mathbf{c}_i)$ and sends $\mathbf{c}_j$ back to $\mathcal{A}$.*
5) *$\mathcal{A}$ submits query (i^*, μ_0, μ_1), then $\mathcal{C}$ in turn samples a bit $b \in \{0,1\}$ and returns $\mathbf{c}_{i^*} \leftarrow \mathsf{Enc}(pk_{i^*}, \mu_b)$ to $\mathcal{A}$. Importantly, the oracle of decryption is only allowed to query once.*

$\mathcal{A}$ outputs $b' \in \{0,1\}$ and denotes the advantage of $\mathcal{A}$ as $\mathbf{Adv} = |\Pr[b'-b] - \frac{1}{2}|$. *If* **Adv** *is negligible for all* PPT $\mathcal{A}$ *then the* PRE *scheme is IND-CPA-secure .*

From the above definition of PRE, we can sketch that once the delegatee and the delegator are chosen, the proxy server can re-encrypt the original ciphertext which is from the delegator, then send the ciphertext to the delegatee. Finally, the delegatee decrypts the re-encryption ciphertext with his own secret key. That's the simple scenario about the single-hop PRE scheme, notably, the proxy doesn't do any evaluation except re-encryption. If the proxy has more ciphertexts from multiple delegators and needs do some computation on these ciphertexts without leaking the information of these ciphertexts, the proxy needs to support homomorphic evaluation. To address this issue, the concept of homomorphic PRE has been proposed [19], [33].

Compared with original single-hop PRE, the following definition of the homomorphic PRE is identical to the general PRE scheme except that the homomorphic evaluation can be supported by our homomorphic PRE scheme.

Definition 2.23 (Homomorphic PRE, From [33]). *The* homomorphic PRE *scheme contains the same algorithms of* FHE *scheme and also has re-key generation and re-encryption algorithms as follows:*

- $rk \leftarrow \mathsf{ReKeyGen}(pk_i, sk_i, pk_j)$: *Takes a key pair (pk_i, sk_i) from input side player i and a public key pk_j from output side player j, outputs the re-encryption key $rk^{(i\to j)}$ from i to j*
- $\mathbf{c}_j \leftarrow \mathsf{ReEnc}(rk, \mathbf{c}_i)$: *Takes rk and $\mathbf{c}_i$ from player i as input, then, outputs re-encryption ciphertexts $\mathbf{c}_j$ for player j;*

meanwhile, the scheme has the following properties,

- Correctness of Re-Encrypted Ciphertexts:

$$m := \mathsf{Dec}\Big(sk_j, \mathsf{ReEnc}\big(rk, \mathsf{Enc}(pk_i, m)\big)\Big).$$

The above equation holds with overwhelming probability.

- Homomorphisms: *The homomorphic property holds if the homomorphic operations for the algorithm* Enc *and* ReEnc *are under the same public key. More concretely, we have that*

$$\mathcal{C}(m_1,\ldots,m_\ell) = \mathsf{Dec}\Big(sk, \big(\mathsf{Eval}\big(pk, (C, \mathsf{ReEnc}(pk, m_1), \ldots, \mathsf{ReEnc}(pk, m_\ell))\big)\big)\Big).$$

for any λ, any $m_1,\ldots,m_\ell \in \{0,1\}^*$, and $C \in \mathcal{C}_\lambda$.

- Security: *The security property of homomorphic* PRE *is denoted by the IND-CPA game which is similar to the security property of general* PRE. *So the detailed of the IND-CPA game is omitted. Thus, if the* PPT *adversary $\mathcal{A}$'s advantage is negligible in λ, i.e., $\mathbf{Adv}_{\mathcal{A}} = |\Pr[b'=b] - \frac{1}{2}| \le \lambda$, the homomorphic* PRE *scheme is IND-CPA-secure.*

3 Optimized Single-Hop PRE Scheme

We note that, the previous key switching mechanisms [23], [30] were expressed in the function Bit and Bit^{-1}. Fortunately, we remark that, the key switching mechanism can be denoted by the language of $\mathbf{G}$ and $\mathbf{G}^{-1}$ in Remark 2.12. Hence, we first give a much simpler key-switching procedure description via gadget matrix in Section 3.1, then utilize the improved key switching procedure to restate the scheme of Chandran et al. [12] in Section 3.2.

3.1 Improved Key Switching

In this section, we first review the Regev scheme [16] before discussing our optimized key switching mechanism. The key generation algorithm of Regev scheme generates a public key $\mathbf{A} := (\mathbf{b}, \mathbf{B}) \in \mathbb{Z}_q^{m\times 1} \times \mathbb{Z}_q^{m\times n}$ and a secret key $\mathbf{t} \leftarrow \mathbb{Z}_q^{n\times 1}$, where we have $\mathbf{b} = \mathbf{B}\cdot\mathbf{t} + \mathbf{e} \pmod q$ for $\mathbf{e} \leftarrow \chi^{m\times 1}$ and the public matrix $\mathbf{B} \in \mathbb{Z}_q^{m\times n}$. Importantly, the encrypter computes $\mathbf{c} = \mathbf{A}^T\cdot\mathbf{r} + \frac{q}{2}(\mu|\mathbf{0}^n)^T \pmod q$ to encrypt $\mu \in \{0,1\}$, where the randomness vector is $\mathbf{r} \leftarrow \mathbb{Z}_q^{(n+1)\times 1}$. Once obtained the ciphertext vector $\mathbf{c}$, the decrypter recovers the message μ by computing $\langle \mathbf{c}, [1, -\mathbf{t}]^T\rangle$. The detailed construction of the important variant of key switching technique is as follows.

- $\mathcal{K}_{sk_I:sk_O} \leftarrow \mathsf{SwitchKeyGen}(sk_I, sk_O)$:
  1) For the following "input/output" secret key $sk_I = [1, -\mathbf{t}_I^T]^T \in \mathbb{Z}_q^{n_I\times 1}$ and $sk_O = [1, -\mathbf{t}_O^T]^T \in \mathbb{Z}_q^{n_O\times 1}$, set $\big((sk_I^T)\cdot\mathbf{G}\big)^T := \mathsf{PowerOf2}_q(sk_I) \in \mathbb{Z}_q^{\hat{n}_I\times 1}$, where $\hat{n_I} = n_I \times \lfloor \log q \rfloor$;
  2) Samples a random matrix $\mathbf{B}_{I:O} \leftarrow \mathbb{Z}_q^{\hat{n}_I\times(n_O-1)}$ and $\mathbf{e}_{I:O}$ from error distribution $\chi^{\hat{n}_I\times 1}$, then

$$\mathbf{b}_{I:O} = \mathbf{B}_{I:O}\cdot\mathbf{t}_O + \mathbf{e}_{I:O} + \big([1, -\mathbf{t}_I^T]\cdot\mathbf{G}\big)^T \in \mathbb{Z}_q^{\hat{n}_I\times 1};$$

  3) Outputs the key switching matrix $\mathcal{K}_{I:O} = [\mathbf{b}_{I:O} \,|\, \mathbf{B}_{I:O}] \in \mathbb{Z}_q^{\hat{n}_I\times n_O}$.
- $\mathbf{c}_O \leftarrow \mathsf{SwitchKey}(\mathcal{K}_{I:O}, \mathbf{G}^{-1}(\mathbf{c}_I))$: where $\mathbf{G}^{-1}(\mathbf{c}_I) = \mathsf{Bit}(\mathbf{c}_I) \in \mathbb{Z}_q^{\hat{n}_I\times 1}$ for $\mathbf{c}_I \in \mathbb{Z}_q^{n_I\times 1}$ and the function $\mathbf{G}^{-1}$ makes the dimension from $\mathbb{Z}_q^{n_I}$ to $\mathbb{Z}_q^{\hat{n}_I}$,
  1) To switch a ciphertext from sk_I to $sk_{I:O}$, runs

$$\begin{aligned}\mathcal{K}_{I:O}\cdot[1, -\mathbf{t}_O^T]^T &= \mathbf{b}_{I:O} - \mathbf{B}_{I:O}\cdot\mathbf{t}_O \\ &= \mathbf{e}_{I:O} + \big([1, -\mathbf{t}_I^T]\cdot\mathbf{G}\big)^T.\end{aligned}$$

  2) Then outputs: $\mathbf{c}_O := \mathcal{K}_{I:O}^T\cdot\mathbf{G}^{-1}(\mathbf{c}_I) \in \mathbb{Z}_q^{n_O\times 1}$.

Below we state the correctness and security of key switching procedure, then present the proofs by the definition. We remark that we can ignore the subscripts when they are apparent from the context.

Lemma 3.1 (Correctness, [29]). *Set $sk_I \in \mathbb{Z}^{n_I}$, $sk_O \in \mathbb{Z}^{n_O}$ and $\mathbf{c}_I \in \mathbb{Z}_q^{n_I}$ be any vectors. Suppose there exist $\mathcal{K}_{I:O} \leftarrow \mathsf{SwitchKeyGen}(sk_I, sk_O)$ and $\mathbf{c}_O \leftarrow \mathsf{SwitchKey}(\mathcal{K}_{I:O}, \mathbf{c}_I)$, then there exists the equality $\langle \mathbf{c}_O, sk_O\rangle = \langle \mathbf{c}_I, sk_I\rangle + \langle \mathbf{G}^{-1}(\mathbf{c}_I), \mathbf{e}_{I:O}\rangle \pmod q$.*

Proof. We give a more detailed proof

$$\begin{aligned}\langle \mathbf{c}_O, sk_O\rangle &= \big(\mathbf{G}^{-1}(\mathbf{c}_I)\big)^T\cdot\underline{\mathcal{K}_{I:O}\cdot\Big([1, -\mathbf{t}_O^T]^T\Big)} \\ &= \langle \mathbf{c}_I, \big([1, -\mathbf{t}_I^T]^T\big)\rangle + \langle \mathbf{G}^{-1}(\mathbf{c}_I), \mathbf{e}_{I:O}\rangle \pmod q.\end{aligned}$$

Completing this proof. □

Lemma 3.2 (Security, [29]). *Set $sk_I \in \mathbb{Z}^{n_I}$ be any vector. Suppose we generate $sk_O \leftarrow \mathsf{SecretKeyGen}(\mathsf{params})$ and $\mathcal{K}_{I:O} \leftarrow \mathsf{SwitchKeyGen}(sk_I, sk_O)$, then the difference between the uniform distribution over $\mathbb{Z}_q^{\hat{n}_I \times n_O}$ and the distribution $\mathcal{K}_{I:O}$ cannot be distinguished by any efficient* PPT *adversary under* $\mathsf{LWE}_{n,q,\chi}$ *assumption.*

3.2 Optimized Single-Hop PRE Scheme

Armed with the optimized key switching program, we first restate the scheme of Chandran et al. [12], which is similar to Aono et al. [11] and Nishimaki et al. [14].

- $\mathsf{params} \leftarrow \mathsf{PRE.Setup}(1^\lambda)$:
  1) Takes as input λ then generates $\mathsf{params} := (n, m, q, \chi)$ as public parameter;
- $(pk, sk) \leftarrow \mathsf{PRE.KeyGen}(\mathsf{params})$:
  1) The algorithm first choose a random matrix $\mathbf{B}$ from $\mathbb{Z}_q^{m\times n}$, a random vector $\mathbf{t} \leftarrow \mathbb{Z}_q^{n\times 1}$, and an error vector $\mathbf{e} \leftarrow \chi^{m\times 1}$ first, then generates $\mathbf{b} = \mathbf{B}\cdot \mathbf{t} + \mathbf{e} \pmod q \in \mathbb{Z}_q^{m\times 1}$;
  2) Outputs $\mathbf{A} = (\mathbf{b}, \mathbf{B}) \in \mathbb{Z}_q^{m\times(n+1)}$ as the public key pk and $\mathbf{s} = (1, -\mathbf{t}^T)^T \in \mathbb{Z}_q^{(n+1)\times 1}$ as the secret key sk.
- $\mathbf{c} \leftarrow \mathsf{PRE.Enc}(\mathsf{params}, pk, \mu \in \{0,1\})$:
  1) Samples a random vector $\mathbf{r}$ from $\{-1,0,1\}^{m\times 1}$ first, then sets $\mathbf{u} := [\mu, 0, \ldots, 0]^T \in \{0,1\}^{(n+1)\times 1}$;
  2) Outputs $\mathbf{c} = \mathbf{A}^T\mathbf{r} + \lfloor \frac{q}{2} \rfloor \mathbf{u} \pmod q \in \mathbb{Z}_q^{(n+1)\times 1}$;
- $rk \leftarrow \mathsf{PRE.ReKeyGen}(\mathsf{params}, pk_i := \mathbf{A}_i, pk_j := \mathbf{A}_j, sk_i := \mathbf{s}_i)$: As the definition of PRE mentioned earlier, in each hop, the re-encryption key generation algorithm takes the public key $pk_i := \mathbf{A}_i := (\mathbf{b}_i, \mathbf{B}) \in \mathbb{Z}_q^{m\times(n+1)}$ and secret key $sk_i := \mathbf{s}_i = (1, -\mathbf{t}_i^T)^T \in \mathbb{Z}_q^{(n+1)\times 1}$ from the input-side player (i.e., delegator) and the public key $pk_j := \mathbf{A}_j := (\mathbf{b}_j, \mathbf{B})$ from the output-side player (i.e., delegatee) as input. Thus, it holds that,
  1) Takes the key pair of the delegator (or input-side player), i.e., pk_i, sk_i, and the public key of the delegatee (or output-side player), i.e., pk_j;
  2) Next, the algorithm chooses a random matrix $\mathbf{R}^{(j)} \leftarrow \{-1,0,1\}^{m\times m}$ and the algorithm computes the matrix $\mathbf{N}^{(j)} := \mathbf{A}_i^T \cdot \mathbf{R}^{(j)} \in \mathbb{Z}_q^{(n+1)\times m}$;
  3) Samples another random matrix $\mathbf{R}^{(i\to j)}$ from $\{-1,0,1\}^{m\times(n+1)\ell_q}$ and computes

$$\mathbf{M}^{(i\to j)} \leftarrow \left[\frac{\mathbf{b}_j^T \cdot \mathbf{R}^{(i\to j)} + \left([1, -\mathbf{t}_i^T]\cdot \mathbf{G}\right)}{\mathbf{B}^T \cdot \mathbf{R}^{(i\to j)}}\right].$$

Remark 3.3. As mentioned above, the key switching matrix is $\mathcal{K}_{I:O} = [\mathbf{b}_{I:O} \,|\, \mathbf{B}_{I:O}] \in \mathbb{Z}_q^{\hat{n}_I \times n_O}$ for $\mathbf{b}_{I:O} = \mathbf{B}_{I:O}\cdot\mathbf{t}_O + \mathbf{e}_{I:O} + \left([1, -\mathbf{t}_I]\cdot\mathbf{G}\right)^T \in \mathbb{Z}_q^{\hat{n}_I\times 1}$, where the dimension of matrix $\mathbf{B}_{I:O}$ is the same as $\left([1, -\mathbf{t}_2^{(i)}]\cdot\mathbf{G}\right)^T$. Notably, in order to re-randomize a converted ciphertext, we need to parse $\mathcal{K}_{I:O}$ into $(\mathbf{B}_{I:O}\cdot\mathbf{t}_O + \mathbf{e}_{I:O} + \left([1, -\mathbf{t}_I]\cdot\mathbf{G}\right)^T)$ and $\mathbf{B}_{I:O}$, then we use $\mathbf{R}^{(i\to j)}$ to multiply them respectively.

  4) Outputs $rk := \mathbf{X} = (\mathbf{M}^{(i\to j)}, \mathbf{N}^{(j)})$.

- $\mathbf{c}_j \leftarrow \mathsf{ReEnc}(\mathsf{params}, rk := (\mathbf{M}^{(i\to j)}, \mathbf{N}^{(j)}), \mathbf{c}_i)$:
  1) Samples a random vector $\hat{\mathbf{r}} \in \{0,1\}^{m\times 1}$, then computes and outputs the re-encryption ciphertexts $\mathbf{c}_j := \mathbf{M}^{(i\to j)}\cdot\mathbf{G}^{-1}(\mathbf{c}) + \mathbf{N}^{(j)}\cdot\hat{\mathbf{r}} \pmod q \in \mathbb{Z}_q^{(n+1)\times 1}$ for $\mathbf{G}^{-1}(\mathbf{c}) \in \mathbb{Z}_q^{(n+1)\ell_q\times 1}$.
- $\mathsf{PRE.Dec}(\mathsf{params}, sk, \mathbf{c})$: Two cases for the *decryption algorithm* will be considered,
  1) *At input side*, the delegator computes $\langle \mathbf{c}, \mathbf{s}\rangle := \mathbf{c}^T \cdot \mathbf{s} = \mathbf{r}\cdot(\mathbf{A}\cdot\mathbf{s}) + \lfloor\frac{q}{2}\rfloor\cdot\mu = \mathbf{r}\cdot\mathbf{e} + \lfloor\frac{q}{2}\rfloor\cdot\mu \pmod q$, if $\|\mathbf{re}\| \le m\cdot B_\chi < q/8$, then outputs $\mu = 1$ and otherwise outputs $\mu = 0$.
  2) *At output side*, the delegatee decrypts the 1-hop re-encryption ciphertexts. Specifically, the delegatee computes $\langle \mathbf{c}_j, \mathbf{s}_j\rangle = \lfloor\frac{q}{2}\rfloor\cdot\mu + \mathbf{error} \pmod q$. Thus we obtain $\mu = 1$ if $\|\mathbf{error}\| < q/8$, otherwise $\mu = 0$.

Actually, the correctness of input side decryption is the same as the general decryption. But the correctness of output side is more complex. Below we present the output side decryption correctness.

Lemma 3.4 (Correctness). *If the following equation holds $\|\mathbf{error}\| < (n+1)\ell_q \cdot mB_\chi$ then the output side decryption of the single-hop* PRE *scheme is correct.*

Proof. Below is the decryption at output side,

$$\begin{aligned}
\langle \mathbf{c}_j, \mathbf{s}_j\rangle &= \left(\mathbf{M}^{(i\to j)}\cdot\mathbf{G}^{-1}(\mathbf{c}_i) + \mathbf{N}^{(j)}\cdot\hat{\mathbf{r}}\right)^T \cdot \mathbf{s}_j \\
&= \left(\left[\frac{\mathbf{b}_j^T\cdot\mathbf{R}^{(i\to j)} + \left([1,-\mathbf{t}_i^T]\cdot\mathbf{G}\right)}{\mathbf{B}_j^T\cdot\mathbf{R}^{(i\to j)}}\right]\cdot\mathbf{G}^{-1}(\mathbf{c}_i) \right.\\
&\quad \left. + \mathbf{A}_j^T\cdot\mathbf{R}^{(j)}\cdot\hat{\mathbf{r}}\right)^T \cdot \left(\frac{1}{-\mathbf{t}_j}\right) \\
&= \left(\frac{\mathbf{b}_j^T\mathbf{R}^{(i\to j)}\mathbf{G}^{-1}(\mathbf{c}_i) + \left([1,-\mathbf{t}_i^T]\mathbf{G}\mathbf{G}^{-1}(\mathbf{c}_i)\right)}{\mathbf{B}_j^T\cdot\mathbf{R}^{(i\to j)}\cdot\mathbf{G}^{-1}(\mathbf{c}_i)}\right)^T \\
&\quad \cdot\left(\frac{1}{-\mathbf{t}_j}\right) + \left(\hat{\mathbf{r}}^T\cdot(\mathbf{R}^{(j)})^T\cdot\mathbf{A}_j\right)\cdot\left(\frac{1}{-\mathbf{t}_j}\right) \\
&= \left([1,-\mathbf{t}_i^T]\mathbf{G}\mathbf{G}^{-1}(\mathbf{c}_i)\right) + \left(\mathbf{R}^{(i\to j)}\mathbf{G}^{-1}(\mathbf{c}_i)\right)^T(\mathbf{b}_j) \\
&\quad - \left(\mathbf{R}^{(i\to j)}\mathbf{G}^{-1}(\mathbf{c}_i)\right)^T(\mathbf{B}_j\mathbf{t}_j) + \left(\hat{\mathbf{r}}^T(\mathbf{R}^{(j)})^T\right)\mathbf{e}_j \\
&= \left\lfloor\frac{q}{2}\right\rfloor\cdot\mu + \mathbf{error} \pmod q,
\end{aligned}$$

where we denote **error** as follows:

$$\mathbf{error} = \mathbf{r}_i\mathbf{e}_i + \left(\left(\mathbf{G}^{-1}(\mathbf{c}_i)\right)^T\left(\mathbf{R}^{(i\to j)}\right)^T + \hat{\mathbf{r}}^T(\mathbf{R}^{(j)})^T\right)\cdot\mathbf{e}_j,$$

hence, we have that

$$\begin{aligned}
\|\mathbf{error}\| &\le \left\|\left(\left(\mathbf{G}^{-1}(\mathbf{c}_i)\right)^T\left(\mathbf{R}^{(i\to j)}\right)^T + \hat{\mathbf{r}}^T(\mathbf{R}^{(j)})^T\right)\mathbf{e}_j\right\| \\
&\quad + \|\mathbf{r}_i\mathbf{e}_i\| \le (n+1)\ell_q m B_\chi.
\end{aligned}$$

This completes the proof. □

Lemma 3.5 (Security). *The single-hop* PRE *scheme is IND-CPA-secure under the decisional* LWE *assumption.*

Proof. Below is the sketch proof of Lemma 3.5. The proof contains three steps, in more detail: first, in the

uncorrupted key-generation oracle case, armed with the decisional LWE assumption, the matrix $\mathbf{A} = (\mathbf{b}, \mathbf{B})$ is replaced by a uniform matrix over $\mathbb{Z}_q^{m\times(n+1)}$. In the corrupted key-generation oracle case, $\mathcal{A}$ will get $\mathbf{s}$; second, for convenience, following analysis in Γ_H, armed with the decisional LWE assumption, the vector $\mathbf{b}_j^T \cdot \mathbf{R}^{(i\to j)}$ is replaced by a uniform distribution over $\mathbb{Z}_q^{1\times(n+1)\ell_q}$. Then we use the Lemma 2.2 (i.e., leftover hash lemma) to replace $\mathbf{X} = (\mathbf{M}^{(i\to j)}, \mathbf{N}^{(j)})$ (i.e., the re-encryption key) with a uniform distribution over $\mathbb{Z}_q^{(n+1)\times(n+1)\ell_q} \times \mathbb{Z}_q^{(n+1)\times m}$ for player i and j are in Γ_H, otherwise aborts. Lastly, we use the Lemma 2.2 to replace $\mathbf{c} = \mathbf{A}^T \cdot \mathbf{r} + \lfloor \frac{q}{2} \rfloor \cdot \mathbf{u} \pmod q$ (i.e., the original ciphertext) with a uniform vector over $\mathbb{Z}_q^{(n+1)\times 1}$. This completes the proof. More details will be found in [12], [14]. □

4 Achieving Multi-Hop PRE Scheme

In this section, we develop a multi-hop homomorphic PRE scheme. Concretely, inspired by the Barington's theorem, we know that the encrypted data can be computed by BP. Moreover, in lattice-based setting, the BP is composed of a series of NAND gates. Actually, the essence of NAND gate is the form of $\mathbf{X} \cdot \mathbf{G}^{-1}(\mathbf{Y})$ which takes two specified matrices $\mathbf{X}$ and $\mathbf{Y}$ as input. The analogies between the form of $\mathbf{X} \cdot \mathbf{G}^{-1}(\mathbf{Y})$ and homomorphic multiplication have been striking. Hence, using BP to compute the PRE's ciphertext, the NAND gate can be assembled by the re-encryption key and ciphertext, then the multi-hop PRE's re-encrypted ciphertext can be obtained. That's the reason why we attempt to design the multi-hop homomorphic PRE scheme.

The high-level technical route of our multi-hop homomorphic PRE scheme is provided as follows: 1). we first optimize the scheme of Chandran et al. (CCLNX) at PKC'14 [12] by using we improved key switching program [34] as described in Section 3. 2). then, we transform the encryption algorithm of optimized PRE scheme into FHE algorithm, namely that $\mathbf{c} = \mathbf{A}^T\mathbf{r} + \lfloor \frac{q}{2} \rfloor \mathbf{u} \pmod q \in \mathbb{Z}_q^{(n+1)\times 1}$ was transformed into the form of GSW-style ciphertext $\mathbf{C} := \mathbf{A}^T \cdot \mathbf{R} + \mu \cdot \mathbf{G} \pmod q \in \mathbb{Z}_q^{(n+1)\times(n+1)\ell_q}$ for gadget matrix $\mathbf{G} \in \mathbb{Z}_q^{(n+1)\times(n+1)\ell_q}$ and random matrix $\mathbf{R} \in \{0,1\}^{m\times(n+1)\ell_q}$. Obviously, the dimension of homomorphic PRE's ciphertext is bigger than the optimized PRE scheme. Hence, it is necessary to shorten the length of ciphertext. To solve this issue, we modify the homomorphic encryption by the form of $\mathbf{c} = \mathbf{A}^T \cdot \mathbf{r} + \mu \cdot 2^{(\ell_q-1)}\mathbf{u}_1 \pmod q \in \mathbb{Z}_q^{(n+1)\times 1}$, where $\mathbf{u}_i$ is the unit vector whose ith position is 1, i.e., $(0,\ldots,1,\ldots,0)$, for $i \in \{0,\ldots,(n+1)\cdot(\ell_q-1)\}$. Hence, we follow the technique of multi-key FHE with short ciphertexts of Brakerski et al. [20] to shorten the ciphertext length. Actually, the essence of this technique is that we choose a fragment of gadget matrix $\mathbf{G}$ instead of the whole matrix $\mathbf{G}$. 3). lastly, the re-encryption keys and ciphertexts are embeded into the NAND gates and the NAND gates are transformed to BP. A more detailed interpretation is as follows.

4.1 Our Construction: Multi-Hop PRE via BP

We present our multi-hop homomorphic PRE scheme in this section. Actually, Aono et al. [11] and Nishimark et al. [14] etc. present some PRE schemes respectively. However, these schemes focus on single-hop, multi-bit encryption and key-private, they seem to fail to take into account the importance of multi-hop.

In order to make our scheme support homomorphic operation, the work of Nishimaki et al. [14] inspire us to adopt the encryption algorithm of the scheme of Brakerski and Perlman [20], which is a variant of Gentry-Sahai-Waters (GSW) [23] scheme with short ciphertext.

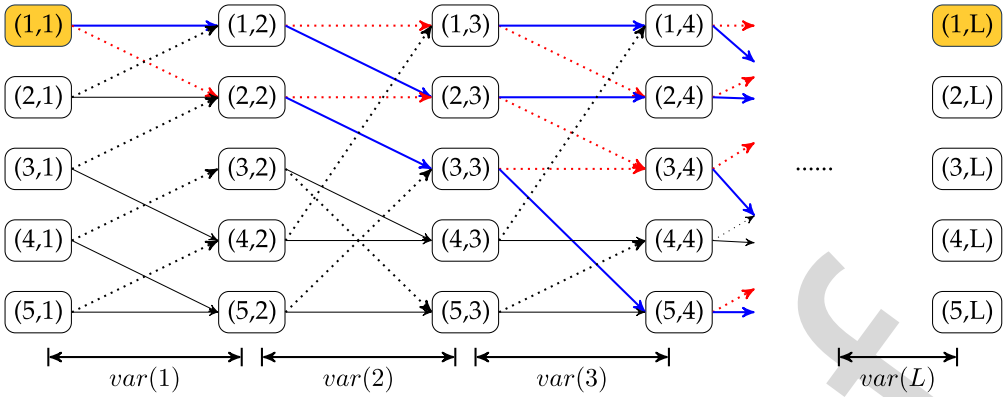
Below we only focus on the Enc, ReKeyGen, ReEnc and Dec algorithms which are different from the above optimized PRE scheme, but we omit the description of Setup and KeyGen since they are same as the optimized PRE scheme. More details are as follows:

- $\mathbf{c}_i \leftarrow \mathsf{Enc}(pk_i, \mu_i \in \{0,1\})$: In order to shorten the GSW style ciphertext $\mathbf{C} = \mathbf{A}^T\mathbf{R} + \mu\mathbf{G} \in \mathbb{Z}_q^{(n+1)\times(n+1)\ell_q}$ for gadget matrix $\mathbf{G} \in \mathbb{Z}_q^{(n+1)\times(n+1)\ell_q}$ and random matrix $\mathbf{R} \in \{0,1\}^{m\times(n+1)\ell_q}$, we follow the method of Brakerski et al. [20] to shorten the ciphertext and give the encryption algorithm with short ciphertext as follows.
  1) A uniform vector $\mathbf{r}_i$ from $\{0,1\}^{m\times 1}$ is first sampled by the algorithm, and lets $\mathbf{u}_1$ as 1th standard basis vector;
  2) Computes and outputs the encryption $\mathbf{c}_i := \mathbf{A}^T \cdot \mathbf{r}_i + \mu_i \cdot (2^{\ell_q-1}\mathbf{u}_1) \pmod q \in \mathbb{Z}_q^{(n+1)\times 1}$.

  Remark 4.1. We set $\mathbf{u}_i$ as the ith unit vector $(0,\ldots,1,\ldots,0)$. We first keep in mind the form of the gadget matrix $\mathbf{G} = \mathbf{g} \otimes \mathbf{I}_{n\times n} \in \mathbb{Z}_q^{n\times n\cdot\ell_q}$, then we consider any column vector $\mathbf{G}[i] \in \mathbb{Z}_q^{n\times 1}$ of $\mathbf{G}$ for $i \in \{0,\ldots,n\cdot\ell_q - 1\}$, we can express $\mathbf{G}[i]$ as $2^j \cdot \mathbf{u}_i = (0,\ldots,2^j,\ldots,0)^T$, where $j := i \bmod \ell_q$. In this case, without loss of generality, we choose $\mathbf{G}[1]$ as an example, (i.e., $i = 1$), then $2^{(\ell_q-1)} \cdot \mathbf{u}_1 = (2^{(\ell_q-1)}, 0, \ldots, 0)^T$. Hence, the new homomorphic encryption with short ciphertext is as follows: $\mathbf{c} = \mathbf{A}^T \cdot \mathbf{r} + \mu \cdot 2^{(\ell_q-1)}\mathbf{u}_1 \pmod q \in \mathbb{Z}_q^{(n+1)\times 1}$. We stress that, in [20], the authors adopted a fragment of gadget matrix $\mathbf{G}$ instead of the whole $\mathbf{G}$.
- $rk \leftarrow \mathsf{ReKeyGen}(\mathsf{params}, pk_i, sk_i, pk_j)$: The algorithm outputs the re-encryption key rk by taking the key pair of the delegator (i.e., player P_i) and the public key of the delegatee (i.e., player P_j) as input. Importantly, the secret key of the delegatee should be encrypted in bit-by-bit manner on the basis of the GSW scheme's encryption algorithm. Hence, the key steps can be processed as follows.
  1) First, the secret key can be decomposed by using the the bit decompose operation Bit and we can obtain $\mathsf{Bit}_k(sk_i)$ for $k \in [n \cdot \ell_q]$, for readability, the kth bit of sk is denoted by $\mathsf{Bit}_k(sk_i)$.
  2) Second, computes

$$\vec{\mathcal{S}}[i] = \frac{\mathbf{b}_j^T \cdot \mathbf{R}^{(i\to j)} + \mathsf{Bit}_k(sk_i) \cdot \mathbf{G}}{\mathbf{B}_j^T \cdot \mathbf{R}^{(i\to j)}},$$

  for a random matrix $\mathbf{R}^{(i\to j)} \leftarrow \{0,1\}^{m\times(n+1)\ell_q}$.
  3) Third, combines all of $\vec{\mathcal{S}}[i]$ and generates a concatenation matrix $\vec{\mathcal{S}} = [\vec{\mathcal{S}}[1], \ldots, \vec{\mathcal{S}}[\ell_q]]$;



- dot arrows: are labeled with 0 and denote $(\mathbf{G} - \mathbf{X}_{var(t)})$;
- solid arrows: are labeled with 1 denote $\mathbf{X}_{var(t)}$;
- blocks (i, j): denotes the state for $i \in [5],\ j \in [L]$;
- yellow block $(1, 1)$: denotes the input (initial) state;
- yellow block $(1, L)$: denotes the output (finial) state;

Here we give an example which has no direct relationship with our scheme, to keep the example simple, we first set $L = 3$, the input length is 2. If we input $x = 10$, i.e., $x_1 = 1, x_2 = 0$, then the branching program starts at node $(1, 1)$, and stops at node $(1, 4)$, namely that: $(1,1) \xrightarrow{1} (1,2) \xrightarrow{0} (1,3) \xrightarrow{1} (1,4)$.

Fig. 2. Branching program.

4) Fourth, generates the following matrix $\mathbf{M}^{(i\to j)} := \mathsf{Extend}(\vec{\mathcal{S}}, (pk_i, pk_j)) \in \mathbb{Z}_q^{(n+1)\times(n+1)\ell_q}$ which satisfies $\mathbf{M}^{(i\to j)} = \sum_i^{\ell_q} \vec{\mathcal{S}}[i]$. We remark that, the Extend algorithm was proposed by Mukherjee and Wicks [35] to design multi-key FHE scheme. In this setting, we can regard $\mathbf{M}^{(i\to j)}$ as

$$\mathbf{M}^{(i\to j)} = \left\lceil \frac{(\mathbf{b}_j^T \cdot \mathbf{R}^{(i\to j)} + ([1, -\mathbf{t}_i^T] \cdot \mathbf{G})}{\mathbf{B}_j^T \cdot \mathbf{R}^{(i\to j)}} \right\rfloor;$$

5) Fifth, computes $\mathbf{N}^{(j)} := \mathbf{A}_j^T \cdot \mathbf{R}^{(j)} \in \mathbb{Z}_q^{(n+1)\times m}$ where $\mathbf{R}^{(j)} \leftarrow \{0,1\}^{m\times m}$ is a random matrix.

6) Last, we can obtain the re-encryption key $rk^{(i\to j)} := \mathbf{X}^{(i\to j)} = (\mathbf{M}^{(i\to j)}, \mathbf{N}^{(j)})$.

- $\mathbf{c}_L \leftarrow \mathsf{ReEnc}((rk^{(0\to 1)}, \ldots, rk^{(L-1\to L)}), \mathbf{c}_0)$: The *multi-hop re-encryption* is attained via BPs by using the ReEnc algorithm. We then homomorphically proceed the BP as follows two steps. In other words, we will show the details how to convert the (augmented) NAND circuits into the BP.

Remark 4.2. We remark that, for the sake of explaining the mechanism of multi-hop re-encryption via BP, the Fig. 2 is present here to help us explain the mechanism.

1) *Initiation:* For readability, we omit the matrix $\mathbf{N}^{(j)}$ and only consider $\mathbf{X}^{(i\to j)} := \mathbf{M}^{(i\to j)}$. Without the loss of generality, we abbreviate $rk^{(i\to j)}$ to rk_j (or $\mathbf{X}^{(i\to j)}$ to $\mathbf{X}_j$) since the context is clear and the subscript and superscript can be omitted. i.e., $rk_1 := \mathbf{X}_1, \ldots, rk_L := \mathbf{X}_L$. Then, we fix $i = 0$ first, and obtain $\mathbf{c}_0 = 2^{\ell_q - 1}\mathbf{u}_1$ and the state vector $\vec{\mathbf{w}}_0 := (2^{\ell_q-1}\mathbf{u}_1, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0})$, here we stress that we set $\mathbf{u}_1 := [1, 0, \ldots, 0]^T \in \mathbb{Z}_q^{1\times nN}$. Next, the $\mathsf{BPOTF}^{\mathsf{Pred}_{\mathcal{C}}}$ is initiated by invoking the predecessor function $\mathsf{Pred}_{\mathcal{C}}$. The next layer of BP is outputted by $\mathsf{Pred}_{\mathcal{C}}$ by taking as input the label i of a gate, i.e., outputs $\Big((\gamma_{t,1,0}, \ldots, \gamma_{t,5,0}), (\gamma_{t,1,1}, \ldots, \gamma_{t,5,1}), var(t)\Big)$.

2) *Iterative Step:* For every step $t = 1, \ldots, L$ it proceeds as follows:

1). Obtains the next layer of the BPs

$$\Big((\gamma_{t,1,0}, \ldots, \gamma_{t,5,0}), (\gamma_{t,1,1}, \ldots, \gamma_{t,5,1}), var(t)\Big),$$

by running $\mathsf{BPOTF}^{\mathsf{Pred}_{\mathcal{C}}}$ and computing the constants $\gamma_{t,i,0}$ and $\gamma_{t,i,1}$.

2). For every $i = 1, \ldots, 5$, $\mathsf{BPOTF}^{\mathsf{Pred}_{\mathcal{C}}}$ evaluates the encrypted next state homomorphically

$$\mathbf{w}_{t,i} = (\mathbf{G} - \mathbf{X}_{vat(t)}) \cdot \mathbf{G}^{-1}(\mathbf{w}_{t-1,\gamma_{t,i,0}}) + \mathbf{X}_{vat(t)} \cdot \mathbf{G}^{-1}(\mathbf{w}_{t-1,\gamma_{t,i,1}}),$$

where we stress that the sum encryption of the secret key $\vec{\mathcal{S}}_{var(t)}$ is defined as $\mathbf{X}_{var(t)} := \sum \vec{\mathcal{S}}_{var(t)}$. Lastly, the results of $\mathsf{BPOTF}^{\mathsf{Pred}_{\mathcal{C}}}$ is the ciphertext $\mathbf{c}_L := \mathbf{w}_{L,1}$.

- $\mathsf{Eval}(\mathsf{params}, pk, \mathbf{c}_1, \ldots, \mathbf{c}_\ell)$: There exist two types of homomorphic operation. For ciphertext $\mathbf{c}_1$ and $\mathbf{c}_2$ under the same public key pk, we have that,
  - *Homomorphic Addition.* $\mathsf{Add}(pk, \mathbf{c}_1, \mathbf{c}_2)$: computes and outputs $\mathbf{c}_1 + \mathbf{c}_2 = (\mu_1 + \mu_2) \cdot (2^{\ell_q-1}\mathbf{u}_1) + \mathbf{A}^T(\mathbf{r}_1 + \mathbf{r}_2) \pmod q$;
  - *Homomorphic Multiplication.* $\mathsf{Mult}(pk, \mathbf{C}_1, \mathbf{c}_2)$: computes and outputs

$$\begin{aligned}\mathbf{C}_1 \cdot \mathbf{G}^{-1}(\mathbf{c}_2) &:= (\mu_1\mathbf{G} + \mathbf{A}^T\mathbf{R}_1) \cdot \mathbf{G}^{-1}(\mathbf{c}_2) \\ &= \mu_1\mu_2 \cdot (2^{\ell_q-1}\mathbf{u}_1) + \mu_1\mathbf{A}^T\mathbf{r}_2 \\ &\quad + \mathbf{A}^T\mathbf{R}_1 \cdot \mathbf{G}^{-1}(\mathbf{c}_2) \pmod q,\end{aligned}$$

  for $\mathbf{C}_1 = \mu\mathbf{G} + \mathbf{A}^T\mathbf{R}_1 \pmod q$;
- $\mathsf{Dec}(\mathsf{params}, sk_L, \mathbf{c}_L)$: Two cases should be considered.
  - The input-side decryption, computes $\langle \mathbf{c}, \mathbf{s} \rangle \pmod q$ and obtains the following results

$$\langle \mathbf{c}, \mathbf{s} \rangle := \mathbf{c}^T \cdot \mathbf{s} = \mathbf{r} \cdot (\mathbf{A} \cdot \mathbf{s}) + \left\lfloor \frac{q}{2} \right\rfloor \cdot \mu$$
$$= \mathbf{r} \cdot \mathbf{e} + \left\lfloor \frac{q}{2} \right\rfloor \cdot \mu \,(\mathrm{mod}\ q);$$

If $\mathsf{noise}_{\mathbf{s},\mu}(\mathbf{c}) := \|\mathbf{re}\| < q/8$, then obtains $\mu = 1$ and otherwise obtains $\mu = 0$. Outputs μ.

- The output-side decryption, in this setting, we focus on the *L-hop decryption algorithm*. In other words, the original ciphertext of player $i \in [L]$ at input-side is re-encrypted L times which is called the ciphertext of player $j \geq i+1$. Thus, the re-encryption ciphertext can be written as $\mathbf{c}_j := \mathbf{X}^{(i \to j)} \cdot \mathbf{G}^{-1}(\mathbf{c}_i)$. In order to obtain the results of $\langle \mathbf{c}_L, \mathbf{s}_L \rangle \,(\mathrm{mod}\ q)$ when $j = L$ after L-hop re-encryptions, then we set $\mathbf{c}_L := \mathbf{X}^{((L-1) \to L)} \cdot \mathbf{G}^{-1}(\mathbf{c}_{(L-1)})$. Lastly computes $\langle \mathbf{c}_L, \mathbf{s}_L \rangle = \mu 2^{\ell_q - 1} + \mathbf{error}$, if $\|\mathbf{error}\| \leq 1/8$, then we obtain $\mu = 1$ and otherwise $\mu = 0$.

4.2 Correctness and Security

We first analyze the two important properties of single-hop homomorphic PRE in this section, including correctness and security, then we analyze the properties of multi-hop homomorphic PRE which contains correctness and security.

Definition 4.3 ([20], [22]). *If the ciphertexts* $\mathbf{c} = \mu 2^{\ell_q - 1}\mathbf{u}_1 + \mathbf{A}^T \cdot \mathbf{r} \in \mathbb{Z}_q^{(n+1)\times 1}$, *along with* $2^{\ell_q - 1}\mathbf{u}_1 \in \mathbb{Z}_q^{(n+1)\times 1}$ *and the secret key* $\mathbf{s} = (1, -\mathbf{t}^T)^T \in \mathbb{Z}_q^{(n+1)\times 1}$, *then the noise of* $\mathbf{c}$ *is the following infinity norm* $\mathsf{noise}_{(\mathbf{s},\mu)}(\mathbf{c}) = \|\mathbf{c} - \mu 2^{\ell_q - 1}\mathbf{u}_1\|_\infty$, *i.e.,* $\mathsf{noise}_{(\mathbf{s},\mu)}(\mathbf{c}) = \|\mathbf{r} \cdot \mathbf{As}\| = \|\mathbf{r} \cdot \mathbf{e}\| \leq \sqrt{m}B \cdot \sqrt{m} \leq m \cdot B \leq \frac{q}{8}$.

Lemma 4.4 ([20], [22], Homomorphic Operation Analysis). *Considering the ciphertexts* $\mathbf{c} = \mu 2^{\ell_q - 1}\mathbf{u}_1 + \mathbf{A}^T \cdot \mathbf{r} \in \mathbb{Z}_q^{(n+1)\times 1}$ *along with* $2^{\ell_q - 1}\mathbf{u}_1 \in \mathbb{Z}_q^{(n+1)\times 1}$ *and* $\mathbf{s} = (1, -\mathbf{t}^T)^T \in \mathbb{Z}_q^{(n+1)\times 1}$, *then the magnitude of noise in addition, multiplication, and negation are as follows. Before analysing the bounded of noise, we first fix all messages* $\mu_1, \mu_2 \in \{0, 1\}$, *then,*

- *For addition, the magnitude is*

$$\mathsf{noise}_{(\mathbf{s},\mu_1+\mu_2)}(\mathbf{c}_1 + \mathbf{c}_2) \leq \mathsf{noise}_{(\mathbf{s},\mu_1)}(\mathbf{c}_1) + \mathsf{noise}_{(\mathbf{s},\mu_2)}(\mathbf{c}_2);$$

i.e., $\mathsf{noise}_{(\mathbf{s},\mu_1+\mu_2)}(\mathbf{c}_1 + \mathbf{c}_2) \leq 2mB \leq q/4$.

- *For multiplication, the magnitude is*

$$\mathsf{noise}_{(\mathbf{s},\mu_1\mu_2)}(\mathbf{C}_1\mathbf{G}^{-1}(\mathbf{c}_2)) \leq (n+1)\ell_q \cdot \mathsf{noise}_{(\mathbf{s},\mu_1)}(\mathbf{C}_1) + \mu_1 \cdot \mathsf{noise}_{(\mathbf{s},\mu_2)}(\mathbf{c}_2),$$

where we stress that $\mathbf{G}^{-1} : \mathbb{Z}_q^{(n+1)\ell_q} \to \mathbb{Z}_q^{n+1}$ *is an efficiently computable function, namely that* $\mathsf{noise}_{(\mathbf{s},\mu_1\mu_2)}(\mathbf{C}_1\mathbf{G}^{-1}(\mathbf{c}_2)) \leq \|\mu_1 \cdot (\mathbf{r}_2\mathbf{e}_2) + (\mathbf{R}_1\mathbf{e}_1) \cdot \mathbf{G}^{-1}(\mathbf{c}_2)\| \leq ((n+1)\ell_q + 1)mB < q/4$.

- *For negation, if all message* $\mu \in \{0, 1\}$,

$$\mathsf{noise}_{(\mathbf{s},1-\mu)}(2^{\ell_q - 1}\mathbf{u}_1 - \mathbf{c}) = \mathsf{noise}_{(\mathbf{s},\mu)}(\mathbf{c}).$$

We note that, NAND *gates constituted the evaluation of the Boolean circuit which has depth L (i.e., L hops). Thus, the noise is amplified by a factor of at most* $((n+1)\ell_q + 1)$, *the final ciphertext after L hops is bounded by* $((n+1)\ell_q + 1)^L mB \leq q/4$.

4.2.1 Single-Hop Homomorphic PRE Analysis

Lemma 4.5 (Single-Hop Correctness). *Considering the results of the single-hop decryption of output-side as follows:*

$$\langle \mathbf{c}_j, \mathbf{s}_j \rangle = \left(\mathbf{M}^{(i \to j)} \cdot \mathbf{G}^{-1}(\mathbf{c}) + \mathbf{N}^{(j)} \cdot \hat{\mathbf{r}}\right)^T \cdot \mathbf{s}_j$$
$$= \mu 2^{\ell_q - 1} + \mathbf{error} \,(\mathrm{mod}\ q),$$

if $\|\mathbf{error}\| < (n+1)\ell_q \cdot mB_\chi$, *then we can say output-side decryption of the single-hop homomorphic* PRE *scheme is correct.*

Proof. The only difference between Lemmas 4.5 and 3.5 is the encryption algorithm. More concretely, we follow the encryption algorithm of the scheme of Brakerski et al. [20] and obtain the following equation:

$$\langle \mathbf{c}_j, \mathbf{s}_j \rangle = \left(\mathbf{M}^{(i \to j)} \cdot \mathbf{G}^{-1}(\mathbf{c}) + \mathbf{N}^{(j)} \cdot \hat{\mathbf{r}}\right)^T \cdot \mathbf{s}_j$$
$$= \mu 2^{\ell_q - 1} + \mathbf{error} \,(\mathrm{mod}\ q).$$

Hence, $\|\mathbf{error}\| \leq (n+1)\ell_q \cdot mB_\chi \leq 1/8$ is based on the Lemma 3.4. This completes the proof. □

Lemma 4.6 (Single-Hop Security). *Assume the single-hop* PRE *scheme is IND-CPA-secure, then the single-hop homomorphic* PRE *with short ciphertext is IND-CPA-secure under* LWE *assumption.*

Proof. Note that the public key and the re-encryption key in the optimized single-hop homomorphic PRE with short ciphertext are the same as the single-hop PRE scheme. However, the encryption algorithm is adopted from the scheme of Brakerski et al. [20] whose ciphertexts are first column of the ciphertexts in Ma et al. [19] scheme. Hence, we just focus on the distribution of ciphertexs. In particular we show that the distribution of ciphertexs is statistically indistinguishable from a uniform vector over $\mathbb{Z}_q^{(n+1)\times 1}$ by using the Lemma 2.2, namely that $\mathbf{A}^T \cdot \mathbf{r}$ is indistinguishable from $\mathbb{Z}_q^{(n+1)\times 1}$ under LWE assumption. □

4.2.2 Multi-Hop Homomorphic PRE Analysis

The correctness of input-side decryption is the same as the correctness of the general decryption. Hence we don't repeat them. The analysis of the output-side decryption's correctness is presented as follows.

Lemma 4.7 (Multi-Hop Correctness). *If the noise magnitude is bounded by the following* $\mathsf{noise}_{sk_L,\mathbf{v}_t[i]}(\mathbf{w}_{t,i}) < 2t(n+1)\ell_q \cdot mB_\chi$ *along with* $t = 0, 1, \ldots, L$ *and* $i \in [5]$, *then the multi-hop homomorphic* PRE *scheme with short ciphertexts is correct after L-hop re-encryptions.*

Proof. We initialize a message without any noises, i.e., $\mathbf{vw}_0$. Hence we set $\mathsf{noise}(\mathbf{w}_{0,i}) = 0$. Then we assume $t' = t - 1$ for $t' < t$. In other words, we use the hypothesis to obtain the following results $\mathsf{noise}_{sk_{t-1},\mathbf{v}_t[i]}(\mathbf{w}_{t-1,\gamma_{t,i}}) = (t-1) \cdot (n+1)\ell_q \cdot mB_\chi$. Then we analyze the noise magnitude by induction on step t, we recall the definition of $\mathbf{w}_t$ as mentioned earlier, we obtain the following results:

$$
\begin{aligned}
&\mathsf{noise}_{sk_L,\mathbf{v}_t[i]}(\mathbf{w}_{t,i}) \\
&= \mathsf{noise}_{sk_L,\mathbf{v}_t[i]}\Big((\mathbf{G}-\mathbf{X}_{vat(t)})\cdot\mathbf{G}^{-1}(\mathbf{w}_{t-1,\gamma_{t,i,0}}) \\
&\quad + \mathbf{X}_{vat(t)}\cdot\mathbf{G}^{-1}(\mathbf{w}_{t-1,\gamma_{t,i,1}})\Big) \\
&= (1-x_{vat(t)})\cdot\mathsf{noise}_{sk_L,\mathbf{v}_t[i]}\Big(\mathbf{w}_{t-1,\gamma_{t,i,0}}\Big) \\
&\quad + 2(n+1)\ell_q\cdot\mathsf{noise}_{sk_L,rk_{var(t)}}\Big(\mathbf{X}_{vat(t)}\Big) \\
&\quad + x_{vat(t)}\cdot\mathsf{noise}_{sk_L,\mathbf{v}_t[i]}\Big(\mathbf{w}_{t-1,\gamma_{t,i,1}}\Big) \\
&\le \max\Big\{\mathsf{noise}_{sk_L,\mathbf{v}_t[i]}\Big(\mathbf{w}_{t-1,\gamma_{t,i,0}}\Big), \\
&\qquad \mathsf{noise}_{sk_L,\mathbf{v}_t[i]}\Big(\mathbf{w}_{t-1,\gamma_{t,i,1}}\Big)\Big\} \\
&\quad + 2(n+1)\ell_q\cdot\mathsf{noise}_{sk_L,rk_{var(t)}}\Big(\mathbf{X}_{vat(t)}\Big) \\
&\le 2(t-1)(n+1)\ell_q\cdot mB_\chi + 2(n+1)\ell_q\cdot mB_\chi \\
&\le 2t(n+1)\ell_q\cdot mB.
\end{aligned}
$$

Obliviously, combining the above result with the previous Lemma 4.7, we can obtain the following result $\mathsf{noise}_{sk_L,\mathbf{v}_L[1]}(\mathbf{w}_{L,1}) < L\cdot 2t(n+1)\ell_q\cdot mB < \frac{q}{8}$ after the L-hop iterations, where the circuit depth is d and $L=4^d$. This completes the proof. □

Theorem 4.8 (Security). *If the single-hop* PRE *scheme is IND-CPA-secure then multi-hop* PRE *is IND-CPA-secure under decisional-*LWE *assumption.*

Proof. The proof contains four steps, for convenience, we set the following analysis for honest players.

1) First, we apply Lemma 2.2 to show that in the uncorrupted key generation oracle case, under decisional LWE assumption, the distribution of public key $\mathbf{A}=(\mathbf{b},\mathbf{B})$ is statistically indistinguishable from a randomly chosen vector over $\mathbb{Z}_q^{m\times(n+1)}$, namely that $\mathbf{b}$ is indistinguishable from uniform over $\mathbb{Z}_q^{m\times 1}$;
2) Second, utilizing the LWE instance, we first sample a uniform distribution over $\mathbb{Z}_q^{1\times(n+1)\ell_q}$ to replace $\mathbf{b}_j^T\cdot\mathbf{R}^{(i\to)}$, then we sample a uniform matrix from $\mathbb{Z}_q^{(n+1)\times(n+1)\ell_q}$ to replace $\mathbf{M}^{(i\to j)}$, we also use a uniform matrix over $\mathbb{Z}_q^{(n+1)\times m}$ to replace $\mathbf{A}_i^T\mathbf{R}^{(j)}$, separately, the above description is for player i and j, they are in Γ_H, otherwise it will abort. Hence, the distribution of re-encryption key $\mathbf{X}$ is computationally indistinguishable from uniform distribution over $\mathbb{Z}_q^{(n+1)\times(n+1)\ell}\times\mathbb{Z}_q^{(n+1)\times m}$;
3) Third, based on Lemma 2.2, we use a uniform vector over $\mathbb{Z}_q^{(n+1)\times 1}$ to replace the original input-side ciphertext $\mathbf{c}=\mathbf{A}^T\cdot\mathbf{r}+2^{\ell_q-1}\mu\cdot\mathbf{u}_1 \pmod q$, namely that a uniform distribution over $\mathbb{Z}_q^{(n+1)\times 1}$ is indistinguishable from $\mathbf{A}^T\cdot\mathbf{r}$.
4) Lastly, we can gain the output-side ciphertexts $\mathbf{c}_2 := \mathbf{X}^{(1\to 2)}\cdot\mathbf{G}^{-1}(\mathbf{c}_1)$ after 1-hop re-encryption. Hence, based on Step 2, we use the Lemma 2.2 to show that a uniform vector over $\mathbb{Z}_q^{(n+1)\times 1}$ is indistinguishable from $\mathbf{c}_2$; similarly, $\mathbf{c}_2$ is indistinguishable from $\mathbf{c}_3 := \mathbf{X}^{(2\to 3)}\cdot\mathbf{G}^{-1}(\mathbf{c}_2)$. Repeating L times via the same method, i.e., after L-hop re-encryptions, we can gain the ciphertexts $\mathbf{c}_L := \mathbf{w}_{L,1}$ which are indistinguishable from uniform vector over $\mathbb{Z}_q^{(n+1)\times 1}$.

This completes the proof. □

4.3 Multi-Bit and Multi-Hop Homomorphic PRE

Aono et al. [11] constructed a multi-bit PRE scheme by encrypting matrix directly. In this paper, we optimize the multi-bit encryption by the method of Li et al. [36] which we can use to encrypt/decrypt multi-bit at one-time.

Inspired by the recent work of Wichs and Zirdelis [37], which studied the seemingly unrelated problem of obfuscating compute-and-compare program. As one of the components of their solution, they encoded t BPs with t-bit output by encoding BP for each output bit separately. Hence, in this paper, we follow the method of [37], we parse the ciphertext into t pieces, and each piece is associated with a BP, separately.

- $(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$:
  1) First, we modified the public key as
  $$pk := \mathbf{A} = \big(\mathbf{b}_1,\ldots,\mathbf{b}_t|\mathbf{B}\big) \in \mathbb{Z}_q^{m\times(n+t)},$$
  where $\mathbf{b}_i := \mathbf{B}\cdot\mathbf{t}_i + \mathbf{e}_i \pmod q \in \mathbb{Z}_q^{m\times 1}$ for $\xi\in[t]$.
  2) We also modified the secret key as
  $$sk := \mathbf{S} = [\mathbf{s}_1^T,\ldots,\mathbf{s}_t^T] \in \mathbb{Z}_q^{(n+t)\times t},$$
  for any $\mathbf{s}_\xi := \big(0,\ldots,1,\ldots,0|-\mathbf{t}_\xi\big) \in \mathbb{Z}_q^{1\times(n+t)}$.
- $\mathbf{c} \leftarrow \mathsf{Enc}(pk,\mathbf{m})$: We still follow the methodology of Brakerski [20] to scale down the dimension of homomorphic ciphertext, but we improve the multi-bit encryption algorithm, in more detail:
  1) For multi-bit messages
  $$\mathbf{m} = (\mu_1,\ldots,\mu_t) \in \mathbb{Z}_q^{1\times t},$$
  2) Invokes the homomorphic encryption with short ciphertexts, we get the ciphertexts over $\mathbb{Z}_q^{(n+t)\times 1}$
  $$\mathbf{c} := \mathbf{A}^T\cdot\mathbf{r} + (2^0\mathbf{u}_1,\ldots,2^t\mathbf{u}_t)\cdot\begin{pmatrix}\mu_1\\ \vdots\\ \mu_t\end{pmatrix} \pmod q,$$
  where $\mathbf{r}\leftarrow\{0,1\}^{m\times 1}$. We remark that, $\mathbf{u}_i := (0,\ldots,1,\ldots,0)^T \in \{0,1\}^{(n+t)\times 1}$ is unit vector and ith position of $\mathbf{u}_i$ is 1 for $\xi\in[t]$.
  Hereafter, we will use the above short ciphertext as the input of re-encryption algorithm.

Remark 4.9. We remark that, for comparison purposes, let us recall Li et al. [36] scheme which is a multi-bit variant of GSW [23] scheme, the encryption algorithm of Li et al. [36] scheme is as follows,

1) For multi-bit messages
$$\mathbf{m} = (\mu_1,\ldots,\mu_t,\mathbf{0}) \in \mathbb{Z}_q^{1\times(n+t)},$$
we transform it into a diagonal matrix over $\{0,1\}^{(n+t)\times(n+t)}$ as follows:

$$\mathbf{M} = \begin{pmatrix} \mathrm{diag}(\mu_1, \ldots, \mu_t) & \mathbf{0}_{t\times n} \\ \mathbf{0}_{n\times t} & \mathrm{diag}(1, \ldots, 1) \end{pmatrix}, \quad (4.1)$$

2) Invokes the homomorphic encryption, we get the following ciphertexts

$$\mathbf{C} := \mathbf{A}^T \cdot \mathbf{R} + \mathbf{M} \cdot \mathbf{G} \,(\mathrm{mod}\ q) \in \mathbb{Z}_q^{(n+t)\times(n+t)\ell_q},$$

we must stress that $\mathbf{A} \cdot \mathbf{S} = [\mathbf{e}_1, \ldots, \mathbf{e}_t] \,(\mathrm{mod}\ q)$.

Obviously, compared with Hiromasa et al. [38] scheme, we use the new method to encrypt multi-bit message and obtain a shorter chiphertext.

- $rk \leftarrow \mathsf{ReKeyGen}(\mathsf{params}, pk_i, sk_i, pk_j)$: The re-key generation algorithm remains unchanged, to obtain the re-encryption key rk in connection with player P_i and P_j, performs the following steps:
  1) First, a random matrix $\mathbf{R}^{(i\to j)} \leftarrow \{-1, 0, 1\}^{m\times(n+t)\ell_q}$ is chosen first and computes

$$\vec{\mathcal{S}}[i] = \frac{\mathbf{b}_j^T \cdot \mathbf{R}^{(i\to j)} + \mathsf{Bit}_k(sk_i) \cdot \mathbf{G}}{\mathbf{B}_j^T \cdot \mathbf{R}^{(i\to j)}},$$

  we note that, the bit decompose operation Bit is denoted in Section 3.1 and the kth bit of sk is defined as $\mathsf{Bit}_k(sk_i)$ for $k \in [(n+t)\cdot\ell_q]$;
  2) Second, the algorithm constructs a concatenation matrix $\vec{\mathcal{S}} = [\vec{\mathcal{S}}[1], \ldots, \vec{\mathcal{S}}[\ell_q]]$ by combining all of $\vec{\mathcal{S}}[i]$;
  3) Third, in order to obtain the key part of re-encryption key, the algorithm computes $\mathbf{M}^{(i\to j)} := \mathsf{Extend}\big(\vec{\mathcal{S}}, (pk_i, pk_j)\big) \in \mathbb{Z}_q^{(n+t)\times(n+t)\ell_q}$ by invoking the Extend algorithm which is proposed by Mukherjee and Wicks [35]. Actually, $\mathbf{M}^{(i\to j)}$ satisfies

$$\mathbf{M}^{(i\to j)} = \left[\frac{(\mathbf{b}_j^T \cdot \mathbf{R}^{(i\to j)} + ([1, -\mathbf{t}_i^T] \cdot \mathbf{G})}{\mathbf{B}_j^T \cdot \mathbf{R}^{(i\to j)}}\right];$$

  4) A random matrix $\mathbf{R}^{(j)} \leftarrow \{-1, 0, 1\}^{m\times m}$ is sampled first and computes $\mathbf{N}^{(j)} := \mathbf{A}_j^T \cdot \mathbf{R}^{(j)} \in \mathbb{Z}_q^{(n+t)\times m}$.
  5) Last, we obtain the re-encryption key $rk^{(i\to j)} := \mathbf{X}^{(i\to j)} = (\mathbf{M}^{(i\to j)}, \mathbf{N}^{(j)})$.
- $\mathbf{c}_L^{(\xi)} \leftarrow \mathsf{ReEnc}\big((rk^{(0\to 1)}, \ldots, rk^{(L-1\to L)}), \mathbf{c}_0^{(\xi)}\big)$: In order to achieve *multi-bit and multi-hop re-encryption* by branching programs, we first parse the ciphertext $\mathbf{c}$ into t pieces, i.e., we set

$$\vec{\mathbf{w}}^{(1)} := (2^1\mathbf{u}_N, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0}),$$
$$\vec{\mathbf{w}}^{(2)} := (2^2\mathbf{u}_{2N}, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0}),$$
$$\cdots$$
$$\vec{\mathbf{w}}^{(\xi)} := (2^\xi\mathbf{u}_{\xi N}, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0}),$$
$$\cdots$$
$$\vec{\mathbf{w}}^{(t)} := (2^t\mathbf{u}_{tN}, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0}),$$

where $\mathbf{u}^{(\xi)} := [1, 0, \ldots, 0]^T \in \mathbb{Z}_q^{1\times(n+t)N}$ for $\xi \in [t]$. Let us denote the re-encryption key as $rk_1 := \mathbf{X}_1, \ldots, rk_L := \mathbf{X}_L$. Next, we initiate $\mathsf{BPOTF}^{\mathsf{Pred}_\mathcal{C}}$.

For every step $t = 1, \ldots, L$, the algorithm $\mathsf{BPOTF}^{\mathsf{Pred}_\mathcal{C}}$ proceeds as follows:

1) Computes the constants $\gamma_{t,i,0}^{(\xi)}$ and $\gamma_{t,i,1}^{(\xi)}$ by executing the algorithm $\mathsf{BPOTF}^{\mathsf{Pred}_\mathcal{C}}$, then uses the constants to obtain the next layer of the BPs

$$\Big(\big(\gamma_{t,1,0}^{(\xi)}, \ldots, \gamma_{t,5,0}^{(\xi)}\big), \big(\gamma_{t,1,1}^{(\xi)}, \ldots, \gamma_{t,5,1}^{(\xi)}\big), var(t)\Big).$$

2) For every $i = 1, \ldots, 5$, the algorithm obtains the encryption of next state by homomorphically computing the following equation:

$$\mathbf{w}_{t,i}^{(\xi)} = (\mathbf{G} - \mathbf{X}_{vat(t)}) \cdot \mathbf{G}^{-1}(\mathbf{w}_{t-1,\gamma_{t,i,0}^{(\xi)}}) + \mathbf{X}_{vat(t)} \cdot \mathbf{G}^{-1}(\mathbf{w}_{t-1,\gamma_{t,i,1}^{(\xi)}}),$$

where $\mathbf{X}_{var(t)} := \sum \vec{\mathcal{S}}_{var(t)}$. We stress that $\mathbf{X}_{var(t)}$ can be viewed as the sum of the encrypted bits of the secret key $\vec{\mathcal{S}}_{var(t)}$. Finally, the algorithm outputs the ciphertext $\mathbf{c}_L^{(\xi)} := \{\mathbf{w}_{L,1}^{(\xi)}\}_{\xi\in[t]}$. Namely that, we have that $\mathbf{c}_L = \{\mathbf{c}_L^{(1)}, \ldots, \mathbf{c}_L^{(t)}\}$.

- $\mathsf{Dec}(\mathsf{params}, sk, \mathbf{c})$: there exist two cases waiting for us to consider.
  - The input-side decryption, for $\xi := 1$ to t, calculates the following results

$$\mu_\xi := \langle \mathbf{c}, \mathbf{S}[\xi] \rangle := \mu_\xi \cdot 2^{\ell_q - 1} + \mathbf{re}_\xi \,(\mathrm{mod}\ q).$$

  If $\mathsf{noise}_{\mathbf{S}[\xi],\mu_\xi}(\mathbf{c}_\xi) := \|\mathbf{re}_\xi\| < q/8$, then obtains $\mu_\xi = 1$ and otherwise obtains $\mu_\xi = 0$. Lastly, repeats t times and outputs $(\mu_1, \ldots, \mu_t)$.
  - The output-side decryption, *L-level decryption algorithm*, after L-hop re-encryptions were finished from player $i \in [L]$ to player $j \geq i+1$, the re-encryption ciphertext is the form of $\mathbf{c}_j := \mathbf{X}^{(i\to j)} \cdot \mathbf{G}^{-1}(\mathbf{c}_i)$. Hence, we can obtain the results of $\langle \mathbf{c}_j, \mathbf{s}_j \rangle \,(\mathrm{mod}\ q)$. Particularly, for each 1-hop re-encryption from player $i := L-1$ to player $j := L$, we can obtain the re-encryption ciphertext as the form of $\mathbf{c}_L := \mathbf{X}^{((L-1)\to L)} \cdot \mathbf{G}^{-1}(\mathbf{c}_{(L-1)})$. Thus, outputs the results of $\langle \mathbf{c}_L, \mathbf{S}_L[\xi] \rangle \,(\mathrm{mod}\ q)$, i.e., $\langle \mathbf{c}_L, \mathbf{S}_L[\xi] \rangle = \mu_\xi 2^{\ell_q - 1} + \mathbf{error}$, if $\|\mathbf{error}\| \leq 1/8$, then outputs $\mu_\xi = 1$ and otherwise $\mu_\xi = 0$. Last, outputs $(\mu_1, \ldots, \mu_t)$.

We can prove the correctness and security by naturally extending the proof of Lemma 4.7 and Theorem 4.8 respectively. Hence, we omit the further details.

4.4 Comparison

In this section, we analyze our scheme's performance. As mentioned earlier in the Table 2, the previous schemes cannot supports homomorphic operations and must encrypt multi-bit. Meanwhile, few works [12], [26] achieve multi-hop but they obtain multi-hop by repeating the single-hop procedure. Our scheme supports single-bit and single-hop homomorphic PRE, single-bit and multi-hop homomorphic PRE, and multi-bit and multi-hop homomorphic PRE. In short, once obtain the single-bit and single-hop PRE scheme, we can adopt the BP to obtain the single-bit and multi-hop PRE and encode t

TABLE 2
The Comparison of LWE-Based PRE and Our Schemes

Scheme	Msg	$\|pk\|$	$\|sk\|$	$\|\mathrm{Ct}\|(\|\text{Re-Ct}\|)$	$\|rk\|$
Kirshanova [13]	nk	$(n+nk)(m+nk)\log q$	$m2nk\log q$	$n^2\log q \cdot m\log 2q$	$m^2\log q$
Fan et al. [26]	nk	$n(m+2k)\log q$	$m2k\log q$	$(2m+nk)\log q$	$2(nk)^2\log q$
Xagawa [10]	t	$(n+t)m\log q$	$nt\log q$	$(n+t)\log q$	$nt\log q$
Aono et al. [11]	t	$nt\log q$	$nt\log q$	$(n+t)\log q$	$(n\ell_q+t)(n+t)\log q$
Nishimaki et al. [14]	t	$mt\log q$	$nt\log q$	$nt\log q$	$n\ell_q(n+t)\log q$
Ma et al. [19]	1	$nn\ell_q\log q$	$n\log q$	$nn\ell_q\log q$	$n(n\ell_q+n^2\ell_q)\log q$
Chandran et al. [12]	1	$m(n+1)\log q$	$n\log q$	$(n+1)\log q$	$n\ell_q(n+1)\log q$
Our scheme	1	$m(n+1)\log q$	$(n+1)\log q$	$(n+1)\log q$	$(n+1)\big((n+1)\ell_q+m\big)\log q$
Multi-Bit Version	t	$m(n+t)\log q$	$t(n+t)\log q$	$(n+t)\log q$	$(n+t)\big((n+t)\ell_q+m\big)\log q$

- $\|\mathrm{Ct}\|$: *the length of the ciphertext;*
- $\ell_q := \lceil \log q \rceil$;
- $\|\text{Re-Ct}\|$: *the length of the re-encrypted ciphertext.*

BPs with t-bit output by encoding BP for each output bit to obtain the multi-bit and multi-hop PRE. A comparison result is provided in the Table 2, where $m \geq \log n + \lambda$, $\ell_q = \lceil \log q \rceil$, n are the dimension of the vector, we define the length of encrypted messages as t.

As is shown by the Table 2, the length of the public key and secret key of our schemes are the same *magnitude* as the previous work, but compared with the multi-hop PRE schemes [13], [26], the length of cipertext (and re-encrypted ciphertext) is shorter. Although the length of our schemes' re-encryption key is longer than others, our schemes can achieve multiple hops and support homomorphic operations. Moreover, we also note that there exist some lattice-based PRE schemes, but in this paper we only focus on some close related PRE schemes.

5 Conclusion

Most existing efficient PRE schemes are designed over the Diffie-Hellman hard problem, whereas these constructions cannot prevent quantum attacks. Moreover, most of lattice-based PRE schemes are single-hop schemes, whereas single-hop PRE schemes tend not to match the real world. Hence, we propose a new efficient method to achieve multi-hop PRE scheme in this paper. We leave many works for future. For example, very recently, a CCA-secure lattice-based FHE scheme was proposed by Canetti et al. [39], their work inspire us to attempt to achieve a CCA-secure multi-hop homomorphic PRE.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (No.61472097).

References

[1] Y. Li, K. Gai, L. Qiu, M. Qiu, and H. Zhao, "Intelligent cryptography approach for secure distributed big data storage in cloud computing," *Inf. Sci.*, vol. 387, pp. 103–115, 2017.
[2] J. Li, W. Yao, Y. Zhang, H. Qian, and J. Han, "Flexible and fine-grained attribute-based data storage in cloud computing," *IEEE Trans Serv. Comput.*, vol. 10, no. 5, pp. 785–796, Sep./Oct. 2017.
[3] J. Li, X. Lin, Y. Zhang, and J. Han, "KSF-OABE: Outsourced attribute-based encryption with keyword search function for cloud storage," *IEEE Trans. Serv. Comput.*, vol. 10, no. 5, pp. 715–725, Sep./Oct. 2017.
[4] H. Wang, D. He, and S. Tang, "Identity-based proxy-oriented data uploading and remote data integrity checking in public cloud," *IEEE Trans. Inf. Forensics Secur.*, vol. 11, no. 6, pp. 1165–1176, Jun. 2016.
[5] M. Blaze, G. Bleumer, and M. Strauss, "Divertible protocols and atomic proxy cryptography," in *Proc. Int. Conf. Theory Appl. Cryptographic Techn.*, 1998, pp. 127–144.
[6] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," in *Proc. Workshop Theory Appl. Cryptographic Techn.*, 1984, pp. 10–18.
[7] R. Canetti and S. Hohenberger, "Chosen-ciphertext secure proxy re-encryption," in *Proc. ACM Conf. Comput. Commun. Secur.*, 2007, pp. 185–194.
[8] J. Shao and Z. Cao, "CCA-secure proxy re-encryption without pairings," in *Proc. Int. Workshop Public Key Cryptography*, 2009, pp. 357–376.
[9] J. Weng, Y. Zhao, and G. Hanaoka, "On the security of a bidirectional proxy re-encryption scheme from PKC 2010," in *Proc. Int. Workshop Public Key Cryptography*, 2010, pp. 284–295.
[10] K. Xagawa, "Cryptography with lattices," Theses, Tokyo Institute of Technology, 2010. [Online]. Available: http://xagawa.net/pdf/2010Thesis.pdf
[11] Y. Aono, X. Boyen, L. T. Phong, and L. Wang, "Key-private proxy re-encryption under LWE," in *Proc. Int. Conf. Progress Cryptology*, 2013, pp. 1–18.
[12] N. Chandran, M. Chase, F.-H. Liu, R. Nishimaki, and K. Xagawa, "Re-encryption, functional re-encryption, and multi-hop re-encryption: A framework for achieving obfuscation-based security and instantiations from lattices," in *Proc. Int. Workshop Public Key Cryptography*, 2014, pp. 95–112.
[13] E. Kirshanova, "Proxy re-encryption from lattices," in *Proc. Int. Conf. Public-Key Cryptography*, 2014, pp. 77–94.
[14] R. Nishimaki and K. Xagawa, "Key-private proxy re-encryption from lattices, revisited," *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.*, vol. 98, no. 1, pp. 100–116, 2015.
[15] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proc. Annu. ACM Symp. Theory Comput.*, 2009, pp. 169–178.
[16] O. Regev, "On lattices, learning with errors, random linear codes, and cryptography," in *Proc. Annu. ACM Symp. Theory Comput.*, 2005, pp. 84–93.
[17] R. Lindner and C. Peikert, "Better key sizes (and attacks) for LWE-based encryption," in *Proc. Int. Conf. Topics Cryptology*, 2011, pp. 319–339.
[18] Y. Ishai and A. Paskin, "Evaluating branching programs on encrypted data," in *Proc. Theory Cryptography Conf.*, 2007, pp. 575–594.
[19] C. Ma, J. Li, and W. Ouyang, "A homomorphic proxy re-encryption from lattices," in *Proc. Int. Conf. Provable Secur.*, 2016, pp. 353–372.
[20] Z. Brakerski and R. Perlman, "Lattice-based fully dynamic multi-key FHE with short ciphertexts," in *Proc. Annu. Int. Cryptology Conf. Advances Cryptology*, 2016, pp. 190–213.
[21] D. A. Barrington, "Bounded-width polynomial-size branching programs recognize exactly those languages in NC1," *J. Comput. Syst. Sci.*

[22] Z. Brakerski and V. Vaikuntanathan, "Lattice-based FHE as secure as PKE," in *Proc. Conf. Innovations Theoretical Comput. Sci.*, 2014, pp. 1–12.
[23] C. Gentry, A. Sahai, and B. Waters, "Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based," in *Proc. Annu. Int. Cryptology Conf. Advances Cryptology*, 2013, pp. 75–92.
[24] X. Liang, Z. Cao, H. Lin, and J. Shao, "Attribute based proxy re-encryption with delegating capabilities," in *Proc. Int. Symp. Inf. Comput. Commun. Secur.*, 2009, pp. 276–286.
[25] S. S. M. Chow, J. Weng, Y. Yang, and R. H. Deng, "Efficient unidirectional proxy re-encryption," in *Proc. Int. Conf. Cryptology Africa*, 2010, pp. 316–332.
[26] X. Fan and F.-H. Liu, "Various proxy re-encryption schemes from lattices," *IACR Cryptology ePrint Archive, Report 2016/278*. [Online]. Available: http://eprint.iacr.org/2016/278
[27] C. Gentry, C. Peikert, and V. Vaikuntanathan, "Trapdoors for hard lattices and new cryptographic constructions," in *Proc. Annu. ACM Symp. Theory Comput.*, 2008, pp. 197–206.
[28] D. Micciancio and C. Peikert, "Trapdoors for lattices: Simpler, tighter, faster, smaller," in *Proc. Annu. Int. Conf. Theory Appl. Cryptographic Techn.*, 2012, pp. 700–718.
[29] Z. Brakerski, "Fully homomorphic encryption without modulus switching from classical GapSVP," in *Proc. Annu. Cryptology Conf. Advances Cryptology*, 2012, pp. 868–886.
[30] Z. Brakerski and V. Vaikuntanathan, "Efficient fully homomorphic encryption from (standard) LWE," in *Proc. Annu. Symp. Found. Comput. Sci.*, 2011, pp. 97–106.
[31] V. Lyubashevsky, "Lattice signatures without trapdoors," in *Proc. Annu. Int. Conf. Theory Appl. Cryptographic Techn.*, 2012, pp. 738–755.
[32] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(Leveled) fully homomorphic encryption without bootstrapping," in *Proc. Innovations Theoretical Comput. Sci. Conf.*, 2012, pp. 309–325.
[33] Y. Aono, T. Hayashi, L. T. Phong, and L. Wang, "Efficient key-rotatable and security-updatable homomorphic encryption," in *Proc. ACM Int. Workshop Secur. Cloud Comput.*, 2017, pp. 35–42.
[34] Z. Li, C. Ma, G. Du, and O. Weiping, "Dual LWE-based fully homomorphic encryption with errorless key switching," in *Proc. IEEE 22nd Int. Conf. Parallel Distrib. Syst.*, 2016, pp. 1169–1174.
[35] P. Mukherjee and D. Wichs, "Two round multiparty computation via multi-key FHE," in *Proc. Annu. Int. Conf. Theory Appl. Cryptographic Techn.*, 2016, pp. 735–763.
[36] Z. Li, C. Ma, E. Morais, and G. Du, "Multi-bit leveled homomorphic encryption via Dual.LWE-based," in *Proc. Int. Conf. Inf. Secur. Cryptology*, 2016, pp. 221–242.
[37] D. Wichs and G. Zirdelis, "Obfuscating compute-and-compare programs under LWE," Cryptology ePrint Archive, Report 2017/276, 2017. [Online]. Available: http://eprint.iacr.org/2017/276
[38] R. Hiromasa, M. Abe, and T. Okamoto, "Packing messages and optimizing bootstrapping in GSW-FHE," in *Proc. Int. Workshop Public Key Cryptography*, 2015, pp. 699–715.
[39] R. Canetti, S. Raghuraman, S. Richelson, and V. Vaikuntanathan, "Chosen-ciphertext secure fully homomorphic encryption," in *Proc. Int. Workshop Public Key Cryptography*, 2017, pp. 213–240.

Zengpeng Li is currently working toward the PhD degree at Harbin Engineering University, China. His primary research interests include cryptography, protocol, and information security. In particular, his research focus is lattice-based cryptography. He is a student member of the IEEE, the CCF, the IEICE, and the CACR.

Chunguang Ma received the PhD degree in cryptography from the College of Computer Science and Technology, Beijing University of Posts and Telecommunications, China, in 2005. He is currently a professor at Harbin Engineering University. His main research interests include cryptography and information security, in particular, cryptographic protocols.

Ding Wang currently is a postdoc at Peking University, China. He has been involved in the community as a TPC member and a reviewer for more than 50 international conferences and journals. His works appear in prestigious venues like ACM CCS, IEEE/IFIP DSN, ESORICS, and the *IEEE Transactions on Dependable and Secure Computing*, and three of them are selected as "ESI highly cited papers". His research interests include cryptography and system security.

Queries to the Author

Q1. There was a discrepancy in the PDF and the source file. We have followed the source file.
Q2. Please provide complete bibliographic details in Ref. [21].
Q3. Please provide the publication year in Ref. [26].