Constant-Size Unbounded Multi-Hop Fully Homomorphic Proxy Re-Encryption from Lattices

Feixiang Zhao[1,2], Huaxiong Wang[2], and Jian Weng[1⋆]

[1] Jinan University, Guangzhou, China, 510632
{fxzhao37, cryptjweng}@gmail.com
[2] Nanyang Technological University, Singapore, 637371
hxwang@ntu.edu.sg

Abstract. Proxy re-encryption is a cryptosystem that achieves efficient encrypted data sharing by allowing a proxy to transform a ciphertext encrypted under one key into another ciphertext under a different key. Homomorphic proxy re-encryption (HPRE) extends this concept by integrating homomorphic encryption, allowing not only the sharing of encrypted data but also the homomorphic computations on such data. The existing HPRE schemes, however, are limited to a single or bounded number of hops of ciphertext re-encryptions. To address this limitation, this paper introduces a novel lattice-based, unbounded multi-hop fully homomorphic proxy re-encryption (FHPRE) scheme, with constant-size ciphertexts. Our FHPRE scheme supports an unbounded number of re-encryption operations and enables arbitrary homomorphic computations over original, re-encrypted, and evaluated ciphertexts. Additionally, we propose a potential application of our FHPRE scheme in the form of a non-interactive, constant-size multi-user computation system for cloud computing environments.

Keywords: Proxy re-encryption · Fully homomorphic encryption · Unbounded multi-hop · Bootstrapping · Lattice-based cryptography.

1 Introduction

Consider a scenario where Alice (delegator) wishes to securely share some encrypted data with Bob (delegatee) from a cloud server. One trivial but risky method is that Alice directly tells Bob her decryption key, compromising the security of all her encrypted data. Alternatively, Alice can download, decrypt, encrypt again with Bob's public key, then re-upload the data. This approach is inefficient and bandwidth-heavy. Proxy re-encryption (PRE), introduced by Blaze et al. [4], offers an elegant solution that allows Alice to delegate the decryption rights of her ciphertext to Bob. Specifically, given a re-encryption key from Alice, a semi-trust third party proxy can transform a ciphertexts under Alice's public key to a ciphertext decryptable by Bob, ensuring the proxy learns no knowledge about the underlying data. PRE has important applications in

⋆ Corresponding author.

cloud computing, secure email forwarding [32], and distributed file system [3]. According to the number of re-encryption can be performed, PRE is categorized into multi-hop PRE (MH-PRE) [9, 22], where ciphertexts can be successively re-encrypted (e.g., from Alice to Bob, then to Carol), and single-hop PRE [2, 19, 21], where further re-encryptions are not supported.

Homomorphic encryption (HE) enables computations on encrypted data without the need of decryption. Gentry proposed the first fully homomorphic encryption (FHE) scheme [17] that supports evaluating arbitrary functions over encrypted data by using the bootstrapping technique. Specifically, bootstrapping allows a ciphertext to be refreshed to a lower error state by homomorphically evaluating the decryption circuit on it with an encrypted secret key.

Homomorphic proxy re-encryption (HPRE) is an extension of PRE that supports homomorphic computation. HPRE realizes not only efficient secure data sharing but also the computability of encrypted data, making HPRE particularly potential in cloud computing, where data security is increasingly important. However, the existing HPRE schemes are limited to support single-hop [23, 29] or bounded multi-hop (with a predefined limit) [25, 26] re-encryptions. An unbounded hop HPRE scheme, offering an unrestricted number of re-delegations and enabling continuous computations, would be much more desirable and practical than a bounded one. Despite the requirement to support homomorphic computation, there are no known lattice-based unbounded multi-hop PRE schemes by far. This gap is an essential factor hindering the development of PRE.

1.1 Our Contributions

To address the problems of current HPRE schemes that are constrained by a limited or bounded number of re-encryption hops, this paper introduces a lattice-based unbounded multi-hop FHPRE (MH-FHPRE) scheme. This scheme has the advantages in: (1) supporting an unbounded number of re-encryptions, (2) enabling arbitrary circuit evaluations on original, re-encrypted and evaluated ciphertexts, and (3) maintaining constant-size ciphertexts throughout the computation process. In the following section, we present an overview of our MH-FHPRE scheme:

- This scheme starts with a learning with errors (LWE) form of public key encryptions:

$$\boldsymbol{c} = (\mathbf{a} = \mathbf{A}\mathbf{r} + \mathbf{e},\ b = \mathbf{r}^\top \mathbf{b} + \frac{q}{4}m + e) \in \mathbb{Z}_q^{n+1},$$

  and the decryption runs as: $m = \left\lfloor \frac{4}{q}(b - \mathbf{a} \cdot \mathbf{s}) \right\rceil$.
- This form supports the FHEW-style homomorphic NAND evaluation as: $\boldsymbol{c}_f = (\mathbf{a}, b) = \mathsf{Eval.NAND}((\mathbf{a}_0, b_0), (\mathbf{a}_1, b_1)) = (-\mathbf{a}_0 - \mathbf{a}_1, \frac{5}{8}q - b_0 - b_1)$.
- A single-hop proxy re-encryption is achieved by a re-encryption key:

$$rk_{i \to j} = \begin{bmatrix} (\mathbf{A}_j \mathbf{R}_j)^\top + (\mathbf{E}'_j)^\top & \mathbf{R}_j^\top \mathbf{b}_j - \mathsf{P2}(\mathbf{s}_i) + (\mathbf{e}'_j)^\top \\ \mathbf{0}_{1 \times n} & 1 \end{bmatrix},$$

which can transform the ciphertext from user i to user j as: $\boldsymbol{c}'_j = (\mathbf{a}'_j, b'_j)$ where $((\mathbf{a}'_j)^\top, b'_j) \leftarrow [\mathsf{BD}(\mathbf{a}_i^\top)|b_i] \cdot rk_{i\to j}$. More details of the algorithms BD and $\mathsf{P2}$ will be introduced in Section 2.5.

- The problem of this single-hop re-encrypted ciphertext $\boldsymbol{c}'_j$ is the accumulation of errors, which is uncontrollable and prevents subsequent re-encryptions and computations. To resolve this problem, we adapt this ciphertext to a new one, $\boldsymbol{c}''_j = (\mathbf{a}''_j, b''_j)$, on which the FHEW-style fast bootstrapping can work.
- The adapted ciphertext can be refreshed by bootstrapping: $\boldsymbol{c}_j = (\mathbf{a}_j, b_j) \leftarrow \mathsf{Bootstrap}(c'')$, with reduced noise for further re-encryptions or computations.

Based on the proposed FHPRE, we suggest a potential application of our scheme in the form of a multi-user computation system in cloud computing environments. This system addresses the scenarios involving k users wishing to collaboratively compute a function f over a set of encrypted data originating from different users, i.e. $\boldsymbol{c}_1, \cdots, \boldsymbol{c}_k$ where $\boldsymbol{c}_i = \mathsf{Enc}_{pk_i}(m_i)$, to obtain $f(m_1, \cdots, m_k)$. Multi-key homomorphic encryption (MKHE) [27, 31, 6, 11] offers a solution to such a task. However, these MKHE schemes face two major limitations: (1) It necessitates interactive protocols like MPC protocol for joined decryption, and (2) the ciphertext size and computation complexity of the joined ciphertexts grows linearly or quadratically with the number of involved users. In Section 7, we present that our FHPRE scheme can be used to construct a non-interactive and scalable multi-user computation system.

1.2 Related Works

The notion of PRE was first introduced by Blaze et al. [4] with a single-hop bidirectional scheme based on DDH assumption. Following this study, Aono et al. [2] and Kirshanova [21] made significant breakthroughs by constructing the first CPA secure and first CCA secure single-hop unidirectional PRE schemes from lattice, which is believed to be quantum secure. This led to further developments by Chandran el al. [10] who proposed a multi-hop PRE scheme that is collusion resistant. Jiang [20] also proposed a lattice-based multi-hop PRE scheme.

Gentry [17] first presents the feasibility of FHE that supports evaluation of arbitrary functions on encrypted data by using the bootstrapping theorem. Following this line, a series of optimized (leveled) HE schemes [5, 7, 8, 13] were developed. Subsequently, Gentry el al. [18] proposed the GSW scheme with "quasi-additive" error growth in homomorphic multiplication, leading to a more practical bootstrapping [1]. A substantial progress was made by Ducas and Micciancio [16], and Chillotti et al. [14], who presented a fast bootstrapping procedure taking less than a second, making an important step towards practical FHE for arbitrary NAND circuits. To improve the bootstrapping efficiency, Micciancio and Sorrell [30] suggested a ring packing and amortized bootstrapping method.

Zhong el al. [35] introduced the first HPRE scheme under GCD assumption. Ma et al. [29] later provided the first lattice-based single-hop unidirectional HPRE scheme. Improved from [29], Li et al. [23] proposed a more flexible single-hop leveled HPRE where the maximum depth of the evaluated circuit had to

be predefined. Following this line, Li et al. [25] gave a lattice-based multi-hop identity-based HPRE via branching program. Recently, [24] proposed a LWE-based multi-hop leveled HPRE scheme with collusion resistance. However, these multi-hop schemes only support a bounded number of re-encryption operations, thereby fixing the maximum number of re-encryptions after setup. This causes a loss of system's flexibility. Table 1 summarizes the results of these classical HPRE schemes.

Table 1: Comparison of the lattice-based HPRE schemes

Schemes	Assumption	FHE	Hop	Security
MLO16 [29]	LWE	✓	Single	CPA
LMZ+19 [23]	LWE	Leveled HE	Single	CPA
LMW17 [25]	LWE	✓	Multi(bounded)	CPA
LQZ+21 [24]	LWE	Leveled HE	Multi(bounded)	CPA
Our Scheme	LWE	✓	Multi(unbounded)	CPA

2 Preliminaries

2.1 Notations

Throughout this paper, we use $[n]$ to denote the set $\{1, 2, \cdots, n\}$ for a positive integer n. We use $x \leftarrow X$ to denote the process of sampling a value x over the distribution X, and $x \xleftarrow{\$} X$ to denote the uniformly random sampling process. We use bold lower-case letters like $\mathbf{a}$ to denote column vectors and bold upper-case letters like $\mathbf{A}$ to denote matrices. We denote the transpose of a vector (matrix) as $\mathbf{a}^\top(\mathbf{A}^\top)$. The horizontal concatenation of elements $a_1, \cdots, a_m$ is denoted as its ordered set like $(a_1, a_2, \cdots, a_m)$ or sometimes as $[a_1|\cdots|a_m]$ for clearer presentation. We define $\|\mathbf{a}\|$ as the l_2 norm of the vector $\mathbf{a}$ and $\|\mathbf{a}\|_\infty$ as $\|a_i\|_{\max}$. We denote the process of rounding x to its nearest integer as $\lceil x \rfloor$, rounding x up as $\lceil x \rceil$, and rounding x down as $\lfloor x \rfloor$. We identify the set of integers modulo q as $\mathbb{Z}_q$ and the group of invertible elements modulo q as $\mathbb{Z}_q^*$.

2.2 Gaussian Distributions

A variable X has sub-Gaussian distribution with a parameter $\sigma > 0$ if for all $t \in \mathbb{R}$, it holds $E[e^{2\pi tX}] \leq e^{\pi\sigma^2/t^2}$. A sub-Gaussian distribution X is called B-bounded if its support is in $[-B, B]$ with a parameter $\sigma = B\sqrt{2\pi}$. A variable X is called sub-Gaussian with a parameter σ if $\Pr[|X| \geq t] \leq 2e^{-\pi t^2/\sigma^2}$ holds for all $t \geq 0$. A vector $\mathbf{x}$ (matrix $\mathbf{X}$) is called sub-Gaussian with parameter σ if for unit vectors $\mathbf{v}, \mathbf{w}$, it satisfies that $\langle \mathbf{v}, \mathbf{x} \rangle$ ($\mathbf{v}^\top \mathbf{X} \mathbf{w}$) is also sub-Gaussian with parameter σ.

2.3 Cyclotomic Rings

Let $\Phi_N(X) = \prod_{i \in \mathbb{Z}_N^*}(X - \omega_N^i)$ be the Nth cyclotomic polynomial for which ω_N is the complex Nth root of unity. We have $\Phi_N(X)$ is a monic polynomial of degree $\varphi(N) = |\mathbb{Z}_N^*|$ and $\Phi_{2N}(x) = X^N + 1$ for any power of 2 integer N. The Nth cyclotomic ring is $\mathcal{R}_N = \mathbb{Z}[X]/\Phi_N(X)$. For an ring element $a \in \mathcal{R}_N = a_0 + a_1 X + \cdots, + a_{\varphi(N)} X^{\varphi(N)}$, we define the Euclidean length of a as $\|a\| = \sqrt{\sum_i |a_i|^2}$ and the spectral norm of a matrix $\mathbf{R} \in \mathcal{R}_N^{n \times m}$ as $s_1(\mathbf{R}) = \sup_{\mathbf{x} \in \mathcal{R}^m \setminus \mathbf{0}} \|\mathbf{R} \cdot \mathbf{x}\| / \|\mathbf{x}\|$. We define $\mathcal{R} = \mathcal{R}_{2N} = \mathbb{Z}[X]/(X^N + 1)$ and write $\mathcal{R}_q$ to denote its residue ring, $\mathbb{Z}_q[X]/(X^N + 1)$.

We define the map η mapping a ring element to its coefficient vector. For a ring element $a \in \mathcal{R} = \sum_{i=0}^{N-1} a_i X^i$, $\eta(a) \to (a_0, \cdots, a_{N-1})$. We also define the map $M : M(a) \to (\eta(a \cdot X^0), \cdots, \eta(a \cdot X^{N-1}))$ for $a \in \mathcal{R}$. We have $M(a) \cdot \eta(b) = \eta(a \cdot b)$, which will be used in the following LWE extraction algorithms.

Fact 1 (Adapted from [15] Fact 6) *If $\mathcal{D}$ is a sub-Gaussian distribution of parameter σ over $\mathcal{R}$, and $\mathbf{R} \leftarrow \mathcal{D}^{m \times k}$ has independents coefficients, we have $s_1(\mathbf{R}) \leq \sigma\sqrt{N} \cdot O(\sqrt{m} + \sqrt{k} + \omega(\sqrt{\log N}))$ with overwhelming probability.*

2.4 (Ring) Learning with Errors Problem

The decisional LWE (DLWE) problem introduced by Regev [33] is defined as:

Definition 1 *For a secret $\mathbf{s} \leftarrow \mathbb{Z}_q^n$, integers n, m, q, and a noise distribution χ over $\mathbb{Z}$, the $\mathsf{DLWE}_{n,q,\chi}$ problem is to distinguish the following two distributions:*

$$(\mathbf{a}_i, b_i = \mathbf{a}_i \cdot \mathbf{s} + e_i) \quad \text{and} \quad (\mathbf{a}_i, u_i)$$

where $\mathbf{a}_i \xleftarrow{\$} \mathbb{Z}_q^n, e_i \leftarrow \chi$, and $u_i \xleftarrow{\$} \mathbb{Z}_q$ are independently chosen. The DLWE assumption is that the $\mathsf{DLWE}_{n,q,\chi}$ problem is infeasible.

The decisional ring-LWE (RLWE) problem [28] is defined as:

Definition 2 *For a secret $s \leftarrow \mathcal{R}_q$, a ring $\mathcal{R}_q$, and a noise distribution χ over $\mathbb{Z}$, the decisional $\mathsf{RLWE}_{\mathcal{R},q,\chi}$ problem is to distinguish the following two distributions:*

$$(a_i, b_i = a_i \cdot s + e_i) \quad \text{and} \quad (a_i, u_i)$$

where $a_i \xleftarrow{\$} \mathcal{R}_q$, $u_i \xleftarrow{\$} \mathbb{Z}_q$, and each coefficient of $e_i \in \mathcal{R}_q$ is chosen from χ. The RLWE assumption is that the $\mathsf{RLWE}_{\mathcal{R},q,\chi}$ problem is infeasible.

2.5 BD and P2 Algorithms

In the following sections, we use two useful algorithms BD and P2 [5] :

- $\mathsf{BD}(\mathbf{x})$: Takes a vector $\mathbf{x} \in \mathbb{Z}_q^n$, outputs the vector $\mathbf{y} = (\mathbf{y}_0, \mathbf{y}_1, \cdots, \mathbf{y}_{\lceil \log q \rceil - 1}) \in \{0,1\}^{n\lceil \log q \rceil}$, such that $\mathbf{x} = \sum_{i=0}^{\lceil \log q \rceil - 1} 2^i \cdot \mathbf{y}_i$.
- $\mathsf{P2}(\mathbf{v})$: Takes a vector $\mathbf{v} \in \mathbb{Z}_q^n$, outputs the vector $\mathbf{w} = (\mathbf{v}, 2\mathbf{v}, \cdots, 2^{\lceil \log q \rceil - 1}\mathbf{v}) \in \mathbb{Z}_q^{n\lceil \log q \rceil}$.

It holds that: for any vectors $\mathbf{x}, \mathbf{y} \in \mathbb{Z}_q^n$, $\mathsf{BD}(\mathbf{x}) \cdot \mathsf{P2}(\mathbf{y}) = \mathbf{x} \cdot \mathbf{y} \pmod q$.

2.6 LWE Public Key Encryption

In this section, we introduce a basic LWE public key encryption scheme on which the bootstrapping works. The LWE encryption public key scheme is parameterized by a dimension n, a ciphertext modulus q, $m = \Theta(n \log q)$, a B-bounded noise distribution χ over $\mathbb{Z}$, and a message modulus $t \leq q$ (we use $t = 4$). The secret key of the scheme is a ternary vector $\mathbf{s} \leftarrow \{-1, 0, 1\}^n$ and the public key is $(\mathbf{A}, \mathbf{b})$, where $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$ and $\mathbf{b} \leftarrow \mathbf{A}^\top \mathbf{s} + \mathbf{e}$ for some $\mathbf{e} \leftarrow \chi^m$. The encryption of a message $\mu \in \{0, 1\}$ is computed by:

$$\mathsf{LWE}_{\mathbf{s}}^{t/q}(\mu) = (\mathbf{a}, b) = (\mathbf{A}\mathbf{r} + \mathbf{e}_1, \mathbf{r}^\top \mathbf{b} + \frac{q}{t}\mu + e_2) \in \mathbb{Z}_q^{n+1}.$$

where $\mathbf{r} \xleftarrow{\$} \{-1, 0, 1\}^m$, $\mathbf{e}_1 \leftarrow \chi^m, e_2 \leftarrow \chi$. The ciphertext can be decrypted by:

$$\mu' = \left\lfloor \frac{t}{q}(b - \mathbf{a} \cdot \mathbf{s}) = \frac{t}{q}(\frac{q}{t}\mu + \mathbf{r}^\top \mathbf{e} + e_2 - \mathbf{e}_1 \cdot \mathbf{s}) \right\rceil \pmod t$$

A correct decryption requires that $\frac{t}{q}(\mathbf{r}^\top \mathbf{e} + e_2 - \mathbf{e}_1 \cdot \mathbf{s}) < \frac{1}{2}$.

2.7 Key Switching

Key switching is a procedure that transforms a LWE ciphertext under a key $\mathbf{s}'$ to a LWE ciphertext under another key $\mathbf{s}$, encrypting the same message. Let B_{ks} be a key switching base, $\mathbf{k}_{h,i,j}^{(\mathcal{S})} \in \mathsf{LWE}_{\mathbf{s}}^{q/q}(hs_i' B_{\mathsf{ks}}^j)$ be an encryption of $hs_i' B_{\mathsf{ks}}^j$ under $\mathbf{s}$ where $h \in [0, B_{\mathsf{ks}} - 1], i \in [n], j \in [0, d_{\mathsf{ks}} - 1]$ for $d_{\mathsf{ks}} = \lceil \log_{B_{\mathsf{ks}}} q \rceil$. Taking $\boldsymbol{c}' = (\mathbf{a}', b') \in \mathsf{LWE}_{\mathbf{s}'}^{t/q}(\mu)$ and an auxiliary key switch key $\mathcal{K}^{(\mathcal{S})} = \{\mathbf{k}_{h,i,j}^{(\mathcal{S})}\}$, the key switching algorithm $\mathsf{KeySW}(\mathcal{K}^{(\mathcal{S})}, (\mathbf{a}', b')) \to (\mathbf{a}, b)$ proceeds as follows:

- For $i \in [n], j \in [0, \cdots, d_{\mathsf{ks}}]$, compute $a_{i,j}'$ such that $a_i' = \sum_j a_{i,j}' B_{\mathsf{ks}}^j$.
- Output $(\mathbf{a}, b) = (0^n, b') - \sum_{i,j} \mathbf{k}_{a_{i,j}', i, j}^{(\mathcal{S})}$.

Fact 2 (Adapted from [16] Lemma 6) *For a ciphertext $\boldsymbol{c}' \in \mathsf{LWE}_{\mathbf{s}'}^{t/q}(\mu)$ with a sub-Gaussian error distribution with parameter α and a key switch key $\mathcal{K}^{(\mathcal{S})} = \{\mathsf{LWE}_{\mathbf{s}}^{q/q}(hs_i' B_{ks}^j)\}$ with a sub-Gaussian error distribution with parameter σ, the output of $\mathsf{KeySW}(\mathcal{K}^{(\mathcal{S})}, \boldsymbol{c}) \in \mathsf{LWE}_{\mathbf{s}}^{t/q}(\mu)$ has a sub-Gaussian error distribution with parameter $\sqrt{\alpha^2 + nd_{\mathsf{ks}}\sigma^2}$.*

Note that key switching is different from re-encryption. Key switching is used for evaluating encrypted data, involving the conversion of ciphertexts between two different decryption keys, both of which are generated by the same user. In contrast, re-encryption is intended to achieve data sharing, involving the transformation of ciphertexts between two keys from different users.

2.8 Modulus Switching

Here we define a scale rounding function as: $[x]_{p/q} = \lfloor (q/p)x \rceil$. We have $|\lfloor (q/p)x \rceil - (q/p)x| < 1/2$. The modulus switching algorithm that transforms a LWE ciphertext from modulus p to modulus q is defined as:

$$\mathsf{ModSW}_{p/q}((\mathbf{a}, b)) = ([a_1]_{p/q}, [a_2]_{p/q}, \cdots, [a_n]_{p/q}, [b]_{p/q}).$$

According to the central limit heuristic theorem, we have the following fact:

Fact 3 (Adapted from [16] Lemma 5) *For any $\mathbf{s} \leftarrow \mathbb{Z}_q^n$, a sub-Gaussian error distribution χ with parameter σ, and a ciphertext $\boldsymbol{c} = (\mathbf{a}, b) \in \mathsf{LWE}_{\mathbf{s}}^{t/p}(m)$, The output of $\mathsf{ModSW}_{p/q}(\boldsymbol{c})$ is a ciphertext in $\mathsf{LWE}_{\mathbf{s}}^{t/q}(m)$ with an error sub-Gaussian distribution with parameter $\sqrt{(q/p)^2\sigma^2 + (\|\mathbf{s}\|^2 + 1)/12}$.*

3 Homomorphic Computation and Bootstrapping

In this section we provide an introduction of the homomorphic computation and bootstrapping in our scheme. Following the line of FHEW-style FHE [16, 14], we consider the NAND gate. We denote an encryption of $m \in \mathbb{Z}_t$ under a secret key $\mathbf{s}$, a modulus q and a δ-bound error distribution as $\mathsf{LWE}_{\mathbf{s}}^{t/q}(m, \delta)$.

3.1 Homomorphic NAND Gate Evaluation

For ciphertexts $\boldsymbol{c}_i = (\mathbf{a}_i, b_i) \in \mathsf{LWE}_{\mathbf{s}}^{4/q}(m_i, q/16)$ where $i = 0, 1$ and $m_i \in \{0, 1\}$, the homomorphic NAND gate function for:

$$(\mathsf{LWE}_{\mathbf{s}}^{4/q}(m_0, q/16), \mathsf{LWE}_{\mathbf{s}}^{4/q}(m_1, q/16)) \rightarrow \mathsf{LWE}_{\mathbf{s}}^{2/q}(m_0 \barwedge m_1, q/4)$$

is computed by

$$\boldsymbol{c}_f = (\mathbf{a}, b) = \mathsf{Eval.NAND}((\mathbf{a}_0, b_0), (\mathbf{a}_1, b_1)) = (-\mathbf{a}_0 - \mathbf{a}_1, \frac{5}{8}q - b_0 - b_1)$$

Correctness The new ciphertext $\boldsymbol{c}_f = (\mathbf{a}, b)$ satisfies:

$$b - \mathbf{a} \cdot \mathbf{s} - \frac{q}{2}(1 - m_0 m_1) = \frac{q}{4}(\frac{1}{2} - (m_0 - m_1)^2) - (e_0 + e_1) = \pm\frac{q}{8} - (e_0 + e_1).$$

Since $|e_i| < \frac{q}{16}$, it holds $\left|\pm\frac{q}{8} - (e_0 + e_1)\right| < \frac{q}{4}$, which proves the correctness.

3.2 Bootstrapping

In this section, we first present an overview of the implemented bootstrapping that produces a refreshed ciphertext with reduced error in our FHPRE scheme.

- For an evaluated ciphertext $(\mathbf{a}', b') \in \mathsf{LWE}_{\mathbf{s}}^{2/q}(m_0 \barwedge m_1, q/4)$, we compute $(\mathbf{a}, b) \leftarrow \mathsf{Bootstrap}((\mathbf{a}', b')) \in \mathsf{LWE}_{\mathbf{s}}^{4/q}(m_0 \barwedge m_1, q/16)$ for further operations.
- For an re-encrypted ciphertext generated as $(\mathbf{a}', b') \in \mathsf{LWE}_{\mathbf{s}}^{4/q}(m, q/8)$, we first transform the ciphertext to be bootstrappable by computing $2(\mathbf{a}', b') \in \mathsf{LWE}_{\mathbf{s}}^{2/q}(m, q/4)$. Then we compute the final re-encrypted ciphertext $\boldsymbol{c}_r = (\mathbf{a}, b) \leftarrow \mathsf{Bootstrap}(2(\mathbf{a}', b')) \in \mathsf{LWE}_{\mathbf{s}}^{4/q}(m, q/16)$ for subsequent operations.

Homomorphic Accumulator [16] The bootstrapping is based on the homomorphic accumulator scheme parameterized by a modulus p, a message modulus q, and a degree $N = 2^{\bar{k}}$ where $q|2N$. We use the rings $\mathcal{R}$ and $\mathcal{R}_p$. For correctness, we use $p = B_g^{d_g}$ where $B_g = 3^w$ for some w, and $u \in \mathbb{Z}_p$ for which $|u - p/8| < 1$. We use a gadget matrix $\mathbf{G} = (\mathbf{I}, B_g\mathbf{I}, \cdots, B_g^{d_g-1}\mathbf{I}) \in \mathcal{R}_p^{2d_g \times 2}$, and $\mathbf{G}^{-1}$ maps a matrix to its B_g-base bit representation. The accumulator use a GSW-style encryption scheme as follows:

- $E_{\bar{s}}(m)$: For a key $\bar{s} \in \mathcal{R}_p$, a message $m \in \mathbb{Z}_q$, $\mathbf{a} \xleftarrow{\$} \mathcal{R}_p^{2d_g}$, and $\mathbf{e} \in \mathcal{R}^{2d_g}$ where each coefficient of e_i is sampled from $\bar{\chi}$ with parameter $\bar{\sigma}$, the encryption is:

$$E_{\bar{s}}(m) = (\mathbf{a}, \mathbf{a} \cdot \bar{s} + \mathbf{e}) + uY^m\mathbf{G} \in \mathcal{R}_p^{2d_g \times 2}, \text{ where } Y = X^{2N/q}.$$

Now we define two algorithms used in the accumulator:

- $\mathsf{Init}(\mathsf{ACC} \leftarrow v)$: On input $v \in \mathbb{Z}_q$, set $\mathsf{ACC} = uY^v \cdot \mathbf{G} \in \mathcal{R}_p^{2d_g \times 2}$.
- $\mathsf{Update}(\mathsf{ACC} \xleftarrow{+} \boldsymbol{C})$: On input $\boldsymbol{C} = E_{\bar{s}}(v')$ and $\mathsf{ACC} = E_{\bar{s}}(v)$, proceeds:
  - Compute $\mathbf{G}^{-1}(u^{-1}\mathsf{ACC}) = \mathbf{H} = [\mathbf{H}_1, \cdots, \mathbf{H}_{d_g}]$: $\mathbf{H}_i \in \mathcal{R}^{2d_g \times 2}$ whose coefficients are in $[(1 - B_g)/2, (B_g - 1)/2]$ and $u^{-1}\mathsf{ACC} = \sum_{i=0}^{d_g-1} B_g^i \mathbf{H}_i$.
  - Output $\mathsf{ACC} = [\mathbf{H}_1, \cdots, \mathbf{H}_{d_g}] \cdot \boldsymbol{C}$.

It satisfies that $\mathsf{Update}(\mathsf{ACC} \xleftarrow{+} \boldsymbol{C}) = E_{\mathbf{s}}(v+v') = (\mathbf{a}'', \mathbf{a}'' \cdot \bar{s} + \mathbf{e}'') + uY^{v+v'}\mathbf{G}$ for which $\mathbf{a}'' = \mathbf{H} \cdot \mathbf{a}' + \mathbf{a} \cdot Y^{v'}$ and $\mathbf{e}'' = \mathbf{e} + [\mathbf{H}_1, \cdots, \mathbf{H}_{d_g}] \cdot \mathbf{e}'$.

Correctness. For $\mathsf{ACC} = (\mathbf{a}, \mathbf{a} \cdot \bar{s} + \mathbf{e}) + uY^v\mathbf{G}$ and $\boldsymbol{C} = (\mathbf{a}', \mathbf{a}' \cdot \bar{s} + \mathbf{e}') + uY^{v'}\mathbf{G}$, it holds that $\mathbf{H}\boldsymbol{C} = \mathbf{H}(\mathbf{a}', \mathbf{a}' \cdot \bar{s} + \mathbf{e}') + \mathbf{G}^{-1}(u^{-1}\mathsf{ACC}) \cdot uY^{v'}\mathbf{G} = \mathbf{H} \cdot (\mathbf{a}', \mathbf{a}' \cdot \bar{s} + \mathbf{e}') + (\mathbf{a}, \mathbf{a} \cdot \bar{s} + \mathbf{e}) \cdot Y^{v'} + uY^{v+v'}\mathbf{G} = (\mathbf{a}'', \mathbf{a}'' \cdot \bar{s} + \mathbf{e}'') + uY^{v+v'}\mathbf{G}$.

We present the bootstrapping algorithm $\mathsf{Bootstrap}$ (**Algorithm 1**) followed by the LWE extraction algorithm ExtLWE (**Algorithm 2**) based on [16].

Algorithm 1 $\mathsf{Bootstrap}(\mathcal{K}^{(\mathcal{S})}, \mathcal{K}^{(\mathcal{B})}, \boldsymbol{c} = (\mathbf{a}, b))$

Input: A key switch key $\mathcal{K}^{(\mathcal{S})} = \{\mathbf{k}_{h,i,j}^{(\mathcal{S})} \in \mathsf{LWE}_{\mathbf{s}}^{q/q}(h\bar{s}_i B_{\mathsf{ks}}^j)\}_{h \in [0, B_{\mathsf{ks}}-1], i \in [0, N-1], j \in [0, d_{\mathsf{ks}}]}$, a bootstrapping key $\mathcal{K}^{(\mathcal{B})} = \{\mathbf{k}_{h,i,j}^{(\mathcal{B})} \in E_{\bar{s}}([hs_i B_{\mathsf{bs}}^j]_q)\}_{h \in [0, B_{\mathsf{bs}}-1], i \in [n], j \in [0, d_{\mathsf{bs}}]}$ where $d_{\mathsf{bs}} = \lceil \log_{d_{\mathsf{bs}}} q \rceil - 1$, and a LWE encryption $\boldsymbol{c} = (\mathbf{a}, b) \in \mathsf{LWE}_{\mathbf{s}}^{2/q}(m, q/4)$.

Output: A LWE encryption $\boldsymbol{c}' = (\mathbf{a}', b') \in \mathsf{LWE}_{\mathbf{s}}^{4/q}(m, q/16)$.

1: **Initialization** $\mathsf{ACC} \leftarrow b + q/4$
2: **for** $i = 1, \cdots, n$ **do**
3: Compute $a_{i,j}$ for B_{bs}-based bit representation of $-a_i$: $-a_i = \sum_j a_{i,j} B_{\mathsf{bs}}^j \pmod q$
4: **for** $j = 0, \cdots, d_{\mathsf{bs}}$ **do**
5: $\mathsf{ACC} \xleftarrow{+} \mathbf{k}_{a_{i,j},i,j}^{(\mathcal{B})}$
6: Output $(\mathbf{a}', b') \leftarrow \mathsf{ExtLWE}(\mathcal{K}^{(\mathcal{S})}, \mathsf{ACC})$.

Theorem 1 (from [16]). *If* $(E, \mathsf{Init}, \mathsf{Update}, \mathsf{ExtLWE})$ *is a correct Homomorphic Accumulator scheme, on input a valid bootstrapping key* $\mathcal{K}^{(\mathcal{B})}$ *and a ciphertext* $\boldsymbol{c} \in \mathsf{LWE}_{\mathbf{s}}^{2/q}(m, q/4)$*, the* Bootstrap *algorithm outputs a* LWE *ciphertext* $\mathsf{Bootstrap}(\boldsymbol{c}) \in \mathsf{LWE}_{\mathbf{s}}^{4/q}(m, \ell(nd_{\mathsf{bs}}))$*, where* ℓ *is a polynomial function.*

Proof. For a correct Homomorphic Accumulator scheme $(E, \mathsf{Init}, \mathsf{Update}, \mathsf{ExtLWE})$ and a ciphertext $\boldsymbol{c} = (\mathbf{a}, b) \in \mathsf{LWE}_{\mathbf{s}}^{2/q}(m, q/4)$, the for loop of the ACC updates outputs an encryption: $E_{\bar{s}}(v) \in \mathcal{R}_p^{2d_g \times 2}$, where

$$v = b + \frac{q}{4} + \sum_{i,j} a_{i,j} s_i B_{\mathsf{bs}}^j = b + \frac{q}{4} - \mathbf{a} \cdot \mathbf{s} = \frac{q}{2}m + e + \frac{q}{4}, \text{ where } |e| < \frac{q}{4}$$

It holds $v \in (0, q/2)$ if $m = 0$, and $v \in (q/2, q)$ if $m = 1$. Thus, the correctness is guaranteed if the extraction algorithm ExtLWE correctly extracts an encryption of $\mathrm{msb}(v)$ as $\mathsf{LWE}_{\mathbf{s}}^{4/q}(\mathrm{msb}(v), \ell(nd_{\mathsf{bs}}))$ satisfying $\ell(nd_{\mathsf{bs}}) < q/16$.

Algorithm 2 $\mathsf{ExtLWE}(\mathcal{K}^{(\mathcal{S})}, \mathsf{ACC})$

Input: A key switch key $\mathcal{K}^{(\mathcal{S})} = \{\mathbf{k}_{h,i,j}^{(\mathcal{S})} \in \mathsf{LWE}_{\mathbf{s}}^{q/q}(h\bar{s}_i B_{\mathsf{ks}}^j)\}$ from $\eta(\bar{s})$ to $\mathbf{s}$, and $\mathsf{ACC} \leftarrow (\mathbf{a}', \mathbf{a}' \cdot \bar{s} + \mathbf{e}') + uY^v\mathbf{G} \in \mathcal{R}_p^{2d_g \times 2}$

Output: A LWE encryption $\boldsymbol{c} \in \mathsf{LWE}_{\mathbf{s}}^{4/q}(\mathrm{msb}(v))$

1: $(\mathbf{a}^\top, \mathbf{b}^\top) \leftarrow (\mathbf{0}^\top, \mathbf{v}^\top, \mathbf{0}^\top, \cdots, \mathbf{0}^\top) \cdot M(\mathsf{ACC}) \in \mathbb{Z}_p^{2N}$ ▷ $\mathbf{v} = (-1, \cdots, -1) \in \mathbb{Z}^N$, and $M(\mathsf{ACC}) \in \mathbb{Z}^{2Nd_g \times 2N}$.

2: Compute $\boldsymbol{c}^{(1)} = (\mathbf{a}, b_0 + u) \in \mathsf{LWE}_{\eta(\bar{s})}^{4/p}(\mathrm{msb}(v))$.

3: Compute $\boldsymbol{c}^{(2)} \leftarrow \mathsf{KeySW}(\mathcal{K}^{(\mathcal{S})}, \boldsymbol{c}^{(1)}) \in \mathsf{LWE}_{\mathbf{s}}^{4/p}(\mathrm{msb}(v))$

4: Output $\boldsymbol{c} \leftarrow \mathsf{ModSW}(\boldsymbol{c}^{(2)}) \in \mathsf{LWE}_{\mathbf{s}}^{4/q}(\mathrm{msb}(v))$

LWE Extraction For Step 2 in Algorithm 2, $(\mathbf{a}^\top, b_0) \leftarrow \mathbf{v}^\top \cdot (M(\bar{a}), \eta(\bar{b}))$, where $(\bar{a}, \bar{b}) \in \mathcal{R}^2$ is the second row of ACC and $\mathbf{v} = (-1, \cdots, -1)$ is a test vector, satisfies $\eta(\bar{b}) = M(\bar{a}) \cdot \eta(\bar{s}) + u \cdot \eta(Y^v) + \bar{\mathbf{e}}$ for some $\bar{\mathbf{e}}$, such that

$$(\mathbf{a}, b_0 + u) = (\mathbf{a}, \mathbf{a} \cdot \eta(\bar{s}) + \mathbf{v} \cdot \bar{\mathbf{e}} + u(1 + \mathbf{v}^\top \cdot \eta(Y^v))$$

As $Y = X^{2N/q}, \eta(Y^v) = \eta(X^{2Nv/q}) = (0, \cdots, x_i, 0, \cdots, 0)$, where $x_i = 1$ for $i = 2Nv/q$ if $v \in (0, q/2)$ and $x_i = -1$ for $i = 2Nv/q - N$ if $v \in (q/2, q)$. So $\mathbf{v}^\top \cdot \eta(Y^v) = -(-1)^{\mathrm{msb}(v)}$ and $u(1 + \mathbf{v}^\top \cdot \eta(Y^v)) = 2u \cdot \mathrm{msb}(v)$. Since $u \approx p/8$,

$$(\mathbf{a}, b_0 + u) \approx (\mathbf{a}, \mathbf{a} \cdot \eta(\bar{s}) + \mathbf{v} \cdot \bar{\mathbf{e}} + \frac{p}{4}\mathrm{msb}(v))$$

is an encryption of $\mathrm{msb}(v)$, where $\mathbf{a} = \mathbf{v}^\top \cdot M(\bar{a})$. Combining the Algorithm 1 and Algorithm 2, and according to the central limit heuristic, we have the theorem:

Theorem 2 (From [16]). *Assuming the hardness of* $\mathsf{RLWE}_{\mathcal{R},p,\bar{\chi}}$*, the error distribution of the output from the algorithm* Bootstrap *has the parameter:*

$$\delta = \ell(nd_{bs}) = \sqrt{\frac{q^2}{p^2}(\bar{\sigma}^2 \frac{B_{bs}^2}{12} d_{bs} qnN(B_g + \frac{p}{B_g^{d_g}} - 1) + \sigma^2 N d_{\mathsf{ks}}) + \frac{\|\mathbf{s}\|^2 + 1}{12}}$$

4 Fully Homomorphic Proxy Re-Encryption (FHPRE)

An unbounded multi-hop fully homomorphic proxy re-encryption (MH-FHPRE) scheme consists of the following seven algorithms:

- $\mathsf{SetUp}(1^\lambda) \to pp$: For a security parameter λ, outputs a public parameter pp.
- $\mathsf{KeyGen}(pp) \to (pk, sk)$: On input a public parameter pp, outputs a public key pk that includes the information of encryption, bootstrapping, and key switching, and a secret decryption key sk.
- $\mathsf{ReKeyGen}(pp, sk_i, pk_j) \to rk_{i\to j}$: On input a secret key sk_i of user i and a public key pk_j of user j, outputs a re-encryption key $rk_{i\to j}$ from i to j.
- $\mathsf{Enc}(pp, pk, \mu) \to \boldsymbol{c}$: On input the public key pk and a message μ, outputs a ciphertext $\boldsymbol{c}$.
- $\mathsf{Dec}(pp, sk, \boldsymbol{c}) \to \mu$: On input the secret key sk and a ciphertext $\boldsymbol{c}$, outputs the message μ.
- $\mathsf{ReEnc}(pp, pk_j, rk_{i\to j}, \boldsymbol{c}^{(i)}) \to \boldsymbol{c}^{(j)}$: On input the public key pk_j of user j, a re-encryption key $rk_{i\to j}$, and a ciphertext $\boldsymbol{c}^{(i)}$ of user i, outputs a ciphertext $\boldsymbol{c}^{(j)}$ that can be decrypted by user j, with the same underlying message.
- $\mathsf{Eval}(pp, f, pk, \{\boldsymbol{c}_1, \cdots, \boldsymbol{c}_n\}) \to \boldsymbol{c}_f$: On input a circuit f, a public key pk, and n ciphertexts under sk, $\boldsymbol{c}_1, \cdots, \boldsymbol{c}_n$ where $\boldsymbol{c}_i \in \mathsf{Enc}_{pk}(\mu_i)$ for $i \in [n]$, outputs a ciphertext $\boldsymbol{c}_f$ that is an encryption of $f(\mu_1, \cdots, \mu_n)$ under sk.

Correctness A correct FHPRE scheme requires that over the choice of $(pk, sk) \leftarrow \mathsf{KeyGen}(pp)$ where $pp \leftarrow \mathsf{SetUp}(1^\lambda)$ for some security parameter λ, and a negligible function $negl(\cdot)$, it holds:

- For a message $\mu \in \mathcal{M}$ and $\boldsymbol{c} \leftarrow \mathsf{Enc}(pp, pk, \mu)$, it holds that $\Pr[\mathsf{Dec}(pp, sk, \boldsymbol{c}) = \mu] > 1 - negl(\lambda)$.
- For a circuit f, messages $\mu_1, \cdots, \mu_n \in \mathcal{M}$, and $\boldsymbol{c}_f \leftarrow \mathsf{Eval}(f, pk, \{\boldsymbol{c}_1, \cdots, \boldsymbol{c}_n\})$, where $\{\boldsymbol{c}_i \leftarrow \mathsf{Enc}(pp, pk, \mu_i)$ for $i \in [n]$, it holds that $\Pr[\mathsf{Dec}(pp, sk, \boldsymbol{c}_f) = f(\mu_1, \cdots, \mu_n)] > 1 - negl(\lambda)$.
- For a message $\mu \in \mathcal{M}$, a re-encrypted ciphertext $\boldsymbol{c}^{(j)} \leftarrow \mathsf{ReEnc}(pp, rk_{i\to j}, \boldsymbol{c}^{(i)})$, where $rk_{i\to j} \leftarrow \mathsf{ReKeyGen}(pp, sk_i, pk_j)$ and $\boldsymbol{c}^{(i)} \leftarrow \mathsf{Enc}(pp, pk_i, \mu)$, it holds that $\Pr[\mathsf{Dec}(pp, sk_j, \boldsymbol{c}^{(j)}) = \mu] > 1 - negl(\lambda)$.

Security of HPRE. Let $\Pi = (\mathsf{SetUp}, \mathsf{KeyGen}, \mathsf{ReKeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{ReEnc}, \mathsf{Eval})$ be a HPRE scheme and λ be a security parameter. The IND-CPA-HPRE security game, $\mathsf{Exp}^{\mathsf{CPA}}_{\mathcal{A},\Pi}(\lambda)$, is defined between a PPT adversary $\mathcal{A}$ and a challenger as:

Initial. Initialize key-value sets $\mathcal{H}$ to store key pairs (pk, sk) of honest users, and $\mathcal{CU}$ for (pk, sk) of corrupted users. $\mathcal{A}$ selects and outputs a target user id i^*.
SetUp. The challenger runs $pp \leftarrow \mathsf{SetUp}(1^\lambda)$ and $(pk_{i^*}, sk_{i^*}) \leftarrow \mathsf{KeyGen}(pp)$, adds $(i^*, (pk_{i^*}, sk_{i^*}))$ to the set $\mathcal{H}$, then outputs pp and pk_{i^*} to $\mathcal{A}$.
Query Phase 1. $\mathcal{A}$ can make polynomial times the following queries:

- Honeset key query $\mathcal{O}_{\mathcal{H}}^{\mathsf{KeyGen}}$: On input an i as a honest id, if $i \in \mathcal{CU}$, output $\perp$; If $i \in \mathcal{H}$, output the value pk_i for the key i in $\mathcal{H}$; Otherwise, run $(pk_i, sk_i) \leftarrow \mathsf{KeyGen}(pp)$, add $(i, (pk_i, sk_i))$ to $\mathcal{H}$, and output pk_i to $\mathcal{A}$.
- Corrupted key query $\mathcal{O}_{\mathcal{CU}}^{\mathsf{KeyGen}}$: On input i as a corrupted id, if $i \in \mathcal{H}$, output $\perp$; If $i \in \mathcal{CU}$, output the value (pk_i, sk_i) for key i in $\mathcal{CU}$; Otherwise, run $(pk_i, sk_i) \leftarrow \mathsf{KeyGen}(pp)$, add $(i, (pk_i, sk_i))$ to $\mathcal{CU}$, and return (pk_i, sk_i) to $\mathcal{A}$.
- Re-encryption key query $\mathcal{O}^{\mathsf{ReKey}}$: On input a pair $((i, pk_i), (j, pk_j))$, if $i \in \mathcal{H}, j \in \mathcal{CU}$, output $\perp$; If both $i, j \in \mathcal{H}$, run $rk_{i\to j} \leftarrow \mathsf{ReKeyGen}(pp, sk_i, pk_j)$ and output $rk_{i\to j}$ to $\mathcal{A}$. Note that for any $i \in \mathcal{CU}, j \in \mathcal{H} \cup \mathcal{CU}$, $\mathcal{A}$ can generate the re-encryption key $rk_{i\to j}$ using sk_i and pk_j by itself.

Challenge. $\mathcal{A}$ chooses and outputs two messages μ_0 and μ_1. The challenger chooses a random bit $\beta \leftarrow \{0, 1\}$ and outputs $C_\beta^* \leftarrow \mathsf{Enc}(pp, pk_{i^*}, \mu_\beta)$ as the challenge ciphertext to $\mathcal{A}$.
Query Phase 2. $\mathcal{A}$ can continue to make queries as in Query Phase 1.
Guess. $\mathcal{A}$ outputs a bit β', this oracle outputs 1 if $\beta' = \beta$ and 0 otherwise.

Definition 3 (CPA Security) *An unidirectional multi-hop fully homomorphic proxy re-encryption scheme is* IND-CPA-HPRE *secure, if any PPT adversary $\mathcal{A}$ wins the game $\mathsf{Exp}_{\mathcal{A},\Pi}^{\mathsf{CPA}}(\lambda)$ with only negligible advantage.*

Circular Security. Like previous FHE schemes, our FHPRE construction requires a circular security assumption introduced in [17], in order to include a public bootstrapping key to achieve ciphertexts refresh.

5 FHPRE Scheme

In this section, we present the algorithms that will be used in our unbounded multi-hop fully homomorphic proxy re-encryption scheme.

- $\mathsf{SetUp}(1^\lambda) \to pp$: On input a security parameter λ, lets $n = poly(\lambda)$ be the dimension of the LWE encryption, $q = 2^k$ be the LWE modulus, $t = 4$ be the message modulus, $m = \Theta(n \log q)$, χ be a B_χ-bounded sub-Gaussian distribution with parameter σ satisfying the hardness of the $\mathsf{DLWE}_{n,q,\chi}$, $\bar{\chi}$ be a sub-Gaussian distribution with parameter $\bar{\sigma}$ satisfying the hardness of $\mathsf{RLWE}_{\mathcal{R},p,\bar{\chi}}$, $p = B_g^{d_g}$ be the RLWE modulus where $B_g = 3^w$ is the gadget base, $N = 2^{\bar{k}}$ be the ring dimension such that $q|2N$, $B_{\mathsf{bs}} \in \mathbb{Z}_q$ be the bootstrapping base, $B_{\mathsf{ks}} \in \mathbb{Z}_q$ be the key switching base, and $u \approx p/8$, sets rings $\mathcal{R} = \mathbb{Z}[X]/(X^N + 1)$ and $\mathcal{R}_p = \mathbb{Z}_p[X]/(X^N + 1)$, outputs the public parameter:
$$pp = (n, m, q, t, \chi, \bar{\chi}, p, N, u, B_{\mathsf{bs}}, B_{\mathsf{ks}}, B_g).$$

- $\mathsf{KeyGen}(pp) \to (pk, sk)$: On input pp, proceeds as follows:
  - Randomly choose $\mathbf{s} \leftarrow \{-1, 0, 1\}^n$ and $\mathbf{A} \leftarrow \mathbb{Z}_q^{n \times m}$.
  - Compute $\mathbf{b} = \mathbf{A}^\top \mathbf{s} + \mathbf{e} \in \mathbb{Z}_q^m$ where $\mathbf{e} \leftarrow \chi^m$.
  - Choose a key $\bar{s} \leftarrow \mathcal{R}_p$, generate $\mathcal{K}^{(\mathcal{B})} = \{\mathbf{k}_{h,i,j}^{(\mathcal{B})} = E_{\bar{s}}([hs_i B_{\mathsf{bs}}^j]_q)\}$, for $h \in [0, B_{\mathsf{bs}} - 1]$, $i \in [n]$, $j \in [0, d_{\mathsf{bs}}]$, where $d_{\mathsf{bs}} = \lceil \log_{B_{\mathsf{bs}}} q \rceil - 1$.
  - Generate a key switch key $\mathcal{K}^{(\mathcal{S})} = \{\mathbf{k}_{h,i,j}^{(\mathcal{S})} = \mathsf{LWE}_{\mathbf{s}}(h\bar{s}_i B_{\mathsf{ks}}^j)\}$ for $h \in [0, d_{\mathsf{ks}}], i \in [0, N-1], j \in [0, d_{\mathsf{ks}} - 1]$, where $d_{\mathsf{ks}} = \lceil \log_{B_{\mathsf{ks}}} q \rceil$.
  - Output $pk = (\mathbf{A}, \mathbf{b}, \mathcal{K}^{(\mathcal{B})}, \mathcal{K}^{(\mathcal{S})})$ and $sk = \mathbf{s}$.

- $\mathsf{ReKeyGen}(pp, sk_i, pk_j) \to rk_{i \to j}$: On input $sk_i = \mathbf{s}_i$ of user i and $pk_j = ((\mathbf{A}_j, \mathbf{b}_j), \mathcal{K}_j^{(\mathcal{B})}, \mathcal{K}_j^{(\mathcal{S})})$ of user j, chooses $\mathbf{R}_j \xleftarrow{\$} \{1, -1\}^{m \times m}$, samples $\mathbf{E}'_j \leftarrow \chi^{n \times m}$ and $\mathbf{e}'_j \leftarrow \chi^m$, computes and outputs a re-encryption key:

$$rk_{i \to j} = \begin{bmatrix} (\mathbf{A}_j \mathbf{R}_j)^\top + (\mathbf{E}'_j)^\top & \mathbf{R}_j^\top \mathbf{b}_j - \mathsf{P2}(\mathbf{s}_i) + (\mathbf{e}'_j)^\top \\ \mathbf{0}_{1 \times n} & 1 \end{bmatrix} \in \mathbb{Z}_q^{(m+1) \times (n+1)}.$$

- $\mathsf{Enc}(pp, pk, \mu) \to \boldsymbol{c}$: On input $pk = ((\mathbf{A}, \mathbf{b}), \mathcal{K}^{(\mathcal{B})}, \mathcal{K}^{(\mathcal{S})})$ and a message $\mu \in \{0, 1\}$, randomly chooses $\mathbf{r} \xleftarrow{\$} \{-1, 0, 1\}^{m \times 1}$, samples $\mathbf{e}_1 \leftarrow \chi^n$ and $e_2 \leftarrow \chi$, computes $\mathbf{a} = \mathbf{A}\mathbf{r} + \mathbf{e}_1 \in \mathbb{Z}_q^n$, $b = \mathbf{r}^\top \mathbf{b} + \frac{q}{4}\mu + e_2 \in \mathbb{Z}_q$, outputs ciphertext $\boldsymbol{c} = (\mathbf{a}, b) \in \mathbb{Z}_q^{n+1}$.

- $\mathsf{Dec}(pp, sk, \boldsymbol{c}) \to \mu$: On input $sk = \mathbf{s}$ and $\boldsymbol{c} = (\mathbf{a}, b)$, outputs:

$$\mu = \left\lfloor \frac{4}{q}(b - \mathbf{a} \cdot \mathbf{s}) \right\rceil \pmod 4.$$

- $\mathsf{ReEnc}(pp, pk_j, rk_{i \to j}, \boldsymbol{c}^{(i)}) \to \boldsymbol{c}^{(j)}$: On input $pk_j = (\mathbf{A}_j, \mathbf{b}_j, \mathcal{K}_j^{(\mathcal{B})}, \mathcal{K}_j^{(\mathcal{S})})$, $rk_{i \to j}$, and $\boldsymbol{c}^{(i)} = (\mathbf{a}_i, b_i) \in \mathsf{LWE}_{\mathbf{s}_i}^{4/q}(\mu, q/16)$, proceeds as follows:
  - Compute: $(\mathbf{a}_j'^\top, b'_j) = \left[\mathsf{BD}(\mathbf{a}_i^\top) \middle| b_i\right] \cdot rk_{i \to j}$ and let $\boldsymbol{c}'^{(j)} = (\mathbf{a}'_j, b'_j) \in \mathsf{LWE}_{\mathbf{s}_j}^{4/q}(\mu, q/8)$.
  - Compute an adapted ciphertext $\boldsymbol{c}''^{(j)} = 2\boldsymbol{c}'^{(j)} \in \mathsf{LWE}_{\mathbf{s}_j}^{2/q}(\mu, q/4)$.
  - Run $\boldsymbol{c}^{(j)} \leftarrow \mathsf{Bootstrap}(\mathcal{K}_j^{(\mathcal{S})}, \mathcal{K}_j^{(\mathcal{B})}, \boldsymbol{c}''^{(j)}) \in \mathsf{LWE}_{\mathbf{s}_j}^{4/q}(\mu, q/16)$, output $\boldsymbol{c}^{(j)}$.

  Note that the last two steps include a bootstrapping process with higher computation complexity to refresh $\boldsymbol{c}''^{(j)}$ for subsequent operations. If only a single-hop of re-encryption is required, i.e. no subsequent re-encryptions or computations is needed, the algorithm can just output $\boldsymbol{c}'^{(j)} \in \mathsf{LWE}_{\mathbf{s}_j}^{4/q}(\mu, q/8)$ as a final single-hop ciphertext, for better re-encryption efficiency.

  In the Eval algorithm, any circuit can be evaluated homomorphically by proper interpretation of the NAND gates. Thus, we describe the computation of a NAND gate.

- $\mathsf{Eval.NAND}(pp, pk, \boldsymbol{c}_1, \boldsymbol{c}_2) \to \boldsymbol{c}_f$: On input $pk = (\mathbf{A}, \mathbf{b}, \mathcal{K}^{(\mathcal{B})}, \mathcal{K}^{(\mathcal{S})})$ and $\boldsymbol{c}_i = (\mathbf{a}_i, b_i) \in \mathsf{LWE}_{\mathbf{s}}^{4/q}(\mu_i, q/16)$ for $i = 0, 1$ and $\mu_i \in \{0, 1\}$, proceeds as follows:

- Compute $\boldsymbol{c}'_f = (\mathbf{a}', b') = (-\mathbf{a}_1 - \mathbf{a}_2, \frac{5}{8}q - b_1 - b_2) \in \mathsf{LWE}^{2/q}_{\mathbf{s}}(\mu_0 \barwedge \mu_1, q/4)$.
- Output $\boldsymbol{c}_f = (\mathbf{a}, b) \leftarrow \mathsf{Bootstrap}(\mathcal{K}^{(\mathcal{S})}, \mathcal{K}^{(\mathcal{B})}, \boldsymbol{c}'_f) \in \mathsf{LWE}^{4/q}_{\mathbf{s}}(\mu_0 \barwedge \mu_1, q/16)$.

Theorem 3 (Correctness). *A multi-hop* FHPRE *scheme is correct with respect to $\mu \in \mathbb{Z}_4$ if the chosen parameters satisfies: $(m^2+nm+2m+n+1)\sigma/\sqrt{2\pi} < q/8$ and $\sqrt{\frac{q^2}{2\pi p^2}(\bar{\sigma}^2 \frac{B^2_{\mathsf{bs}}}{12} d_{\mathsf{bs}} qnN(B_g + \frac{p}{B_g^{d_g}} - 1) + \sigma^2 N d_{\mathsf{ks}}) + \frac{n+1}{24\pi}} < \frac{q}{16}$.*

Proof. We consider the correctness of decryptions for the original ciphertexts, evaluated ciphertexts, and re-encrypted ciphertexts.

- For an original ciphertext $(\mathbf{a} = \mathbf{A}\mathbf{r} + \mathbf{e}_1, b = \mathbf{r}^\top(\mathbf{A}^\top \mathbf{s} + \mathbf{e}) + \frac{q}{4}\mu + e_2)$ where $\mathbf{e} \leftarrow \chi^m, \mathbf{e}_1 \leftarrow \chi^n, e_2 \leftarrow \chi$ for a χ with bound $B_\chi = \sigma/\sqrt{2\pi}$. We have $b - \mathbf{a}\cdot\mathbf{s} = \frac{q}{4}\mu + (\mathbf{r}\cdot\mathbf{e} + e_2 - \mathbf{e}_1 \cdot \mathbf{s})$, such that the correct decryption requires
$$|\mathbf{r}\cdot\mathbf{e} + e_2 - \mathbf{e}_1 \cdot \mathbf{s}| < (n+m+1)B_\chi < q/8. \tag{1}$$
- For an evaluated ciphertext $(\mathbf{a}, b)$, we consider the ciphertext after NAND gate evaluation since the algorithm $\mathsf{Bootstrap}$ refreshes the ciphertext to the original state after each gate evaluation. As section 3.2 shows, the correct decryption requires: (1) The noise of an original ciphertext before computations satisfies
$$|\mathbf{r}\cdot\mathbf{e} + e_2 - \mathbf{e}_1 \cdot \mathbf{s}| < (m+n+1)B_\chi < (m+n+1)\sigma/\sqrt{2\pi} < q/16 \tag{2}$$
(2) The noise Δ of a refreshed ciphertext produced by $\mathsf{Bootstrap}$ satisfies
$$\Delta \le \sqrt{\frac{q^2}{2\pi p^2}(\bar{\sigma}^2 \frac{B^2_{\mathsf{bs}}}{12} d_{\mathsf{bs}} qnN(B_g + \frac{p}{B_g^{d_g}} - 1) + \sigma^2 N d_{\mathsf{ks}}) + \frac{n+1}{24\pi}} < \frac{q}{16} \tag{3}$$
- For a multi-hop re-encrypted ciphertext, let $(\mathbf{a}'_j, b'_j) \in \mathsf{LWE}^{4/q}_{\mathbf{s}_j}(\mu, \delta')$ where
$$(\mathbf{a}'^\top_j, b'_j) = \left[\mathsf{BD}(\mathbf{a}_i^\top) \middle| b_i\right] \cdot \begin{bmatrix} (\mathbf{A}_j\mathbf{R}_j)^\top + (\mathbf{E}'_j)^\top & \mathbf{R}_j^\top \mathbf{b}_j - \mathsf{P2}(\mathbf{s}_i) + (\mathbf{e}'_j)^\top \\ \mathbf{0}_{1\times n} & 1 \end{bmatrix}$$
satisfying $b'_j - \mathbf{a}'_j \cdot \mathbf{s} = (q/4)\mu + e'$ and $|e'| < \delta'$. It holds that for the adapted ciphertext $(\mathbf{a}''_j, b''_j) = 2(\mathbf{a}'_j, b'_j)$, $b''_j - \mathbf{a}''_j \cdot \mathbf{s} = (q/2)\mu + e''$ where $|e''| < 2\delta'$. Thus, (1) A correct ciphertext for bootstrapping requires: $(\mathbf{a}''_j, b''_j) \in \mathsf{LWE}^{2/q}_{\mathbf{s}}(\mu, q/4)$, which means $\delta' = q/8$. (2) A single-hop re-encrypted ciphertext, $(\mathbf{a}'_j, b'_j)$, has the same correctness requirement: $\delta' = q/8$. So that for $\mathbf{a}'_j = (\mathsf{BD}(\mathbf{a}_i^\top)(\mathbf{A}_j\mathbf{R}_j)^\top + \mathsf{BD}(\mathbf{a}_i^\top)(\mathbf{E}'_j)^\top)^\top$ and $b'_j = \mathsf{BD}(\mathbf{a}_i^\top)\mathbf{R}_j^\top(\mathbf{A}_j^\top \mathbf{s}_j + \mathbf{e}_j) - \mathsf{BD}(\mathbf{a}_i^\top)\mathsf{P2}(\mathbf{s}_i) + \mathsf{BD}(\mathbf{a}_i^\top)(\mathbf{e}'_j)^\top + b_i$, it holds: $b'_j - \mathbf{a}'_j \cdot \mathbf{s}_j = (q/4)\mu + (\mathsf{BD}(\mathbf{a}_i^\top)\mathbf{R}^\top \mathbf{e}_j + \mathsf{BD}(\mathbf{a}_i^\top)(\mathbf{e}'_j)^\top - \mathbf{e}_{i,1}^\top \mathbf{s}_i + \mathbf{r}_i^\top \mathbf{e}_i + e_{i,2} - \mathsf{BD}(\mathbf{a}_i^\top)(\mathbf{E}'_j)^\top \mathbf{s}_j)$. Therefore, the correctness of a re-encrypted ciphertext requires:
$$\begin{aligned} e' &= \mathsf{BD}(\mathbf{a}_i^\top)\mathbf{R}_j^\top \mathbf{e}_j + \mathsf{BD}(\mathbf{a}_i^\top)(\mathbf{e}'_j)^\top - \mathbf{e}_{i,1}^\top \mathbf{s}_i + \mathbf{r}_i^\top \mathbf{e}_i + e_{i,2} - \mathsf{BD}(\mathbf{a}_i^\top)(\mathbf{E}'_j)^\top \mathbf{s}_j \\ &< m^2 B_\chi + mB_\chi + nB_\chi + mB_\chi + B_\chi + nmB_\chi \\ &< (m^2 + nm + 2m + n + 1)\sigma/\sqrt{2\pi} < \delta' = q/8 \end{aligned} \tag{4}$$

In summary, an unbounded multi-hop FHPRE scheme is correct if the parameters chosen satisfying (3) and (4), which completes the proof.

6 Security Proof

Theorem 4. *The above* FHPRE *scheme is* CPA*-secure assuming circular security, the hardness of* $\mathsf{DLWE}_{n,q,\chi}$*, and the hardness of* $\mathsf{RLWE}_{\mathcal{R},p,\bar{\chi}}$.

Proof. The proof proceeds with a sequence of games as follows:

Game 0. This game is the original IND-CPA-HPRE game from Definition 3.
Game 1. In this game, the challenger changes the way in answering the key queries for honest users as: $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n\times m}$, $\mathbf{b} \xleftarrow{\$} \mathbb{Z}_q$.

We note that the secret key $\mathbf{s}$ is not required in key queries for honest users. Game 1 and Game 0 are indistinguishable for any PPT $\mathcal{A}$. This is because the distributions $(\mathbf{A}, \mathbf{b} = \mathbf{A}^\top\mathbf{s} + \mathbf{e})$ and $(\mathbf{A}, \mathbf{u})$ are indistinguishable for some $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n\times m}, \mathbf{s} \leftarrow \{-1, 0, 1\}^n, \mathbf{e} \leftarrow \chi^m$ and $\mathbf{u} \xleftarrow{\$} \mathbb{Z}_q^m$ based on the hardness assumptions of $\mathsf{DLWE}_{n,q,\chi}$.
Game 2. In this game, the challenger changes the way in generating the challenge ciphertext $C^*_\beta = (\mathbf{a}^*, b^*)$ in Challenge phase as: $\mathbf{a}^* = \mathbf{A}_{i^*}\mathbf{r}^* + \mathbf{e}_1^* \in \mathbb{Z}_q^n$ and $b^* \xleftarrow{\$} \mathbb{Z}_q$, where $\mathbf{r}^* \xleftarrow{\$} \{-1, 0, 1\}^{m\times 1}$ and $\mathbf{e}_1^* \leftarrow \chi^n$.

Game 2 and Game 1 are also indistinguishable with the only difference: the generation of b^*. b^* of the challenge ciphertext in Game 1 is generated by $b^* \leftarrow \langle \mathbf{b}^\$, \mathbf{r}^*\rangle + \frac{q}{4}\mu_\beta + e_2^*$, where $\mathbf{r}^* \xleftarrow{\$} \{-1, 0, 1\}^{m\times 1}$, $b^\$ \xleftarrow{\$} \mathbb{Z}_q$ and $e_2^* \leftarrow \chi$. According to the hardness assumption of $\mathsf{DLWE}_{n,q,\chi}$, the distributions $\langle \mathbf{b}^\$, \mathbf{r}^*\rangle + e_2^*$ and $u \xleftarrow{\$} \mathbb{Z}_q$ are indistinguishable. Therefore, $\langle \mathbf{b}^\$, \mathbf{r}^*\rangle + \frac{q}{4}\mu_\beta + e_2$ in Game 1 is also indistinguishable from any random element in $\mathbb{Z}_q$, as b^* in Game 2. Because the challenge ciphertext generated in Game 2 is completely independent of the bit β, any PPT adversary has 0 advantage in winning the game. Also, the circular security assumption guarantees the security with the usage of bootstrapping key and key switch key. Therefore, any PPT adversary will have negligible advantage in the original IND-CPA-HPRE game while any two consecutive games are indistinguishable, which completes the proof.

7 Multi-User Computation System based on FHPRE

In this section we present a multi-user computation system that is built upon the proposed FHPRE scheme. This system is designed for scenarios involving a group of users, namely, user 1, $\cdots$, user L, wishing to collaboratively compute a function f over their encrypted data, $\boldsymbol{c}_1, \cdots, \boldsymbol{c}_L$, where $\boldsymbol{c}_i$ is an encryption of m_i under pk_i from user i. The goal is to compute the result: $f(m_1, \cdots, m_L)$, without disclosing each user's underlying message. To accomplish this computation task, the system (Fig 1) proceeds with the following phases:

1. **Preparation.** The system first selects an intermediate user (i.e. L). Each user i in $[L-1]$, runs $rk_{i\to L} \leftarrow \mathsf{ReKeyGen}(pp, sk_i, pk_L)$, then sends the re-encryption key $rk_{i\to L}$ to the proxy. For each user i in $[L-1]$, user L runs $rk_{L\to i} \leftarrow \mathsf{ReKeyGen}(pp, sk_L, pk_i)$ then sends $\{rk_{L\to i}\}_{i\in[L-1]}$ to the proxy.
2. **Unified Re-Encryptions.** For each $\boldsymbol{c}_i$, where $i \in [L-1]$, the proxy runs $\boldsymbol{c}_{i,L} \in \mathsf{Enc}_{pk_L}(m_i) \leftarrow \mathsf{ReEnc}_{rk_{i\to L}}(pp, pk_L, \boldsymbol{c}_i)$, for subsequent computations.
3. **Computation.** The cloud server computes $\boldsymbol{c}_f \in \mathsf{Enc}_{pk_L}(f(m_1, \cdots, m_L)) \leftarrow \mathsf{Eval}(pp, f, pk_L, \boldsymbol{c}_L, \{\boldsymbol{c}_{i,L}\}_{i\in[L-1]})$, where $\boldsymbol{c}_{i,L} \in \mathsf{Enc}_{pk_L}(m_i)$.
4. **Distribution.** After computing $\boldsymbol{c}_f$, for each user i for $i \in [L-1]$, the proxy runs $\boldsymbol{c}_{f,i} \leftarrow \mathsf{ReEnc}_{rk_{L\to i}}(pp, pk_i, \boldsymbol{c}_f)$ where $\boldsymbol{c}_{f,i} \in \mathsf{Enc}_{pk_i}(f(m_1, \cdots, m_L))$. Thus, $\boldsymbol{c}_{f,i}$ can be downloaded from the cloud and decrypted independently by user i with its own secret key to recover the message $f(m_1, \cdots, m_L)$.

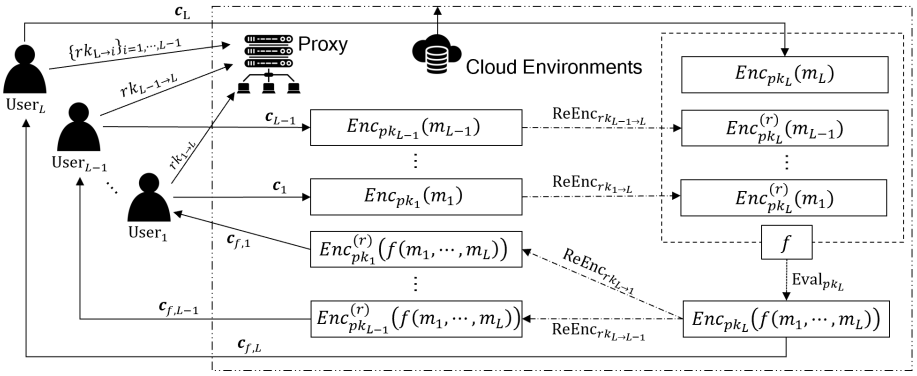

Fig. 1: The Multi-User Computation System Based on FHPRE.

Our proposed system effectively addresses the mentioned limitations inherent in MKHE schemes: (1) the need for interactive joined decryption, and (2) the growth of ciphertexts is, at least linearly, with the number of involved users. Our solution to the first limitation involves one round of re-encryptions in phase 4, where the evaluated ciphertext is re-encrypted from the intermediate user to all other participants, allowing each user to independently recover the result by their own keys, thus eliminating the joined decryption process. As noted in phase 3, all ciphertexts are evaluated under a single key of the intermediate user, negating the need for computing a joined ciphertext as in MKHE. This property, combined with the bootstrapping, remains the ciphertext a constant size after computations, reducing storage and computational costs. Table 2 summarizes the comparison of the ciphertext size and computational costs based on the number of scalar operations, between MKHE schemes and our FHPRE system. Notably, the NAND gate evaluation, that takes $O(n^2)$ scalar operations in our system, includes the bootstrapping procedure, while other MKHE schemes treat

bootstrapping as a separate process. If taking bootstrapping into account, these MKHE schemes would have a rise in the time complexity per evaluation.

The security of this system is based on the FHPRE scheme. However, we note that it places reliance on a semi-trust third-party proxy. Potential risks like the collusion of a compromised proxy and malicious delegatees should be considered.

Table 2: Comparison of ciphertext size and computational costs (number of scalar operations) between MKHE schemes and our FHPRE multi-user computation system. k is the number of users and n is the dimension of LWE/RLWE.

Scheme	Interactive Decryption	Ciphertext Size	Evaluation Complexity
BP16 [6] (MKHE)	Required	$O(kn)$	$O(kn)$ (NAND)
CCS19 [11] (MKHE)	Required	$O(kn)$	$O(k^2n^2)$ (NAND)
CDKS19 [12] (MKHE)	Required	$O(kn)$	$O(k^2n)$ (Mult)
Our FHPRE System	Not required	$O(n)$	$O(n^2)$ (NAND)

8 Performance Analysis

In this section, we present the space and time complexity performance analysis of the proposed FHPRE scheme. The asymptotic data size of ciphertext, re-encryption key, and the evaluation keys, and the time complexity of homomorphic NAND evaluation, re-encryption, and evaluation key generation are summarized in Table 3. As shown, the size of ciphertext is a linear function of the dimension n of the LWE encryption. The asymptotic size of the re-encryption key is $O(n^2 \log q)$ where q is the LWE encryption modulus. The asymptotic size of the evaluation keys is a linear function in $n, N, \log q$, and $\log p$, where N and p are the degree and the modulus of the RLWE encryption respectively.

Regarding time performance, the homomorphic NAND computation is dominated by the loop of $O(n \log q)$ ACC Update operations, $\mathsf{ACC} \xleftarrow{+} \mathbf{k}^{(\mathcal{B})}_{a_{i,j},i,j}$, in Bootstrap algorithm. Each $\mathsf{ACC} \xleftarrow{+} \mathbf{k}^{(\mathcal{B})}_{a_{i,j},i,j}$ takes $O(\log^2 p)$ ring multiplications, where each multiplication takes $O(N \log N)$ using FFT techniques. The re-encryption operation, dominated by the computation: $\left[\mathsf{BD}(\mathbf{a}_i^\top) \middle| b_i\right] \cdot rk_{i \to j}$, requires $(n+1)(m+1)$ multiplications in $\mathbb{Z}_q$, taking $O(n^2 \log q)$ time. And it takes $O(n \log q)$ RLWE encryptions, cumulatively amounting to $O(n \log q \log p)$ ring multiplications in total to generate the evaluation keys.

Optimization by Ring Packing and Amortized Bootstrapping It is worth mentioning that the ring packing and amortized bootstrapping techniques, as in [30], can be naturally applied to our FHPRE scheme for optimization. Amortized bootstrapping allows to refresh multiple ciphertexts simultaneously in one

Table 3: Asymptotic Space and Computation Performance of FHPRE

Space		Time	
Type	Complexity	Type	Complexity
Ciphertext	$O(n)$	Hom NAND	$O(nN \log N \log q \log^2 p)$
Re-Encryption Key	$O(n^2 \log q)$	Re-Encryption	$O(n^2 \log q)$
Evaluation Key	$O(nN \log q \log p)$	Evaluation KeyGen	$O(nN \log N \log q \log p)$

bootstrapping operation, reducing the total number of executions. This process amortizes the bootstrapping cost over a large number, yielding a cost reduction per message of $O(n^{1-\epsilon})$ for some $\epsilon < 1/2$. This property is particularly beneficial for the FHPRE multi-user computation system as described in Section 7. For example, $L-1$ ciphertexts must be refreshed for subsequent computations after being re-encrypted to an intermediate user in phase 2, Unified Re-Encryptions. In such cases, amortized bootstrapping can be an effective optimization approach.

9 Conclusions and Future Works

To address the problem of the existing homomorphic proxy re-encryption schemes that support only a single or bounded number of re-encryption operations, we introduce a novel lattice-based constant-size unbounded multi-hop FHPRE scheme. This scheme supports an unbounded number of re-encryption operations and enables arbitrary homomorphic computations with constant-size ciphertexts. Utilizing our FHPRE scheme, we have conceptualized a potential application in the form of a non-interactive multi-user computation system in cloud computing environments. This system facilitates a group of users to collaboratively homomorphically evaluate a function over a set of ciphertexts from them. Compared to the current multi-key homomorphic encryption schemes, our system obviates the need for interactive joint decryption and maintains constant size ciphertexts throughout the computations process, leading to a cost reduction in both storage requirements and computational complexity.

However, it is worthy to acknowledge that our scheme relies on a semi-trust third-party proxy to perform re-encryptions. This introduces risks like collusion between a compromised proxy and dishonest delegatees. Therefore, developing a collusion-resistant unbounded FHPRE scheme would be a subsequent direction. Exploring strategies like threshold PRE [34], can also be a potential approach to mitigate collusion risks. Another interesting problem is the construction of a FHPRE scheme that is secure against honest re-encryption attacks (HRA) or chosen ciphertext attacks (CCA).

Acknowledgements This work is supported by Major Program of Guangdong Basic and Applied Research Project under Grant No. 2019B030302008, National Natural Science Foundation of China under Grant Nos. 61825203, 62332007

and U22B2028, Science and Technology Major Project of Tibetan Autonomous Region of China under Grant No. XZ202201ZD0006G, Guangdong Provincial Science and Technology Project under Grant No. 2021A0505030033, National Joint Engineering Research Center of Network Security Detection and Protection Technology, Guangdong Key Laboratory of Data Security and Privacy Preserving, Guangdong Hong Kong Joint Laboratory for Data Security and Privacy Protection, and Engineering Research Center of Trustworthy AI, Ministry of Education.

References

1. Alperin-Sheriff, J., Peikert, C.: Faster bootstrapping with polynomial error. In: Advances in Cryptology–CRYPTO 2014. pp. 297–314 (2014)
2. Aono, Y., Boyen, X., Phong, L.T., Wang, L.: Key-private proxy re-encryption under LWE. In: Progress in Cryptology - INDOCRYPT 2013 - 14th International Conference on Cryptology in India. vol. 8250, pp. 1–18 (2013)
3. Ateniese, G., Fu, K., Green, M., Hohenberger, S.: Improved proxy re-encryption schemes with applications to secure distributed storage. ACM Trans. Inf. Syst. Secur. **9**(1), 1–30 (2006)
4. Blaze, M., Bleumer, G., Strauss, M.: Divertible protocols and atomic proxy cryptography. In: Advances in Cryptology - EUROCRYPT '98. vol. 1403, pp. 127–144 (1998)
5. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (leveled) fully homomorphic encryption without bootstrapping. ACM Trans. Comput. Theory **6**(3), 13:1–13:36 (2014)
6. Brakerski, Z., Perlman, R.: Lattice-based fully dynamic multi-key FHE with short ciphertexts. In: Advances in Cryptology - CRYPTO 2016. vol. 9814, pp. 190–213 (2016)
7. Brakerski, Z., Vaikuntanathan, V.: Efficient fully homomorphic encryption from (standard) LWE. Electron. Colloquium Comput. Complex. **TR11-109** (2011)
8. Brakerski, Z., Vaikuntanathan, V.: Fully homomorphic encryption from ring-lwe and security for key dependent messages. In: Advances in Cryptology - CRYPTO 2011. vol. 6841, pp. 505–524 (2011)
9. Canetti, R., Hohenberger, S.: Chosen-ciphertext secure proxy re-encryption. In: Proceedings of the 2007 ACM Conference on Computer and Communications Security, CCS 2007. pp. 185–194 (2007)
10. Chandran, N., Chase, M., Liu, F.H., Nishimaki, R., Xagawa, K.: Re-encryption, functional re-encryption, and multi-hop re-encryption: A framework for achieving obfuscation-based security and instantiations from lattices. In: International Workshop on Public Key Cryptography. pp. 95–112 (2014)
11. Chen, H., Chillotti, I., Song, Y.: Multi-key homomorphic encryption from TFHE. In: Advances in Cryptology - ASIACRYPT 2019. vol. 11922, pp. 446–472 (2019)
12. Chen, H., Dai, W., Kim, M., Song, Y.: Efficient multi-key homomorphic encryption with packed ciphertexts with application to oblivious neural network inference. In: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019. pp. 395–412 (2019)
13. Cheon, J.H., Kim, A., Kim, M., Song, Y.S.: Homomorphic encryption for arithmetic of approximate numbers. In: Advances in Cryptology - ASIACRYPT 2017. vol. 10624, pp. 409–437 (2017)

14. Chillotti, I., Gama, N., Georgieva, M., Izabachène, M.: Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. In: Advances in Cryptology - ASIACRYPT 2016. vol. 10031, pp. 3–33 (2016)
15. Ducas, L., Micciancio, D.: Improved short lattice signatures in the standard model. In: Advances in Cryptology - CRYPTO 2014. vol. 8616, pp. 335–352
16. Ducas, L., Micciancio, D.: FHEW: bootstrapping homomorphic encryption in less than a second. In: Advances in Cryptology - EUROCRYPT 2015. vol. 9056, pp. 617–640 (2015)
17. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009. pp. 169–178 (2009)
18. Gentry, C., Sahai, A., Waters, B.: Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In: Advances in Cryptology–CRYPTO 2013. pp. 75–92 (2013)
19. Ivan, A., Dodis, Y.: Proxy cryptography revisited. In: Proceedings of the Network and Distributed System Security Symposium, NDSS 2003 (2003)
20. Jiang, M., Hu, Y., Wang, B., Wang, F., Lai, Q.: Lattice-based multi-use unidirectional proxy re-encryption. Security and Communication Networks **8**(18), 3796–3803 (2015)
21. Kirshanova, E.: Proxy re-encryption from lattices. In: Public-Key Cryptography - PKC 2014. vol. 8383, pp. 77–94 (2014)
22. Lai, J., Huang, Z., Au, M.H., Mao, X.: Constant-size cca-secure multi-hop unidirectional proxy re-encryption from indistinguishability obfuscation. Theor. Comput. Sci. **847**, 1–16 (2020)
23. Li, J., Ma, C., Zhang, L., Yuan, Q.: Unidirectional FHPRE scheme from lattice for cloud computing. Int. J. Netw. Secur. **21**(4), 592–600 (2019)
24. Li, J., Qiao, Z., Zhang, K., Cui, C.: A lattice-based homomorphic proxy re-encryption scheme with strong anti-collusion for cloud computing. Sensors **21**(1), 288 (2021)
25. Li, Z., Ma, C., Wang, D.: Towards multi-hop homomorphic identity-based proxy re-encryption via branching program. IEEE Access **5**, 16214–16228 (2017)
26. Li, Z., Ma, C., Wang, D.: Achieving multi-hop PRE via branching program. IEEE Trans. Cloud Comput. **8**(1), 45–58 (2020)
27. López-Alt, A., Tromer, E., Vaikuntanathan, V.: On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In: Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012. pp. 1219–1234 (2012)
28. Lyubashevsky, V., Peikert, C., Regev, O.: On ideal lattices and learning with errors over rings. In: Advances in Cryptology–EUROCRYPT 2010: 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, May 30–June 3, 2010. Proceedings 29. pp. 1–23. Springer (2010)
29. Ma, C., Li, J., Ouyang, W.: A homomorphic proxy re-encryption from lattices. In: Provable Security - 10th International Conference, ProvSec 2016. vol. 10005, pp. 353–372 (2016)
30. Micciancio, D., Sorrell, J.: Ring packing and amortized FHEW bootstrapping. In: 45th International Colloquium on Automata, Languages, and Programming, ICALP 2018. vol. 107, pp. 100:1–100:14 (2018)
31. Mukherjee, P., Wichs, D.: Two round multiparty computation via multi-key FHE. In: Advances in Cryptology - EUROCRYPT 2016. vol. 9666, pp. 735–763 (2016)

32. Pareek, G.: Proxy visible re-encryption scheme with application to e-mail forwarding. In: Proceedings of the 10th international conference on security of information and networks. pp. 212–217 (2017)
33. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: Proceedings of the 37th Annual ACM Symposium on Theory of Computing. pp. 84–93 (2005)
34. Zhao, F., Weng, J., Xie, W., Li, M., Weng, J.: Hra-secure attribute-based threshold proxy re-encryption from lattices. Inf. Sci. **655**, 119900 (2024)
35. Zhong, H., Cui, J., Shi, R., Xia, C.: Many-to-one homomorphic encryption scheme. Security and Communication Networks **9**(10), 1007–1015 (2016)

A Homomorphic Gates Evaluation

In this section, we describe the basic logic gate functions that are compatible with the bootstrapping algorithm.

- **NOT Gate** The homomorphic NOT gate for $\boldsymbol{c}' \in \mathsf{LWE}_{\mathbf{s}}^{4/q}(m, q/16)$ where $m \in \{0,1\}$, is defined as: Let $\boldsymbol{c}' = (\mathbf{a}', b')$, $(\mathbf{a}, b)$ is computed by

$$\mathsf{Eval.NOT}((\mathbf{a}', b')) = (-\mathbf{a}', \frac{q}{4} - b') \in \mathsf{LWE}_{\mathbf{s}}^{4/q}(\neg m, \frac{q}{16}).$$

  It satisfies: $b - \mathbf{a} \cdot \mathbf{s} - \frac{q}{4}(1-m) = -e'$, with $|-e'| < \frac{q}{16}$. No subsequent bootstrapping is needed for a NOT gate since there is no error increase.

- **AND Gate** The homomorphic AND gate for $\boldsymbol{c}_i \in \mathsf{LWE}_{\mathbf{s}}^{4/q}(m_i, q/16)$, where $i = 0, 1, m_i \in \{0,1\}$, is defined as: Let $\boldsymbol{c}_i = (\mathbf{a}_i, b_i)$, $(\mathbf{a}, b)$ is computed by

$$\mathsf{Eval.AND}((\mathbf{a}_0, b_0), (\mathbf{a}_1, b_1)) = (\mathbf{a}_0 + \mathbf{a}_1, -\frac{q}{8} + b_0 + b_1) \in \mathsf{LWE}_{\mathbf{s}}^{2/q}(m_0 \wedge m_1, \frac{q}{4}).$$

  It satisfies: $b - \mathbf{a}\cdot\mathbf{s} - \frac{q}{2}(m_0 m_1) = \frac{q}{4}(m_0 - m_1)^2 + (e_0 + e_1) - \frac{q}{8} = \pm\frac{q}{8} + (e_0 + e_1)$, with $\left|\pm\frac{q}{8} + (e_0 + e_1)\right| < \frac{q}{4}$.

- **OR Gate** The homomorphic OR gate for $\boldsymbol{c}_i \in \mathsf{LWE}_{\mathbf{s}}^{4/q}(m_i, q/16)$, where $i = 0, 1, m_i \in \{0,1\}$, is defined as: Let $\boldsymbol{c}_i = (\mathbf{a}_i, b_i)$, $(\mathbf{a}, b)$ is computed by

$$\mathsf{Eval.OR}((\mathbf{a}_0, b_0), (\mathbf{a}_1, b_1)) = (\mathbf{a}_0 + \mathbf{a}_1, \frac{q}{8} + b_0 + b_1) \in \mathsf{LWE}_{\mathbf{s}}^{2/q}(m_0 \vee m_1, \frac{q}{4}).$$

  It satisfies: $b - \mathbf{a}\cdot\mathbf{s} - \frac{q}{2}(m_0 + m_1 - m_0 m_1) = -\frac{q}{4}(m_0 - m_1)^2 + (e_0 + e_1) + \frac{q}{8} = \pm\frac{q}{8} + (e_0 + e_1)$, with $\left|\pm\frac{q}{8} + (e_0 + e_1)\right| < \frac{q}{4}$.

- **XOR Gate** The homomorphic XOR gate for $\boldsymbol{c}_i \in \mathsf{LWE}_{\mathbf{s}}^{4/q}(m_i, q/16)$, where $i = 0, 1, m_i \in \{0,1\}$, is defined as: Let $\boldsymbol{c}_i = (\mathbf{a}_i, b_i)$, $(\mathbf{a}, b)$ is computed by

$$\mathsf{Eval.XOR}((\mathbf{a}_0, b_0), (\mathbf{a}_1, b_1)) = (2\mathbf{a}_0 + 2\mathbf{a}_1, 2b_0 + 2b_1) \in \mathsf{LWE}_{\mathbf{s}}^{2/q}(m_0 \oplus m_1, \frac{q}{4}).$$

  It satisfies: $b - \mathbf{a}\cdot\mathbf{s} - \frac{q}{2}(m_0 + m_1 - 2m_0 m_1) = q(m_0 m_1) + 2(e_0 + e_1)$, with $|q(m_0 m_1) + 2(e_0 + e_1)| < \frac{q}{4} \pmod q$.