International Journal of Cloud Computing and Services Science (IJ-CLOSER)
Vol.1, No.4, October 2012, pp. 200~209
ISSN: 2089-3337

More Practical Fully Homomorphic Encryption

Gu Chun-sheng*,, Wu Fang-sheng*, Jing Zheng-jun*,***, Yu Zhi-min***
* School of Computer Engineering, Jiangsu Teachers University of Technology, China Changzhou 213001
** School of Computer Science and Technology, University of Science and Technology of China, China Hefei 230027
*** School of Computer Science, Nanjing University of Posts and Telecommunications, Jiangsu Nanjing 210003

Article Info

Article history:

Received Augt 30th, 2012
Accepted Sept 23th, 2012

Keyword:

Fully Homomorphic Encryption
Polynomial Coset Problem
Factoring Integer
Diophantine Equation Problem
Approximate GCD

ABSTRACT

In this paper, we first modify the Smart-Vercauteren's fully homomorphic encryption scheme [1] by applying self-loop bootstrappable technique. The security of the modified scheme only depends on the hardness of the polynomial coset problem, removing the assumption of the sparse subset sum problem in the original paper in [1]. Moreover, we construct a non-self-loop in FHE by using cycle keys. Then, we further improve our scheme to make it be practical. The securities of our improving FHE's are respectively based on the hardness of factoring integer problem, solving Diophantine equation problem, and finding approximate greatest common divisor problem.

Corresponding Author:

Gu Chun-sheng,
School of Computer Engineering,
Jiangsu Teachers University of Technology,
1801 Zhongwu Main Road, Zhonglou District, Changzhou City, Jiangsu Province, China, 213001.
Email: guchunsheng@gmail.com

1. INTRODUCTION

In this work, we first present a fully homomorphic encryption scheme (FHE) by modifying the FHE in [1], which merely uses the elementary theory of algebraic number fields. Then, we improve this scheme to make it be practical by hiding the ideal lattice in the public key. The public key in their scheme consists of a prime p and an integer α modulo p. The private key is an integer s_0. To encrypt a bit m, one selects a small random polynomial $r(x)$, and output a ciphertext $c = (m + 2r(\alpha)) \bmod p$. To decrypt a ciphertext c, one computes the message bit as follows: $m = (c - \lfloor c \times s_0 / p + 0.5 \rfloor) \bmod 2$. To implement FHE, they introduce the hardness assumption of the sparse subset sum problem. Moreover, to obtain the FHE in [1], they must set lattices of dimension at least $n = 2^{27}$, which is beyond practical usability [2].

The aim of this paper is to remove the hardness assumption of SSSP in the Smart-Vercauteren's FHE [1], and improves our scheme to make it be practical. The advantage of their scheme is simple, since the public key in their SHE scheme merely implies two integers (p, α). However, it is well know that (p, α) forms a basis of an ideal lattice, and is equivalent to its corresponding HNF representation. So, when the dimension of an ideal lattice is small, one can find the secret key by using lattice reduction algorithm. This is the reason why the dimension of an ideal lattice n cannot be taken too small in their FHE.

1.1 Our Contribution

The difference between our scheme and their work is mainly located on fully homomorphic encryption scheme. We use the approach of re-randomizing the secret key to squash decryption polynomial, whereas they introduce the hardness assumption of SSSP to squash decryption polynomial. The security of

our scheme is only based on the hardness of the polynomial coset problem. What is more, we further construct several variant FHEs to make our scheme be practical. The securities of our variant schemes respectively depend on the hardness assumption of factoring integer problem, solving Diophantine equation problem, and approximate GCD problem. The start point of our work is to provide the public key of other forms to hide the public key (p,α).

1.2 Related work

Rivest, Adleman, and Dertouzos [3] first investigated a privacy homomorphism, which now is called the fully homomorphic encryption (FHE). Many researchers [4-7] have worked at this open problem. Until 2009, Gentry [8] constructed the first fully homomorphic encryption using ideal lattice. In Gentry's scheme, the public key is approximately n^7 bits, the computation per gate costs $O(n^6)$ operations. Smart and Vercauteren [1] presented a fully homomorphic encryption scheme with both relatively small key $O(n^3)$ bits , ciphertext size $O(n^{1.5})$ bits and computation per gate at least $O(n^3)$ operations, which is in some sense a specialization and optimization of Gentry's scheme. Dijk, Gentry, Halevi, and Vaikuntanathan [9] proposed a simple fully homomorphic encryption scheme over the integers, whose security depends on the hardness of finding an approximate integer GCD. Stehle and Steinfeld [10] improved Gentry's fully homomorphic scheme and obtained to a faster fully homomorphic scheme, with $O(n^{3.5})$ bits complexity per elementary binary addition/multiplication gate, but the hardness assumption of the security of the scheme in [10] is stronger than that in [8].

1.3 Organization

Section 2 recalls notations and related defintions, and gives Smart-Vercauteren's somewhat homomorphic encryption. Section 3 first transforms the somewhat homomorphic encryption into a FHE by applying self-loop bootstrappable technique, then constructs a non-self-loop FHE by using the method of cycle keys. Section 4 further improves our FHE to make it be practical. Section 5 gives a concrete implementation of our scheme. Section 6 concludes this paper.

2. Preliminaries

2.1 Notations

Let n be a security parameter, $[n]=\{0,1,...,n\}$. Let R be the ring of integer polynomials modulo $f(x)$, i.e., $R=Z[x]/f(x)$, where $f(x)$ is an integer monic and irreducible polynomial of degree n. For $f\in R$, we denote by $\|f\|_\infty$ the infinity norm of its coefficient vector, sometimes denoted $\|f\|$, $[f]_2$ the polynomial of its coefficient modulo 2. For R, its expansion factor γ_{mul} is n, that is, $\|u\times v\|_\infty \le n\cdot\|u\|_\infty\cdot\|v\|_\infty$, where $\times$ is multiplication in R.

2.2 Ideal Lattices

In this paper, we take $f(x)=x^n+1$ with n a power of 2. Let I be a principal ideal of R, namely, it only has a single generator. For the coefficient vector $\bar{u}=(u_0,u_1,...,u_{n-1})^T$ of $u\in R$, we define the cyclic rotation $rot(\bar{u})=(-u_{n-1},u_0,...,u_{n-2})^T$, and the corresponding circulant matrix $Rot(u)=(\bar{u},rot(\bar{u}),...,rot^{n-1}(\bar{u}))^T$. $Rot(u)$ is called the rotation basis of the ideal lattice (u). For $\forall f,u\in R$, $[f]_u$ is the coefficient vector of f modulo the rotation basis of u, namely, $\vec{f} \bmod Rot(u)$. So, we consider each element of R as being both a polynomial and a vector.

2.3 Smart-Vercauteren's Somewhat Homomorphic Encryption (SHE)

Smart and Vercauteren's in the SHE of [1] first select a principal ideal $u(x)$ with $u(x)=1 \bmod 2$ over the ring R such that the determinant of the circulant matrix of $u(x)$ is a prime. It is easy to verify that this ideal can be represented by either two integers (p,α), where $p=\det(Rot(u))$ and α is a common root of $u(x)$ and $f(x)$ modulo p. Then, they evaluate a polynomial $s(x)$ such that $u(x)\times s(x)=p \bmod f(x)$. They finally output the public key (p,α), the secret key $s_0=s(x) \bmod x$. To encrypt a message bit

$m \in \{0,1\}$, the scheme chooses a random small polynomial $r(x)$, computes its value at α, and outputs a ciphertext $c = Enc(m) \overset{def}{=} (2r(\alpha) + m) \bmod p$. Given a ciphertext c and the secret key s_0, the decryption algorithm deciphers the message bit $m = Dec(c) \overset{def}{=} (c - \lfloor c \cdot s_0 / p + 0.5 \rfloor) \bmod 2$. Given two ciphertexts c_1, c_2, addition operation $Add(c_1, c_2) \overset{def}{=} (c_1 + c_2) \bmod p$, and multiplication operation $Mul(c_1, c_2) \overset{def}{=} (c_1 \cdot c_2) \bmod p$.

3. Fully Homomorphic Encryption (FHE)

In this section, we firstly construct a self-loop FHE based on the Smart-Vercauteren's SHE, and simply analyze the security of our scheme. Then, we present a non-self-loop FHE by using the approach of cycle keys.

3.1 Construction of Self-loop FHE

Since our SHE is same as that of [1], we only need to give a new Recrypt algorithm, which freshens a 'dirty' ciphertext c into a new ciphertext c_{new} with the 'smaller' error term and the same plaintext of c. To do this, we generate the ciphertexts of the secret key and add them to the public key.

KeyGen Algorithm:

(1) Assume $f(x) = x^n + 1$ with n a power of 2. Choose a random polynomial $u(x)$ with $u(x) = 1 \bmod 2$ and $\|u(x)\|_\infty = 2^\eta$ such that $p = \det(Rot(u(x))) \approx 2^{n\eta}$ is a prime. Evaluate a common root α of $u(x)$ and $f(x)$ modulo p, and $s(x) = \sum_{i=0}^{n-1} s_i x^i \in Z[x]$ such that $u(x) \times s(x) = p \bmod f(x)$.

(2) Set $s_0 = s(x) \bmod x \bmod (2p)$.

(3) Choose at random an integer $s_{0,1} = \sum_{i=0}^{\lambda} z_i 2^i$ with $w = w(s_{0,1}) = \omega(\log n)$ and $\lambda = \lfloor \log(2p) \rfloor$, where $w(s_{0,1})$ is hamming weight of $s_{0,1}$, and take $s_{0,2} = s_0 - s_{0,1}$.

(4) Encrypt each bit z_i of $s_{0,1}$: $\bar{z}_i = Enc(z_i) = (2r_i(\alpha) + z_i) \bmod p$ such that $\|r_i(x)\|_\infty = n/2$ for each $r_i(x)$. Let $\bar{s}_{0,1} = \sum_{i=0}^{\lambda} \bar{z}_i 2^i$.

(5) Output the public key $pk = (n, p, \alpha, w, \bar{s}_{0,1}, s_{0,2})$, and the secret key $sk = (p, s_0)$.

All Enc, Dec, Add, Mul algorithms are same as those in [1].

Remark 3.1: To simplify our description, we take $u(x) = 1 \bmod 2$ same as that in [1]. In fact, we can select an arbitrary polynomial $u(x)$. In this case, we need to modify the above SHE scheme as follows. The public key is $pk = (n, p, \alpha, w, \overline{[u(x)]_2}, \{\bar{s}_{i,1}, s_{i,2}\}_{i=0}^{n-1})$ such that $s_i = s_{i,1} + s_{i,2}$ and $\overline{[u(x)]_2}$ is the polynomial of encrypted $u(x) \bmod 2$, the secret key is $sk = (p, u(x) \bmod 2, s(x))$. The decryption algorithm is changed into $m = c - \lfloor c * s(x) + 0.5h \rfloor \times [u(x)]_2$, where $h = \sum_{i=0}^{n-1} x^i$.

Remark 3.2: We may also select an odd composite number $p = \det(Rot(u(x)))$. To obtain root α, we need to factor p, then solve α_i for each prime factor by using extending GCD, and compute α by applying Chinese Remainder Theorem. So, we can choose many polynomial $u_i(x)$ with different prime $p_i = \det(Rot(u_i(x)))$, and set $u(x) = \prod u_i(x)$. The advantage of this scheme is easily to generate key.

Remark 3.3: It is obvious that our scheme can also use larger message space as that of [1].

Recrypting algorithm (Recrypt-1(*pk*, *c*)):

(1) Set $v_i = (c \times 2^i) / p$, $i \in [\lambda]$, keeping only $k = \lceil \log(w+1) \rceil + 3$ bits of precision after the binary point for each v_i.

(2) Evaluate $\bar{g}_i = v_i \times \bar{z}_i$ and $g_{0,2} = c \times s_{0,2} / p$, and sum $\lambda + 1$ encrypted rational numbers with $w + 1$ non-zero numbers, denoted as $\bar{x} = (g_{0,2} + \sum_{i=0}^{\lambda} \bar{g}_i) \bmod 2$.

(3) Assume $\bar{x} = \bar{x}_0 . \bar{x}_{-1} \bar{x}_{-2} \cdots \bar{x}_{-k}$. Evaluate $\bar{x}_{0,-1} = Add(\bar{x}_0, \bar{x}_{-1})$ and $u = c \bmod 2$.

(4) Output a new ciphertext $c_{new} = Add(u, \bar{x}_{0,-1})$.

Theorem 3.1. Recrypt algorithm correctly generates a 'fresh' ciphertext c_{new} with the same message of c, and supports a product of two 'fresh' ciphertexts when $t \le \frac{\eta - 1 + \log n}{8\log n + 2\log \eta}$, where $t = w + 1$.

Proof: By the Dec algorithm, we know

$$\begin{aligned}
&(c - \lfloor c \times s_0 / p + 0.5 \rfloor) \bmod 2 \\
&= c \bmod 2 + \lfloor c \times (s_{0,1} + s_{0,2}) / p + 0.5 \rfloor \bmod 2 \\
&= c \bmod 2 + \lfloor c / p \times s_{0,1} + c \times s_{0,2} / p + 0.5 \rfloor \bmod 2 \\
&= c \bmod 2 + \left\lfloor (c / p) \times \sum_{i=0}^{\lambda} z_i 2^i + c \times s_{0,2} / p + 0.5 \right\rfloor \bmod 2 \\
&= c \bmod 2 + \left\lfloor \sum_{i=0}^{\lambda} (c \times 2^i) / p \times z_i + c \times s_{0,2} / p + 0.5 \right\rfloor \bmod 2 \\
&= c \bmod 2 + \left\lfloor \sum_{i=0}^{\lambda} v_i \times z_i + c \times s_{0,2} / p + 0.5 \right\rfloor \bmod 2
\end{aligned}$$

Assume $\rho = \lambda + 2 \approx n\eta$. Since there are t non-zero encrypted rational numbers among ρ encrypted rational numbers, we can use symmetric polynomials with degree t to evaluate the sum of these rational numbers. The number of symmetric polynomials is at most $\binom{\rho}{t} \approx \rho^t$. The number of degree t monomials in the polynomial representing our rational addition algorithm is equal to $\binom{t}{\lceil t/2 \rceil} \times \binom{t}{\lceil t/4 \rceil} \times \ldots \times \binom{t}{1}$, which is less than t^t (see [2] for details). So, the polynomial $r(x)$ corresponding to a new generating ciphertext is satisfied to $\|r(x)\|_\infty \le \rho^t t^t n^{2^{\log t + 1} - 1} = \rho^t t^t n^{2t-1}$, where n is the infinity norm of the polynomial corresponding to $\vec{z}_i$. Moreover, to support one multiplication for two 'fresh' ciphertexts, we require $\|(r(x))^2 s(x) / p\|_\infty < 1/2$ according to [1]. Hence, we have $\|(r(x))^2\|_\infty \le (\rho t n^2)^{2t} / n \le 2^{\eta - 1}$, namely, $t \le \frac{\eta - 1 + \log n}{8\log n + 2\log \eta}$. ■

3.2 Security of FHE

For the semantic secure of our scheme, we follow the security analysis of [1]. The following definition is from that in [1].

Definition 3.1. (Polynomial Coset Problem (PCP) [1]). Given (r, pk), the problem is to guess whether $b = 0$ or $b = 1$, where r is computed from either $r = u(\alpha) \bmod p$ for $b = 0$, where $u(x)$ is a random polynomial with $\|u\| \le \beta$, or uniformly selected from $r \leftarrow_U F_p$ for $b = 1$.

Theorem 3.2. (Theorem 1 [1]). Suppose there is an algorithm A which breaks the semantic security of our SHE with advantage ε. Then there is a distinguishing algorithm D, which decides the PCP with advantage $\varepsilon / 2$.

3.3 Construction of Non-Self-loop FHE

According to [8], the above FHE can not prove to be semantically secure by a standard hybrid argument when using self-loop. In fact, the FHE in [1, 8] also reveals the encrypted secret key bits, although it is not direct. The advantage of applying cycle keys is to maximize possible distribution of the ciphertexts of encrypted secret key. In the following, we present a non-self-loop FHE by using method of cycle keys. But the drawback of our non-self-loop scheme is to require calling Recrypt two times to refresh ciphertext.

Non-self-loop-KeyGen Algortihm

(1) Call Step (1) of KeyGen in Section 3.1 two times to generate the public keys $pk_1 = (p_1, \alpha_1)$, $pk_2 = (p_2, \alpha_2)$ and the secret keys $sk_1 = (p_1, s^1)$, $sk_2 = (p_2, s^2)$.

(2) Set $s_0^1 = s^1 \bmod x \bmod (2p_1)$, and $s_0^2 = s^2 \bmod x \bmod (2p_2)$.

(3) Choose at random an integer $s_{0,1}^j = \sum_{i=0}^{\lambda} z_i^j 2^i$, $j = 1, 2$ with $w = w(s_{0,1}^j) = \omega(\log n)$ and $\lambda = \max\{\lfloor \log(2p_1) \rfloor, \lfloor \log(2p_2) \rfloor\}$, and take $s_{0,2}^j = s_0^j - s_{0,1}^j$.

(4) Encrypt $s_{0,1}^1$ under $pk_2 = (p_2, \alpha_2)$ as $\vec{z}_i^1 = Enc(z_i^1) = (2r_i(\alpha_2) + z_i^1) \bmod p_2$ with random $r_i^1(x)$, and $s_{0,1}^2$ under $pk_1 = (p_1, \alpha_1)$. Let $\vec{s}_{0,1}^j = \sum_{i=0}^{\lambda} \vec{z}_i^j 2^i$.

(5) Output the public key $pk^* = \{n, p_j, \alpha_j, w, \vec{s}_{0,1}^j, s_{0,2}^j\}_{j=1}^2$, and the secret key $sk = \{p_j, s_0^j\}_{j=1}^2$.

Assume we use $pk_1 = (p_1, \alpha_1)$ as the public key when encrypting message. To refresh a ciphertext c, we first call Recrypt with c to generate an intermediate ciphertext c_1 under pk_2, then again call Recrypt with c_1 to obtain a new ciphertext c_{new} under pk_1.

4. Improvement of FHE

In the following, we only discuss how to improve the self-loop FHE scheme in Section 3.1. Indeed, it is not difficult to verify that our improvement scheme can be transform into the corresponding non-self-loop scheme.

Since the public key in our FHE is $pk = (n, p, \alpha, w, \vec{s}_{0,1}, s_{0,2})$, one can construct a principal ideal lattice with HNF form by (p, α). So, the dimension n of an ideal lattice must be set large enough to guarantee the security of FHE. As a result, for practical values of n, the origin scheme [1] can not be made FHE. The start point of our work is to hide the parameters (p, α) to avoid the lattice reduction attack. Hence, we can choose a small or even a constant n to make our scheme be practical. To describe simplicity, in this section we denote by τ security parameter, n the dimension of an ideal lattice.

4.1. FHE Based on Factoring Integer Problem

To hide integer α, we substitute α by the ciphertexts $\{b_i = Enc(0)\}_{i=0}^{O(\tau)}$ of many 0-bits, namely, the public key becomes $pk = (n, p, w, \{b_i\}_{i=0}^{O(\tau)}, \vec{s}_{0,1}, s_{0,2})$. To attack this scheme, one can factor $x^n + 1 \bmod p$, then guess α among n roots of $x^n + 1 \bmod p$. However, we observe that $x^n + 1 \bmod p$ can be efficiently factored only when p is a prime or has been factored. So, if we take $q = pq_0$ such that $p = \det(Rot(s(x)))$ is a prime, q_0 is another prime, then one cannot factor $x^n + 1$ when hidden modulo p.

KeyGen-1 Algorithm:

(1) Generate (p, α) and s_0 as KeyGen in Section 3.1.

(2) Choose a random prime q_0, and take $q = pq_0$ with $|p| \approx |q_0|$, where $|y|$ is the length of y.

(3) Encrypt $\theta = O(\tau)$ 0-bits: $b_i = 2r_i(\alpha) \bmod q$ with $\|r_i(x)\|_\infty = 2^{\tau - 1}$.

(4) Choose a random fraction $s_{0,1} = \sum_{i=0}^{\lambda} z_i 2^{-i}$ with $w(s_{0,1}) = \omega(\log \tau)$ and $\lambda = \lceil \log q \rceil$, and set $s_{0,2} = \lfloor s_0 / p - s_{0,1} \rfloor_2$, keeping only λ bits of precision after the binary point, and encrypt each bit z_i as $\vec{z}_i = (2r_i(\alpha) + z_i) \bmod q$ with $\|r_i(x)\|_\infty = 2^{\tau - 1}$. Let $\vec{s}_{0,1} = \sum_{i=0}^{\lambda} \vec{z}_i 2^{-i}$.

(5) Output the public key $pk = (n, q, w, \{b_i\}_{i=0}^{\theta}, \vec{s}_{0,1}, s_{0,2})$ and the secret key $sk = (p, s_0)$.

Encryption Algorithm (Enc). Given the public key pk and a message bit $m \in \{0,1\}$, choose a small random subset $T \subset [\theta]$ and a random integer $|e| < 2^{\tau - 1}$, output a ciphertext $c = (\sum_{i \in T} b_i + 2e + m) \bmod q$.

The Dec, Mul, Add algorithms are identical to that in Section 3.1 except with replacing p with q. The Recrypt algorithm is modified into $c_{new} = \lfloor c \vec{s}_{0,1} + c s_{0,2} + 0.5 \rfloor + \lfloor c \rfloor_2$.

Correctness and Security: One can easily check that this scheme is correctness. To break this scheme, one first considers to factor $q = pq_0$ and $x^n + 1 \bmod p$, then guess α among n roots and solve $u(x)$ and $s(x)$. However, as far as I know, there is not an efficient algorithm which given q, factors $x^n + 1$.

Theorem 4.1. Given $pk = (n, q = pq_0, w, \{b_i\}_{i=0}^{\theta}, \vec{s}_{0,1}, s_{0,2})$, suppose factoring $x^n + 1$ is hard. Then our scheme based on factoring integer problem is semantic secure when $n \geq 2$.

Remark 4.1: For $n = 2$, there is an interesting example. It is well known that there is $p = a^2 + b^2$ for a prime $p = 1 \bmod 4$. So, for $f(x) = x^2 + 1$ and a prime $p = 1 \bmod 4$ large enough, we can set

$u(x)=ax+b$ with $\det(Rot(u(x)))=p$, and construct a scheme based on factoring integer problem. This special example can also adapt to our following schemes.

4.2. FHE Based on Diophantine Equation Problem

There is an efficient quantum algorithm which factors integers [11]. So in this subsection, we construct a new variant of our scheme, whose security depends on the hardness assumption of solving Diophantine equation problem.

4.2.1 Costruction

KeyGen-2 Algorithm:

(1) Choose a random polynomial $u(x)$ such that $p=\det(Rot(u(x)))\geq 2^{n\eta}$ is a prime, $u(x)=1 \bmod 2$, and $\|u(x)\|_\infty=2^\eta$. Evaluate a common root α of $u(x)$ and x^n+1 under modulo p, and $s(x)=\sum_{i=0}^{n-1}s_i x^i \in Z[x]$ such that $u(x)\times s(x)=p \bmod (x^n+1)$.

(2) Select a list of integers $d_j=2r_j(\alpha) \bmod p^2$ such that $d_j/p\approx 2^{\tau(j+1)}$ for $j=[\mu]$, $\mu=\lceil(\log p)/\tau\rceil-1$, where $\|r_j(x)\|_\infty\leq 2^{\tau-1}$. Recall that d_j is a ciphertext of 0-bit.

(3) Encrypt a list of 0-bits: $b_i=2r_i(\alpha) \bmod p$ such that $r_i(x)$ is a random polynomial, $\|r_i(x)\|_\infty\leq 2^{\tau-1}$, b_0 is an odd integer, and $|b_i|\leq|b_0|$ for all $i=[\theta], \theta=O(\tau)$.

(4) Choose a random fraction $s_{0,1}=\sum_{i=0}^{\lambda}z_i 2^{-i}$ with $w(s_{0,1})=\omega(\log\tau)$ and $\lambda=\lfloor\log 2p\rfloor+2$, and set $s_{0,2}=[s_0/p-s_{0,1}]_2$, keeping only λ bits of precision after the binary point. Encrypt each bit z_i of $s_{0,1}$: $\bar{z}_i=Enc(z_i)=(2r_i(\alpha)+z_i) \bmod p$ with $\|r_i(x)\|_\infty\leq 2^{\tau-1}$. Let $\bar{s}_{0,1}=\sum_{i=0}^{\lambda}\bar{z}_i 2^{-i}$.

(5) Output the public key $pk=(n,w,\{d_j\}_{j=0}^{\mu},\{b_i\}_{i=0}^{\theta},\bar{s}_{0,1},s_{0,2})$ and the secret key $sk=(p,s_0)$.

Remark 4.2: To generate d_j, we first choose at random a list of integers $d_j=2r_j(\alpha) \bmod p^2$, and remain all qualified d_j, and then for other non-qualified d_j, evaluate $d_j=2r_j(\alpha) \bmod p+q_j p$, where $q_j\approx 2^{\tau(j+1)}$. In fact, all d_j can be obtained by computing $d_j=2r_j(\alpha) \bmod p+q_j p$. We observe that d_j, $j=[\mu]$ do not reveal any information about p except with the length of p, if suppose $2r_j(\alpha) \bmod p$ is distinguishing from the uniform distribution over the set $[p]$.

Remark 4.3: In the KeyGen-2 algorithm, we can also replace p by $q=q_0 p$ with an odd integer q_0 when computing $b_i=2r_i(\alpha) \bmod p$ to further hide modulo p.

Encryption Algorithm (Enc). Given the public key pk and a message bit $m\in\{0,1\}$, choose a small random subset $T\subset[\theta]$ and a random integer $|e|<2^{\tau-1}$, output a ciphertext $c=(\sum_{i\in T}b_i+2e+m) \bmod b_0$.

Add Operation (Add). Given the public key pk, and two ciphertexts c_1,c_2, evaluate a ciphertext $c=(c_1+c_2) \bmod b_0$.

Multiplication Operation (Mul). Given the public key pk and two ciphertexts c_1,c_2, evaluate a new ciphertext $c=(c_1\times c_2) \bmod d_\mu \bmod d_{\mu-1}\ldots \bmod d_0 \bmod b_0$, denoted as $c=Opt(c_1\times c_2)$.

Recall here that the quotient of each optimization is at most 2^τ, that is, each optimization only increase the coefficient of the polynomial corresponding to a ciphertext at most $2^{2\tau}$.

Decryption Algorithm (Dec). Given the secret key sk and a ciphertext c, decipher the message bit $m=(c-\lfloor c\times s_0/p+0.5\rfloor) \bmod 2$.

Recrypting algorithm (Recrypt-2(*pk*, *c*)).

(1) Evaluate $\bar{g}_i=c\times\sum_{i=1}^{\lambda}\bar{z}_i 2^{-i}$ and $g_{0,2}=c\times s_{0,2}$, and sum $\lambda+1$ encrypted rational numbers with $t=w+1$ non-zero rational numbers: $\bar{x}=(g_{0,2}+\sum_{i=0}^{\lambda}\bar{g}_i) \bmod 2$.

(2) Assume $\bar{x}=\bar{x}_0.\bar{x}_{-1}\bar{x}_{-2}\cdots\bar{x}_{-k}$. Evaluate $\bar{x}_{0,-1}=Add(\bar{x}_0,\bar{x}_{-1})$ and $u=c \bmod 2$.

(3) Output a new ciphertext $c_{new}=Add(u,\bar{x}_{0,-1})$.

4.2.2 Correctness

According to [1], it is not difficult to verify that KeyGen, Enc, Add, Dec Algorithms are correct. For Mul algorithm, the reason we use d_j is to reduce length of ciphertext to a fixed length, and remain the infinity norm of the polynomial corresponding to new generating ciphertext to be controllable small.

Now, we determine for what parameters all above algorithms are correct. First, we know that the infinity norm in each polynomial corresponding to b_i, d_j is at most 2^τ according to KeyGen algorithm. For the Enc algorithm, the polynomial $r'(x)$ corresponding to $c' = (\sum_{i\in T} b_i + 2e + m)$ is satisfied to $\|r'(x)\|_\infty \le |T| \times 2^\tau + 2^\tau = \theta 2^\tau$, and taking modulo b_0 increases the norm of $r'(x)$ at most $\theta 2^\tau$. Since $c = (\sum_{i\in T} b_i + 2e + m) - \sigma b_0$ with $|\sigma| < \theta$, hence the infinity norm of $r(x)$ corresponding to $c = (\sum_{i\in T} b_i + 2e + m) \bmod b_0$ is at most $\theta 2^{\tau+1}$. For the Dec algorithm, we know that if $\|r(x) \times s(x) / p\|_\infty < 1/2$, namely, $\|r(x)\|_\infty < 2^{\eta-1}$ according to [1], Dec decrypts will be correct.

For the Add operation, $r(x)$ corresponding to $c = (c_1 + c_2) \bmod b_0$ is subject to $\|r(x)\|_\infty \le 3 \times 2^\tau$. For the Mul operation, we use the optimization technique in [9]. Without loss of generality, assume that $r_i(x), i = 1,2$ are corresponding to the polynomial of ciphertext c_i and satisfied to $\|r_i(x)\| \le 2^\kappa$. Because $c' = c_1 \times c_2 \le b_0^2 < p^2$, we need to evaluate modulo d_j at most $\mu = \lceil n\eta / \tau \rceil$ times, and the infinity norm of polynomial related to ciphertext c' increases at most $2^{2\tau}$ for each time. So, the polynomial $r(x)$ corresponding to $c = Opt(c_1 \times c_2)$ is satisfied to $\|r(x)\|_\infty \le n \times 2^{2\kappa} + 2^{2\tau} \times n\eta / \tau$. When $\kappa = \tau, \eta \le \tau^2$, $\|r(x)\|_\infty \le n(\tau+1)2^{2\tau} < 2^{3\tau}$.

Theorem 4.2. Recrypt-2 algorithm correctly generates a 'fresh' ciphertext c_{new} with the same message of c, and supports a product of two 'fresh' ciphertexts when $t \le \dfrac{\eta - 1 + \tau}{6\tau + 2\log\eta}$, where $t = w + 1$ and $\tau \ge \log n + \log t$.

Proof: By using same method in the proof of Theorem 3.1, we can obtain $\|(r(x))^2\|_\infty \le (\rho^t t^t (2^\tau)^{2t-1})^2 \le (\rho t (2^\tau)^2)^{2t} / 2^\tau \le 2^{\eta-1}$. So, we have $t \le \dfrac{\eta - 1 + \tau}{6\tau + 2\log\eta}$. ■

4.2.3 Security

In our scheme, we hide (p, α) by replacing it with the ciphertexts of many 0-bits. So, the security of our scheme relies on the following hidden polynomial coset problem.

Definition 4.1 (Hidden Polynomial Coset Problem (HPCP)) The challenger generates the public key $pk = (n, w, \{d_j\}_{j=0}^{\mu}, \{b_i\}_{i=0}^{\theta}, \vec{s}_{0,1}, s_{0,2})$ and chooses a random bit $\beta \leftarrow_U \{0,1\}$. If $\beta = 0$ then the challenger calls Enc algorithm to generate $c = (\sum_{i\in T} b_i + 2e) \bmod b_0$. If $\beta = 1$ then the challenger select a random number $c \leftarrow_U [b_0]$. Given (c, pk), the problem is to guess whether $\beta = 0$ or $\beta = 1$.

Indeed, the above HPCP is equivalent to the following Diophantine equation problem.

Definition 4.2 (Diophantine Equation Problem (DEP)) Given $(c, \{d_j\}_{j=1}^{\mu}, \{b_i\}_{i=0}^{\varphi})$, we construct a Diophantine equation system as follows:

$$\begin{cases} 2\sum_{k=0}^{n-1} r_{j,k} x^k - q_j p - d_j = 0, \ j = 1, \ldots, \mu \\ 2\sum_{k=0}^{n-1} r_{i,k} x^k - t_i p - b_i = 0, \ i = 1, \ldots, \varphi \\ 2\sum_{k=0}^{n-1} c_k x^k - vp - c = 0 \end{cases} \quad (1)$$

Its equivalent formula is:

$$\sum_{j=1}^{\mu} (2\sum_{k=0}^{n-1} r_{j,k} x^k - q_j p - d_j)^2 + \sum_{i=0}^{\varphi} (2\sum_{k=0}^{n-1} r_{i,k} x^k - t_i p - b_i)^2 + (2\sum_{k=0}^{n-1} c_k x^k - vp - c)^2 = 0 \quad (2)$$

The problem is to decide whether there is a solution in integers for the equation (1) or (2), such that $|r_{j,k}| \le \delta, |r_{i,k}| \le \delta, |c_k| \le \delta, \delta = O(\sqrt[n]{b_0})$.

For an arbitrary polynomial equation with integer coefficients, called Hilbert's Tenth Problem, this problem is undecidable [12]. Since the Diophantine equation problem we define is a bounded version problem by using the public key pk. So, it is decidable and in NP. An interesting open problem is to determine the hardness of the above DEP problem.

Theorem 4.3. Suppose there is an algorithm A which breaks the semantic security of our SHE with advantage ε. Then there is an algorithm D for solving HPCP and DEP with advantage at least $\varepsilon/2$. The running time of D is polynomial in the running time of A, and $1/\varepsilon$.

Proof: The proof of theorem is same as the proof of Theorem 1 in [1], except with substitute the modulo p with b_0.■

4.2.4 Extension of Large Message Space

We now extend our scheme to support large message space as that in [1]. We first use KeyGen-2 to get the public key $pk = (n, w, \{d_j\}_{j=0}^{\mu}, \{b_i\}_{i=0}^{\varphi}, \vec{s}_{0,1}, s_{0,2})$ and the secret key $sk = (p, s)$, and generate $(\vec{s}_{j,1}, s_{j,2})$ as $(\vec{s}_{0,1}, s_{0,2})$. Then we encrypt the ciphertexts of n 0-bits $l_j^{'} = 2r_j(\alpha) \bmod p$, and compute $l_j = (l_j^{'} + \alpha^j) \bmod p$. Finally, we output the public key $pk = (n, w, \{d_j\}_{j=0}^{\mu}, \{b_i\}_{i=0}^{\varphi}, \{\vec{s}_{j,1}, s_{j,2}\}_{j=1}^{n-1}, \{l_j\}_{j=0}^{n-1})$ and the secret key $sk = (p, s)$. Now Given the public key pk and a message $m \in \{0,1\}^n$, Enc choose a small random subset $T \subset [\theta]$ and a random integer $|e| < 2^{\tau-1}$, evaluate a ciphertext $c = (\sum_{i \in T} b_i + 2e + \sum_{j=0}^{n-1} l_j m_j) \bmod b_0$. According to analysis in [1], we decipher the message $m = (c - \lfloor c \times s / p + 0.5 \rfloor) \bmod 2$. We observe that there is a minor error for the decryption algorithm of Section 6 in [SV10]. When refreshing ciphertext, we first get a ciphertext c_j of each bit of message m by applying Recrypt algorithm, then evaluate $c_{new} = (\sum_{j=0}^{n-1} Opt(c_j \times l_j)) \bmod b_0$. When performing homomorphic operations, we first obtain each encrypted bit of m, then perform appropriately homomorphic operations for each bit, and finally combine each encrypted bits into a ciphertext of n bits message by using same approach of computing c_{new}.

By using similar approach, it is not difficult to verify that all schemes in this paper support large message space.

4.3 FHE Based on Approximate GCD

For the FHE based on factoring integer problem, we may replace $q = pq_0$ with a list of approximate multiple integers of p. So, we design a variant scheme of FHE, whose security is based on the hardness of approximate GCD problem. Different from the scheme in [vDGHV10], our scheme has larger message space.

Indeed, this scheme is a special case in Section 4.2. Namely, we can choose a constant polynomial $2r_j(x) = 2e_j$ and take $d_j = pq_j + 2e_j$, when computing $d_j = 2r_j(\alpha) \bmod p^2$. If we further substitute $\{b_i\}_{i=0}^{\theta}$ by $\{b_i = q_i p + 2e_i\}_{i=0}^{\theta}$, then this scheme becomes the scheme of [9]. Here we omit other details.

Definition 4.4. (Approximate-GCD over the Integers (AGCD)) Given a list of approximate multiples of p: $\{d_i = pq_i + e_i : a_i, e_i \in Z, s.t. |e_i| < 2^{\tau-1}\}_{i=0}^{\mu}$, find p.

Theorem 4.4. (Theorem 4.2 [9]) Suppose there is an algorithm A which breaks the semantic security of our SHE with advantage ε. Then there is an algorithm D for solving AGCD with advantage at least $\varepsilon/2$. The running time of D is polynomial in the running time of A, and $1/\varepsilon$.

5 Implementation

For simplicity, we discuss a concrete implementation of the FHE in Section 4.2. Take $n = 4$, $\tau = 50$, $w = 14$, $\eta = 2200$, $\lceil \log p \rceil = 8800$, $\lambda = 8192$, $\varphi = 1024$, $\mu = 200$. We now analyze how to implement our FHE and its security.

For $s_{0,1} = \sum_{i=1}^{\lambda} z_i 2^{-i}$ and encrypted $\vec{s}_{0,1}$, assume we can continuously separate it to 1023 groups, each group has 8 binary bits, but at most single 1-bit in them, and only $w = 14$ groups with single 1-bit among these groups.

To refresh ciphertext c, we evaluate $\vec{g}_1 = c \times \sum_{i=1}^{\lambda} \vec{z}_i 2^{-i} = \sum_{j=0}^{\lambda/8-1} (\sum_{i=1}^{8} c \times 2^{-(8j+i)} \vec{z}_{8j+i})$ and $g_2 = c \times s_{0,2}$ to get 1024 encrypted rational numbers with $w+1=15$ non-zero numbers. Hence, we can apply symmetric polynomial technique to sum these encrypted rational numbers. According to analysis of [GH10], it is not hard to verify that the decryption polynomial is about $2^{34} \times 1024^{15} = 2^{184}$ degree-15 monomials. What is more, we need to support a product of two refreshing ciphertexts, our scheme requires to evaluate polynomials with 2^{368} degree-30 monomials.

Since $2r_i(x)$ corresponding to encrypted z_i is satisfied to $\|2r_i(x)\|_\infty \le 2^\tau = 2^{50}$, so $r(x)$ corresponding to the ciphertext of each bit of $\vec{g}_{1,j}$ is satisfied to $\|r(x)\|_\infty \le 2^{50} \times 10 < 2^{54}$. It is easy to verify that the infinity norm of a degree-32 monomial is at most 2^{1821}. So, $\log\|u(x)\|_\infty > 1821 + 368 = 2189$. Thus, $\eta = \log\|u(x)\|_\infty = 2200$ is feasible for our scheme. When taking above parameters, the expansion rate of ciphertext in our scheme is about $8800 / 4 = 2200$.

6 CONCLUSION

By using self-loop bootstrappable technique, we modify the fully homomorphic encryption scheme in [SV10], whose security only depends on the hardness of the polynomial coset problem, removing the assumption of the sparse subset sum problem. Then to obtain better performance, we construct three variant schemes based on hardness assumption of different problems. In addition, we assume our scheme is KDM-secure, since the public key in our scheme implies the ciphertexts $\vec{s}$ of the secret key s to implement FHE.

ACKNOWLEDGEMENTS

This work is supported by National Natural Science Foundation of China Grants (No. 61142007), by Foundation of Jinagsu Province 'Qinglang Project' (No.KYQ09002), and by Foundation of Jiangsu Teachers University of Technology (No. KYY11055).

REFERENCES

[1] Nigel P. Smart and Frederik Vercauteren. Fully homomorphic encryption with relatively small key and ciphertext sizes. PKC 2010, LNCS 6056, pp. 420–443.
[2] Craig Gentry and Shai Halevi. Implementing Gentry's fully-homomorphic encryption scheme. Eurocrypt 2011, LNCS 6632, pp. 129–148.
[3] R. Rivest, L. Adleman, and M. Dertouzos. On data banks and privacy homomorphisms. In Foundations of Secure Computation, pp. 169-180, 1978.
[4] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-DNF formulas on ciphertexts. LNCS, 2005, Volume 3378, pp. 325-341, 2005.
[5] C. Aguilar Melchor, G. Castagnos, and G. Gaborit. Lattice-based homomorphic encryption of vector spaces. In IEEE International Symposium on Information Theory, ISIT'2008, pp. 1858-1862, 2008.
[6] T. Sander, A. Young, and M. Yung. Non-interactive CryptoComputing for NC1. In 40th Annual Symposium on Foundations of Computer Science (FOCS '99), pp. 554-567, 1999.
[7] A. C. Yao. Protocols for secure computations (extended abstract). In 23rd Annual Symposium on Foundations of Computer Science (FOCS '82), pp. 160-164, 1982.
[8] C. Gentry. Fully homomorphic encryption using ideal lattices. STOC 2009, pp. 169-178, 2009.
[9] M. van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan. Fully homomorphic encryption over the integers. Eurocrypt 2010, LNCS 6110, pp. 24-43.
[10] D. Stehle and R. Steinfeld. Faster Fully Homomorphic Encryption. Asiacrypt 2010, LNCS 6477, pp. 377-394.
[11] Shor P. W., Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM Journal on Computing 26(5), 1484–1509 (1997); Extended abstract in FOCS 1994 (1994).
[12] M. Davis. Hilbert's tenth problem is unsolvable. American Mathematical Monthly, 80:233–269, 1973.

BIOGRAPHY OF AUTHORS

Gu Chun-sheng received his Ph.D. Degree from University of Science and Technology of China in 2005. Since 2008 he has been an associate professor in the School of Computer Engineering, Jiangsu Teachers University of Technology. His research interests are in the cryptanalysis and design of cryptography.