A fast integer-based batch full-homomorphic encryption scheme over finite field

Long Zhang[1] and Qiuling Yue[2]

[1]*School of Mathematical Sciences, Heilongjiang University, Harbin, 150080, China*
lzhang@hlju.edu.cn
[2]*School of Mathematical Sciences, Heilongjiang University, Harbin, 150080, China*
yueqiuling@hotmail.com

November 19, 2013

Abstract

In view of the problems that the plaintext space is too small in the existing schemes. In this paper, a new improved scheme is presented by improving the DGHV scheme. The plaintext space of the improved scheme is extended from finite prime field F_2 in the original scheme to finite prime field F_p. Combine and apply the method of encryption in the batch encryption scheme was proposed in 2013, and the plaintext space is further extended to finite fields F_q. The new improved scheme encrypts the message by applying the modular mathematical operation and the Chinese remainder theorem, and the security of the scheme is based on the the difficulty of approximate greatest common divisor problem and the spare subset sum problem. The improved scheme we got has the advantages of encrypt fast, and the size of ciphertext is small. So compared with the original scheme, it is better for practical application.

Key words: Fully homomorphic encryption; Public key; Finite fields; Chinese Remainder Theorem; Squash decryption circuit

1 Introduction

The ideas of fully homomorphic encryption was proposed in 1978 and has been highly concerned by the researchers(see[1]). The homomorphic makes the scheme has a lot of applications in many fields. Especially it has a significant impact on untrusted platform for trusted computing, So many problems in fact can be applied fully homomorphic encryption to solve. Over the years it has been a beautiful dream the cryptographers longing for. But little progress in their studies, until 2009 Gentry (see[2, 3]) presented the first fully homomorphic encryption scheme, and the research of fully homomorphic encryption scheme just walked into the new era. DGHV is a classic fully homomorphic encryption scheme following the scheme based on ideal lattice that was proposed by Gentry(see[4]).

The advantages of this scheme are the required mathematical tool has simple structure and is easy to understand and computer-implemented. But now there are still many problems in fully homomorphic encryption, where efficiency is important and these problems seriously affect its application in practice. Therefore, how to improve the whole efficiency of the scheme is crucial. DGHV scheme can only encrypt a single bit of message, and encryption and decryption is slow and the size of ciphertext is too large. In this paper, According to these problems to improve and the improved scheme can be encrypted message from the finite prime fieldF_p, not the finite prime field F_2. Then improve anther scheme was proposed by Cheon et al.(see[5]), and eventually get a scheme which plaintext space is arbitrary finite fieldF_q. This expansion undoubtedly will shorten the time of encryption and the size of ciphertext consumedly, and these advantages will be conducive to fully homomorphic filtering scheme in practical application.

2 Preliminaries

2.1 Symbols

For a real number r, we denote by $\lceil z \rceil$ if $\lceil z \rceil \in [z, z+1)$; denote by $\lfloor z \rfloor$ if $\lfloor z \rfloor \in (z-1, z]$; denote by $\lfloor z \rceil$ if $\lfloor z \rceil \in (z-1/2, z+1/2]$; For a real number z and an integer p, the quotient of z with respect to p is denoted by $q_p(z)$ and the remainder is denoted by$r_p(z) \in (-p/2, p/2]$, namely $q_p(z) = \lfloor z/p \rceil$,$r_p(z) = z - q_p(z) \cdot p$. We also denote the remainder by $[z]_p$ or$r_p(z)$, we use these notations interchangeably throughout the paper.

2.2 DGHV scheme

DGHV scheme changes the mathematical tool fully homomorphic scheme based on from the ideal lattice which was put forward by Gentry to integer ring. So it makes fully homomorphic scheme more easy to understand. Dijk et al. who wanted to show that so complex problems can also be achieved through the simple integers. The construction ideas of the scheme as same as Gentry's, Just using simpler tool. Therefore the whole scheme looks very concise and clear.

The way of DGHV is to construct a somewhat scheme which can achieve the finite orders of fully homomorphic operations firstly, then converts it into the bootstrapping scheme by squashing the decryption circuit. Finally, gets the fully homomorphic encryption scheme that we expect using Bootstrap transformation theorem. The whole thought is relatively simple, but there is no lack of nodus, such as how to squash the decryption circuit, and how to determine the times the somewhat scheme can evaluate. Its security is based on mathematical calculations difficult problems: Approximate greatest common divisor problem. More details about the security see [4].

3 Improved DGHV scheme

The improved DGHV scheme in this paper mainly aims at plaintext space in the original scheme is too small(i.e. plaintext space only has two elements 0 and 1)to improve, and makes plaintext space of the improved scheme extending to the finite prime field F_p. To encrypt a number n, if apply DGHV scheme firstly, we should convert n into binary bits and then encrypt per-bit. It will encrypt $\lceil \log n \rceil$ times. The time of encryption determines the size of ciphertext and the time encryption and decryption require. Thus reducing the times of encryption will improve the efficiency of fully homomorphic encryption scheme.

3.1 Somewhat scheme

3.1.1 Parameters

γ is the bit-length of the integers in the public key;

η is the bit-length of the secret key;

ρ is the bit-length of the noise;

ε is the bit-length of the plaintext;

ξ is the bit-length of p in the finite prime field.

τ is the number of the integers in the public key;

And these parameters are not casual to set, they must satisfy the following conditions:

$\rho = \omega(\log \lambda)$, in order to resistance the brute force attack of the noise;

$\eta \geq \rho \cdot \Theta(\lambda \log^2 \lambda)$, in order to evaluate the squashed decryption circuit;

$\gamma = \omega(\eta^2 \log \lambda)$, in order to be able to defeat all kinds of the attacks based on lattice;

$\tau \geq \gamma + \omega(\log \lambda)$, for using the hash leftover lemma;

$\xi \leq \Theta(\eta \log \lambda)$, to ensure the noise does not exceed the threshold so that the decryption is correct;

$\varepsilon \leq \xi/2$, for applying the improved scheme to encrypt the message.

We also use another noise parameter $\rho' = \rho + \omega(\log \lambda)$, a simple parameter setting is $\rho = \lambda$, $\rho' = 2\lambda$, $\eta = \Theta(\lambda^2)$, $\gamma = \Theta(\lambda^5)$, $\tau = \gamma + \lambda$ [4].

3.1.2 Concrete scheme

$KeyGen(\lambda)$: Select a η-bit prime number s from $(2\mathbb{Z}+1) \cap [2^{\eta-1}, 2^{\eta})$ as the secret key. Randomly select τ primes q_i from $[0, 2^{\gamma}/s]$, $r_i \in (-2^{\rho}, 2^{\rho})$, and compute $x_i = sq_i + r_i$, with $1 \leq i \leq \tau$. Output the public key $pk = (N, x)$, secret key $sk = s$.

$Encrypt(pk, m)$: Randomly choose a subset $S \subseteq \{1, 2, \cdots, \tau\}$ and a integer $r \in (-2^{\rho'}, 2^{\rho'})$. Let $c \leftarrow [m + pr + p\sum_{i \in S} x_i]_{x_0}$ and output c.

$Decrypt(sk, c)$: Compute $m \leftarrow (c mods) mod p$ and output s.

$Evaluate(pk, C, c_1, \cdots, c_g)$: For a given circuit C with g input, and ciphertext $c_1, c_2, \cdots, c_g$, firstly modulo 2 multiplication gate in circuit is switched to modulo x_0 multiplication gate, at the same time mode 2 addition gate in circuits is replaced by addition gate of module x_0, Then input the ciphertext to expansion of circuit, and execute all operations. Finally, output the result of the operations.

3.1.3 Homomorphic verification

Note in order to decrypt correctly, it need to ensure that the values of *cmods* in between $(-s/2, s/2)$. Now we verify its homomorphic. Let $c_1 \leftarrow [m_1 + pr_1 + p\sum_{i\in S} x_i]_{x_0}$, $c_2 \leftarrow [m_2 + pr_2 + p\sum_{i\in S'} x_i]_{x_0}$, then

$$\begin{aligned}
E(m_1) + E(m_2) = c_1 + c_2 &= [m_1 + pr_1 + p\sum_{i\in S} x_i]_{x_0} + [m_2 + pr_2 + p\sum_{i\in S'} x_i]_{x_0} \\
&= [m_1 + pr_1 + p\sum_{i\in S} x_i + m_2 + pr_2 + p\sum_{i\in S'} x_i]_{x_0} \\
&= [m_1 + m_2 + p(r_1 + r_2) + p\sum_{i\in S''} x_i]_{x_0} = E(m_1 + m_2)
\end{aligned}$$

$$\begin{aligned}
E(m_1) \times E(m_2) = c_1 \times c_2 &= [m_1 + pr_1 + p\sum_{i\in S} x_i]_{x_0} \times [m_2 + pr_2 + p\sum_{i\in S'} x_i]_{x_0} \\
&= [(m_1 + pr_1 + p\sum_{i\in S} x_i) \times (m_2 + pr_2 + p\sum_{i\in S'} x_i)]_{x_0} \\
&= [m_1 \times m_2 + pA + p\sum_{i\in S''} x_i]_{x_0} = E(m_1 \times m_2)
\end{aligned}$$

Where A is a integer.

Then consider the correctness of the decryption, and analyze the noise.

3.1.4 Correctness

The noise of above scheme is *cmods*, and we know $c = m + p(r + \tilde{r}) + sp\tilde{q}$. So $\lfloor c/s \rceil$ needs to calculate firstly. In fact, it is equivalent to solve the integer $p\tilde{q}$ which is the quotient$c$ divided by s. Due to $c/s = p(r + \tilde{r})/s + p\tilde{q}$, in order to satisfy $\lfloor c/s \rceil = p(r + \tilde{r})$, need to satisfy the following conditions firstly:

$$|\frac{m + p(r + \tilde{r})}{s}| \leq \frac{1}{2}$$

Namely $|m + p(r + \tilde{r})| \leq s/2$.

Furthermore $c - \lfloor c/s \rceil = |m + p(r + \tilde{r})|$, next step is to calculate $c - \lfloor c/s \rceil$ modulo p. Because $m \leq p/2$, hence $(p(r + \tilde{r}))modp = m$. So we can properly recover the plaintext m.

3.1.5 Noise and the times of evaluate polynomial

Observe the noise how to change in above verification, when in addition situations the noise is $m_1 + m_2 + p(r_1 + r_2 + \tilde{r})$, while in the multiplication circumstances, the noise is $[m_1 + p(r_1 + \tilde{r})] \times [m_1 + p(r_1 + \tilde{r})]$.

According to the changes above, easy to know in the add operation circumstances, the noise is in a linear growth and increases slowly, but in the multiplicative circumstances, the noise is in quadratic growth and increases sharply. So impact on the capacity of evaluation mainly is the degree of polynomials and the depth of multiply circuit. As we know, once the noise exceeds the threshold value, it would not be able to decrypt correctly. So if want to achieve fully homomorphic encryption scheme, it requires for reducing noise. A method of refresh the noise was presented in Gentry's papers, namely *Recrypt* algorithm, more details about *Recrypt* seeing [3].

After discussing the noise, and then take a look at how many times the scheme can evaluate. According to the definition of noise, we know the noise is $m_1 m_2 + pR$, where R is an integer.

Let circuit C as the circuit need to evaluate. Circuit C can be expressed as a g element d times function f. For a given plaintext sequence $m_1, m_2, \cdots, m_g$, and the corresponding ciphertext sequence is $c_1, c_2, \cdots, c_g$, where $c_i = Encrypt\ (pk, m_i)$, $i = 1, 2, \cdots, g$. The value range of a g element d times function f can be used to measure by an elementary symmetric polynomial.

$$|f(x_1, x_2, \cdots, x_g)| \leq C_g^d M^d \leq g^d M^d$$

where $x_i \leq M$.

Let $x_i = m_i + p(r_i + \tilde{r}_i)$, then $M \sim 2^{\xi+\rho'+2}$, $s \sim 2^{\eta}$, $\xi \sim (\alpha' - 1)\rho'$ can be deduced by $M \sim 2^{\xi+\rho'+1} \sim 2^{\alpha'(\rho'+2)}$, where α' is a constant.

We have

$$d \leq \frac{\alpha'(\eta - 4)}{\log gM}.$$

So the times of polynomial in the scheme can evaluate satisfies

$$d \leq \frac{\alpha'(\eta - 4)}{\log g + \alpha'(\rho' + 2)}.$$

3.2 Bootstrapping scheme

In order to compress we still need to introduce some other parameters.

$\kappa = \gamma\eta/\rho'$, $\Theta = \omega(\kappa \log \lambda)$, $\lambda_{sub} = \lambda$, $\kappa' = \gamma k'/\rho'$.

KeyGen: Choose a random integer from $[2^{\eta-\xi}, 2^{\eta-\xi-1})$, and compute $s = pt + 1$, and ask $s/2$ for a prime, or re-select t. Invoke the *KeyGen* algorithm in above scheme to generate public key $pk = \langle x_0, x_1, \cdots, x_\tau \rangle$, and secret key $sk = s$. Let $x_s = \lfloor 2^\kappa / s \rceil$, and randomly select a Θ-dimensional vector $\overrightarrow{s} = \langle s_1, s_2, \cdots, s_\Theta \rangle$ which the hamming weight is λ_{sub}. Then get a set $S = \{i : s_i = 1\}$ according to the vector $\vec{s}$, and we know the number of non-zero elements is λ_{sub}. Randomly choose integers u_i from $[0, 2^\kappa + 1)$, where $i = 1, \cdots, \Theta$. Make $\sum_{i \in S} u_i = x_s (mod 2^{\kappa+1})$. Let $y_i = u_i / 2^\kappa$, and get a vector $\vec{y} = \langle y_1, \cdots, y_\Theta \rangle$, where

$$\begin{aligned}
[\sum_{i\in S} y_i]_p &= [\sum_{i \in S} u_i / 2^\kappa]_p \\
&= [(\sum_{i\in S} u_i)/2^\kappa]_p \\
&= [(x_s mod 2^{\kappa+1})/2^\kappa]_p \\
&= [(\lfloor 2^\kappa / s \rceil mod 2^{\kappa+1})/2^\kappa]_p
\end{aligned}$$

$$= [(2^{\kappa}/s)/2^{\kappa}]_p = [1/s]_p$$
$$= (1/s) - |\Delta_s|$$

Encrypt: Apply the *Encrypt* algorithm in above scheme to generate the ciphertext c^* of plaintext m. Then let $z_i \leftarrow [c^* \cdot y_i]_p$, where $i = 1, \cdots, \Theta$. Each z_i retains n bits accuracy: $n = \lceil \log \theta \rceil + 3$, and output c^* and $\vec{z} = \langle z_1, \cdots, z_{\Theta} \rangle$.

Decrypt: Input c^*, $\vec{z}$ and secret key $\vec{s}$. Compute $m \leftarrow [c^* - \lfloor \sum_i s_i z_i \rceil]_p$and finally output m.

Evaluate: For a given circuit C with g inputs, and ciphertext $c_1, c_2, \cdots, c_g$, firstly modulo 2 multiplication gate in circuit is switched to modulo x_0 multiplication gate, at the same time mode 2 addition gate in circuits is replaced by addition gate of module x_0, and then input the ciphertext into expansion of circuit. In order to keep the correctness of decryption, it need to extract the main ciphertext c^* from ciphertext c before operation. Then refresh the main ciphertext c^* to get $c^{*'}$, and input it to the operation gate. Output $c^{*''}$, and $c' = (c^{*''}, z'')$ is extended by it. This continues until the end of the operations, and the final result will be outputted.

To achieve better results of reducing the complexity of decryption, According to [4], the decryption circuit is divided into three steps.

1.Compute $a_i = \sum s_i z_i$,

2.Generate $m+1$ rational numbers $\{w_j\}_{j=1}^{m}$ according to λ_{sub} rational numbers $\{a_i\}_{i=1}^{\Theta}$, and satisfies $\sum w_j = \sum a_i (modp)$.

3.Compute and output $c^* - \sum w_i$.

Now analyze the times of polynomials in above three steps.

Step 1. The times of polynomial required is 2;

Step 2. The times of polynomial required is λ_{sub};

Step 3. The times of $\sum w_j$ required is $32 \log^2 \lambda$. Take $\lambda_{sub} = \lambda$, then the times of the decryption polynomial required approximately is $2\lambda_{sub} \cdot 32 \log^2 \lambda = 64\lambda \log^2 \lambda$, and the times of corresponding expansion decryption circuit required approximately is $128\lambda \log^2 \lambda$. Since $\log g$ is very small relative to η, it can be neglected. If want the decryption circuit can be evaluated ,it must satisfy

$$128\lambda \log^2 \lambda \leq \frac{\eta - 4}{\rho' + 2}.$$

Let $\eta = \rho' \cdot 128\lambda \log^2 \lambda$, and the times of polynomial the scheme can evaluate is $128\lambda \log^2 \lambda \leq \eta - 4/\rho' + 2$. At the same time the decryption circuit belongs to the permission circuits, therefore the above scheme is a bootstrapping scheme. According the bootstrap transformation theorem to guarantee that the fully homomorphic scheme can be converted by a bootstrapping scheme. Thus we get a fully homomorphic encryption scheme whose plaintext space is the finite prime field F_P.

4 Introduce [5]

Coron et al. proposed [5] in 2013, and it got a batch fully homomorphic scheme with Chinese remainder theorem. Its main idea is to select l relatively prime integers $p_0, p_1, \cdots, p_{l-1}$ and gets the ciphertext $c \leftarrow q \cdot \Pi_{i=0}^{l-1} p_i + CRT(2r_0 + m_0, \cdots, 2r_{l-1} + m_{l-1})$ through encrypting the plaintext $m_0, m_1, \cdots, m_{l-1}$.

Hence a plaintext vector $\vec{m} = \langle m_0, m_1, \cdots, m_{l-1} \rangle$ can be encrypted a ordinary ciphertext:

$$c = q[\sum_{i=0}^{l-1} m_i \cdot x'_i + 2r + 2\sum_{i \in S} x_i]_{x_0}.$$

where x_i satisfies $x_i mod p_j = r_{i,j}$, and the additional public key satisfies $x'_i mod p_j = \delta_{i,j} + 2r'_{i,j}$; where $\delta_{i,j}$ is Kronecker function. So these can guarantee $[c mod p_j]_2 = m_j$. The following is the scheme.

4.1 Parameters

$\rho \geq 2\lambda$, for resisting brute-force attack against noise;

$\eta \geq \alpha' + \rho' + 1 + \log_2(l)$, keep the correctness of decryption;

$\eta \geq \rho \cdot \Theta(\lambda \log^2 \lambda)$, in order to evaluate the squashed decryption circuit;

$\gamma = \omega(\eta^2 \log \lambda)$, in order to be able to defeat all kinds attack method based on lattice;

$\rho' \geq \rho + \lambda$ and $\alpha' \geq \alpha + \lambda$, to keep the semantic security;

$\alpha\tau \leq \gamma + \lambda$ and $\tau \leq l(\rho' + 2) + \lambda$, to ensure the noise does not exceed the threshold value so that the decryption is correct;

$\varepsilon \leq \xi/2$, for applying the hash leftover lemma.

The parameter setting is $\rho = 2\lambda$, $\rho' = 3\lambda$, $\eta = \Theta(\lambda^2)$, $\alpha = \Theta(\lambda^2)$, $\tau = \Theta(\lambda^3)$, $\alpha' = \Theta(\lambda^2)$, $l = \Theta(\lambda^2)$, $\gamma = \Theta(\lambda^5)$ [5].

4.2 Somewhat scheme

KeyGen(λ): Select l η-bit prime numbers $p_0, p_1, \cdots, p_{l-1}$ and let $\pi = p_0 p_1 \cdots p_{l-1}$, and $x_0 = q_0\pi$, where $q_0 \leftarrow [0, 2^\gamma/\pi]$. According to the following method to generate x_i, x'_i and Π_i:

$1 \leq i \leq \tau$, $x_i mod p_j = 2r_{i,j}$,

$1 \leq i \leq \tau$, $x'_i mod p_j = \delta_{i,j} + 2r'_{i,j}$,

$1 \leq i \leq \tau$, $\Pi_i mod p_j = 2\varpi_{i,j} + \delta_{i,j} \cdot 2^{\rho'+1}$,

where $r'_{i,j}$, $\varpi_{i,j} \leftarrow \mathbb{Z} \cap (-2^\rho, 2^\rho)$, $r_{i,j} \leftarrow \mathbb{Z} \cap (-2^{\rho'-1}, 2^{\rho'-1})$.

Finally, output the public key $pk = \langle x_0, (x_i)_{0 \leq i \leq \tau}, (x'_i)_{0 \leq i \leq l-1}, (\Pi_i)_{0 \leq i \leq l-1} \rangle$, and the secret key $sk = (p_j)_{0 \leq j \leq l-1}$.

Encrypt(pk, m): Randomly choose vectors $\vec{b} = (b_i)_{0 \leq i \leq \tau} \in (-2^\alpha, 2^\alpha)^\tau$ and $\vec{b'} = (b'_i)_{0 \leq i \leq \tau} \in (-2^{\alpha'}, 2^{\alpha'})^l$. Output

$$c = [\sum_{i=0}^{l-1} m_i \cdot x'_i + \sum_{i=0}^{l-1} b'_i \cdot \Pi_i + \sum_{i=1}^{\tau} b_i \cdot x_i]_{x_0}.$$

.

Decrypt(sk, c): Compute $m_j \leftarrow (c mod p_j) mod 2$ and output $\vec{m} = (m_0, m_1, \cdots, m_{l-1})$.

4.3 Bootstrapping scheme

Using the same compression method as [4].

KeyGen: Invoke the *KeyGen* algorithm in above scheme to generate public key pk^*, and secret key $sk^* = (p_0, p_1, \cdots, p_{l-1})$. Randomly select a θ-dimensional vector $\overrightarrow{s_j} = \langle s_{j,0}, s_{j,1}, \cdots, s_{j,\theta-1} \rangle$ and Θ integers u_i from $[0, 2^\kappa + 1)$, where $i = 1, \cdots, \Theta$, such that $\sum_{i=0}^{\Theta-1} s_{j,i} u_i = x_{p_j} (mod 2^{\kappa+1})$. Let $y_i = u_i / 2^\kappa$, and get a vector $\overrightarrow{y} = \langle y_1, \cdots, y_\Theta \rangle$. We have $1/p_j = \sum_{i=0}^{\Theta-1} s_{j,i} y_i + \varepsilon_j mod 2$, where $|\varepsilon_j| < 2^{-\kappa}$ and output $pk = (pk^*, \vec{y})$, $sk = \vec{s}$.

Encrypt: Apply the *Encrypt* algorithm in above scheme to generate the ciphertext c^* of plaintext m. Then let $z_i \leftarrow [c^* \cdot y_i]_2$, where $i = 1, \cdots, \Theta$. Each z_i retains n bits accuracy: $n = \lceil \log \theta + 1 \rceil + 3$. Define $\vec{z} = (z_i)_{i=0,\cdots,\Theta-1}$ and output $c = (c^*, \vec{z})$.

Decrypt: Compute $m_j \leftarrow [\lfloor \sum_{i=0}^{\Theta-1} s_{j,i} z_i \rceil]_2 \oplus (c mod 2)$and finally output $\vec{m}$.

There is no longer the duplicate explanation for the correctness safety of the scheme, more details see [5].

5 Finite field scheme

According to finite field theory, the arbitrary finite field can be expressed as a prime field n times extension field. The thought in section 3 is mainly to improve [5]. Finally we will get a fully homomorphic scheme whose plaintext space is the finite field F_q, the concrete scheme is following:

$\rho \geq 2\lambda$, for resisting brute-force attack against noise;

$\eta \geq \alpha' + \rho' + 1 + \log_2(l)$, keep the correctness of decryption;

$\eta \geq \rho \cdot \Theta(\lambda \log^2 \lambda)$, in order to evaluate the squashed decryption circuit;

$\gamma = \omega(\eta^2 \log \lambda)$, in order to be able to defeat all kinds attack method based on lattice;

$\rho' \geq \rho + \lambda$ and $\alpha' \geq \alpha + \lambda$, to keep the semantic security;

$\alpha\tau \leq \gamma + \lambda$ and $\tau \leq l(\rho' + 2) + \lambda$, to ensure the noise does not exceed the threshold value so that the decryption is correct;

$\varepsilon \leq \xi/2$, for applying the hash leftover lemma;

$\xi \leq \Theta(\eta \log \lambda)$, to also ensure the noise does not exceed the threshold value so that the decryption is correct;

$\varepsilon \leq \xi/2$, for applying the improved scheme.

The parameter setting is $\rho = 2\lambda$, $\rho' = 3\lambda$, $\eta = \Theta(\lambda^2)$, $\alpha = \Theta(\lambda^2)$, $\tau = \Theta(\lambda^3)$, $\alpha' = \Theta(\lambda^2)$, $l = \Theta(\lambda^2)$, $\gamma = \Theta(\lambda^5)$ [5].

5.1 Somewhat scheme

KeyGen(λ): Select l η-bit prime numbers $p_0, p_1, \cdots, p_{l-1}$ and let $\pi = p_0 p_1 \cdots p_{l-1}$, and $x_0 = q_0 \pi$, where $q_0 \leftarrow [0, 2^\gamma / \pi]$. According to the following method to generate x_i, x'_i and Π_i:

$1 \leq i \leq \tau$, $x_i mod p_j = pr_{i,j}$,

$1 \leq i \leq \tau$, $x'_i mod p_j = \delta_{i,j} + pr'_{i,j}$,

$1 \leq i \leq \tau$, $\Pi_i mod p_j = p\varpi_{i,j} + \delta_{i,j} \cdot 2^{\rho'+1}$,

where $r'_{i,j}$, $\varpi_{i,j} \leftarrow \mathbb{Z} \cap (-2^\rho, 2^\rho)$, $r_{i,j} \leftarrow \mathbb{Z} \cap (-2^{\rho'-1}, 2^{\rho'-1})$.

Finally, output the public key $pk = \langle x_0, (x_i)_{0\leq i\leq\tau}, (x'_i)_{0\leq i\leq l-1}, (\Pi_i)_{0\leq i\leq l-1}\rangle$, and the secret key $sk = (p_j)_{0\leq j\leq l-1}$.

$Encrypt(pk, m)$: Randomly choose a vector $\vec{b} = (b_i)_{0\leq i\leq\tau} \in (-2^\alpha, 2^\alpha)^\tau$ and the other vector $\vec{b'} = (b'_i)_{0\leq i\leq\tau} \in (-2^{\alpha'}, 2^{\alpha'})^l$. Output

$$c = [\sum_{i=0}^{l-1} m_i \cdot x'_i + \sum_{i=0}^{l-1} b'_i \cdot \Pi_i + \sum_{i=1}^{\tau} b_i \cdot x_i]_{x_0}.$$

.

$Decrypt(sk, c)$: Compute $m_j \leftarrow (c mod p_j) mod p$ and output $\vec{m} = (m_0, m_1, \cdots, m_{l-1})$.

5.2 Bootstrapping scheme

Using the similar squash method as [4].

$KeyGen$: Choose a prime number t. Calculate $s = pt + 1$ and it satisfies $s/2$ is still a prime, or reselect t. Invoke the $KeyGen$ algorithm in above scheme to generate public key pk^*, and secret key $sk^* = (p_0, p_1, \cdots, p_{l-1})$. Randomly select a θ-dimensional vector $\overrightarrow{s_j} = \langle s_{j,0}, s_{j,1}, \cdots, s_{j,\theta-1}\rangle$ and Θ integers u_i from $[0, 2^\kappa + 1)$, where $i = 1, \cdots, \Theta$. Make $\sum_{i=0}^{\Theta-1} s_{j,i}u_i = x_{p_j}(mod 2^{\kappa+1})$. Let $y_i = u_i/2^\kappa$, and get a vector $\overrightarrow{y} = \langle y_1, \cdots, y_\Theta\rangle$. We have $1/p_j = \sum_{i=0}^{\Theta-1} s_{j,i}y_i + \varepsilon_j mod p$, where $|\varepsilon_j| < 2^{-\kappa}$ and output $pk = (pk^*, \vec{y})$, $sk = \vec{s}$.

$Encrypt$: Apply the $Encrypt$ algorithm in above scheme to generate the ciphertext c^* of plaintext m. Then let $z_i \leftarrow [c^* \cdot y_i]_p$, where $i = 1, \cdots, \Theta$. Each z_i retains n bits accuracy: $n = \lceil \log\theta + 1\rceil + 3$. Define $\vec{z} = (z_i)_{i=0,\cdots,\Theta-1}$ and output $c = (c^*, \vec{z})$.

$Decrypt$: Compute $m_j \leftarrow (c - \lfloor\sum_{i=0}^{\Theta-1} s_{j,i}z_i\rceil mod s_j) mod p$, and finally output $\vec{m}$.

6 Comparison and analysis

In this paper, the encryption algorithm is similar to [4], but due to the expansion of plaintext space, the running time of encryption and the size of ciphertext are varied considerably.

Now there is a toy instance, it just want to illustrate how concise and fast the improved scheme, so it do not take account of security.

The plaintext is "We are in China".

Which corresponds to the set of integers Z_{26} is

22040017040813020708130O,

which converts into binary number is

1010010000111101000100111010001110O.

Apply different schemes to encrypt the message, we can get the Table 1 below.

The data in above table is simulated on a desktop computer(Intel Core i3-2120 at 3.30GHz, 2.0GB RAM)

Table 1: The comparison among these scheme

scheme	size of ciphertext	time of encryption	time of decryption
DGHV	11445	0.19s	0.16s
$F_p = F_{21192287}$	1336	0.022s	0.018s
$F_q = F_{211922287^4}$	336	0.019s	0.015s

According to the above comparison of the data in Table 1, the size of ciphertext has reduced in the improved scheme drastically, and the time of encryption is less than before. The security parameters for this example is 5, and certainly the security parameters will be much larger than 5 in practical applications. This makes the size of public key and secret key larger, so the size of ciphertext and the computational complexity will become very large. At that time, the advantages of improved scheme will be more obvious.

7 Summary

For researching and improving the DGHV scheme, a different kind of improved ideas is proposed. Integrate the ideas in [5], Then the plaintext space extended to a wider range: arbitrary finite field. The difficult problems based on in the new improved scheme are as the same as the original DGHV scheme. The selection of parameter is also in accordance with the requirements of the original scheme. So the safety of the improvement scheme and the security of DGHV scheme are the same. Though the security of the proposed improvement is as the same as the the original scheme, the speed of encryption is greatly faster and dramatically reduces the size of the ciphertext. These advantages are more obvious in the situation that the message is very long.

Although the homomorphic encryption was proposed in earlier time, the progress is little. There are many problems still to be solved, such as how to reduce the size of secret key, or how to shorten the size of ciphertext and at the same time ensure the security and so many problems. It still remains to be further research.

References

[1] Rivest R L, Adleman L, Dertouzos M L. On data banks and privacy homomorphisms[J]. Foundations of secure computation, 1978, 32(4): 169-178.

[2] Gentry C. Fully Homomorphic Encryption Using Ideal Lattices[A]. ACM STOC[C] 2009: 169-178.

[3] Gentry C. A fully homomorphic encryption scheme[D]. Stanford University, 2009.

[4] Van Dijk M, Gentry C, Halevi S, et al. Fully homomorphic encryption over the integers[A].Advances in CryptologyCEUROCRYPT 2010. Springer Berlin Heidelberg, 2010: 24-43.

[5] Cheon J H, Coron J S, Kim J, et al. Batch fully homomorphic encryption over the integers[A].Advances in CryptologyCEUROCRYPT 2013. Springer Berlin Heidelberg, 2013: 315-335.