A Fully Homomorphic Encryption Scheme over Finite Prime Field

Liwang Bai[1, b], Qiqi Zhao[1, c] and Yuqing Lan[1, a]

[1]School of Computer Science and Engineering, Beihang University, Beijing 100191, China;

[a] lanyuqing@buaa.edu.cn, [b] bailiwang@buaa.edu.cn, [c] kiki_pri@163.com

Keywords: data security; fully homomorphic encryption; finite prime field; recrypt.

Abstract. Data security of small & medium-sized banks in transmission, storage and even calculation in cloud environment challenges traditional encryption technology. Nevertheless, homomorphic encryption technology could meet users' needs in data privacy and cloud computing resources simultaneously. Current homomorphic encryption schemes have limited plaintext space and thus limited application scenarios. Based on the homomorphic encryption scheme over integer proposed by Dijk et al., this paper proposes an enhanced scheme which expands the plaintext space from monobit range to a finite prime field, and verifies the correctness and efficiency of it, effectively promoting the practicability of homomorphic encryption.

Introduction

Due to limited manpower, material and financial resources, small & medium-sized commercial banks tend to construct their IT service system with cloud hosting, in which case, all the data from the bank customers is managed by the cloud service provider, for whom data security must be the primary consideration. Traditional encryption technology plays an essential role in data transmission and storage, but when part of the bank's core data, including the account balance, transaction records and other data, requires invisibility while they are calculated, counted and updated in real time by the cloud service provider, traditional encryption technology fails to calculate it in encrypted state and return the correct result. Is it possible to have a kind of technology with which the bank can keep data private and secure, and can also make full use of the cloud computing resources?

Fully homomorphic encryption (FHE for short) is a form of encryption that allows computations to be carried out freely on ciphertext without the information of the cipher code, i.e., for any effective function f and plaintext m, the equation $Dec(f(Enc(m))) = f(m)$ is true. With this special property, fully homomorphic encryption is widely used in theory and practice, such as cloud computing security, ciphertext retrieval and secure multiparty computation, etc. [1]. In 2009, Gentry proposed the idea of homomorphic decryption and constructed the first fully homomorphic encryption scheme based using ideal lattice [3], and then another scheme was constructed based on the assumption of LWE (Ring --LWE) [4,5], which promotes the efficiency and practicality of homomorphic encryption. However, plaintext space is basically F_2 (only two elements: 0, 1), or F_2^n (a vector of monobit data) in the existing mature scheme while the messages to be encrypted and computed are not only 0 or 1 but also many other numbers in reality. With the current scheme, the data need to be converted into binary number, then be encrypted bit by bit, i.e., number n needs to be encrypted $\lceil log\ n \rceil$ times, resulting larger ciphertext, and longer encryption and decryption time. Aiming at the problem of too limited plaintext space, this paper improves the existing scheme, expands the plaintext space from prime field F_2 to F_p, and therefore increases the applicability of the scheme in bank data hosting and other secure cloud computing scenarios.

Related Works

Fully homomorphic encryption, presented by Rivest et al. in 1978, is known as the "Holy Grail" in the cryptography field and remains an unsolved problem. Subsequently, many schemes tackling this problem had been proposed, some of which either satisfy the additive homomorphism or the

multiplicative homomorphism, and some of which can satisfy both the addition and multiplication finitely.

In 2009, Gentry constructed the first fully homomorphic encryption scheme utilizing ideal lattice [3], which broke through the morass. Many researchers in the world proposed their improved solutions in different aspects [6-12], which are essentially based on the ideals of all sorts of rings.

The general framework for constructing FHE schemes are as follows. (1)Construct a somewhat homomorphic encryption (SHE) scheme which can merely carry out the computation of lower-order polynomials. (2) "Squash" the decrypt circuit to lower down its multiplicative degree and make sure the circuit can be evaluated by the scheme. Then the scheme is "bootstrappable" because it can prevent the noise of ciphertext from increasing after every evaluation. (3) Apply Gentry's transformation to get an FHE scheme from the bootstrappable scheme.

Some representative schemes should be noticed. In 2010, Dijk, Gentry, Halevi and Vaikuntanathan proposed the fully homomorphic encryption scheme over integers (DGHV scheme for short) [10], of which the concept is simple; in 2013, Cheon et al. proposed a multi-bit fully homomorphic encryption scheme [11]. They used the Chinese Remainder Theorem to encrypt multi bits at one time, which improved the efficiency of FHE significantly ; in 2011, based on the "partially approximate common divisor problem" [13], Tang Dianhua, Zhu Shixiong and Cao Yunfei improved the DGHV scheme by proposing a FHE scheme[14] faster in key generation and encryption.

Preliminaries

Homomorphic Encryption Scheme. A homomorphic encryption scheme contains four algorithms: the key generation algorithm $KeyGen$, the encryption algorithm Enc, the decryption algorithm Dec and the ciphertext computation algorithm $Evaluate(pk,\ f,\ c_1,\ \dots,\ c_t)$. Since the homomorphic encryption is designed to calculate the ciphertext, $Evaluate$ is the core of the scheme, while others are its base to provide encryption and decryption functions.

$KeyGen.\ (pk, sk) \leftarrow KeyGen\ (k)$, choose a parameter K, and generate the public key pk and the secret key sk.

$Enc.\ c \leftarrow Enc_{pk}\ (m)$, encrypt plaintext m with public key pk, and output ciphertext c.

$Dec.\ m \leftarrow Dec_{sk}(c)$, decrypt the ciphertext c with secret key sk, and output plaintext m.

$Evaluate$. Input public key pk, function f with t inputs and a set of ciphertext $\vec{c} = (\mathrm{c}_1, \mathrm{c}_2, \dots, \mathrm{c}_t)$, in which c_i is the ciphertext of m_i, output $c^* = Evaluate(pk, f, \vec{c})$, and meet the equality of $Dec(sk, c^*) = f(m_1, m_2, \dots m_t)$.

Homomorphic encryption scheme must be able to evaluate the following basic functions.

$Add(c_1, c_2)$

$Mult(c_1, c_2)$

Homomorphic Decryption. The noise of ciphertext increases rapidly in evaluation, especially in the multiplicative computation, which limits the times of evaluation and leads to possible failure to decrypt the result. Thus, noise control becomes the crucial problem to realize fully homomorphic encryption. Gentry uses an important technique named homomorphic decryption to solve the problem. He respectively encrypts the ciphertext output and the corresponding key bit by bit by evaluation , and executes decryption algorithm onto them, which (called ciphertext refresh) will produce a new ciphertext with the same noise scale as the ciphertext has before evaluation while the corresponding plaintext can be decrypted exactly. If the noise of the new ciphertext is low enough to permit another multiplication, there will be an immediate following evaluation after the evaluation of the ciphertext by refreshing ciphertext through homomorphic decryption. Through this recursive process,the ciphertext can be evaluated infinitely and thus the fully homomorphic encryption.

The process of homomorphic decryption are as follows.

Assume that $Encrypt(pk_1, m) \rightarrow c_1, Encrypt(pk_2, sk_{1j}) \rightarrow \overrightarrow{sk_1}$, every element of $\overrightarrow{sk_1}$ is the ciphertext from every bit of sk_1, and is encrypted with public key pk_2.

The algorithm can be expressed as:

$recypt(pk_2,\ Dec,\ \overrightarrow{sk_1},\ c_1)$:

$encrypt(pk_2,\ c_{1j}) \rightarrow \overrightarrow{c_1}$;

$evaluate(pk_2,\ Dec,\ \overrightarrow{sk_1},\ \overrightarrow{c_1}) \rightarrow c_2$.

c_1 is the ciphertext with double encryption from m, the first layer is encrypted with pk_1, and the second layer is encrypted with pk_2; c_2 is the result of homomorphic decryption, the same as ciphertext from m with pk_2.

The significance of homomorphic decryption lies in noise reduction. We can get the plaintext by decrypting the ciphertext with large noise (eliminate noise), and then encrypt it again with a new key (import new noise), which makes the noise smaller. This process only requires one single operation on ciphertext, and does not expose the plaintext. If a multiplication operation can be executed with small-enough noise, the purpose of homomorphic decryption is achieved.

DGHV Scheme. In June 2010, Dijk, Gentry, Halevi and Vaikuntanathan published a paper entitled *Fully Homomorphic Encryption over the Integers* [10].The solution in the paper is an improved scheme of Gentry's by replacing ideal lattice with integer ring, and using the addition and multiplication on the integer ring instead of the operation on the ideal lattice. The scheme is easy to understand for its simple concept.

Firstly, a symmetric encryption scheme was proposed by Dijk *et al*.

$KeyGen_\varepsilon$. The key p is a prime number and $p \in (2^{\eta-1}, 2^\eta]$;;

$Encrypt_\varepsilon$. Input plaintext m, the key p, output $c \leftarrow m + pq + 2r$, while q, r are random integer and $m + 2r < p/2$;

$Decrypt_\varepsilon$. Input ciphertext c and the key p, output plaintext $m \leftarrow (c\ mod\ p) mod 2$;

The scheme above can decrypt correctly while noise $m + 2r$ is smaller than $p/2$.

Then the asymmetric encryption schemes was given. Its public key is composed of some encryption result from 0, i.e. $x_i = q_i p + 2r_i$, while q_i and r_i are chosen randomly in the above way.

$KeyGen_\varepsilon$: $\mathrm{K}_p = \langle x_0, x_1, \dots, x_\tau, \rangle$, $K_s = p$;

$Encrypt_\varepsilon$. Input the plaintext $m \in \{0,1\}$, the public key K_p, output ciphertext $c \leftarrow m + 2r + \sum x_i$;

$Decrypt_\varepsilon$. Input ciphertext c and the secret key K_s, output plaintext $m \leftarrow (c\ mod\ p) mod 2$;

A somewhat homomorphic encryption scheme was formed by Dijk *et al*. through adding parameters and introducing $Evaluate$ algorithm.

$KeyGen_\varepsilon$. The secret key p is a prime number and $p \in (2^{\eta-1}, 2^\eta)$,the public key $\mathrm{K}_p = \langle x_0, x_1, \dots, x_\tau \rangle$, while $x_i \leftarrow \mathrm{D}_{\tau,\rho}(p), \mathrm{D}_{\tau,\rho}(p) = \left\{ x_i \middle| x_i = pq_i + r_i; q_i \leftarrow \frac{\mathrm{Z} \cap [0, 2^\gamma]}{p} \right\}, r \leftarrow \mathrm{Z} \cap [-2^\rho, 2^\rho]$, let $x_0 = \max\{x_0, x_1, \dots, x_\tau\}$, and x_0 is an odd integer, $r_p(x_0)$ is an even integer;

$Encrypt_\varepsilon$. Input plaintext $m \in \{0,1\}$, the public key K_p, choose a random subset $S \subseteq \{1,2,\cdots,\tau)$, and a random integer r $\in (-2^\rho, 2^\rho)$, output $c \leftarrow [m + 2r + 2\sum_{i \in S} x_i]_{x_0}$;

$Decrypt_\varepsilon$. Output $m \leftarrow (c\ mod\ p) mod 2$, while the input are ciphertext c and the secret key K_s;

$Evaluate$. Given the circuit C_ε, the public key K_p and ciphertexts c_j, apply the (integer) addition and multiplication gates of C_ε to the ciphertexts, performing all the operations over the integers, and return the resulting integer.

Make sure that$|m + 2r| < p/2$ to ensure the correctness of decryption. Note that $(c\ mod\ p) = c - p \cdot \lfloor c/p \rceil$, and as p is odd we can instead decrypt using the formula

$m \leftarrow (c\ mod\ p) mod 2 = (c\ mod\ 2) \oplus (\lfloor c/p \rceil mod 2)$.

According to the above formula of decryption circuit, the complexity of the decryption circuit is mainly derived from $(\lfloor c/p \rceil mod 2)$. Unfortunately, the complexity of $(\lfloor c/p \rceil mod 2)$ has exceeded d, which is the depth of permitted $Evaluate$ circuit in the scheme, so it is necessary to "squash the decrypt circuit " [3].

Firstly, the related parameters are given.

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;

τ is the number of integers in the public key.

These parameters must be set under the following constraints for the sake of security:

$\rho = \omega(log\ \lambda_\varepsilon)$, to protect against brute-force attacks on the noise;

$\eta \geq \rho \cdot \Theta(\lambda_\varepsilon log^2 \lambda_\varepsilon)$, in order to support homomorphism to evaluate the "squashed decryption circuit";

$\gamma = \omega\ (\eta^2 log\ \lambda_\varepsilon)$, to thwart various lattice-based attacks on the underlying approximate-gcd problem.

Then 3 other parameters are introduced, $\kappa = \gamma\eta/\rho'$, $r_{set} = \omega(\kappa\ log\lambda_\varepsilon)$, $r_{sub} = \lambda_\varepsilon$.

They add to the public key a set $\vec{y} = \{y_1, y_2, \dots, y_\Theta\}$ of rational numbers in [0, 2) with k bits of precision, the details of the modified encryption scheme are as follows:

$KeyGen_\varepsilon$. Generate $K_s^* = p$ and K_p^* as before. Set $x_p = \lfloor 2^\kappa/p \rceil$, choose at random a Θ-bit vector $\vec{s} = (s_1, s_2, \dots, s_\Theta)$ with Hamming weight θ, and let $S = \{i | s_i = 1\}$, choose at random integers $u_i \in Z \bigcap (0, 2^{\kappa+1})$, $i = 1, \dots, \Theta$, subject to the condition that $\sum u_i = x_p (mod\ 2^{\kappa+1})$, let $y_i = u_i/2^k, \vec{y} = \{y_1, y_2, \dots, y_\Theta\}$, output $K_p = (K_p^*, \vec{y})$, $K_s = \vec{s}$;

$Encrypt_\varepsilon$.Generate a ciphertext c^* as before. Then for $i = 1,2, \dots, r_{set}$, let $z_i \leftarrow [c^* \times y_i]_2$, keeping only $n = \lfloor log\ r_{sub} \rceil + 3$ bits of precision after the binary point for each z_i, output $\vec{c} = (c^*, z_1, z_2, \dots, z_{\text{set}})$.

$Decrypt_\varepsilon$. Output $m' \leftarrow [c^* - \lfloor \sum_{i=1}^{\tau_{set}} s_i z_i \rceil]_2$ with c^*,z_i and the secret key K_s.

Homomorphic Encryption Scheme over F_p

In order to increase the practicability, this section provides an improvement of the DGHV scheme in Reference [10], expanding the plaintext space from prime field F_2 to F_p, then squashing the decrypt circuit, and making the scheme bootstrappable.

Design of Scheme. We first give a somewhat homomophic encryption scheme, and then change it into the full homomophic encryption scheme.

A. SHE scheme

The choice of parameters is crucial to the scheme, the parameters of this paper are set according to the DGHV scheme.

a) 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 prime number p.
These parameters must be set under the following constraints:
$\rho = \omega(log\ \lambda_\varepsilon)$, to protect against brute-force attacks on the noise;
$\eta \geq \rho \cdot \Theta(\lambda_\varepsilon log^2 \lambda_\varepsilon)$, in order to support homomorphism for deep enough circuits to evaluate the "squashed decryption circuit";
$\gamma = \omega\ (\eta^2 log\ \lambda_\varepsilon)$, to thwart various lattice-based attacks on the underlying approximate-gcd problem.
$\xi \leq \Theta(\eta log\ \lambda_\varepsilon)$, in order to ensure that the noise does not exceed the threshold, and the scheme is correct.
$\varepsilon \leq \xi/2$, in order to ensure that the plaintext m is a element of finite prime field F_p so that the scheme here can be used for encrypt .
We also use another noise parameter $\rho' = \rho + \omega(log\ \lambda_\varepsilon)$, a simple parameter setting method is setting $\rho = \lambda_\varepsilon$, $\rho' = 2\lambda_\varepsilon$, $\eta = \Theta(\lambda_\varepsilon^2)$, $\gamma = \Theta(\lambda_\varepsilon^5)$.

b) symmetric encryption scheme

A simple symmetric encryption scheme is given in document [14], and this paper adopts a similar encryption algorithm to document [10] and [14] in order to illustrate clearly the improvements. The

improved scheme can encrypt any integer less than p by expanding the plaintext space from prime field F_2 to F_p instead of converting the integer into binary form and encrypting it bit by bit, therefore, and thus the efficiency of the scheme is greatly increased. In this section we consider the plaintext space as prime field F_p, i.e., the plaintext $m \in \{0,1,\cdots,p-1\}$, in which p is a prime number, and the ciphertext space is $\mathbb{C}$.

$KeyGen(\lambda_\varepsilon)$. For the secret key, choose a η-bits prime number s in $(2\mathbb{Z}+1)\cap[2^{\eta-1},2^{\eta})$.

$Encrypt(K_p,m)$, Given arbitrary integer $m \in \{0,1,\cdots,p-1\}$, calculate $c = m + sq + pr$, in which $q \in (2^{\gamma}-1, 2^{\gamma}]$ and $r \in (2^{\rho-1}, 2^{\rho})$ are random integers.

$Decrypt(K_s,c)$. Output $m \leftarrow (c \bmod s) \bmod p$.

Does the modified scheme still maintain the homomorphic properties on multiplication and addition over integers? Verification is as follows.

Assume that $c_1 = m_1 + sq_1 + pr_1, c_2 = m_2 + sq_2 + pr_2$, then

$$\begin{aligned}\mathrm{E}(m_1)+\mathrm{E}(m_2) &= c_1 + c_2 = m_1 + sq_1 + pr_1 + m_2 + sq_2 + pr_2 \\ &= m_1 + m_2 + s(q_1+q_2) + p(r_1+r_2) = \mathrm{E}(m_1+m_2)\end{aligned}$$

$$\begin{aligned}E(m_1)\times E(m_2) &= c_1 \times c_2 = (m_1 + sq_1 + pr_1)\times(m_2 + sq_2 + pr_2) \\ &= m_1 \times m_2 + s(m_2q_1 + m_1q_2 + pr_1q_2 + pr_2q_1 + sq_1q_2) + p(m_1r_2 + m_2r_1 \\ &\quad + pr_1r_2) = E(m_1 \times m_2)\end{aligned}$$

The noise is $m + pr$ in the scheme. If $m + pr < s/2$, the decryption algorithm is $m \leftarrow (c \bmod s) \bmod p = (m + pr) \bmod p$, we can get the plaintext m from ciphertext c, or we will fail to obtain correct result.

c) Asymmetric encryption scheme

$KeyGen(\lambda_\varepsilon)$. Generate the secret key K_s by choosing a prime number $s \in [2^{\eta-1}, 2^{\eta}]$, choose a prime number $q \in \left(2^{\theta-1}, 2^{\theta}\right)$ randomly, then choose at random two integers $l \in [0, 2^{\gamma}/s]$ and $h \in (-2^{\rho}, 2^{\rho})$, set $N = sq$, calculate $x = sl + ph$. Output the public key $K_p = (N, x)$ and the secret key $K_s = s$.

Choose at random two integers $r_1 \in (-2^{\rho'}, 2^{\rho'})$ and $r_2 \in (-2^{\rho}, 2^{\rho})$, output $c = (m + pr_1 + r_2x) \bmod N$ while $Kp = (N, x)$.

$Decrypt(K_s,c)$. Output $m \leftarrow (c \bmod s) \bmod p$.

$Evaluate\left(K_p, C, c_1, \cdots, c_g\right)$. For a given circuit C_ε and g ciphertexts c_i, replace the addition and multiplication gates in original circuit with gates for addition and multiplication mod N, after inputting g ciphertexts, perform all the operations and output the result eventually. Note that in order to decrypt correctly, we should have $-s/2 < c \bmod s < s/2$.

d) Correctness of decryption

The noise is $c \bmod s$ in the scheme, and $c = m + p(r_1 + r_2h) + s(r_2l - kq)$, $\lfloor c/s \rceil$ is calculated firstly in decryption, is equal to calculating integer quotient $r_2l - kq$ of c/s, since

$$c/s = \frac{m + p(r_1 + r_2h)}{s} + r_2l - kq$$

In order to satisfy $\lfloor c/s \rceil = r_2l - kq$, the following conditions need to be satisfied

$$\left|\frac{m + p(r_1 + r_2h)}{s}\right| < \frac{1}{2}$$

That is

$$|\, m + p(r_1 + r_2h)| < \frac{s}{2}$$

Then we have $c - \lfloor c/s \rceil = m + p(r_1 + r_2h)$, the next stage of decryption is perform the operation modulo p on $c - \lfloor c/s \rceil$. Since $m < p/2$, there is $\left(m + p(r_1 + r_2h)\right) \bmod p = m$, thus the plaintext is recovered correctly.

Analysis on noise and degree of evaluable polynomial:

Observe the change of noise in the verification on homomorphism above, the noise in homomorphic addition is

$$|m_1 + m_2 + p(r_1 + r_2 h + r'_1 + r'_2 h)| \le |m + p(r_1 + r_2 h)| + |m + p(r'_1 + r'_2 h)|$$

Represent homomorphic multiplication as $c_1 \times c_2 = m_1 \times m_2 + pA + sB$, then the noise is

$$|m_1 \times m_2 + p(A)| \le |m + p(r_1 + r_2 h)| \times |m + p(r'_1 + r'_2 h)|$$

According to the change rule of noise above, we know that noise has a linear growth in homomorphic addition, but has a square growth in homomorphic multiplication. It is showed that the noise increases much more rapidly in homomorphic multiplication than in homomorphic addition. Therefore, the factor impacting scheme's evaluation ability is mainly the degree of polynomials or the depth of the multiplication of circuit. Since decryption will be incorrect once the noise exceeds the threshold value, noise reduction is required to achieve fully homomorphism. Gentry gives a method that can refresh the ciphertext, that is, Recrypt algorithm, detailed in Reference [3].

The degree of evaluable polynomial in the scheme is analyzed here. According to the definition of noise, the noise $|m + p(r_1 + r_2 h)| \le 2^{\varepsilon} + 2^{\xi}(2^{\rho'} + 2^{\rho'}) \approx 2^{\xi+\rho'+1}$. Set circuit C be the circuit to be evaluated, then it can be expressed as a function f with g-variable and d-degree, for given plaintexts $m_1, m_2, \cdots, m_g$ and corresponding cipertxets $c_1, c_2, \cdots, c_g$, the value range of function f with g-variable and d-degree can be measured with an elementary symmetric polynomial.

$$|f(x_1, x_2, \ldots, x_g)| \le C_g^d M^d \le g^d M^d$$

In which $x_i \le M$.

Set $\mathrm{x_i} = \mathrm{m_i} + \mathrm{p}(\mathrm{r_{1i}} + \mathrm{r_{2i}x})$, then $\mathrm{M}\sim 2^{\varepsilon+\rho'+1}, \mathrm{s}\sim 2^{\eta}$, with $\mathrm{M}\sim 2^{\varepsilon+\rho'+1}\sim 2^{\alpha'(\rho'+2)}$, we get $\xi\sim(\alpha'-1)\rho'$, in which α' is a constant.

Therefore, the degree of evaluable polynomial needs to meet the following conditions.

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

B. FHE scheme

We have analyzed the variation of the noise after operation above. Since noise increases continuously after homomorphic operation and will eventually lead to the incorrect decryption of ciphertext, Gentry's Recrypt algorithm is applied to solve this problem. However, the application has an important prerequisite that the decryption circuit must be an evaluable circuit, that is, the degree of the decryption circuit must be less than the degree of evaluable polynomial. According to the parameters setting in this section, the maximum degree of evaluable polynomial in this somewhat scheme is $\alpha\lambda_{\varepsilon}\log^2\lambda_{\varepsilon}$, when α is a constant.

Having known the maximum degree of evaluable polynomial, we need to calculate the degree of decryption circuit and determine whether the latter is less than the former. The scheme will be bootstrappable if the answer is yes, and will also be a FHE scheme according to the following theorem, otherwise we need to squash the decrypt circuit.

Theorem 4.1 [2]. Any one of the bootstrappable homomorphic encryption schemes can be transformed into a full homomorphic encryption scheme.

We round up and adjust the result by judging its sign when deal with the decrypt circuit. Thus, the decrypt circuit above is decomposed into

$$Dec(c, K_s) = (c \bmod s) \bmod p = (c - s\lfloor c/s \rceil) \bmod p$$

Since $s = pt + 1$, there is $s \bmod p = 1$, so the circuit is changed into

$$Dec(c, K_s) = (c \bmod s) \bmod p = (c - \lfloor c/s \rceil) \bmod p$$

It is notable that the decryption circuit is very complex, mainly due to the large amount of calculation on $\lfloor c/s \rceil$, and the amount of calculation on $mod\ p$ is negligible because p is small. The complexity of the decryption circuit depends on the modular operation inside. c/s can be expressed as $\mathrm{c} \times \mathrm{s}^{-1}$, then it can be transformed into multiplication of two N-bits integers. According to the existing conclusion in [14]

Performing multiplication of two N-bits integers is equivalent to finding the sum of n binary numbers, the result can be expressed as a quadratic polynomial of the input bit length.

By applying Gentry's "three-for-two" method, sum of n binary numbers can be transformed into sum of 2 binary numbers after $\log_{3/2}$ n operations. Then finding sum of 2 n-bits integers need n times

of operations. Therefore, the result polynomial of the input bit length is at most $2 * 2^{log_{3/2} n} * n = 2 * 2^{(log n)/(log 3 - log 2)} = 2 * n^{1/(log 3 - log 2)} * n \approx 2 * n^{1.71} * n = 2n^{2.71}$ degree.

Note that $1/s$ needs to keep the $log\, s$ bits of precision to ensure the correctness while calculating c/s. So the final degree of c/s is $2log^{2.71}\, s \approx 2(\lambda_\varepsilon^2)^{2.71} = 2\lambda_\varepsilon^{5.42}$, which obviously exceeds the degree of evaluable polynomial, i.e., the decryption circuit is not included in the permitted circuit because of its complexity. The too large calculated amount of c/s is the root cause, so we need to solve the problem by dealing with the decryption circuit. The same method to squash the decryption circuit as in document [10] is applied here so that the degree of decryption polynomial is less than evaluable polynomial.

We need to introduce some other parameters to accomplish the squash.

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

$KeyGen_\varepsilon$. Choose at random a prime number $t \in [2^{\eta-\xi-1}, 2^{\eta-\xi})$, compute $s = pt + 1$, and set the secret key $K_s = s$ if $s/2$ is still a prime number, or, rechoose t until satisfying this constraint. Generate the public key $K_p = (N, x)$ as above. Set $x_s = [2^\kappa/s]$, choose at random a Θ-bit vector $\vec{s} = (s_1, s_2, \ldots, s_\Theta)$ with Hamming weight θ, and get a set $S = \{i | s_i = 1\}$ based on $\vec{s}$, then the number of non-zero elements in S is λ'_ε.Choose at random integers $u_i \in Z \cap (0, 2^{\kappa+1})$, $i = 1, \ldots, \Theta$, subject to the condition that $\sum u_i = x_p (mod\ 2^{\kappa+1})$, and let $y_i = u_i/2^k, \vec{y} = \{y_1, y_2, \ldots, y_\Theta\}$, in which

$$\left[\sum_{i\in S} y_i\right]_p = \left[\sum_{i\in S} \frac{u_i}{2^\kappa}\right]_p = [(\sum_{i\in S} u_i)/2^\kappa]_p = [(x_s\ mod\ 2^{\kappa+1})/2^\kappa]_p$$
$$= [([2^\kappa/s]\ mod\ 2^{\kappa+1})/2^\kappa]_p = [(2^\kappa/s)\ 2^\kappa]_p = [1/s]_p = (1/s) - |\Delta_s|$$

$Encrypt_\varepsilon$. Generate the ciphertext c^* from plaintext m as before. Then for $i = 1,2, \ldots, \Theta$, let $z_i \leftarrow [c^* \times y_i]_p$, keeping only $n = \lfloor log\, \Theta \rceil + 3$ bits of precision after the binary point for each z_i, output $\vec{c} = (c^*, z_1, z_2, \ldots, z_{\text{set}})$.

$Decrypt_\varepsilon$. Output $m' \leftarrow [c^* - \lfloor \sum_{i=1}^{\Theta} s_i z_i \rceil]_p$ with c^*,z_i and the secret key K_s.

$Evaluate$. For a given circuit C_ε, replace the addition and multiplication gates mod 2 in the original circuit with gates for addition and multiplication mod N, input ciphertexts $c_1, c_2, \cdots, c_g$. To ensure the correctness of decryption after operation, we need to extract the main ciphertext c^* before entering into the operation gates, get $c^{*\prime}$ by refreshing c^*, then apply the (integer) addition and multiplication gates of C_ε, output $c^{*\prime\prime}$, and then refresh $c^{*\prime\prime}$ if necessary, repeat this process until all the operations have been performed, output the result finally.

To reduce the computational complexity better, the decryption algorithm is divided into three steps according to the method in Reference [10].

- Compute$a_i = \sum s_i z_i$;
- Generate $m + 1$ rational numbers based on λ'_ε rational numbers $\{a_i\}_{i=1}^{\Theta}$ in the previous step;
- Compute and output $c^* - \sum w_j$.

The degree of polynomial in the three steps above is analyzed as follows.

- The degree of polynomial that correspond to Step 1 is 2.
- The degree of polynomial that correspond to Step 2 is λ'_ε.
- Step 3. The degree of $\sum w_j$ is $32\, log^2\, \lambda_\varepsilon$, let $\lambda_\varepsilon = \lambda'_\varepsilon$, then the degree of decryption circuit is approximately $2\lambda'_\varepsilon \cdot 32 \log^2 \lambda_\varepsilon = 64\lambda_\varepsilon \log^2 \lambda_\varepsilon$, and the degree of corresponding extended decryption circuit is approximately $128\lambda_\varepsilon \log^2 \lambda_\varepsilon$. Since $log\, g$ is very small for η, it can be ignored. To make sure that the decryption circuit is evaluable, there must be

$$128\lambda_\varepsilon \log^2 \lambda_\varepsilon \le \frac{\eta - 4}{\rho' + 2}$$

Let $\eta = \rho' \cdot 128\lambda_\varepsilon \log^2 \lambda_\varepsilon$, then the scheme can evaluate polynomials of degree up to $128\lambda_\varepsilon \log^2 \lambda_\varepsilon \le (\eta - 4)/(\rho' + 2)$. At this point, the decryption circuit belongs to permit circuits, and the scheme above is bootstrappable. According to theorem 4.1, this scheme can be transformed to a homomorphic encryption scheme, then we have the FHE scheme over F_p.

Verification of the Encryption Scheme. In order to verify the scheme above, a decimal number is encrypted and then recover from the ciphertext. Decimal number 1001, for example, 1111101001 in binary, needs to be encrypted 10 times according to the scheme Reference [10], while only one encryption is taken in this scheme. Let $p = 2237$, then the public key is

N=176065201490437834028558746467531808290665332054504591589017047515976152491
973531498999588958905529328495837707865989841782600199801705674294954959671425860171460541 153344939779098130169748959718637668106731795 1

z=261289035462370819399874365778192013946697495748967925881140723897132381152
422235780116678103076803968290149543483564624405697044885380070280710422001827185822759615730226600017429130843140716250257765487122383364947302058628947085250541050929643315183278848130157703898606944055082083738990470182127059262996457900405284925763373704825141691011138993475085133149325286616920886088132845772950425301944779132635497577048603966547000405414899901606784865214907210963714134528564161792703122645097111685183809791282979331980968121266096538997158609067875132063212266221894272642183972274269817034229222656515068492949892065699664468373755833126798043820671593333082889257162246840905359388447329607741601850265172685410559995984974741442165353979810181963024152204823297854164762009593678731061806962788863878037742313104580715410090716787854265260999663517992882519076930255542181495277646057488574076128554948258186535063317624732798978680721

The ciphertext corresponding to 1001 is

c=689947529242732533296416753279087811506227168592870446462867888441844102636777968050882208060967873068113077367630181952544741822453829172881876991054631780168469295836324501150102559549682842523391003226605851

The required time for encryption is 0.008 seconds, and the decryption time is 0.002 seconds. If we apply the scheme in literature [10], the ciphertext size will be 10 times of c, while the encryption time is about 10 times long as here, so we consider that the scheme has advantages on ciphertext size and the encryption and decryption time to the scheme in literature [10]. Data experimental environment in this section:

$c++$ 6.0 with NTL library
64-bit operating system
Installed memory (RAM):4.0GB
Processor: Intel® Core(TM) i5-2320 CPU @3.00GHz.

Conclusion

For cloud hosting of bank data and other application scenarios, this paper presents an improved scheme based on the DGHV scheme, which expands the plaintext space from prime field F_2 to F_p, and promotes the practicality of FHE. It is proved that the proposed scheme can correctly encrypt and decrypt data, and is more efficient than the existing DGHV scheme.

References

[1] Brenner M, Wiebelitz J, Von Voigt G, et al. Secret program execution in the cloud applying homomorphic encryption[C]// Digital Ecosystems and Technologies Conference (DEST), 2011 Proceedings of the 5th IEEE International Conference on. IEEE, 2011:114-119.

[2] Gentry C. A fully homomorphic encryption scheme[J]. Dissertations & Theses - Gradworks, 2009.

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

[4] Brakerski Z. Fully Homomorphic Encryption without Modulus Switching from Classical GapSVP[M]// Advances in Cryptology – CRYPTO 2012. Springer Berlin Heidelberg, 2012:868-886.

[5] Lyubashevsky V, Peikert C, Regev O. On ideal lattices and learning with errors over rings[C]. Proc ofthe 29th International Conference on Theory and Application of Cryptographic Techniques．Berlin:Springer,2010:1—23．

[6] Craig Gentry, Shai Halevi. Implementing Gentry's Fully-Homomorphic Encryption Scheme[C]// International Conference on Theory and Applications of Cryptographic Techniques: Advances in Cryptology. Springer-Verlag, 2011:129--148.

[7] Ogura N, Yamamoto G, Kobayashi T, et al. An Improvement of Key Generation Algorithm for Gentry's Homomorphic Encryption Scheme[M]// Advances in Information and Computer Security. Springer Berlin Heidelberg, 2010:70-83.

[8] Chen Y, Nguyen P Q. Faster Algorithms for Approximate Common Divisors: Breaking Fully-Homomorphic-Encryption Challenges over the Integers[M]// Advances in Cryptology – EUROCRYPT 2012. Springer Berlin Heidelberg, 2015:502-519.

[9] Brakerski Z, Gentry C, Vaikuntanathan V. (Leveled) Fully Homomorphic Encryption without Bootstrapping[J]. Acm Transactions on Computation Theory, 2011, 18(3):169--178.

[10] VanDijk M，Gentry C，Halevi S，et al．Fully homomorphic encryption over the Integers[C]. Proc of the 29th International Conference on Theory and Application of Cryptographic Techniques．Berlin：Springer，2010:24—43．

[11] J. H. Cheon, J.-S. Coron, J. Kim, M. S. Lee, T. Lepoint, M. Tibouchi, A. Yun: Batch Fully Homomorphic Encryption over the Integers. In: EUROCRYPT 2013,LNCS 7881, pp.315-335, 2013.

[12] JeanSébastien Coron, Avradip Mandal, David Naccache, et al. Fully Homomorphic Encryption over the Integers with Shorter Public Keys[C]// Conference on Advances in Cryptology. Springer-Verlag, 2011:487--504.

[13] Howgrave-Graham N. Approximate Integer Common Divisors[M]// Cryptography and Lattices. Springer Berlin Heidelberg, 2001:51-66

[14]. TANG Dianhua, ZHU Shixiong, CAO Yunfei. Faster fully homomorphic encryption scheme over integer. Computer Engineering and Applications, 2012, 48（28）：117-122.