Efficient Fully Homomorphic Encryption with Large Plaintext Space

NingBo Li¹, TanPing Zhou¹, XiaoYuan Yang¹, YiLiang Han¹ and YuJuan Sun²

¹Key Laboratory of Network & Information Security under the People’s Armed Police, College of Cryptography Engineering, Engineering University of People’s Armed Police, Xi’an, 710086, People’s Republic of China; ²Department of Electrical and Computer Engineering University of Toronto, Toronto, ON, Canada

ABSTRACT

The security of multimedia content and personal privacy for big data has triggered widespread concern in the society. Fully homomorphic encryption (FHE), which can homomorphically compute arbitrary functions on the encrypted data without knowing the secret key, is valuable in protecting user’s data security. However, most of the FHE schemes only take single-bit of ciphertext as the input, which makes the evaluation process complicated. In EUROCRYPT’2015, Ducas and Micciancio proposed an FHE scheme FHEW with the plaintext space $\mathbb{Z}_2$, and gave an assumption of extending the plaintext space to $\mathbb{Z}_t$. In this paper, we optimize the decryption algorithm in bootstrapping, and propose an FHE scheme with large plaintext space $\mathbb{Z}_t$. Firstly, we optimize the rounding function of the decryption algorithm in FHEW to the msdExtract algorithm, which can homomorphically extract the most significant digit of the plaintext. Secondly, we design the msdExtract algorithm by employing the homomorphic accumulator, and present the process of general bootstrapping. Finally, based on the msdExtract algorithm, we extend the plaintext space of our scheme to $\mathbb{Z}_t$, comparing to $\mathbb{Z}_2$ in FHEW. The security of our scheme is based on the basic LWE scheme and FHEW. What’s more, our scheme can perform the evaluation more conveniently with large plaintext space, and can be applied to more scenarios.

1. INTRODUCTION

With the rapid increase of capacity for big data, more and more multimedia data are handled in the cloud, and the security of multimedia content and personal privacy have become a hot topic in big data. Fully homomorphic encryption (FHE) allows a user to perform arbitrary computation on the encrypted data without leaking the secret keys, and the decryption of the output is equal to the result that doing the same computations on the original message. In other words, FHE has inherent exchangeable properties for encryption and homomorphic computation, and is very valuable for protecting the user’s information security [1] and secret communication [2] in the environment of cloud computing [3] and big data [4]. Since Gentry’s breakthrough work in 2009 [5], many researchers have done deep research on FHE [6–13], and most of their works concentrate on the FHE scheme’s construction, efficiency, and security [14–23]. Efficiency is the main obstacle to the application of FHE schemes, and there are mainly two methods to improve the efficiency. One is applying some optimal technologies to the construction of FHE, such as SMID (Single Instruction Multiple Data), modulus switching [17], etc. The other way is to reduce the computational complexity of the bootstrapping process and has been developed rapidly after GSW13 [18] proposed in 2013 by Gentry et al.

In ITCS’2014, Brakerski and Vaikuntanathan presented an optimal vision of GSW13 called BV14 [22]. According to Barrington’s Theorem [15], it can be proved that BV14 reduces the noise expansion during bootstrapping to a polynomial level of the security parameter. In CRYPTO’2014, Alperin and Peikert proposed an efficient scheme AP14 [24], in which the noise only increases in polynomial scale during bootstrapping. The prominent contribution of AP14 lies in the construction of an efficient external scheme for bootstrapping. The external scheme, which is used to evaluate the internal decryption process, has an efficient decryption process. What’s more, it supports an efficient transformation from the external large ciphertext to the original ciphertext in the internal scheme.

In EUROCRYPT’2015, Ducas and Micciancio proposed a more efficient FHE scheme called FHEW [25] based on AP14. FHEW transforms the rounding process during bootstrapping to the msdExtract algorithm which
can homomorphically extract the MSB (Most Significant Bit) of the plaintext from the external ciphertext. By constructing a faster msbExtract process, FHEW can perform bootstrapping within a second. However, only if the plaintext space is $\mathbb{Z}_2$ that the msbExtract algorithm can homomorphically extract the MSB of the plaintext.

In LATINCRIPT’2015, Biasse and Ruiz proposed a new scheme BR15 [26], which introduces homomorphic member tests and parallelizes the bootstrapping process. However, BR15 is based on a special cyclotomic polynomial ring, which will affect the performance of FFT/NTT (Fast Fourier Transformation/Number Theoretic Transform) during bootstrapping. However, comparing to Helib [20], BR15 increases the number of parallels, which results in nearly 100 times improvement in performance.

**Contributions:** The main improvement of our work on the FHEW scheme lies in two parts. First, we optimize the decryption algorithm in basic LWE encryption scheme to msbExtract algorithm, which can homomorphically extract the MSB of the plaintext, thus allow the bootstrapping procedure to refresh the encryptions of message modulo an integer $t \geq 2$.

Second, we give a detailed process about how to implement the msdExtract algorithm by employing the homomorphic accumulator and present the process of general bootstrapping via msdExtract algorithm and homomorphic accumulator. As operations during the general bootstrapping is implemented directly on encrypted data, and the encryption algorithm follows the basic LWE scheme and external scheme in FHEW, the security of our scheme is the same as FHEW, and our scheme has wide applications for large plaintext space such as the protection of DNA: A,T,G,C.

**Organization:** The rest of the paper is organized as follows. Some background knowledge is provided in Section 2. In Section 3, we firstly introduce the basic LWE symmetric encryption scheme, then analyze the optimization of decrypting process from rounding function to msdExtract function, and give the construction of the general bootstrapping via a homomorphic accumulator. In Section 4, we analyze the construction and components in homomorphic accumulator detailedly. Section 5 analyzes our scheme in aspects of the correctness of the general bootstrapping, scheme’s computation complexity and security. The conclusion is provided in Section 6.

## 2. MATHMATICAL PRELIMINARIES

This section introduces some symbols, basic concepts, and definitions used in our scheme. For a list of vectors or matrices $a_1, a_2, \ldots, a_l$, we use $(\ldots)$ for vertical concatenation, and $[\ldots]$ for horizontal concatenation.

### 2.1 Distribution

We define the randomized rounding function $\psi : \mathbb{R} \rightarrow \mathbb{Z}_q$, which satisfies $\psi(x + n) = \psi(x) + n$ for arbitrary $x \in \mathbb{R}$ and $n \in \mathbb{Z}$. When input $x \in \mathbb{Z}$, we have $\psi(x) = \psi(0) + x$ as a special case, which means that a certain noise $\psi(0)$ is added to the input $x$. We define $\psi(x) - x$ as the rounding error of $\psi(x)$, and typically the error distribution $\psi(x) - x < q/2t$.

We define that the random variable $x \in \mathbb{R}$ is a sub-Gaussian distribution of parameter $\alpha$, if its momentgenerating function (MGF) satisfies $E[exp(2\pi i t x)] \leq \exp(\pi \alpha^2 t^2)$ for all $t \in \mathbb{R}$, and then we have $\Pr[|t| \geq t] \leq 2 \exp(-\pi t^2/\alpha^2)$ for $t \geq 0$. The concept of sub-Gaussian distribution can be extended to vector $x$ and matrice $X$. For unit vector $u$, if $u'X > 0$ is a sub-Gaussian distribution, then $x$ is also a sub-Gaussian distribution. For all the unit vectors $u$ and $v$, if $u'Xv$ is a sub-Gaussian distribution, so is $X$. From the definition, we can see that if concatenate multiple variables which are sub-Gaussian with parameter $\alpha$, the resulting vectors and matrices are also sub-Gaussian with parameter $\alpha$.

### 2.2 Cyclotomic Ring

For security parameter $\lambda$, let $\Phi_{2N}(X) = X^N + 1$ be the $2N$-th cyclotomic polynomial, where $N = N(\lambda)$ is a power of 2. Let the cyclotomic ring $\mathcal{R} = \mathbb{Z}[X]/(X^N + 1)$, and $\mathcal{R}_Q = \mathcal{R}/(Q\mathcal{R})$ denotes the residue ring of $\mathcal{R}$ modulo an integer $Q$, and then there is an isomorphic mapping $\mathcal{R}_Q \simeq \mathbb{Z}_Q^N$ which is defined as $f : a \rightarrow \bar{a}$, where $a = a_0 + a_1X + \ldots + a_{N-1}X^{N-1} \in \mathcal{R}, \bar{a} = (a_0, a_1, \ldots, a_{N-1}) \in \mathbb{Z}_Q^N$. The notation $\bar{\cdot}$ can be extended to vector and matrix over $\mathcal{R}$. We define the Euclidean length of ring elements as $||a|| = ||\bar{a}|| = \sqrt{\sum |a_i|^2}$, and the spectral norm of a matrix $R \in \mathcal{R}^{w \times k}$ of ring elements as $s_1(R) = \sup_{x \in \mathcal{R}^{k \times 0}} ||R \cdot x||/||x||$. Given the ring element $r \in \mathcal{R}$, we define the function $r \Rightarrow$ as

$$
\Rightarrow r = \begin{bmatrix} r_0 & r_{N-1} & \ldots & r_1 \\
r_1 & r_0 & r_2 \\
\vdots & \ddots & \ddots & \vdots \\
r_{N-1} & r_{N-2} & \ldots & r_0
\end{bmatrix} \in \mathbb{Z}^{N \times N}
$$

(1)
And we extend the function $\rightarrow$ to vectors and matrices over $\mathcal{R}$. Given a matrix $R \in \mathcal{R}^{w \times k}$, we expand it to matrix $R \in \mathcal{R}^{Nw \times Nk}$ with $w \times k$ blocks $R_{ij} \in \mathcal{R}^{N \times N}$. Notice that $s_1(\rangle) = s_1(\langle)$ and $s_1(R) = s_1(\langle R)$ are established.

For ease of easily understanding, two functions $(-1)$ and $(1)_{(x,y)}^I$ are defined in this paper:

$$\begin{align*}
(-1)_{l,k} & = \begin{cases} 
1 & l < k \\
0 & l \geq k
\end{cases} \\
(1)_{(x,y)}^v & = \begin{cases} 
0 & v \notin (x,y) \\
1 & v \in (x,y)
\end{cases}
\end{align*}$$

(2) (3)

where $l, k, x, y$ and $v$ are all integers.

### 2.3 The General Learning with Errors (GLWE) Problem

The basic homomorphic encryption scheme in this paper is based on the hardness of (Ring) learning with errors (LWE) problem, which can be summarized as the GLWE problem [17], and it’s defined as follows.

**GLWE problem:** For the security parameter $\lambda$, let $f(x) = x^d + 1$, where $d = d(\lambda)$ is a power of 2. Let the modulus $q = q(\lambda)$ and dimension $n$ be integer, let $R = \mathbb{Z}[x]/f(x)$ and $R_q = R/qR$. Let $\chi = \chi(\lambda)$ be a distribution on $R$. The GLWE problem is to distinguish the following two distributions: First distribution is the uniform samples $(a_i, b_i) \in R_q^{n+1}$. In the second distribution, sampled $a_i \leftarrow R_q^n$ and $s \leftarrow R_q^s$ uniformly, $e_i \leftarrow \chi$, and the second distribution is the samples $(a_i, b_i) \in R_q^{n+1}$ where $b_i = a_i \cdot s + e_i$. The GLWE assumption is that the GLWE problem is infeasible.

**LWE problem:** The LWE problem is simply GLWE problem instantiated with $d = 1$.

**RLWE problem:** The RLWE problem is GLWE problem instantiated with $n = 1$.

### 3. THE NEWLY DESIGNED SCHEME

According to Gentry’s theory, a “noisy” ciphertext can be refreshed by running the bootstrapping procedure to reduce the noise involved in it. In earlier FHE scheme, the decryption algorithms for ciphertext decryption and bootstrapping are the same. For optimizing the efficiency of FHE, since AP14, many FHE schemes are divided into internal and external schemes, and the decryption algorithms are usually different. We write enc($\cdot$) (E($\cdot$)) for the encryption scheme of internal (external) scheme. Ciphertext in internal (external) scheme is called internal (external) ciphertext. We can homomorphically run the internal decryption program on the external ciphertext to refresh the “noisy” ciphertext, and get a more efficient bootstrapping process. This type of bootstrapping is called general bootstrapping in this paper.

FHE scheme usually contains three parts: basic encryption scheme, homomorphic evaluation and bootstrapping. In this paper, the basic encryption scheme and the homomorphic evaluation follow the basic LWE symmetric encryption scheme in [25]. The main difference and innovation of our scheme lie in the bootstrapping process.

As bootstrapping process needs to run the decryption procedure in internal scheme homomorphically, we firstly introduce the basic encryption scheme, then present the optimization of the decryption process, and introduce the design of the general bootstrapping via homomorphic accumulator at last.

### 3.1 Basic Scheme and Techniques

In this section, we will introduce the basic LWE symmetric encryption scheme used in the internal scheme and two basic techniques (Modulus switching and Key switching [17]). The external encryption scheme will be introduced in Section 4.1.

**LWE Symmetric Encryption Scheme:** Let $n$ denote the dimension of the ring $\mathcal{R}$. $[1]_q$ denotes the reduction modulus $q$ into the interval $(-q/2, q/2)$. Set the plaintext modulus $t \geq 2$ and the ciphertext modulus $q = q(\lambda)$.

Define the randomized rounding function $\psi : \mathbb{R} \rightarrow \mathbb{Z}$ (For the sake of decrypting correctly, usually the noise distribution of the rounding function satisfies $|\psi(x) - x| < q/(2t)$). The keys $s \in \mathbb{Z}_q^n$ of the scheme are randomly selected vectors (or short vectors). The encryption of plaintext $m \in \mathbb{Z}_t$ under the key $s$ is

\[
LWE_{s/q}^{m/q}(m) = c = (a, b) = (a, \psi(a \cdot s + mq/t) \bmod q) \in \mathbb{Z}_q^{n+1}
\]

(4)

where $a \leftarrow \mathbb{Z}_q^n$ is chosen uniformly at random.

Let $err(a, b) = (b - a \cdot s - mq/t) \bmod q$ denotes the error of the ciphertext $c$, and $LWE_{s/q}^{m/q}(m, B)$ denotes $|err(a, b)| < B$. For the sake of correct decryption, we need the ciphertext $c \in LWE_{s/q}^{m/q}(m, q/(2t))$. As $|b - a \cdot s|$
= ψ(mq/t), and |ψ(mq/t − mq/t| ≤ q/2t, we have |t · ψ(mq/t)/q − m| ≤ 1/2, therefore

\[ m = [t(b − a ⋅ s)/q] \mod t \]

The homomorphic computations in the basic scheme follow BGV scheme:

\[ c_+ = \text{bootstrapping}_s(c_1 + c_2) \]

\[ c_\times = \text{bootstrapping}_s \otimes_s (c_1 \otimes c_2), \]

where “⊗” means tensor product.

Modulus switching and Key switching are two techniques often used in current FHE [24], and here we give a brief introduction to them.

**Modulus switching:** Modulus switching technique is an efficient way to manage the noise involved in the during homomorphic operations. It can convert a ciphertext \( c \in \text{LWE}_{\sigma/Q}(m) \) with modulus Q to another ciphertext \( c' \in \text{LWE}_{\sigma/q}(m) \) with a new modulus \( q \), while keeping the corresponding plaintext unchanged.

\[ \text{ModSwitch}(c \in \text{LWE}_{\sigma/Q}(m), q) = c' \in \text{LWE}_{\sigma/q}(m) \]

**Lemma 1:** [13] Given the secret key \( s \in \mathbb{Z}_q \), message \( m \in \mathbb{Z}_t \) and ciphertext \( c \in \text{LWE}_{\sigma/Q}(m) \) with sub-Gaussian error of the parameter \( \sigma \), the output of \( \text{ModSwitch}(c) \) is a \( \text{LWE}_{\sigma/q}(m) \) ciphertext with a sub-Gaussian error of parameter \( \sqrt{(q\sigma/Q)^2 + 2\pi(||s||^2 + 1)} \).

**Key switching:** Key switching technique can convert a ciphertext \( c \in \text{LWE}_{\sigma/q}(m) \) under the key \( s \) into another ciphertext \( c' \in \text{LWE}_{\sigma/q}(m) \) under the key \( s' \), while keeping the corresponding plaintext unchanged. Let \( k_{s,j,v} \) be the encryption of \( vs \cdot B_{ks}^j \), where \( B_{ks} \) is the base, \( v = 0, \ldots, B_{ks} \) and \( j = 0, \ldots, [\log B_{ks}, q] \). The switching key \( \mathcal{R} \) can be denoted as \( \mathcal{R} = \{k_{s,j,v} \in \text{LWE}_{\sigma/q}(v \cdot s_j B_{ks}^j)\} \), and the key switching procedure can be represented as:

\[ \text{KeySwitch}(c \in \text{LWE}_{\sigma/q}(m), \mathcal{R}) = c' \in \text{LWE}_{\sigma/q}(m) \]

The key switching technique can reduce the dimension of the expanded ciphertext after the homomorphic computation (mainly homomorphic multiplication) to the original dimension, thus can save the storage cost for ciphertext.

**Lemma 2:** Given the ciphertext \( c \in \text{LWE}_{\sigma/q}(m) \) with a sub-Gaussian error of parameter \( \alpha \), the switching key \( \mathcal{R} = \{k_{s,j,v}\} \), and a sub-Gaussian error \( \sigma \) with parameter \( \sigma \). The key switching procedure outputs the ciphertext \( \text{KeySwitch}(c, \mathcal{R}) \) in which the noise is a sub-Gaussian distribution of parameter \( \sqrt{\alpha^2 + Nd_{ks}\sigma^2} \).

### 3.2 Optimization of the Decryption Process

Given the ciphertext \( (a, b) \in \text{LWE}_{\sigma/q}(m, q/2t) \), the decrypting process of the basic LWE scheme is \( [t(b − a \cdot s)/q] \mod t \). As the rounding operation is not easy to be implemented to the ciphertext, some conversion is needed. Let \( \text{msd}(\cdot) \) (\( \text{msdExtract}(\cdot) \)) denotes extracting the most significant digit of the plaintext (ciphertext). We found the rounding function in the decryption process can be converted to \( \text{msd}(\cdot) \) or \( \text{msdExtract}(\cdot) \), and the analysis is as follows.

First, we use \( t = 4 \) as an example to analyze the basic decryption process. As shown in the Figure 1, the points on the circle represent \( \mathbb{Z}_q \), and the value of \( \rho = b − a \cdot s \) can be regarded as a point on the circle. Regardless of the noise \( e \), as \( b − a \cdot s = mq/t + e \), there are four possible positions of \( \rho \) on the circle: 0, q/4, 2q/4, 3q/4. Due to the interference of noise \( e \), the corresponding position of \( \rho \) is uncertain on the circle. By limiting the noise \( e \) to \( |e| ≤ q/8 \), we define a mapping \( \pi : \mathbb{Z}_q \rightarrow \mathbb{Z}_t \) from \( \rho \) to the corresponding plaintext \( m \):

\[ \pi : \mathbb{Z}_q \rightarrow \mathbb{Z}_t \]

\[ \begin{align*}
\rho & \in [-q/2t, q/2t) \rightarrow 0 \\
\rho & \in [q/2t, 3q/2t) \rightarrow 1 \\
& \vdots \\
\rho & \in [5q/2t, 7q/2t) \rightarrow t − 1
\end{align*} \]

![Figure 1: The mapping π from ρ to the corresponding plaintext m (t = 4)](image-url)
According to the mapping \( \pi \) between the points on the circle \( (\rho = b - a \cdot s) \) and the corresponding plaintext \( m \), the correct decryption is achieved.

When adding \( q/2t \) to \( \rho = b - a \cdot s \), the new mapping \( \pi' : \mathbb{Z}_q \rightarrow \mathbb{Z}_t \) from \( (\rho + q/2t) \) to \( m \) can be expressed as

\[
\pi' : \mathbb{Z}_q \rightarrow \mathbb{Z}_t \\
(\rho + q/2t) \rightarrow m
\]

where

\[
\begin{align*}
\rho & \in [0, q/t) \rightarrow 0 \\
\rho & \in [q/t, 2q/t) \rightarrow 1 \\
\rho & \in [(t - 1)q/t, q) \rightarrow t - 1 
\end{align*}
\]

and the new mapping \( \pi' \) can be represented visually in Figure 2.

When \( q = t^4 \), \( (\rho + q/2t) \) can be represented by \( (\rho + q/2t)_t \) which has \( k - 1 \) digits in base:

\[
\begin{align*}
(\rho + \frac{q}{2t}) & \in [0, \frac{q}{t^2}) \rightarrow \cdots (k - 1 \text{ digits}) \\
(\rho + \frac{q}{2t}) & \in [\frac{q}{t^2}, \frac{2q}{t^2}) \rightarrow \cdots (k - 1 \text{ digits}) \\
\vdots \\
(\rho + \frac{q}{2t}) & \in [\frac{(t - 1)q}{t^2}, q) \rightarrow \cdots (k - 1 \text{ digits})
\end{align*}
\]

As we can see in (10), the plaintext \( m \) in mapping \( \pi' \) can be represented by \( \text{msd}((\rho + q/2t)_t) \). Naturally, the mapping \( \pi' \) can be expressed as

\[
\begin{align*}
\pi' : \mathbb{Z}_q & \rightarrow \mathbb{Z}_t \\
(\frac{q}{2t} + \rho) & \rightarrow \text{msd}((\rho + q/2t)_t)
\end{align*}
\]

Then we successfully convert the decryption process of the basic LWE encryption scheme to \( \text{msd}() \), and we will give the detailed steps for homomorphically extracting the most significant digit of ciphertext by \( \text{msdExtract}() \) in Section 3.3.

### 3.3 General Bootstrapping via Homomorphic Accumulator

In order to refresh the “noisy” ciphertext \( c = (a, b) \in \text{LWE}_{\epsilon}^{l/q}(m) \), we need to run the homomorphic decryption procedure on \( \text{E}(b + q/2t - a \cdot s) \). As the decryption algorithm of the internal scheme has been optimized to \( \text{msd}(\cdot) \), we construct a mapping \( \zeta \), and the refreshing process can be expressed as

\[
\zeta : \text{E}(v) \xrightarrow{\text{msdExtract}(\cdot)} \text{enc}(\text{msd}(v)) \\
\text{E}(b - a \cdot s + \frac{q}{2t}) \xrightarrow{\text{msdExtract}(\cdot)} \text{enc}(\text{msd}(b - a \cdot s + \frac{q}{2t})) = \text{enc}(m)
\]

Before giving detailed steps about how to implement the mapping, we first introduce the homomorphic accumulator \( \text{ACC} \), which is employed to support the construction of the mapping, and the specific process will be introduced in Section 4.1.

**Definition 1:** (homomorphic accumulator \( \text{ACC} \)) The homomorphic accumulator \( \text{ACC} \) contains five functions \( (\text{E}(\cdot), \text{Init}(\cdot), \text{Incr}(\cdot), \text{uMult}(\cdot), \text{msdExtract}(\cdot)) \). \( \text{E}(\cdot) \) is the encryption algorithm of the external scheme, \( \text{Init}(\cdot) \) is the initialization function, \( \text{Incr}(\cdot) \) is the homomorphic adjustment function, \( \text{uMult}(\cdot) \) is the parameter adjustment function, \( \text{msdExtract}(\cdot) \) is the function which homomorphically extract the most significant digit.

For brevity, we write \( \text{ACC} \leftarrow v \) for \( \text{ACC} \leftarrow \text{Init}(v) \), and \( \text{ACC} \leftarrow E(v) \) for \( \text{ACC} \leftarrow \text{Incr}(\text{ACC}, E(v)) \). For \( v_0, v_1, \ldots, v_l \in \mathbb{Z}_q \), after the sequences of operations

\[
\text{ACC} \leftarrow v_0; \text{ACC} \leftarrow \text{E}(v_l)_{l=1,\ldots,l}
\]

the output of the ACC is called the \( l \)-encryption of \( v \), where \( v = \sum v_i \bmod q \).

The homomorphic accumulator \( \text{ACC} \) is said \( \epsilon \)-correct, only if for arbitrary \( l \)-encryption \( \text{ACC} \) of \( v \), \( c \leftarrow \text{msdExtract}(\text{ACC}) \) ensures \( \text{LWE}_{\epsilon}^{l/q}(\text{msd}(v), \epsilon(l)) \) with overwhelming probability.

Next, we give a brief introduction to general bootstrapping and how to homomorphically implementing the \( \text{msdExtract}(\cdot) \) function.

Using the bootstrapping keys and homomorphic accumulator, the general bootstrapping is constructed in algorithm 1.
Algorithm 1 Getbootstrapping($\mathcal{K}$, $(a, b)$, where $\mathcal{K} = \{K_{ij} = E(c \cdot s_j b_i^j)\}_{i \leq j}$).

1. ACC ← $b + \frac{a}{2t}$
2. for $i = 1, \ldots, n$, do
   Compute $-a_i = \sum b_i \cdot a_i (\text{mod} q)$ and get $a_{ij}$
   for $j = 0, \ldots, d_r - 1$, do
   ACC $\leftarrow K_{ai,j} = E(a_{ij} \cdot s_j b_i^j)$
3. output msdExtract(ACC)

Where $\mathcal{K} = \{K_{ij} = E(c \cdot s_j b_i^j)\}_{i \leq j}$ are bootstrapping keys, $B_i$ is the decomposing base, $d_r = \lceil \log_B q \rceil, c \in \{0, 1, \ldots, B_r - 1\}$.

- For the “noisy” ciphertext $c = (a, b)$, $E(b + q/2t - a \cdot s)$ can be computed by $\sum_{i=0}^n \sum_{j=0}^{d_r-1} ((\text{ACC} \leftarrow \text{Init}(q/2t + b)) \leftarrow K_{ai,j})$. The analysis is as follows:

$$E \left( \frac{q}{2t} + b - a \cdot s \right) = E \left( \frac{q}{2t} + b \right) \oplus E(-a \cdot s)$$
$$= (\text{ACC} \leftarrow \text{Init}(q/2t + b)) \oplus E(-a \cdot s)$$
$$= \text{ACC} \oplus \sum_{i=1}^n E(-a_i \cdot s_i)$$
$$= \text{ACC} \oplus \sum_{i=0}^n \sum_{j=0}^{d_r-1} E(a_{ij} \cdot s_j b_i^j)$$
$$= \text{ACC} \oplus \sum_{i=0}^n \sum_{j=0}^{d_r-1} K_{ai,j}$$
$$= \sum_{i=0}^n \sum_{j=0}^{d_r-1} (\text{ACC} \leftarrow K_{ai,j})$$

(12)

The msdExtract($\cdot$) function can extract the most significant digit of external ciphertext, i.e.

$$\text{msdExtract}(E(b + q/2t - a \cdot s))$$
$$= \text{enc}(E(b + q/2t - a \cdot s))$$

Then the ciphertext $c = (a, b)$ is refreshed, and a flowchart of our scheme is provided in Figure 3.

4. THE DETAILED CONSTRUCTION OF THE SCHEME

In this section, we present the details of the functions in the homomorphic accumulator, and analyze the components in the homomorphic accumulator.

4.1 The Construction of Homomorphic Accumulator

For security parameter $\lambda$, the parameters used in homomorphic accumulator include the ciphertext modulus $Q = Q(\lambda)$ and plaintext modulus $q = q(\lambda)$ in the external scheme ($q$ is also the ciphertext modulus in internal LWE scheme), the plaintext modulus $t = t(\lambda)$ in the internal scheme, and the decomposing base $B_g$ of external ciphertext. For simplicity of the analysis, we assume that $Q = B_g d_f$ for some integer $d_f$.

The ring in the external scheme is $\mathcal{R}_Q = \mathcal{R}/Q\mathcal{R}$, where $\mathcal{R} = \mathbb{Z}[X]/(X^N + 1)$. The integer $u \in \mathbb{Z}_Q$ is a reversible element closest to $Q/t$ in $\mathbb{Z}_Q$. In order to decrypt correctly, the parameter $\delta = u - Q/t$ is at most $|\delta| < 1$. The condition for parameter selection is that there are integers $d_r \neq 0$, $t$ and $k$ which ensure that $q/t \cdot t = N + d_r \cdot \tau$, $\tau = d_h r^2 (t + 1)$, and the modulus $q = t^k$.

The plaintext $m \in \mathbb{Z}_q$ needs to be encoded as a unit root $y^m \in \mathcal{R}$ before being encrypted in the external scheme, where $Y = X^r$. Noticing that the unit roots $G = \{X\}$ is a cyclic group, and the group $\langle Y \rangle = \{Y^0, \ldots, Y^{t-1} \}$ is a subgroup of $G$, we will be given in 4.2.

The homomorphic accumulator contains five parts: $E_z(m)$, Init(ACC $\leftarrow v$), Incr(ACC $\leftarrow C$), mUltcontant (C $\leftarrow k$), msdExtract(ACC).

- $E_z(m)$. Input the message $m$ and key $z \in \mathcal{R}$, pick $a \overset{R}{\longleftarrow} \mathcal{R}_{Qd_f}$ uniformly at random, and $e \in \mathcal{R}_{d_f} \simeq \mathbb{Z}_{d_f}^{\mathbb{Z}_{d_f}^N}$ with a sub-Gaussian distribution $\chi$ of the

Figure 3: A brief description of the scheme
parameter \( z \), output the external ciphertext

\[
E_c(m) = [a, a \cdot z + e] + uY^m G
\]

\[
= \begin{bmatrix}
a_1 & a_1 \cdot z + e_1 \\
a_2 & a_2 \cdot z + e_2 \\
\vdots & \vdots \\
a_{2d-1} & a_{2d-1} \cdot z + e_{2d-1} \\
a_{2d} & a_{2d} \cdot z + e_{2d} \\
\end{bmatrix}
\]

where \( G = (I, B_1, \ldots, B_{d-1}) \in \mathcal{R}^{2^{d-2} \times 2} \), and \( I \in \mathcal{R}^{2 \times 2} \) is the identity matrix.

- \( \text{Init}(\text{ACC} \leftarrow v) \). Input \( v \in \mathbb{Z}_q \), and output the initialized accumulator \( \text{ACC} := uY^v \cdot G \in \mathcal{R}^{2^{d-2} \times 2} \).
- \( \text{Incr}(\text{ACC} \leftarrow C) \). Input \( \text{ACC} \in \mathcal{R}^{2^{d-2} \times 2} \) and the external ciphertext \( C \in \mathcal{R}^{2^{d-2} \times 2} \), decompose the accumulator \( \text{ACC} \) based on \( B_g \) as

\[
u^{-1} \text{ACC} = \sum_{i=1}^{d} B_i^{-1} D_i \in \mathcal{R}^{2^{d-2} \times 2}
\]

where each \( D_i \) has entries with coefficients in \( \{1 - B_g, \ldots, (B_g - 1)/2\} \), and then update the accumulator \( \text{ACC} := [D_1, \ldots, D_d] \cdot C \in \mathcal{R}^{2^{d-2} \times 2} \).

- \( \text{uMultCont}(C \leftarrow k) \). Input a constant \( k \in \mathbb{Z}_q \) and ciphertext \( C \in \mathcal{R}^{2^{d-2} \times 2} \) with parameter \( u \in \mathbb{Z}_q \), output a ciphertext \( C_{k u} \leftarrow kC \in \mathcal{R}^{2^{d-2} \times 2} \) with the parameter \( ku \in \mathbb{Z}_q \). For brevity, we denote \( C_{ku} \in \mathcal{R}^{2^{d-2} \times 2} \) as \( C(k) \).
- \( \text{msdExtract}(\cdot) \). Firstly the plaintext \( m \in \mathbb{Z}_q \) is encoded as a monomial \( Y^m \) in \( (Y) \). Secondly, the operation of plaintext is achieved by operating the position of the term in the monomial.

Input the switching key \( \mathcal{R} \) and a set of vectors \( T = \{t(1), t(2), \ldots, t(t)\} \), where \( t(\mathcal{k}) = (-1)k+1 \sum_{i=0}^{\beta} Y^i \), \( \alpha = (k - 1)q/t \beta = kq/t - 1 \). The details of the function \( \text{msdExtract}(\cdot) \) are shown in Algorithm 2.

where \( \text{msd}(k)(v) = \begin{cases} 1 & \beta \leq v < \alpha \\ 0 & \text{else} \end{cases} \), \( b(\mathcal{k}) \) is a vector consisting of the last \( N \) components of the 2N-dimension vector \( [\tilde{0}, t(\mathcal{k})]^T, \tilde{0}, \ldots, \tilde{0}] \cdot \text{ACC}(k) \), and \( b(\mathcal{k}) \) is the first component of \( b(\mathcal{k}) \).

**4.2 Analysis of Homomorphic Accumulator Components**

This section analyzes the functions in the homomorphic accumulator and their properties. The plaintext \( m \in \mathbb{Z}_q \) needs to be encoded as \( Y^m \in \mathcal{R} \) before being encrypted in the external scheme, so we need to prove that \( < Y > = \{Y^0, \ldots, Y^{q-1}\} \) forms a group. As ciphertext computing is operated in \( \mathcal{R}_Q \), we also need to prove that the elements in \( < Y > \) are contained in \( \mathcal{R} \). Actually, group \( < Y > \) is a subgroup of \( G^- = < X > \), and the distribution of \( < Y > = \{Y^0, \ldots, Y^{q-1}\} \) in \( \mathcal{R} \) is shown in Figure 4.

As is shown in Figure 4, the \( i \)-th column in the matrix represents for \( \pm X^i \), and the \( j \)-th row in the matrix indicates that the power of \( X \) in the row is greater than \( (j - 1)N \) and less than \( jN \), and most of the zeros in the graph are not marked. Analysis of the matrix shows that if there is an integer \( d_{tr} \neq 0 \) ensure that \( \tau = td_{tr}, \tau [(N + d_{tr}) and q = tN + d_{tr}] \), then each column will only have one non-zero element. This ensures that the elements of \( < Y > \) are different. More importantly, for \( m \in \mathbb{Z}_q \), \( Y^m \) just appear in the \( \text{msd}(m) \)-th row, and this property is the key to the construction of \( \text{msdExtract}(\cdot) \).

**Lemma 3:** If there is an integer \( d_{tr} \neq 0 \) ensures that \( \tau = td_{tr}, \tau [(N + d_{tr}) and q = tN + d_{tr}] \) and \( (t + 1) \), then \( < Y > = \{Y^0, \ldots, Y^{q-1}\} \) is a subgroup of cyclic group \( G^- = < X > = \{X\} \), where \( Y = X^\tau \).

**Proof:** Firstly we prove that the set \( < Y > \) forms a group. As the elements in \( < Y > \) inherit the associative law
and identity element of the ring \( R \), it only needs to prove that the set \( < Y > \) satisfies the closure, and the elements in the set are unique and have the inverse element.

**Closure:**

\[
Y^q = X^{(N+td_r)\frac{t}{\tau}} = X^{(l+1)N} = Y^0
\]

(16)

**Inverse element:** As \( Y^m \cdot Y^{-m} = 1 \), then the inverse element of \( Y^m \) is \( Y^{-m} \).

**Uniqueness:** Here we use proof by contradiction.

If \( i \leq j \) and \( Y^i = Y^j \), then \( Y^j / Y^i = X^{(j-i)} = 1 \) and \( 2N|\tau (j-i) \). As \( 0 \leq j-i \leq q-1 \), then \( \tau (j-i) \leq \tau (q-1) < tN \), which is contradictory with \( 2N \) \( \tau (j-i) \). So there is \( Y^j \neq Y^i \) established and ensure the elements in set \( Y \) are different.

Secondly, the elements in \( < Y > \) are contained in \( G^- = < X > \), then \( < Y > \) is a subgroup of the cyclic group \( G^- \).

**FACT 5 (Analysis to the addition function of the accumulator)** For two plaintexts \( m, m' \in \mathbb{Z}_q \), if \( \text{ACC} = [a, a \cdot z + e] + uY^m G \) and \( \text{C} = [a', a \cdot z + e'] + uY^m G \), then \( \text{ACC} + C \) has the form of \( \text{ACC} = [a'', a'' \cdot z + e''] + uY^{m+m'} G \), where \( e'' = Y^m \cdot e + [D_1, \ldots, D_{d_y}] \cdot e' \).

**Lemma 4:** (Analysis of parameter adjustment function) For plaintext \( m \in \mathbb{Z}_q \) and constant \( k \in \mathbb{Z}_q \), if \( \text{C} = [a, a \cdot z + e] + uY^m G \), then the output of \( u\text{Multcontant}(C \leftarrow k) \) has the form of \( \text{C}_{ku} = [a'', a'' \cdot z + ke] + ukY^m G \) where \( e \) is the error in \( C \).

**Proof:** As \( u\text{Multcontant}(C \leftarrow k) \) outputs the ciphertext \( \text{C}_{ku} \leftarrow kC \), let \( a'' = ka \), then

\[
\text{C}_{ku} = k[a, a \cdot z + e] + kuY^m G = [ka, ka \cdot z + ke] + kuY^m G = [a'', a'' \cdot z + ke] + ukY^m G
\]

(17)

So the ciphertext has form of \( \text{C}_{ku} = [a'', a'' \cdot z + ke] + ukY^m G \), where \( e \) is the noise in \( C \).

5. ANALYSIS OF OUR SCHEME

In this section, we analyze the correctness of the general bootstrapping process and the scheme's security.

5.1 Analysis of the Correctness of the Generalized Bootstrapping

Based on the construction and analysis of the homomorphic accumulator, this section analyzes the correctness of the bootstrapping process in Algorithm 1.

**FACT 7 (Spectral Norm of Decomposition Matrix)** Let \( C(i) \leftarrow E_z(v(i)) \) be the fresh ciphertext of \( v(i) \in \mathbb{Z}_q \), where \( i \leq l = \sqrt{\log n} \). Firstly compute \( \text{ACC} \leftarrow v(0) \), secondly define the value of accumulator after computing \( \text{ACC} \leftarrow C(i) \). Let \( D(i) = [D_1, \ldots, D_{d_y}] \) be based on \( B_g \) and decompose the output of accumulator \( u^{-1}\text{ACC}(i) = \sum_{j=1}^{d_y} B_{i,j}^{-1} D_j \) then there is \( s_1(D(0), D(1), \ldots, D(l-i)) = O(B_g \sqrt{Nd_g} \cdot l) \) with a large probability.

Lemma 8 analyzes the first four steps in msdExtract(\(-\)) function. Theorem 9 analyzes the impact of key switching and modulus switching process on ciphertext based on Lemma 8.

**Lemma 5** (Intermediate noise): Supposing that the RLWE\(_{R,Q,X}\) problem is hard, let accumulator \( \text{ACC} \) be the...
Proof: Run \( l \) times of homomorphic addition on the initial accumulator ACC, and the output of the accumulator is called ACC. According to Fact 5, ACC ciphertext has the form of the external ciphertext \( \text{ACC} = [a, a \cdot z + e] + uY^vG \), and satisfies

\[
e = \sum_{i=1}^{l} D^{(i-1)} e^{(i)} = \sum_{i=1}^{l} D^{(i-1)} y^{w(i)} e^{(i)} = Y^{w(0)} D^{(0)} e^{(1)} + Y^{w(1)} D^{(1)} e^{(2)} + \ldots + D^{(l-1)} y^{w(l)} e^{(l)}
\]

where \( e^{(i)} = Y^{w(i)} e^{(i)} + w^{(i)} \in \mathbb{Z}_q \), and \( e^{(i)} \) is the noise used in the encryption of \( C^{(i)} = E_x(t^{(i)}) \).

Line 1: According to Lemma 4, ciphertext \( \text{ACC}(k) = \text{uMultcontant(ACC} \rightarrow k - 1) \) has the form of the external scheme \( \text{ACC}(k) = [a(k), a(k) \cdot z + e(k)] + u(k-1) Y^v G \), where noise \( e(k) = (k-1)e \).

Line 2: Suppose that \( [a(k), b(k)] \in \mathbb{R}^{l \times 2} \) is the second line of the accumulator \( \text{ACC}(k) \in \mathbb{R}^{2d_g \times 2} \), thus \( b'(k) = a(k) \cdot z + e(k) + (k-1)uY^v \), where \( e(k) = (k-1)e \) and \( e \) is the item of noise \( e \) involved in the second line of the original ACC ciphertext. According to the definition of \( \rightarrow \) and \( \Rightarrow \), we have \( b'(k) \rightarrow = a(k) \rightarrow \cdot z + (k-1)uY^v \).

As vector \( \overrightarrow{[\hat{0}, (t_{(k)})^f, \hat{0}, \ldots, \hat{0}]} \) is 0 except for the second item, we have

\[
[(a(k))^f, (b(k))^f] = [\overrightarrow{0}, (t(k))^f, \overrightarrow{0}, \ldots, \overrightarrow{0}] \rightarrow \Rightarrow
\]

\[
= (t(k))^f \cdot [(a(k), b(k)]
\]

Line 3: As \( [(a(k))^f, (b(k))^f] = (t(k))^f \cdot [a(k), b'(k)] \), there is \( a(k)^f = (t(k))^f \cdot a(k) \) and \( b'(k) = (t(k))^f \cdot b'(k) \). As \( b'(k) = \overrightarrow{a(k) \cdot z + (k-1)uY^v + (k-1)e} \), we can further have

\[
b(k)^f = (k-1)u \cdot (1)_{[u,v]} + (t(k))^f \cdot (k-1)e
\]

where \( t(k) \cdot \overrightarrow{y^v} = (1)_{[u,v]}^v \). So \( (a(k), b(k)] \) has the form of LWE ciphertext, and the noise contained in ciphertext \( c(k) = \langle a(k), b'(k) \rangle \) is \( e(k) = (k-1)(t(k))^f \cdot e \), where \( e \) is the item of noise \( e \) involved in the second line of the original ACC ciphertext. Then

\[
e(k) = (k-1)(t(k))^f \cdot e = (k-1)\overrightarrow{[\hat{0}, (t(k))^f, \overrightarrow{0}, \ldots, \overrightarrow{0}] \cdot e}
\]

As \( e = \sum_{i=1}^{l} D^{(i)} y^{w(i)} e^{(i)} \), thus the noise \( e(k) \) can be represented as

\[
e(k) = (k-1) \cdot \overrightarrow{[\hat{0}, (t(k))^f, \overrightarrow{0}, \ldots, \overrightarrow{0}] \cdot e}
\]

As the number of items in vector \( t \) does not exceed \( q/t \), then \( ||t|| \leq q/t \). According to Fact 7 we can see that the spectral norm of \( [D^{(0)}, D^{(1)}, \ldots, D^{(l-1)}] \) is \( O(B_g \sqrt{qN_d}) \), hence the noise \( e(k) \) can be represented as \( e(k) = v \cdot (\overrightarrow{e}, \ldots, \overrightarrow{e}) \), where \( |v| = O((k-1)B_g \sqrt{qN_d} \cdot l/t) \), and \( (\overrightarrow{e}, \ldots, \overrightarrow{e}) \) is a sub-Gaussian vector of the parameter \( \varsigma \), so the noise \( e(k) \) is a sub-Gaussian distribution of the parameter \( \phi = O((k-1) \sqrt{qN_d} \cdot l^3) \).

Line 4: Add \( t \) LWE ciphertexts and get

\[
c = \sum_{k=1}^{k=t} c(k) = \sum_{k=1}^{k=t} (a(k), a(k) \cdot z) + t(k) \cdot (k-1)e + (k-1)u \cdot (1)_{[u,v]} \]

\[
= \sum_{k=1}^{k=t} (a(k), a(k) \cdot z) + \sum_{k=1}^{k=t} t(k) \cdot (k-1)e + u \cdot \text{msb}(v)
\]

where \( \sum_{k=1}^{k=t} ((k-1)1_{[u,v]}^v) = \sum_{k=1}^{k=t} ((k-1) \cdot (1)_{[u,v]}^v) = \text{msb}(v) \).

The noise in the ciphertext \( c \) is \( e = \sum_{k=1}^{k=t} e(k) \), which is a sub-Gaussian distribution of the parameter \( \beta = O((k-1)B_g \sqrt{qN_d} \cdot l/t) \). As \( 1 \leq k \leq t \), then noise \( e \) is a sub-Gaussian distribution of parameter \( \beta = O(B_g \sqrt{qN_d} \cdot l^3) \).

Considering the process of key switching and modulus switching in Algorithm 2 on the basis of Lemma 8, and analyze the noise function in the accumulator to obtain the Theorem 9.

Theorem 9: Supposing that RLWE_{\mathbb{R},Q,X} problem is hard and the homomorphic accumulator above is \( e \)-correct. When inputting the switching keys \( \mathbb{R} = \{k_{ij}, w_{ij} \} \leq N, j \leq B_{k_3}, w \leq d_{k_3} \) and the \( l \)-encryption of \( v \) to accumulator
ACC, \( k_{ij,w} \in \text{LWE}_s^{q'/q}(w \cdot z_i B_{k,j}^j) \), and \( \text{ACC} \) outputs the \( \text{LWE}_s^{q'/q}(\text{msd}(v)) \) ciphertext with the noise parameter \( \varepsilon(l) = w \cdot \sqrt{\log n} \cdot \sqrt{\frac{q^2}{Q}} (\varepsilon^2 B_g^2 \cdot l \cdot q \cdot N_{ds} t^3 + \sigma^2 N_{ds}) + ||s||^2 \) (23)

**Proof:** According to Lemma 8, ciphertext \( c = \sum_{k=1}^{k=q} c_{(k)} \) belongs to \( \text{LWE}_s^{q'/q}(\text{msd}(v)) \), which contains the noise \( \text{err}(c) = \text{err}(\sum_{k=1}^{k=q} c_{(k)}) \), which is a sub-Gaussian distribution of parameter \( \phi \) and mean \( 2\delta \), where \( \phi = O(\varepsilon B_g \sqrt{q \cdot N_{ds} \cdot l^3}) \).

According to Lemma 2, when input the ciphertext instance \( c \) of \( \text{LWE}_s^{q'/q}(\text{msd}(v)) \), the key switching process will output \( \text{LWE}_s^{q'/q}(\text{msd}(v)) \) ciphertext \( c' \leftarrow \text{KeySwitch}(c, R) \), where the noise is a sub-Gaussian distribution of parameter

\[
\sqrt{\phi^2 + N_{ds} \sigma^2} = \sqrt{(O(\varepsilon B_g \sqrt{q \cdot N_{ds} \cdot l^3})^2 + N_{ds} \sigma^2)}
\] (24)

According to Lemma 1, when input the ciphertext instance \( c' \) of \( \text{LWE}_s^{q'/q}(\text{msd}(v)) \), the modulus switching process will output \( \text{LWE}_s^{q'/q}(\text{msd}(v)) \) ciphertext \( c' \leftarrow \text{ModSwitch}(c') \), where the noise is a sub-Gaussian distribution of parameter

\[
\sqrt{\frac{q^2}{Q}} (O(\varepsilon B_g \sqrt{q \cdot N_{ds} \cdot l^3}))^2 + 2\pi ||s||^2 + 1)
\]

\[
= w \cdot \sqrt{\log n} \cdot \sqrt{\frac{q^2}{Q}} (\varepsilon^2 B_g^2 \cdot l \cdot q \cdot N_{ds} t^3 + \sigma^2 N_{ds}) + ||s||^2
\] (25)

**FACT 10** If the homomorphic accumulator is \( \varepsilon \)-correct, then input the ciphertext \( c \in \text{LWE}_s^{q'/q}(m, q/2t) \) and bootstrapping key \( K = (K_{i,j} = E(c \cdot s_i B_j))_{i,j} \), and the general bootstrapping process will output ciphertext \( \text{Gbootstrapping}_{K}(c) \leftarrow \text{LWE}_s^{q'/q}(\text{msd}(m), \varepsilon(\text{nd}_r)) \).

**Proof:** The proving process analyzes the specific steps of each row in Algorithm 1.

Line 1: According to the definition of function \( \text{Init} \), when input \( b + q/(2t) \), then output the initialized accumulator \( \text{ACC} = uY^{b+q/2t} \cdot G \in \mathcal{F}_{Q}^{2d_{g} \times 2} \).

Line 2: Computing \( l = nd_r \) times homomorphic additions to accumulator \( \text{ACC} \). The new input \( K_{i,a_{ij},j} = E_c(a_{ij} \cdot s_i B_j) \) of each addition is the fresh ciphertext of the external scheme.

According to Definition 3 we can see that ciphertext \( \text{ACC} \) is the \( l \)-encryption of \( v = b + q/2t + \sum_{ij} a_{ij}s_i B_j \), i.e. the ciphertext form of \( \text{ACC} \) is \( \text{ACC} = [a, a \cdot z + e] + uY^v G \), and the bound of noise \( e \) doesn’t exceed \( \varepsilon(\text{nd}_r) \). Analyze the plaintext \( v \):

\[
v = b + q/2t + \sum_{ij} a_{ij}s_i B_j = b + q/2t - \sum_{i=1}^{n} a_{i}s_i
\] (26)

Line 3: According to Theorem 9, when input the \( l \)-encryption of \( v \) to accumulator \( \text{ACC} \), the accumulator outputs the ciphertext \( \text{LWE}_s^{q'/q}(\text{msd}(v)) \), in which the bound of the noise is \( \varepsilon(l) \). As

\[
\text{msd}(v) = \text{msd}(b + q/2t - as) = [t(b - a \cdot s)/q] \mod t = m
\] (27)

then the output of the homomorphic accumulator is the ciphertext \( \text{Gbootstrapping}_{K}(c) \) of \( \text{LWE}_s^{q'/q}(\text{msd}(m), \varepsilon(\text{nd}_r)) \).

### 5.2 Scheme’s Computational Complexity and Security

**Computational Complexity.** In this paper, the basic encryption scheme and the homomorphic evaluation follow the basic LWE symmetric encryption scheme in [25]. The main difference and innovation of our scheme lie in the bootstrapping process, so we mainly analyze the computational complexity of the bootstrapping process as follows.

The computational complexity of step1-step2 in algorithm 1 is \( \tilde{O}(n \log^3 Q \cdot N \log N) \).

The computational complexity of step3 in algorithm 1 (i.e. the computational complexity of algorithm 2) is \( \tilde{O}(N^3 \log^3 Q) \).

So the computational complexity of the bootstrapping process is \( \tilde{O}(N^3 \log^3 Q) \). As the complexity of the switching process is relatively high, the computational complexity of the bootstrapping process in FHEW and BR15 is the same as ours. What’s more, the process of encryption, decryption, and key generation in our scheme is the
same as FHEW and BR15, so the complexities of these processes don’t change.

Security. Based on FHEW scheme, the main work of this paper is to extend the plaintext space of FHE to \( \mathbb{Z}_t \). There are two main differences between our scheme and FHEW. On the one hand, we optimize the rounding function during the decryption process to msdExtract function, thus get a basic LWE symmetric encryption scheme with plaintext space. The scheme’s security is related to the parameters such as noise, polynomial dimension, and plaintext modulus. In 2008, Regev gave specific relationships and instances between scheme’s security and these parameters [27], and the extension of plaintext space does not reduce the scheme’s security at the exponential level. On the other hand, the encryption algorithm follows the basic LWE scheme and RGSW scheme in FHEW, and the evaluations during bootstrapping are implemented directly on the encrypted data without knowing the secret keys, so it does not reduce the basic scheme’s security.

6. CONCLUSION

The security of multimedia content and personal privacy has become a hot topic in big data. FHE has been one of the important tools for personal privacy. In this paper, we proposed an FHE scheme with large plaintext space \( \mathbb{Z}_t \). Comparing to the plaintext space \( \mathbb{Z}_2 \) in FHEW, our scheme can perform the evaluation process more conveniently in large plaintext space, and can be applied to more scenarios. We will focus on optimizing the efficiency of bootstrapping and the applications of FHE in practice in our future work.

FUNDING

This work was supported by National Key R&D Program of China [grant number 2017YFB0802000]; National Natural Science Foundation of China [grant number U1636114, 61772550, 61572521, 61872384]; National Cryptography Development Fund of China [grant number MMJJ20170112].

REFERENCES

1. Y. Cao, Z. Zhou, X. Sun and C. Gao, “Coverless information hiding based on the molecular structure images of material,” Comput. Mater. Continua, Vol. 54, no. 2, pp. 197–207, 2018.

2. Q. K. Nie, Jian Weng, X. B. Xu and B. W. Feng, “Defining embedding distortion for intra prediction mode-based video steganography,” Comput. Mater. Continua, Vol. 55, no. 1, pp. 59–70, 2018.

3. Z. Yan, X. Yu and W. Ding, “Context-aware verifiable cloud computing,” IEEE Access, Vol. 5, pp. 2211–27, 2017.

4. Y. L. Liu, H. Peng and J. Wang, “Verifiable diversity ranking search over encrypted outsourced data,” Comput. Mater. Continua, Vol. 55, pp. 37–57, 2018.

5. C. Gentry, “Fully homomorphic encryption using ideal lattices,” Stoc, Vol. 9, no. 4, pp. 169–78, 2009.

6. L. Kuang, L. Yang, J. Feng and M. Dong, “Secure tensor decomposition using fully homomorphic encryption scheme,” IEEE Trans. Cloud Comput., Vol. PP, no. 99, 1, 2015.

7. D. Cousins, K. Rohloff and D. Sumorok, “Designing an FPGA-accelerated homomorphic encryption co-processor,” IEEE Trans. Emerging Top. Comput., Vol.5, no. 2, pp. 193–206, 2016.

8. H. Chen, Y. Hu, Z. Lian, H. Jia and X. A. Wang, “An additively homomorphic encryption over large message space,” Int. J. Information Technol. Web Eng. (IJITWE), Vol. 10, no. 3, pp. 82–102, 2015.

9. M. Yi, L. Wang and Y. Ma, “Efficient security sequencing problem over insecure channel based on homomorphic encryption,” China Commun., Vol.13, no. 9, pp. 195–202, 2016.

10. M. Yasuda, T. Shimoyama, J. Kogure, K. Yokoyama and T. Koshiba, “New packing method in somewhat homomorphic encryption and its applications,” Secur. Commun. Netw., Vol. 8, no. 13, pp. 2194–213, 2015.

11. D. Grigoriev and I. Ponomarenko, “Homomorphic public-key cryptosystems over groups and rings,” Quaderni di Matematica, Vol. 13, pp. 305–325, 2004.

12. C. Gentry, S. Halevi and N. P. Smart, Homomorphic Evaluation of the AES Circuit. Advances in Cryptology – CRYPTO 2012. Berlin: Springer, 2012, pp. 850–67.

13. S. Halevi and V. Shoup, Algorithms in HElib. Advances in Cryptology – CRYPTO 2014. Berlin: Springer, 2014, pp. 554–71.

14. M. V. Dijk, C. Gentry, S. Halevi and V. Vaikuntanathan, Fully Homomorphic Encryption Over the Integers. Advances in Cryptology – EUROCRYPT 2010. Berlin: Springer, 2010, pp. 24–43.

15. Z. Brakerski and V. Vaikuntanathan, “Fully homomorphic encryption from (standard) LWE,” IEEE Foundations of Computer Science, 2011, pp. 97–106.

16. T. Zhou, X. Yang, W. Zhang and L. Wu, “Efficient fully homomorphic encryption with circularly secure key switching process,” Int. J. High Perform. Comput. Netw., Vol. 9, no. 5/6, pp. 417, 2015.

17. Z. Brakerski, C. Gentry and V. Vaikuntanathan, “(Leveled) fully homomorphic encryption without bootstrapping,” in Proc of the 3rd Innovations in Theoretical Computer Science Conf. New York: ACM, 2012, pp. 309–25.
18. C. Gentry, A. Sahai and B. Waters, *Homomorphic Encryption From Learning with Errors*. Advances in Cryptology – CRYPTO 2013. Berlin: Springer, 2013, pp. 75–92.

19. J. Alperin-Sheriff and C. Peikert, *Practical Bootstrapping in Quasilinear Time*. Advances in Cryptology – CRYPTO 2013. Berlin: Springer, 2013, pp. 1–20.

20. S. Halevi and V. Shoup, “Bootstrapping for HElib,” IACR Cryptology ePrint Archive, 2014. Available: http://eprint.iacr.org/2014/873.

21. C. Gentry, S. Halevi and N. P. Smart, “Better bootstrapping in fully homomorphic encryption,” in International Conference on Practice and Theory in Public Key Cryptography, Springer-Verlag, 2012, pp. 1–16.

22. Z. Brakerski and V. Vaikuntanathan, “Lattice-based FHE as secure as PKE,” in Conference on Innovations in Theoretical Computer Science, 2014, pp. 1–12.

23. D. A. Barrington, “Bounded-width polynomial-size branching programs recognize exactly those languages in NC1,” in ACM Symposium on Theory of Computing, May 28–30, 1986, Berkeley, California, USA, DBLP, 1986, pp. 1–5.

24. J. Alperin-Sheriff and C. Peikert, *Faster Bootstrapping with Polynomial Error*. Advances in Cryptology – CRYPTO 2014. Berlin: Springer, 2014, pp. 297–314.

25. L. Ducas and D. Micciancio, *FHEW: Bootstrapping Homomorphic Encryption in Less than a Second*. Advances in Cryptology – EUROCRYPT 2015. Berlin: Springer, 2015, pp. 617–40.

26. J. F. Biasse and L. Ruiz, *FHEW with Efficient Multibit Bootstrapping*. Progress in Cryptology – LATINCRYPT 2015. Berlin: Springer International Publishing, 2015, pp. 119–35.

27. O. Regev, “Lattice-based cryptography,” in International Cryptology Conference, Springer Berlin Heidelberg, 2006, pp.131–41.

---

**Authors**

**NingBo Li** was born in China, in 1992. He received the BS and MS degrees from Engineering University of People’s Armed Police. Now he is a PhD candidate. His main research interests include fully homomorphic encryption, encryption scheme based on lattice.

**TanPing Zhou** was born in China, in 1989. He received the PhD degree from Engineering University of People’s Armed Police. His main research interests include fully homomorphic encryption, encryption scheme based on lattice.

**YiLiang Han** was born in 1978. PhD and PhD supervisor. His main research interests include information security and cryptology.

**XiaoYuan Yang** was born in 1959. PhD supervisor and master. His main research interests include information security and cryptology.

**YuJuan Sun** received the PhD degree in mathematics from Lanzhou University, Lanzhou, China, in 2010. She is currently with Xidian University, Xi’an, China. Her research interests include cryptography and coding theory.

---

**Corresponding author. E-mail:** 850301775@qq.com

**E-mail:** 372726936@qq.com

**E-mail:** yilianghan@hotmail.com

**E-mail:** sunyj2005@foxmail.com