Extend FHEW to General Case

Zhou Tanping*, Liu Longfei, Yang Xiaoyuan, Han Yiliang
Key Laboratory of Network & Information Security under the Chinese People's Armed Police, Electronic Department, Engineering College of the Armed Police Force, Xi'an, 710086, China

Abstract: When talking about FHE, refresh process is a little different from bootstrapping process. Bootstrapping always means that a scheme homomorphic decrypting its process, while refresh imply that use another scheme, always in large scale, to perform its decryption process. In EUROCRYPT'2015, Ducas and Micciancio proposed a FHE which can perform refresh process in less than a second, called DM14, while the scheme only support bite plaintext space, which is cumbersome for many applications. Extending DM14 to a large plaintext space becomes an open problem. In order to solve it, we improved the msbExtract process to endure a large base, by mapping the element to position. As a result, we constructed an efficient FHE with large plaintext space and quickly refresh process. We implemented our scheme in computer, and made a comparison between our performance and DM14. The result is that the running time is almost same, when extend the plaintext space from 2 to 8.

Key words: public key encryption; fully homomorphic encryption; refresh process; large plaintext space

1 Introduction

Fully homomorphic encryption (FHE) is a secure homomorphic mapping from plaintext space to ciphertext space, which allows us to operate directly on ciphertexts, and the output is the ciphertext correspond to the output of the same operation among the plaintexts. The substantial progress was achieved by Gentry's first FHE scheme based on ideal lattices in STOC'2009[1]. Since then there are many works about basing the security of FHE on more standard and removing the circular security , and improving the efficiency of Gentry's initial solution[2-9]. Among them, the works on the efficiency of FHE are outstanding, since the efficiency is the bottleneck of its practicality. There are two major ways to improvement the efficiency of FHE. One of them is the improvements of the construction in order to apply many optimize technologies, such as batching , modular switching and so on. The second way is to reduce the computation complexity of bootstrapping process, which was developed rapidly after GSW[6]. In 2014, many wonderful results, about how to improvement the performance of bootstrapping process, was presented. In ITCS 2014, Brakerski and Vaikuntanathan reached the major milestone of FHE based on GSW, called BV14 [10]. BV14 decreased the noise of the bootstrapping process to polynomial of the security parameter λ by Barrington's Theorem[11]. In CRYPTO'2014, Alperin and Peikert constructed a scheme with a more efficient bootstrapping, with polynomial error, called AP14[12]. The wonderful contribution of AP14 is that it uses a different big scheme, which is constructed to supports efficient computation of decryption process and efficient computation of transform from big ciphertext back to the ciphertext of origin scheme, to perform the decryption process of the origin scheme. In EUROCRYPT'2015, Ducas and Micciancio succeed to the framework of AP14, and constructed a more efficient FHE, called DM14[13]. DM14 can perform bootstrapping homomorphic encryption in less than a second, by constructing the efficiency $\mathrm{msbExtract}$ process, which can extract the msb (most significant bit) of the plaintext, giving the corresponding big ciphertext. While there is a drawback about this scheme that the $\mathrm{msbExtract}$ process can extract the msb of the plaintext, only when the base of plaintext is 2. It is an open problem proposed on DM14 that extend the $\mathrm{msbExtract}$ process to large plaintext spaces, which may improve the performance of FHE to a high level by combine other optimization technique, such as batching.

We solved the problem of improving the DM14 to large plaintext spaces, by generalizing the $\mathrm{msbExtract}$

This work was supported by the National Natural Science Foundation of China (Grant No. 61272492. 61572521), Natural Science Basic Research Plan In Shaanxi Province Of China (Program No.2015JM6353) and the Program for New Century Excellent Talents in University(NCET-12-0620).

procedure , which can only extract the most significant bit of the plaintext, when the base of plaintext is 2, to $\mathbb{Z}_{t_2}$, which can only extract the most significant bit of the plaintext, when the base of plaintext is t_2 . The new idea in our scheme is that we constructed a map from the plaintext $m \in \mathbb{Z}_q$ to the set of big ciphertext, a cyclic group, where each m corresponding to a unique position. Thus, we can get some of the special functions of the message by just operate on the corresponding position, which will be more efficient. What's more, we implemented our scheme and the source code is reasonably concise and simple, consisting of about 700 lines of C++ code.

2 Preliminaries

For a list of vectors and matrices $\boldsymbol{a}_1, \boldsymbol{a}_2, ..., \boldsymbol{a}_l$, we set $(...) = \begin{bmatrix} \boldsymbol{a}_1 \\ \vdots \\ \boldsymbol{a}_l \end{bmatrix}$ for vertical concatenation, and $[...] = \begin{bmatrix} \boldsymbol{a}_1 & ... & \boldsymbol{a}_1 \end{bmatrix}$ for horizontal concatenation.

2.1 Distributions[13]

We define a randomized rounding function $\chi : \mathbb{R} \to \mathbb{Z}$, which can be regarded as the round off function combine the integer errors. In detail $\chi(x) = \lfloor x \rceil + e_0$, where $e_0 \in \mathbb{Z}$. Similarly, we define $\chi(x) - x$ as the rounding error of $\chi(x)$.

We defined the subgaussian variable as the random variable X over $\mathbb{R}$, where the (scaled) moment-generating function satisfies $E[\exp(2\pi t X)] \le \exp(\pi \alpha^2 t^2)$, for all $t \in \mathbb{R}$. If X is subgaussian, then it satisfies that $\Pr\{|X| \ge t\} \le 2\exp(-\pi t^2 / \alpha^2)$ for all $t \ge 0$. We extend the nation of subgaussian to vector $\boldsymbol{x}$ and matrix $\boldsymbol{X}$, if $\boldsymbol{x}$ satisfies that $< \boldsymbol{u}, \boldsymbol{x} >$ are subgaussian for all unit vectors $\boldsymbol{u}$, and if $\boldsymbol{X}$ satisfies that $\boldsymbol{u}^t \boldsymbol{X} \boldsymbol{v}$ are subgaussian for all unit vectors $\boldsymbol{u}, \boldsymbol{v}$. Notice that the concatenation of independent subgaussian vectors or matrixes with common parameter α is subgaussian with parameter α .

2.2 The Cyclotomic Ring[13]

In order to simple the FFT process and operations on element, we set the cyclotomic ring as $\mathcal{R} = \mathbb{Z}[X]/(X^N + 1)$, where $X^N + 1$ is the the $(2N)$ -th cyclotomic polynomial $\Phi_{2N}(X) = \prod_{i \in \mathbb{Z}_{2N}^*} (X - \omega_{2N}^i) = X^N + 1$, and N is a power of 2. The $\mathcal{R}_Q = \mathcal{R}/(Q\mathcal{R})$, where Q is a integer.

By the definition, we know that $\mathcal{R}_Q \simeq \mathbb{Z}_Q^N$. The isomorphic mapping can be defined as $f : a \to \vec{a}$, where $a = a_0 + a_1 x + ... + a_{N-1} x^{N-1} \in \mathcal{R}, \vec{a} = (a_0, a_1, ..., a_{N-1}) \in \mathbb{Z}^N$. We extend the notation $\vec{\cdot}$ to vector and matrix over $\mathcal{R}$. We define the Euclidean length of a ring element as $\| a \| = \| \vec{a} \| = \sqrt{\sum_i | a_i |^2}$, and the spectral norm of a matrix $\boldsymbol{R} \in \mathcal{R}^{w \times k}$ of ring elements as $s_1(\boldsymbol{R}) = \sup_{\boldsymbol{x} \in \mathcal{R}^k \setminus \{0\}} \| \boldsymbol{R} \cdot \boldsymbol{x} \| / \| \boldsymbol{x} \|$.

We denote $\overset{\Rightarrow}{r} = \begin{bmatrix} r_0 & r_{N-1} & r_1 \\ r_1 & r_0 & r_2 \\ \vdots & \vdots & \vdots \\ r_{N-1} & r_{N-2} & r_0 \end{bmatrix} \in \mathbb{Z}^{N \times N}$, giving a ring element $r \in \mathcal{R}$. We extend the notation $\overset{\Rightarrow}{\cdot}$ to vectors and matrices over $\mathcal{R}$. When giving a matrix $\boldsymbol{R} \in \mathcal{R}^{w \times k}$, we expand it to a big matrix $\overset{\Rightarrow}{\boldsymbol{R}} \in \mathcal{R}^{Nw \times Nk}$, with $w \times k$ blocks, each of the block is $\overset{\Rightarrow}{\boldsymbol{R}}_{i,j} \in \mathcal{R}^{N \times N}$. Notice that we have $s_1(r) = s_1(\overset{\Rightarrow}{r})$ and $s_1(\boldsymbol{R}) = s_1(\overset{\Rightarrow}{\boldsymbol{R}})$[13].

What's more, we proposed two special functions to make our scheme easy to understand. For two integers l, k, we let $\overset{l,k}{(-1)}$ be the function that $\overset{l,k}{(-1)} = \begin{cases} -1 & l < k \\ 1 & l \geq k \end{cases}$; For three integers x, y, v, we let $(1)_{(x,y)}^v$ be the function that $(1)_{(x,y)}^v = \begin{cases} 0 & v \notin (x,y) \\ 1 & v \in (x,y) \end{cases}$;

3 The Small LWE Symmetric Encryption

In this section, we will introduce the small LWE Symmetric Encryption and two basic operations, Modulus switching and Key Switching, which was first presented in BGV.

3.1 the small LWE Symmetric Encryption[13]

We recall the definition of the most basic LWE symmetric encryption scheme (see [4,32,3]). LWE symmetric encryption is parameterized by a dimension n, a message-modulus $t \geq 2$, a ciphertext modulus $q = n^{O(1)}$ and a randomized rounding function $\chi : \mathbb{R} \to \mathbb{Z}$. The message space of the scheme is $\mathbb{Z}_t$. (Typically, the rounding function has error distribution $| \chi(x) - x | < \frac{q}{2t}$.) The key of the encryption scheme is a vector $\boldsymbol{s} \in \mathbb{Z}_q^n$, which may be chosen uniformly at random, or as a random short vector. The encryption of a message $m \in \mathbb{Z}_t$ under key $\boldsymbol{s} \in \mathbb{Z}_q^n$ is

$$\mathrm{LWE}_{s}^{t/q}(m) = (\boldsymbol{a}, \chi(\boldsymbol{a}\cdot\boldsymbol{s} + m\frac{q}{t}) \bmod q) \in \mathbb{Z}_q^{n+1}$$

where $\boldsymbol{a} \leftarrow \mathbb{Z}_q^n$ is chosen uniformly at random. Notice that when t divides q, the encryption of m equals $(\boldsymbol{a}, \boldsymbol{a}\cdot\boldsymbol{s} + m\frac{q}{t} + e \bmod q)$, where the error e is chosen according to a fixed noise distribution $\chi(0)$. A ciphertext $(\boldsymbol{a}, b) \in \mathbb{Z}_q^{n+1}$ is decrypted by computing

$$m' = \lfloor t(b - \boldsymbol{a}\cdot\boldsymbol{s}) / q \rceil \bmod t \in \mathbb{Z}_t .$$

We write $\mathrm{LWE}_{\boldsymbol{s}}^{t/q}(m, B)$ to denote the set of all possible encryptions of m under $\boldsymbol{s}$ with error bounded by $|\operatorname{err}(\boldsymbol{a}, b)| < B$, where $\operatorname{err}(\boldsymbol{a}, b) = (b - \boldsymbol{a}\cdot\boldsymbol{s} - m\frac{q}{t}) \bmod q \in \mathbb{Z}_q$ describing the rounding error, reduced modulo q to the centered interval $[-\frac{q}{2}, \frac{q}{2}]$. It is easy to check that for all $(\boldsymbol{a}, b) \in \mathrm{LWE}_{s}^{t/q}(m, \frac{q}{2t})$, the decryption procedure correctly recovers the encrypted message.

$$\lfloor t(b - \boldsymbol{a}\cdot\boldsymbol{s}) / q \rceil \bmod t = \lfloor \frac{t}{q}\cdot(\frac{q}{t}m + e) \rceil = \lfloor m + \frac{t}{q}e \rceil = m \bmod t$$

Modulus switching. LWE ciphertexts can be converted from one modulus Q to another q using the (scaled) randomized rounding function $[\cdot]_{Q:q}; \mathbb{Z}_Q \to \mathbb{Z}_q$ defined as $[x]_{Q:q} = \lfloor \frac{q}{Q} x \rceil$, where $\lfloor\cdot\rceil$ is the round off function. In particular, the rounding error $[x]_{Q:q} - \frac{q}{Q}x$ is subgaussian of parameter $\sqrt{2\pi}$. Given a ciphertext $(\boldsymbol{a}, b) \in \mathrm{LWE}_{\boldsymbol{z}}^{t/Q}(m)$ with modular Q, and a new molular q, the key modulus procedure outputs

$\mathrm{ModSwitch}(\boldsymbol{a}, b) = [\boldsymbol{a}, b]_{Q:q} = ([a_1]_{Q:q}, ..., [a_n]_{Q:q}, [b]_{Q:q})$.

Lemma 1[13] For any $\boldsymbol{s} \in \mathbb{Z}_q^n$, $m \in \mathbb{Z}_t$ and ciphertext $\mathrm{LWE}_{s}^{t/Q}(m)$ with subgaussian error of parameter σ, the rounding $\mathrm{ModSwitch}(\boldsymbol{c})$ is a $\mathrm{LWE}_{s}^{t/q}(m)$ ciphertext with subgaussian error of parameter $\sqrt{(\frac{q}{Q}\sigma)^2 + 2\pi(\|\boldsymbol{s}\|^2 + 1)}$.

Key Switching. Key switch is designed to convert an LWE encryption $\boldsymbol{c} = (\boldsymbol{a}, b) \in \mathrm{LWE}_{\boldsymbol{z}}^{t/q}(m)$ to another LWE encryption $\boldsymbol{c}' \in \mathrm{LWE}_{s}^{t/q}(m)$. The process is similar to bootstrapping, since we need to homomorphic operate

$b-\boldsymbol{a}\cdot\boldsymbol{s}$, which is a part of decryption process $\lfloor t(b-\boldsymbol{a}\cdot\boldsymbol{s})/q\rceil \bmod t$. We have $\mathrm{enc}_{\boldsymbol{z}}(b-\boldsymbol{a}\cdot\boldsymbol{s})=b-\boldsymbol{a}\cdot\mathrm{enc}_{\boldsymbol{z}}(\boldsymbol{s})=\mathrm{enc}_{\boldsymbol{z}}(\frac{q}{t}m+e)$, because of the property of LWE encryption. The $\boldsymbol{s}$ is encrypted as $\mathfrak{K}=\{\boldsymbol{k}_{i,j,w}\}$, where $\boldsymbol{k}_{i,j,v}\in\mathrm{LWE}_{\boldsymbol{s}}^{q/q}(w\cdot z_i B_{ks}^j)$, for a base B_{ks} and all $i=1,\dots,N, v\in\{0,\dots,B_{ks}\}$ and $j=0,\dots,d_{ks}-1$, where $d_{ks}=\lceil\log_{B_{ks}} q\rceil$. The reason of why it is encrypted in this way is similar to the reason about $E(\boldsymbol{s})$ analyzed on section 4.

Given the switching key $\mathfrak{K}=\{\boldsymbol{k}_{i,j,w}\}$ and a ciphertext $(\boldsymbol{a},b)\in\mathrm{LWE}_{\boldsymbol{z}}^{t/q}(m)$. Firstly, the key switching procedure computes the base-B_{ks} expansion of each coefficient $a_i=\sum_{j=0}^{d_{ks}-1}a_{i,j}B_{ks}^j$. Secondly, outputs

$\mathrm{KeySwitch}((\boldsymbol{a},b),\mathfrak{K})=(0,b)-\sum_{i=1}^{N}\sum_{j=0}^{d_{ks}-1}\boldsymbol{k}_{i,j,a_{i,j}}$.

Lemma 2[13] Given a ciphertext $\boldsymbol{c}\in\mathrm{LWE}_{\boldsymbol{z}}^{t/q}(m)$ with subgaussian error of parameter α, and switching keys $\boldsymbol{k}_{i,j,v}\in\mathrm{LWE}_{\boldsymbol{s}}^{q/q}(v\cdot z_i B_{ks}^j)$ with subgaussian error of parameter σ. The key switching procedure outputs an encryption $\mathrm{KeySwitch}(\boldsymbol{c},\{\boldsymbol{k}_{i,j,v}\})$ with subgaussian error of parameter $\sqrt{\alpha^2+Nd_{ks}\sigma^2}$.

4 Refreshing

In this section, we will description of the refreshing function, which is used to get an LWE ciphertext $\mathrm{LWE}_{\boldsymbol{s}}^{t/q}(m)$ with smaller noise, giving an LWE valid ciphertext $(\boldsymbol{a},b)\in\mathrm{LWE}_{\boldsymbol{s}}^{t_2/q}(m)$ with a big noise. It may looks like bootstrapping to some reader, but refresh process may be much fast then it, since in the bootstrapping we always require use the original LWE ciphertext to homomorphic decrypt itself. In the refresh process, an intermediate encryption scheme (big scheme) E with message space $\mathbb{Z}_q$ was always been used. The big scheme E is designed to homomorphic decrypt the original LWE ciphertext (small ciphertext.)

In this paper, the LWE decryption procedure for an given ciphertext $(\boldsymbol{a},b)\in\mathrm{LWE}_{\boldsymbol{s}}^{t_2/q}(m,\frac{q}{2t_2})$, where

$b-\boldsymbol{a}\cdot\boldsymbol{s}=m\frac{q}{t_2}+e+kq$ for some e and k, is $\lfloor t_2(b-\boldsymbol{a}\cdot\boldsymbol{s})/q\rceil \bmod t_2$. Notice that

$$\lfloor t_2(b - \boldsymbol{a}\cdot\boldsymbol{s})/q \rceil \bmod t_2 = \lfloor t_2(m\frac{q}{t_2} + e + kq)/q \rceil \bmod t_2 \xrightarrow{e<q/2t_2} (m + t_2 k) \bmod t_2 = m$$

$$= msb(\frac{q}{2t_2} + m\frac{q}{t_2} + e) = msb(\frac{q}{2t_2} + m\frac{q}{t_2} + e + kq) = msb(\frac{q}{2t_2} + b - \boldsymbol{a}\cdot\boldsymbol{s}).$$

It is clear that an efficient refresh process was given, if we can design an efficient msb process for some special big scheme E. We will discuss the detail of the special big scheme in section 5, which can efficient extract the msb of big ciphertext C. It will be more efficient when the plaintext of E is $\mathbb{Z}_q$, since the homomorphic operations are on elements belongs to $\mathbb{Z}_q$.

Here we focus on other part of the homomorphic $msb(\frac{q}{2t_2} + b - \boldsymbol{a}\cdot\boldsymbol{s})$, which contains homomorphic addition and homomorphic multiplication. We can transform homomorphic multiplication to homomorphic addition by the way that we present all the possible result of $\boldsymbol{k}\cdot\boldsymbol{s}$ for all $\boldsymbol{k}\in\mathbb{Z}_q^n$ and an fixed $\boldsymbol{s}\in\mathbb{Z}_q$, when the small ciphertext $(\boldsymbol{a}, b)$ was given, we will pick up $\boldsymbol{a}\cdot\boldsymbol{s}$ from $\boldsymbol{k}\cdot\boldsymbol{s}$. In fact, in order to control the error, we will encrypt $\boldsymbol{k}\cdot\boldsymbol{s}$ in the way $E(\boldsymbol{k}\cdot\boldsymbol{s}) = E(\sum_{i=1}^{n} k_i \cdot s_i) = E(\sum_{i=1}^{n} (\sum_{j=0}^{d_r-1} B_r^j k_{i,j}) \cdot s_i) = E(\sum_{i=1}^{n} \sum_{j=0}^{d_r-1} k_{i,j}(B_r^j s_i))$, where B_r is an integer and $d_r = \lceil \log_{B_r} q \rceil$, $k_{i,j} \in \{0,\ldots,B_r - 1\}$. Then, we define a refreshing key $\mathcal{K} = \{K_{i,c,j} = E(c\cdot \boldsymbol{s}_i B_r^j)\}_{i,c,j}$ for $c \in \{0,1,\ldots,B_r - 1\}$, $j = 0,\ldots,d_r - 1$ (where $d_r = \lceil \log_{B_r} q \rceil$) and $i = 1,\ldots,n$ ($\mathcal{K} = \{K_{i,c,j} = E(c\cdot \boldsymbol{s}_i B_r^j)\}_{i,c,j}$ is different from switch key $\mathfrak{K} = \{\boldsymbol{k}_{i,j,w} \mathrm{LWE}_{\boldsymbol{s}}^{q/q}(w\cdot z_i B_{ks}^j)\}$.)

Notice that there is only two types of operation left, many times homomorphic additions for elements which belongs to $\mathbb{Z}_q$ and one time homomorphic msb for big ciphertext.

Definition 1 (Homomorphic Accumulator). A Homomorphic Accumulator Scheme is a quadruple of algorithms $(E, \mathrm{Init}, \mathrm{Incr}, \mathrm{msbExtract})$ together with moduli t, q, where E and $\mathrm{msbExtract}$ may require key material related to an LWE key $\boldsymbol{s}$. For brevity, we write $\mathrm{ACC} \leftarrow v$ for $\mathrm{ACC} \leftarrow \mathrm{Init}(v)$, and $\mathrm{ACC} \xleftarrow{+} E(v)$ for $\mathrm{ACC} \leftarrow \mathrm{Incr}(\mathrm{ACC}, E(v))$. For any $v_0, v_1, \ldots, v_l \in \mathbb{Z}_q$, after the sequence of operations

$$\mathrm{ACC} \leftarrow v_0; \text{ for } i = 1 \text{ to } l \text{ do } \mathrm{ACC} \xleftarrow{+} E(v_i)$$

We say that ACC is an l-encryption of v, where $v = \sum v_i \bmod q$.

A Homomorphic Accumulator Scheme is said ε-correct for some function ε if, for any l-encryption

ACC of v, computing $\boldsymbol{c} \leftarrow \mathrm{msbExtract}(\mathrm{ACC})$ ensures $\boldsymbol{c} \leftarrow \mathrm{LWE}_{\boldsymbol{s}}^{t/q}(\mathrm{msb}(v), \varepsilon(l))$ with overwhelming probability.
It then proceeds as described in Algorithm 1

Algorithm 1 $\mathrm{Refresh}_{\mathcal{K}}(\boldsymbol{a}, b)$ for $\mathcal{K} = \{K_{i,c,j} = E(c \cdot \boldsymbol{s}_i B_r^j)\}_{i \le n, c \le B_r, j \le d_r}$

```
$\mathrm{ACC} \leftarrow b + \frac{q}{2t_2}$
for $i = 1, \ldots, n$ do
    Compute the base-$B_r$ representation of $-a_i = \sum_j B_r^j \cdot a_{i,j} (\bmod q)$
    for $j = 0, \ldots, d_r - 1$ do $\mathrm{ACC} \xleftarrow{+} K_{i, a_{i,j}, j}$
end for
Output $\mathrm{msbExtract}(\mathrm{ACC})$
```

Theorem 8 If $(E, \mathrm{Init}, \mathrm{Incr}, \mathrm{msbExtract})$ is a correct Homomorphic Accumulator Scheme, then the Refresh procedure, on input any ciphertext $\boldsymbol{c} \in \mathrm{LWE}_{\boldsymbol{s}}^{t_2/q}(m, \frac{q}{2t_2})$ and a valid refreshing key $\mathcal{K} = \{K_{i,c,j} = E(c \cdot \mathbf{s}_i B_r^j)\}_{i,c,j}$, outputs a ciphertext $\mathrm{Refresh}_{\mathcal{K}}(\boldsymbol{c}) \leftarrow \mathrm{LWE}_{\boldsymbol{s}}^{t/q}(\mathrm{msb}(v), \varepsilon(nd))$.

5 The Construction Of Big Scheme

In this paper, three types of scheme and corresponding ciphertexts are involved. We will call the origin LWE scheme $\mathrm{LWE}_{\boldsymbol{s}}^{t_2/q}$ as small scheme, the scheme, which is used to operation the decryption of small scheme, as big scheme, and the output LWE scheme $\mathrm{LWE}_{\boldsymbol{s}}^{t/q}$ of $\mathrm{msbExtract}$ as the final output scheme. In the same way, we call the ciphertext of small scheme as small ciphertext, the ciphertext of big scheme as big ciphertext, the ciphertext of final output scheme as final output ciphertext.

Our construction of big scheme follows the DM14. Essentially, we present our more powerful $\mathrm{msbExtract}$. Compared with the $\mathrm{msbExtract}$ proposed by DM14, ours can extract the most significant bit of the plaintext, when the base of plaintext is t_2, the DM14 constrains the base of plaintext to 2. What's more, we proposed some functions to make our $\mathrm{msbExtract}$ more efficient and easy to understand.

5.1 Description of the construction of big scheme

The big scheme is parameterized by a modulus Q, a plaintext space $\mathbb{Z}_q$, where q is the modular of small LWE encryption system, a base B_g to decompose the ciphertext, and a dimension $N = 2^K$. In order to successful decryption, we require that $q \mid t_2(N + d_{tr})$ and $d_{tr} \cdot (q-1) < N$, for some integer $d_{tr} \neq 0$. We use the rings $\mathcal{R} = \mathbb{Z}[X]/(X^N + 1)$ and $\mathcal{R}_Q = (\mathcal{R}/Q\mathcal{R})$, and an additional parameter u, which should be an invertible element of $\mathbb{Z}_Q$ close to Q/t, where t is the plaintext of the output $\mathrm{LWE}_{\boldsymbol{s}}^{t/q}$ encryption of $\mathrm{msbExtract}$ process. For the reason of the successful decryption, we require that $\delta = u - Q/t$ is at most $|\delta| < 1$, for the reason of the successful decryption. In order to make the finally output LWE ciphertext to express the original message $m \in \mathbb{Z}_{t_2}$, we require $t_2 < t$. Messages $m \in \mathbb{Z}_q$ are encoded as q-th roots of unity $Y^m \in \mathcal{R}$ where $Y = X^{\frac{t_2(N+d_{tr})}{q}}$. Notice that the $2N$-th roots of unity $\mathcal{G} = < X > = \{1, X, ..., X^{N-1}, X^N = -1, X^{N+1} = -X, ..., X^{2N-1} = -X^{N-1}\}$ form a cyclic group, and the message space $< Y >$ is a subgroup of $G \simeq \mathbb{Z}_{2N}$, we will analysis the relationship between $< Y >$ and G in 5.2.

Our big scheme contains five parts, $E_z(m)$, $\mathrm{Init}(\mathrm{ACC} \leftarrow v)$, $\mathrm{Incr}(\mathrm{ACC} \xleftarrow{+} \boldsymbol{C})$, $\mathrm{uMultcontant}(\boldsymbol{C} \leftarrow k)$, $\mathrm{msbExtract}$. The last two $\mathrm{uMultcontant}(\boldsymbol{C} \leftarrow k)$, $\mathrm{msbExtract}$ parts are different from DM14.

$-E_z(m)$, giving a message m and a key $z \in \mathcal{R}$, picks $\boldsymbol{a} \xleftarrow{R} \mathcal{R}_Q^{2d_g}$ uniformly at random, and $\boldsymbol{e} \in \mathcal{R}^{2d_g} \simeq \mathbb{Z}^{2d_g N}$ with a subgaussian distribution χ of parameter ς, and outputs

$$E_z(m) = [\boldsymbol{a}, \boldsymbol{a} \cdot z + \boldsymbol{e}] + uY^m \boldsymbol{G} \in \mathcal{R}_Q^{2d_g \times 2}$$

where $\boldsymbol{G} = (\boldsymbol{I}, B_g \boldsymbol{I}, ..., B_g^{d_g - 1} \boldsymbol{I}) \in \mathcal{R}_Q^{2d_g \times 2}$, $\boldsymbol{I} \in \mathcal{R}_Q^{2\times 2}$ is the identity matrix.

- $\mathrm{Init}(\mathrm{ACC} \leftarrow v)$, giving $v \in \mathbb{Z}_q$, and sets $ACC := uY^v \cdot \boldsymbol{G} \in \mathcal{R}_Q^{2d_g \times 2}$.

- $\mathrm{Incr}(\mathrm{ACC} \xleftarrow{+} \boldsymbol{C})$, giving the current accumulator content $ACC \in \mathcal{R}_Q^{2d_g \times 2}$ and a ciphertext $\boldsymbol{C} \in \mathcal{R}_Q^{2d_g \times 2}$, first computes the base-B_g decomposition of $u^{-1} ACC = \sum_{i=1}^{d_g} B_g^{i-1} \boldsymbol{D}_i \in \mathcal{R}^{2d_g \times 2}$ (where each $\boldsymbol{D}_i \in \mathcal{R}_Q^{2d_g \times 2}$ has entries with coefficients in $\{\frac{1-B_g}{2}, ..., \frac{B_g - 1}{2}\}$, and then updates the accumulator to $ACC = [\mathrm{D}_1, ..., \mathrm{D}_{d_g}] \cdot \boldsymbol{C} \in \mathcal{R}^{2d_g \times 2}$.

-uMultcontant$(\boldsymbol{C} \leftarrow k)$ giving a constant $k \in \mathbb{Z}_q$ and a ciphertext $\boldsymbol{C} \in \mathcal{R}_Q^{2d_g \times 2}$ with the parameter $u \in \mathbb{Z}_q$, output a ciphertext $\boldsymbol{C}_{ku} \in \mathcal{R}_Q^{2d_g \times 2}$ with the parameter $ku \in \mathbb{Z}_q$.

First computes the base- B_g decomposition $contant(k) = uk \cdot \boldsymbol{G}$, then set $\boldsymbol{C}_{ku} \leftarrow \mathrm{Incr}(\boldsymbol{C} \xleftarrow{+} contant(k))$. For brevity, we sometimes write $\boldsymbol{C}_{(k)}$ for $\boldsymbol{C}_{ku} \in \mathcal{R}_Q^{2d_g \times 2}$.

- msbExtract giving a key-switching auxiliary input $\mathfrak{K}$ (as defined in Section 2) and a set of vectors $T = \{\boldsymbol{t}_{(1)}, \boldsymbol{t}_{(2)}, ..., \boldsymbol{t}_{(k)}\}$,where

$\boldsymbol{t}_{(k)} = \sum_{i=(k-1)q/t_2}^{i=kq/t_2 - 1} \overrightarrow{Y^i}$, $\mathcal{R} = \mathbb{Z}[\mathrm{X}] / (\mathrm{X}^N - 1)$, $Y = X^{\frac{t_2(N+d_{tr})}{q}}$, $t_2(N + d_{tr}) = o \cdot q$, $d_{tr} \cdot (q-1) < N$, $t_2 \mid q$. The main idea of the extraction of msb is that $msb_{(k)} = \boldsymbol{t}_{(k)} \cdot \overrightarrow{Y^v} = 1$, if $\frac{(k-1)q}{t_2} \le v < \frac{kq}{t_2}$, otherwise $msb_{(k)} = \boldsymbol{t}_{(k)} \cdot \overrightarrow{Y^v} = 0$.(i.e. $msb_{(k)} = \boldsymbol{t}_{(k)} \cdot \overrightarrow{Y^v} = (1)^v_{(\frac{(k-1)q}{t_2}, \frac{kq}{t_2})}$) Notice that, $msb(v) = 0 \cdot msb(v)_{(1)} + 1 \cdot msb(v)_{(2)} + ... + (t_2 - 1) \cdot msb(v)_{(t_2)}$. For noise management, we also note that $\| \boldsymbol{t} \|^2 \le \frac{q}{t_2}$. The detail follows.

Algorithm 2 - msbExtract$_{\mathfrak{K}}$(ACC) for $\mathfrak{K} = \{k_{i,j,w}\}_{i \le N, j \le B_{ks}, w \le d_{ks}}$

Require: a switching key $\mathfrak{K} = \{k_{i,j,w}\}_{i \le N, j \le B_{ks}, w \le d_{ks}}$ from $\boldsymbol{z}$ to $\boldsymbol{s}$, where $k_{i,j,w} \in LWE_{\boldsymbol{s}}^{q/q}(w \cdot z_i B_{ks}^j)$. An accumulator ACC that is an l-encryption of v.

```
1: $\mathrm{ACC}_{(k)} = \mathrm{uMultcontant}(\mathrm{ACC} \leftarrow k-1)$, for $1 \le k \le t_2$      $\in \mathrm{LWE}_{\vec{z}}^{t/Q}$
2: $[(\boldsymbol{a}_{(k)})^t, (\boldsymbol{b}_{(k)})^t] \leftarrow [\vec{0}^t, (\boldsymbol{t}_{(k)})^t, \vec{0}^t, ..., \vec{0}^t] \cdot \overset{\Rightarrow}{\mathrm{ACC}}_{(k)}$, for $1 \le k \le t_2$      // $\overset{\Rightarrow}{\mathrm{ACC}}_{(k)} \in \mathbb{Z}^{2d_g N \times 2N}$
3: $\boldsymbol{c}_{(k)} \leftarrow (\boldsymbol{a}_{(k)}, b_{(k)0})$      $\in \mathrm{LWE}_{\vec{z}}^{t/Q}(\mathrm{msb}(v)_{(k)})$
4: $\boldsymbol{c} = \sum_{k=1}^{k=t_2} \boldsymbol{c}_{(k)}$      $\in \mathrm{LWE}_{\vec{z}}^{t/Q}(\mathrm{msb}(v))$
5: $\boldsymbol{c}' \leftarrow \mathrm{KeySwitch}(\boldsymbol{c}, \mathfrak{K})$      $\in \mathrm{LWE}_{\boldsymbol{s}}^{t/Q}(\mathrm{msb}(v))$
```

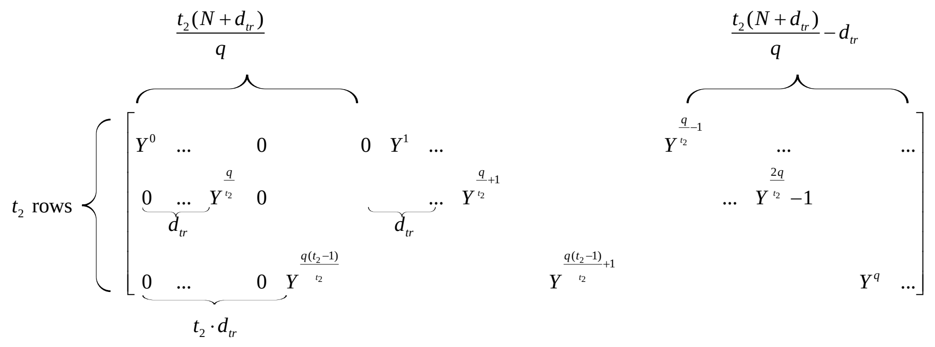
```
6: $\boldsymbol{c}'' \leftarrow \mathrm{ModSwitch}(\boldsymbol{c}')$        $\in \mathrm{LWE}_s^{t/q}(\mathrm{msb}(v))$
7: Return $\boldsymbol{c}''$
```

5.2 Correctness

In this subsection we prove that ACC is a correct Homomorphic Accumulator Scheme for an appropriate error function ε . The main result is given in Theorem 10. But first, let us detail the behavior of individual operations.

lemma 3. If $q \mid t_2(N+d_{tr})$ and $d_{tr}\cdot(q-1)<N$, for some integer $d_{tr}\neq 0$, $Y^m=\left(X^{\frac{t_2(N+d_{tr})}{q}}\right)^m \in \mathcal{R}$ where $m\in\mathbb{Z}_q$ and $\mathcal{G}=<X>=\{1,X,...,X^{N-1},X^N=-1,X^{N+1}=-X,...,X^{2N-1}=-X^{N-1}\}$, then the message space $<Y>$ is a subgroup of G .

Proof. Notice that $Y^m=\left(X^{\frac{t_2(N+d_{tr})}{q}}\right)^m = X^{\frac{t_2(N+d_{tr})m}{q}} = (-1)^{\lfloor\frac{t_2m}{q}\rfloor\cdot N}\cdot X^{\frac{t_2(N+d_{tr})m}{q}-\lfloor\frac{t_2m}{q}\rfloor\cdot N}$. And $\frac{t_2(N+d_{tr})m_1}{q}-\lfloor\frac{t_2m_1}{q}\rfloor\cdot N$ is different from $\frac{t_2(N+d_{tr})m_2}{q}-\lfloor\frac{t_2m_2}{q}\rfloor\cdot N$ when $m_1\neq m_2$. It's easy to understand by this following big matrix, most of the elements 0 are not writhed in the matrix.



$\frac{t_2(N+d_{tr})}{q}$
$\frac{t_2(N+d_{tr})}{q}-d_{tr}$
t_2 rows
Y^0 ... 0 0 Y^1 ... $Y^{\frac{q}{t_2}-1}$ ... ...
0 ... $Y^{\frac{q}{t_2}}$ 0 ... $Y^{\frac{q}{t_2}+1}$ ... $Y^{\frac{2q}{t_2}-1}$
d_{tr}
d_{tr}
0 ... 0 $Y^{\frac{q(t_2-1)}{t_2}}$ $Y^{\frac{q(t_2-1)}{t_2}+1}$ Y^q ...
$t_2\cdot d_{tr}$

We know that the message space $<Y>$ has q different element in $\mathcal{R}=\mathbb{Z}[X]/(X^N+1)$, with the form X^i, which induce that the message space $<Y>$ is a subgroup of G . What's more, we can find that each $m\in\mathbb{Z}_q$ corresponds to only one element in $\mathcal{G}$. The main idea in our scheme is that we constructed a map from the plaintext $m\in\mathbb{Z}_q$ to the cyclic group $\mathcal{G}$. If we maps X^i to a position $i \bmod N$ in $\mathbb{Z}_N$, and the element in the position is defined as the coefficient of $X^i \bmod (X^N+1)$. Then, each $m\rightarrow Y^m$ corresponding to a unique position in $\mathbb{Z}_N$, where the element of this position is 1 or -1. What's more, if the msb of m is i under the base t_2 , then the position corresponding to m belongs to the i -th row of the upper matrix. At last, we can find out the msb of m under the base t_2, by check if the position of m is belongs to the row 1 or row 2 or … or row t_2.

Fact 4 For any messages $m, m' \in \mathbb{Z}_q$, if $\mathrm{ACC} = [\boldsymbol{a}, \boldsymbol{a} \cdot z + \boldsymbol{e}] + uY^m \boldsymbol{G}$ and $\boldsymbol{C} = [\boldsymbol{a}', \boldsymbol{a} \cdot z + \boldsymbol{e}'] + uY^{m'} \boldsymbol{G}$, then $\mathrm{ACC} \xleftarrow{+} \boldsymbol{C}$ has the form $\mathrm{ACC} = [\boldsymbol{a}'', \boldsymbol{a}'' \cdot z + \boldsymbol{e}''] + uY^{m+m'} \boldsymbol{G}$ for $\boldsymbol{e}'' = Y^{m'} \cdot \boldsymbol{e} + [\mathrm{D}_1, ..., \mathrm{D}_{d_g}] \cdot \boldsymbol{e}'$.

lemma 5 For any messages $m \in \mathbb{Z}_q$ and constant $k \in \mathbb{Z}_q$, if $\boldsymbol{C} = [\boldsymbol{a}, \boldsymbol{a} \cdot z + \boldsymbol{e}] + uY^m \boldsymbol{G}$, then $\mathrm{uMultcontant}(\boldsymbol{C} \leftarrow k)$ has the form $\boldsymbol{C}_{ku} = [\boldsymbol{a}''', \boldsymbol{a}''' \cdot z + \boldsymbol{e}]$ for, where $\boldsymbol{e}$ is the error corresponding to $\boldsymbol{C}$.

Proof. Notice that $\boldsymbol{C}_{ku} \leftarrow \mathrm{Incr}(\boldsymbol{C} \xleftarrow{+} \mathit{contant}(k))$, then we can apply the result of Fact 9. We get $\boldsymbol{C}_{ku}$'s error $\boldsymbol{e}'' = Y^{m'} \cdot \boldsymbol{e} + [\boldsymbol{D}_1, ..., \boldsymbol{D}_{d_g}] \cdot \boldsymbol{e}'$. since $\mathit{contant}(k)$ can be regarded as a ciphertext $\boldsymbol{C}'$, where $Y^{m'} = 1$ and $\boldsymbol{e}' = 0$. Substituting them to $\boldsymbol{e}'' = Y^{m'} \cdot \boldsymbol{e} + [\boldsymbol{D}_1, ..., \boldsymbol{D}_{d_g}] \cdot \boldsymbol{e}'$, we can get $\boldsymbol{e}'' = \boldsymbol{e}$.

The last operation $\mathrm{msbExtract}$ is slightly more intricate. Let us put aside the key and modulus switching steps, and consider the former 4 steps.

First of all, we will prove that the output of $\mathrm{msbExtract}$ has the right form of LWE scheme. In the first step, lemma 3 tells us, the $\mathrm{uMultcontant}(\mathrm{ACC} \leftarrow k-1)$ output a valid big ciphertext. In the second and third step, we let $[a_{(k)}, b'_{(k)}] \in \mathcal{R}^{1\times 2}$ be the second row of the accumulator $\mathrm{ACC}_{(k)} \in \mathcal{R}^{2d_g \times 2}$. If $\mathrm{ACC}_{(k)}$ is a GSW encryption of a value v with $u(k-1)$, then $[a_{(k)}, b'_{(k)}]$ will satisfies $\overrightarrow{b'_{(k)}} = \overset{\Rightarrow}{a_{(k)}} \cdot \vec{z} + (k-1)uY^v + \boldsymbol{e}$ for some small error $\boldsymbol{e}$. Then, we can get the follow equality $[\vec{0}^t, (\boldsymbol{t}_{(k)})^t, \vec{0}^t, ..., \vec{0}^t] \cdot \overset{\Rightarrow}{\mathrm{ACC}_{(k)}} = (\boldsymbol{t}_{(k)})^t \cdot [\overset{\Rightarrow}{a_{(k)}}, \overset{\Rightarrow}{b'_{(k)}}]$. In the third step, we define $\boldsymbol{a}_{(k)} = (\boldsymbol{t}_{(k)})^t \cdot \overset{\Rightarrow}{a_{(k)}}$. Then, $\boldsymbol{c}_{(k)} \leftarrow (\boldsymbol{a}_{(k)}, b_{(k)0})$ can deduce to $((\boldsymbol{a}_{(k)})^t, b_{(k)0}) \leftarrow (\boldsymbol{t}_{(k)})^t \cdot [\overset{\Rightarrow}{a_{(k)}}, \overrightarrow{b'_{(k)}}]$, and

$$(\boldsymbol{t}_{(k)})^t \cdot \overrightarrow{b'_{(k)}} = (\boldsymbol{t}_{(k)})^t \cdot \overset{\Rightarrow}{a_{(k)}} \cdot \vec{z} + (\boldsymbol{t}_{(k)})^t (k-1) \cdot uY^v + (\boldsymbol{t}_{(k)})^t \cdot \boldsymbol{e} = \boldsymbol{a}_{(k)} \cdot \vec{z} + (\boldsymbol{t}_{(k)})^t \cdot \boldsymbol{e} + (k-1) \cdot u \cdot (1)^v_{(\frac{(k-1)q}{t_2}, \frac{kq}{t_2})}.$$

Notice that $msb_{(k)} = \boldsymbol{t}_{(k)} \cdot \overrightarrow{Y^v} = (1)^v_{(\frac{(k-1)q}{t_2}, \frac{kq}{t_2})}$. In the fourth step,

$$\boldsymbol{c} = \sum_{k=1}^{k=t_2} \boldsymbol{c}^{(k)} = \sum_{k=1}^{k=t_2} (\boldsymbol{a}^{(k)}, \boldsymbol{a}^{(k)} \cdot \vec{z} + \boldsymbol{t}^{(k)} \cdot \boldsymbol{e} + (k-1) \cdot u \cdot (1)^v_{(\frac{(k-1)q}{t_2}, \frac{kq}{t_2})}) = (\sum_{k=1}^{k=t_2} \boldsymbol{a}^{(k)}, \sum_{k=1}^{k=t_2} \boldsymbol{a}^{(k)} \cdot \vec{z} + \sum_{k=1}^{k=t_2} \boldsymbol{t}^{(k)} \cdot \boldsymbol{e} + u \cdot msb(v))$$

, which has the right form of LWE scheme. Secondly, we will analyze the error when operation the $\mathrm{msbExtract}$.

Fact 6 (Spectral Norm of Decomposed Matrices) Let $\boldsymbol{C}^{(i)} \leftarrow E_z(v^{(i)})$ be fresh encryptions of $v^{(i)} \in \mathbb{Z}_q$ for all $i \le l = w\sqrt{\log n}$, and assume that the $\boldsymbol{C}^{(i)}$'s are indistinguishable from random without the knowledge of z. Consider $\mathrm{ACC}^{(l)}$ as the value of ACC after the sequence of operations:

$\mathrm{ACC} \leftarrow v^{(0)}$; for $i = 1,\dots,l$ do $\mathrm{ACC} \xleftarrow{+} \boldsymbol{C}^{(i)}$

Set $\boldsymbol{D}^{(i)} = [\boldsymbol{D}_1^{(i)},\dots,\boldsymbol{D}_{d_g}^{(i)}]\cdot C$ to be the decomposition of $u^{-1}ACC^{(i)} = \sum_{i=1}^{d_g} B_g^{i-1}\boldsymbol{D}_i^{(i)}$. Then, with overwhelming probability we have

$$s_1(\boldsymbol{D}^{(0)},\boldsymbol{D}^{(1)},\dots,\boldsymbol{D}^{(l-1)}) = O(B_g\sqrt{Nd_g \cdot l}).$$

Lemma 7 (Intermediate error) Assume the hardness of $\mathrm{RLWE}_{\mathcal{R},Q,\chi}$, and let ACC be an l-encryption of v where $l \ge w\sqrt{\log N}$. Then the ciphertext $\boldsymbol{c}_{(k)} \leftarrow \mathrm{LWE}_{\vec{z}}^{t/Q}(\mathrm{msb}(v))$ as defined in line 3 of algorithm 2 while computing $\mathrm{msbExtract(ACC)}$ has an error $\mathrm{err}(\boldsymbol{c}) = \mathrm{err}(\sum_{k=1}^{k=t_2}\boldsymbol{c}_{(k)})$ which is, under the randomness used in the calls to $E_z(\cdot)$, a subgaussian variable of parameter β and mean 2δ where $\beta = O(t_2 \cdot \varsigma B\sqrt{q\cdot Nd_g \cdot l})$.

Proof. Applying l times Fact 9 $\boldsymbol{e}'' = Y^{m'}\cdot\boldsymbol{e} + [\boldsymbol{D}_1,\dots,\boldsymbol{D}_{d_g}]\cdot\boldsymbol{e}'$, we can show that ACC^l has the form

$\mathrm{ACC}^l = [\boldsymbol{A}, \boldsymbol{A}\cdot z + \boldsymbol{e}] + u\,\mathrm{Y}^v\boldsymbol{G}$ with

$\boldsymbol{e} = \sum_{i=1}^{l}\boldsymbol{D}^{(i-1)}\tilde{\boldsymbol{e}}^{(i)} = \sum_{i=1}^{l}\boldsymbol{D}^{(i-1)}Y^{w^{(i)}}\boldsymbol{e}^{(i)} = Y^{w^{(1)}}\boldsymbol{D}^{(0)}\boldsymbol{e}^{(1)} + Y^{w^{(2)}}\boldsymbol{D}^{(1)}\boldsymbol{e}^{(2)} + \dots + \boldsymbol{D}^{(l-1)}Y^{w^{(l)}}\boldsymbol{e}^{(l)}$ where $\tilde{\boldsymbol{e}}^{(i)} = Y^{w^{(i)}}\boldsymbol{e}^{(i)}$ for some $w^{(i)} \in \mathbb{Z}_q$ and $\boldsymbol{e}^{(i)}$ is the error used in the encryption $\boldsymbol{C}^{(i)} \leftarrow E_z(v^{(i)})$. The $\mathrm{uMultcontant(ACC} \leftarrow k)$ will not increase the noise in ACC, then $\mathrm{ACC}_{(k)}$ with the noise $\boldsymbol{e}_{(k)}^{\mathrm{ACC}^l} = \boldsymbol{e}$, and $\boldsymbol{c}_{(k)}$ with noise $e_{(k)} = [\vec{0}^t, \boldsymbol{t}^t, \vec{0}^t,\dots,\vec{0}^t]\cdot\boldsymbol{e}$. The final error in $\boldsymbol{c}$ is $e = \sum_{k=1}^{k=t_2} e_{(k)} = [\vec{0}^t, (\boldsymbol{t}_{(k)})^t, \vec{0}^t,\dots,\vec{0}^t]\cdot\sum_{k=1}^{k=t_2}\vec{\boldsymbol{e}} = t_2\cdot[\vec{0}^t, (\boldsymbol{t}_{(k)})^t, \vec{0}^t,\dots,\vec{0}^t]\cdot\vec{\boldsymbol{e}}$. We rewrite

$e = t_2\cdot[\vec{0}^t, \boldsymbol{t}^t, \vec{0}^t,\dots,\vec{0}^t]\cdot[\boldsymbol{D}^{(0)}, \overset{\Rightarrow}{\boldsymbol{D}^{(1)}},\dots,\boldsymbol{D}^{(l-1)}]\cdot\overrightarrow{(\tilde{\boldsymbol{e}}^{(1)},\dots,\tilde{\boldsymbol{e}}^{(l)})}.$

Recall that $\|\boldsymbol{t}\| = \sqrt{q/t_2}$, and by Fact 12, we have that $[\boldsymbol{D}^{(0)}, \overset{\Rightarrow}{\boldsymbol{D}^{(1)}},\dots,\boldsymbol{D}^{(l-1)}]$ has spectral norm $O(B_g\sqrt{Nd_g \cdot l})$. We can rewrite $e = \boldsymbol{v}\cdot\overrightarrow{(\tilde{\boldsymbol{e}}^{(1)},\dots,\tilde{\boldsymbol{e}}^{(l)})}$ where $\|\boldsymbol{v}\| = O(B_g\sqrt{q\cdot Nd_g \cdot l})$ } and $\overrightarrow{(\tilde{\boldsymbol{e}}^{(1)},\dots,\tilde{\boldsymbol{e}}^{(l)})}$

is a subgaussian vector of parameter ς . We conclude that the final error is subgaussian of parameter $\beta = O(t_2 \cdot \varsigma B_g \sqrt{qNd_g \cdot l})$.

Theorem 8 Assuming the hardness $\mathrm{RLWE}_{\mathcal{R},Q,\chi}$ the above Homomorphic Accumulator Scheme is ε-correct with error function

$$\varepsilon(l) = w \cdot \sqrt{\log n} \cdot \sqrt{\frac{q^2}{Q^2}(t_{t_2}^2 \varsigma^2 B_g^2 \cdot l \cdot q \cdot Nd_g + \sigma^2 Nd_{ks}) + \|\boldsymbol{s}\|^2}$$

By Lemma 7 we have $\mathrm{err}(\boldsymbol{c}) = \mathrm{err}(\sum_{k=1}^{k=t_2} \boldsymbol{c}_{(k)})$ which is a subgaussian variable of parameter β and mean 2δ where $\beta = O(t_2 \cdot \varsigma B \sqrt{q \cdot Nd_g \cdot l})$. By Lemmata 2, the KeySwitch process outputs an encryption $\mathrm{KeySwitch}(\boldsymbol{c}, \{\boldsymbol{k}_{i,j,v}\})$ with subgaussian error of parameter $\sqrt{\beta^2 + Nd_{ks}\sigma^2} = \sqrt{(O(t_2 \cdot \varsigma B \sqrt{q \cdot Nd_g \cdot l})^2 + Nd_{ks}\sigma^2}$. By Lemma 3, the ModSwitch process outputs is a $\mathrm{LWE}_{\boldsymbol{s}}^{t/q}(m)$ ciphertext $\mathrm{ModSwitch}(\boldsymbol{c})$ with subgaussian error of parameter

$$\sqrt{(\frac{q}{Q}\sqrt{(O(t_2 \cdot \varsigma B \sqrt{q \cdot Nd_g \cdot l})^2 + Nd_{ks}\sigma^2})^2 + 2\pi(\|\boldsymbol{s}\|^2 + 1)} = w \cdot \sqrt{\log n} \cdot \sqrt{\frac{q^2}{Q^2}(t_{t_2}^2 \varsigma^2 B_g^2 \cdot l \cdot q \cdot Nd_g + \sigma^2 Nd_{ks}) + \|\boldsymbol{s}\|^2}.$$

5.3 Security

We made the improvement based on the DM14 schemes, and the improvement is focused on enhancing the domain of the plaintext. There are 2 main differences between our scheme and DM14. On one hand, we replace the origin LWE scheme $\mathrm{LWE}_{\boldsymbol{s}}^{2/q}$ to a $\mathrm{LWE}_{\boldsymbol{s}}^{4/q}$. The change will not affect the security, since both of the two schemes are secure. On the other hand, we extend the msbExtract to $\mathbb{Z}_{t_2}$, by do some operates on the big ciphertexts which will not reduce the security.

5.4 Parameters, Implementation

We implement our scheme with the follow parameters: $n = 500$, $\sigma = 2^{17}$, $\varsigma = 1.4$, $N = 1024$, $q = 342$, $Q = 2^{32}$, $t_2 = 8$, $u = \frac{Q}{8} - 1 = 2^{29} + 1$, $B_g = 2^{11}$, $d_g = 3$, $B_r = 23$, $d_r = 2$, $B_{ks} = 23$, $d_{ks} = 2$. During the implementation of our scheme, we used the FFTW library http://www.fftw.org/install/windows.html, which is a free portable C library, allowing the rapid prototyping of crypto-systems. The implementation results are recorded in Table 4 and Fig. 1. The tests run on a LENOVO ThinkCenter M8000T computer with the Windows 7 operation system, equipped with Intel(R) Core(TM) i5-2400 CPU, 3.10GHz and 2.00GB Memory.

Table 1 The running time comparison between our scheme and DM14

	SETUP	KeyGen	Encrypt	Refesh	Decrypt
Our scheme	858ms	29484ms	0ms	6848ms	16ms

DM14[13]	842	29952ms	0ms	7285ms	15ms

6 Summary and Future Directions

We solve the problem of extending DM14 to large plaintext space. It is the foundation of combining the fast refresh with other optimal technologies, which is a meaningful job.

Reference

1 Gentry C. Fully homomorphic encryption using ideal lattices [C]//Proc of the 41st Annual ACM Symposium on Theory of Computing, STOC. New York: ACM, 2009: 169-178

2 Van Dijk M, Gentry C, Halevi S, et al. Fully homomorphic encryption over the integers [G] // LNCS 6110: Advances in Cryptology–EUROCRYPT 2010. Berlin: Springer, 2010: 24-43

3 Brakerski Z, Vaikuntanathan V. Efficient fully homomorphic encryption from (standard) LWE [C]// Proc of 52nd Annual Symp on Foundations of Computer Science. Los Alamitos,CA:IEEE Computer Society, 2011: 97-106

4 Tanping Z, Xiaoyuan Y, Wei Z, et al. Efficient Fully Homomorphic Encryption with Circularly Secure Key Switching Process[J]. a a, 1: 1.

5 Brakerski Z, Gentry C, Vaikuntanathan V. (Leveled) fully homomorphic encryption without bootstrapping [C]//Proc of the 3rd Innovations in Theoretical Computer Science Conf. New York: ACM, 2012: 309-325

6 Gentry C, Sahai A, Waters B. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based [G] // LNCS 8042: Advances in Cryptology–CRYPTO 2013. Berlin: Springer, 2013: 75-92

7 Alperin-Sheriff J , Peikert C. Practical bootstrapping in quasilinear time. In CRYPTO 2013. 2013: 1–20.

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

9 C. Gentry, S. Halevi, and N. P. Smart. Better bootstrapping in fully homomorphic encryption. In M. Fischlin,J. Buchmann, and M. Manulis, editors, PKC 2012, volume 7293 of LNCS, pages 1–16. Springer, May 2012.

10 Brakerski Z, Vaikuntanathan V. Lattice-based FHE as secure as PKE. In ITCS, pages 1,2014.

11 Barrington D. Bounded-width polynomial-size branching programs recognize exactly those languages in NC1. In STOC, pages 1–5. 1986.

12 Alperin-Sheriff J, Peikert C. Faster bootstrapping with polynomial error[M]//Advances in Cryptology–CRYPTO 2014. Springer Berlin Heidelberg, 2014: 297-314.

13 Ducas L, Micciancio D. FHEW: Bootstrapping Homomorphic Encryption in less than a second[M]//Advances in Cryptology--EUROCRYPT 2015. Springer Berlin Heidelberg, 2015: 617-640.

Biographies

Zhou Tanping, born in 1989. PhD candidate. His main research interests include fully homomorphic encryption and lattice-based cryptology. *The corresponding author. Email: **850301775@qq.com**.

Liu Longfei, born in 1990. master. His main research interests include fully homomorphic encryption and lattice-based cryptology.

Yang Xiaoyuan, born in 1959. PhD supervisor. His main research interests include information security and cryptology.

Han Yiliang, born in 1977. PhD supervisor and assistant professor. His main research interests include information security and cryptology