Multi-integer Somewhat Homomorphic Encryption Scheme with China Remainder Theorem

CHAO FENG[1,2,*], YANG XIN[2], YIXIAN YANG[2], HONGLIANG ZHU[2]
[1]School of Information Science and Engineering
Shandong University
Shanda South Road 27, Jinan, Shandong
CHINA
[*]chaojuan99@hotmail.com
[2]National Engineering Laboratory for Disaster Backup and Recovery
Beijing University of Posts and Telecommunications
Xitucheng Road 10, Beijing
CHINA

Abstract: - As an effective solution to protect the privacy of the data, homomorphic encryption has become a hot research topic. Existing homomorphic schemes are not truly practical due to their high computational complexity and huge key size. In 2013, Coron *et al*. proposed a batch homomorphic encryption scheme, i.e. a scheme that supports encrypting and homomorphically evaluating several plaintext bits as a single ciphertext. Based on china remainder theorem, we propose a multi-integer somewhat homomorphic encryption scheme. It can be regarded as a generalization of Coron's scheme with larger message space. Furthermore, we put forward a new hardness problem, which is called the random approximation greatest common divisor (RAGCD). We prove that RAGCD problem is a stronger version of approximation greatest common divisor (AGCD) problem. Our variant remains semantically secure under RAGCD problem. As a consequence, we obtain a shorter public key without sacrificing the security of the scheme. The estimates are backed up with experiment data. It is expected that, the proposed scheme makes the encrypted data processing practical for suitable applications.

Key-Words: - Information Security, Cryptography, Somewhat Homomorphic Encryption, Multi-Integer, China Remainder Theorem, Random Approximation Greatest Common Divisor (RAGCD)

1 Introduction

In 1978, Rivest *et al.* introduced the basic concept of homomorphic encryption that allowed computation on ciphertexts without decryption [1]. Shortly after its publication, major security problems were found in the original scheme. In the past thirty years, many additively or multiplicatively homomorphic encryption schemes were proposed by the researchers, unfortunately, none of them could supports both addition and multiplication on ciphertexts simultaneously. During this period, the best result was the Boneh–Goh–Nissim cryptosystem, which supports evaluation of an unlimited number of addition operations but at most one multiplication [2]. In 2009, Gentry came up with the first fully homomorphic encryption scheme, i.e. a scheme that supports both addition and multiplication on ciphertexts simultaneously [3, 4]. First, Gentry constructed a somewhat homomorphic scheme, which only supports a limited number of multiplications. The second step of Gentry's framework consisted in squashing the decryption procedure so that it could be expressed as a low degree polynomial in the bits of the ciphertext and the secret key. Then, a key idea, called "bootstrapping", was to evaluate this decryption polynomial homomorphically on the encryption of the ciphertext bits and the secret-key, which given another ciphertext of the same plaintext. If the degree of the decryption polynomial was small enough, the noise in the new ciphertext was smaller than that in the original ciphertext. So this new ciphertext could be used again in a subsequent homomorphic operation (either addition or multiplication). Using this "ciphertext refresh" procedure the number of permissible homomorphic operations became unlimited and a fully homomorphic encryption scheme can be obtained. However, Gentry's scheme involved a relatively untested hardness assumption, e.g., the hardness of problems on ideal lattices. Soon after Gentry's original paper appeared, Smart and Vercauteren presented a refinement of Gentry's scheme giving shorter public key size and ciphertext size, but

which was still not practical [5]. One obstacle in Smart's scheme was the complexity of key generation of the somewhat homomorphic scheme. Gentry and Halevi presented an optimized version [6]. In particular, the optimized version had an efficient key generation procedure and a simpler decryption circuit. Stehle also proposed an improved scheme of Gentry's scheme, which introduced decryption errors to reduce computation cost [7]. Meanwhile, Ogura proposed a somewhat homomorphic encryption with short public key size [8]. Based on multilinear mapping, Garg proposed a homomorphic encryption that was based on ideal lattice [9].

Different from Gentry's "blueprint", a homomorphic encryption scheme based on learning with error (LWE) assumption was proposed by Brakerski and Vaikuntanathan, which does not contain complex operations on ideals lattice [10]. The security of Brakerski's scheme was reduced to the worst-case hardness of "short vector problems" on arbitrary lattices (rather than ideal lattice). Soon afterwards, Brakerski proposed a radically new approach to fully homomorphic encryption that dramatically improves performance and bases security on weaker assumptions [11]. A central conceptual contribution in this work was a new way of constructing leveled fully homomorphic encryption schemes (capable of evaluating arbitrary polynomial-size circuits), without Gentry's bootstrapping procedure. Meanwhile, Gentry proposed some efficient homomorphic schemes based on LWE [12-15]. Lopez proposed a multi-key homomorphic filtering scheme, which was useful for multi-party computation [16].

In 2010, Dijk proposed a somewhat homomorphic encryption scheme (DGHV scheme), which was a variant of Gentry's scheme and relied purely on the arithmetic of the integers [17]. The main appeal of the scheme (compared to the original Gentry's scheme) was its conceptual simplicity; namely all operations were done over the integers instead of ideal lattices. Coron presented an optimized version of the DGHV scheme [18]. The public key size of the scheme was reduced to $O(\lambda^7)$. Based on china remainder theorem, Kim *et al*. proposed a homomorphic encryption with message space $\mathbf{Z}_{2^\lambda}$ instead of $\mathbf{Z}_2$, and the public key size was $O(\lambda^{10})$ [19]. In 2013, Coron *et al*. proposed a batch homomorphic encryption scheme that supports encrypting and homomorphically processing a vector of bits as a single ciphertext [20]. Recently, Coron *et al*. proposed a variant of the DGHV scheme with the same scale-invariant property [21]. In [21], Coron *et al*. proved the equivalence between the (error-free) decisional AGCD problem and the classical computational AGCD problem. This equivalence allowed to get rid of the additional noise in all the integer-based fully homomorphic encryption schemes.

Although any computation can be expressed as a Boolean circuit, we need to design an efficient homomorphic encryption scheme with larger message space. In this paper, we propose a multi-integer somewhat homomorphic encryption scheme with shorter key size. Besides, the message space is $\mathbf{Z}_{\min(n_i)}$ instead of $\mathbf{Z}_2$.

1.1 Comparisons with related works

In this section, we compare our scheme with some related works. Dijk *et al*. proposed a homomorphic encryption scheme [17]. The scheme was conceptually simpler than Gentry's scheme, because it operated on integers instead of ideal lattices. The size of public key was $O(\lambda^{10})$ which was too large for any practical application. It was shown in [18] how to reduce the public key size by storing only a small subset of the original public key and generating the full public key on the fly by combining the elements in the small subset multiplicatively. The public key size was reduced to $O(\lambda^7)$. However, the improvement came at the expense of the overall complexity. Based on china remainder theorem, Kim *et al*. described a multi-bit homomorphic scheme [19]. The ciphertext expansion rate and the overall complexity were improved. Unfortunately, the public key was $O(\lambda^{10})$. Coron *et al*. extended the fully homomorphic encryption scheme to batch fully homomorphic encryption, i.e., a scheme that supports encrypting and homomorphically processing a vector of plaintext bits as a single ciphertext [20]. The public key size was reduced to $O(\lambda^7)$. Coron *et al*. described a variant of DGHV scheme with the same scale-invariant property [21]. It had a single secret modulus whose size was linear in the multiplicative depth of the circuit to be homomorphically evaluated, instead of exponential. A main problem with the version of DGHV scheme was that the ciphertext expansion rate was bigger than before [19, 20].

In this paper, we propose a variant of DGHV scheme with the batch property, which can be regarded as an extension of Coron's scheme [20]. The message space is $\mathbf{Z}_{\min(n_i)}$ instead of $\mathbf{Z}_2$. We put forward a new hardness problem, which is called the

random approximation greatest common divisor (RAGCD) problem. Theorem 4 shows that RAGCD problem is a stronger version of approximation greatest common divisor (AGCD) problem. As a consequence, we obtain a shorter public key without sacrificing the security of the scheme. The public key size is reduced to $O(\lambda^4)$ which is suitable for practical application. Meanwhile, the ciphertext expansion rate is reduced to $O(\lambda^3)$, and the overall complexity is reduced to $O(\lambda^8)$. Table 1 shows a comparison of our scheme with related works. We provide in Table 3 timings for our batch DGHV scheme, and provide in Table 4 concrete key sizes for our batch DGHV scheme. Our estimates are backed up with experimental data. It is worth noting that we concentrates on the somewhat homomorphic encryption scheme, which is more applicable than fully homomorphic encryption scheme.

Table 1. Comparisons of Our Scheme with Related Works

	[17]	[18]	[19]	[20]	[21]	Our scheme
Parallel computing	NO	NO	YES	YES	YES	YES
Size of pk	$O(\lambda^{10})$	$O(\lambda^7)$	$O(\lambda^{10})$	$O(\lambda^7)$	$O(\lambda^4)$	$O(\lambda^4)$
ciphertext expansion rate	$O(\lambda^5)$	$O(\lambda^5)$	$O(\lambda^2)$	$O(\lambda^3)$	$O(\lambda^4)$	$O(\lambda^3)$
Overall complexity	$O(\lambda^{12})$	$O(\lambda^{15})$	$O(\lambda^{10})$	$O(\lambda^8)$	$O(\lambda^8)$	$O(\lambda^8)$
Hardness problem	AGCD	AGCD	AGCD	AGCD	AGCD	RAGCD

1.2 Organization

The remainder of the paper is organized as follows. In section 2, we briefly recall the notations and definitions. In section 3, we formally describe the main technical of this work. In section 4, we discuss the performance and security issues. Section 5 describes two known attacks for the AGCD problem. Section 6 reports the main results of our experiments. Finally, section 7 concludes the paper and presents the open problem.

2 Preliminary

2.1 Notations

Throughout this paper we use λ to indicate the security parameter. Real numbers and integers are denoted by lowercase letters, vectors are denoted by lowercase bold letters, and sets are denoted by capital bold letters. Particularly, we denote the integer set by $\mathbf{Z}$, and denote the module-N integer residue class by $\mathbf{Z}_N$. For a set $\mathbf{A}$, $a \xleftarrow{R} \mathbf{A}$ denotes that a is sampled uniformly from $\mathbf{A}$. For a real number x, we denote by $\lfloor x \rfloor$, $\lceil x \rfloor$, $\lceil x \rfloor$ the rounding of a down, up, or to the nearest integer. Namely, these are the unique integers in the half open intervals $(x-1, x]$, $[x, x+1)$ and $(x-0.5, x+0.5]$, respectively. For two integers z and p, we denote the quotient and remainder of z with respect to p by $q_p(z)$ and $r_p(z)$. Meanwhile, for two integers z and p, we denote the reduction of z modulo p by $(z \bmod p)$ or $[z]_p$ with $-p/2 < [z]_p \le p/2$.

According to DGHV scheme, we denote the evaluated polynomial on ciphertexts by function f corresponding to the evaluated circuit C. For example, the arithmetic circuit of $f(x, y) = x^2 + xy + y^2$ is described in Fig. 1.

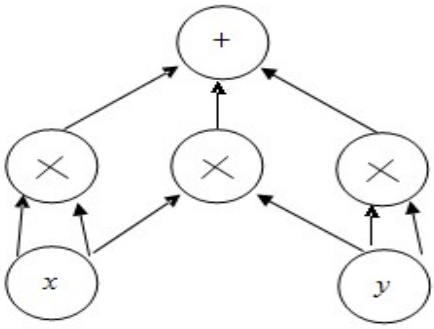

Fig. 1. the arithmetic circuit of $f(x,y)=x^2+xy+y^2$

2.2 Homomorphic encryption

A homomorphic encryption scheme consists of four algorithms: the key generation algorithm KeyGen , the encryption algorithm Enc , the decryption algorithm Dec , and an additional algorithm Eval . Compared with the usual private/public key encryption, homomorphic encryption can supports arbitrary operation on the ciphertexts without decryption. Roughly speaking , KeyGen takes the security parameter λ as input, and outputs the secret key sk and the public key pk; Enc takes the public key pk and the message m as input, and outputs the corresponding ciphertext c; Dec takes the secret key sk and the ciphertext c as input, and outputs the corresponding message m'; Eval takes the public key pk, a τ-variable function f , and a τ -tuple ciphertext $\mathbf{c}=(c_1,c_2,...,c_\tau)$, where $c_i=\mathrm{Enc}(sk,m_i)$ as input, and outputs an evaluated ciphertext $c=\mathrm{Eval}(pk,f,\mathbf{c})$.

Definition 1 (Correctness). A homomorphic encryption is correct for a given τ-variable function f if, for any key-pair (pk,sk) generated by KeyGen , and any ciphertexts $\mathbf{c}=(c_1,c_2,...,c_\tau)$ with $c_i=\mathrm{Enc}(sk,m_i)$, it is the case that:

$$\mathrm{Dec}(sk,\mathrm{Eval}(pk,f,(c_1,c_2,...,c_\tau)))=f(m_1,m_2,...,m_\tau) \quad (1)$$

Definition 2 (Somewhat Homomorphic Encryption). Let $d=\deg(f)$, and f_{Dec} be the decryption function. The scheme is correct for f . The scheme is a somewhat homomorphic encryption, it is the case that: $d\le 2\deg(f_{\mathrm{Dec}})$.

Definition 3 (Fully Homomorphic Encryption). Let $d=\deg(f)$, and the scheme be correct for f . The scheme is a fully homomorphic encryption, it is the case that: $d\in(2\deg(f_{\mathrm{Dec}}),+\infty)$.

2.3 China Remainder Theorem

The China Remainder Theorem is a result of congruence in number theory [22]. It has a great many applications in cryptology.

Theorem 1 (China Remainder Theorem). Suppose $n_1,n_2,...,n_k$ are positive integers that are pair-wise coprime. For any given sequence of integers $\pi_1,\pi_2,...,\pi_k$, there exists an integer m solving the following system of simultaneous congruence:

$$\begin{aligned} m&=\pi_1(\mathrm{mod}\,n_1)\\ m&=\pi_2(\mathrm{mod}\,n_2)\\ &...\\ m&=\pi_k(\mathrm{mod}\,n_k)\end{aligned} \quad (2)$$

Proposition 1. For a mapping: $\sigma: m\leftrightarrow(\pi_1,\pi_2,...,\pi_k)$, and $\pi_i\in\mathbf{Z}_{n_i}$, $m\in\mathbf{Z}_n$. σ is bijective , and $\sigma:\mathbf{Z}_n\leftrightarrow\mathbf{Z}_{n_1}\times\mathbf{Z}_{n_2}\times\cdots\times\mathbf{Z}_{n_k}$ is a homomorphic mapping, if it is the case that: for $i=1,2,...,k$, $m=\pi_i \bmod n_i$.

Proof: As $m=\pi_i \bmod n_i$, we compute $\pi_i=m-\lfloor m/n_i\rceil n_i$, and outputs a sequence of integers $\pi_1,\pi_2,...,\pi_k$. Similarly, for $i=1,2,...,k$, let $l_i=n/n_i=\prod_{j\ne i}n_j$, and compute $v_i=l_i(l_i^{-1}\bmod n_i)$. It is worth mentioning that, for $\gcd(l_i,n_i)=1$, it is reasonable to compute v_i . Then compute $m=\sum_{i=1}^{k}\pi_i\cdot v_i(\mathrm{mod}\,n)$. According to the basic number theory, it can be conclude that σ is bijective. Assume that $n\in\mathbf{Z}$, $n_i\in\mathbf{Z}$, and $n_i\mid n$, we can conclude that σ is homomorphic. □

Proposition 2 (An Instance of σ). For $m=\pi_i \bmod n_i$, $i=1,2,...,k$, we can compute

$$\pi_i=m-\lfloor m/n_i\rceil n_i \quad (3)$$

, and produce a sequence of integers $\pi_1,\pi_2,...,\pi_k$. Similarly, we assume that $l_i=n/n_i$, $v_i=l_i(l_i^{-1}\bmod n_i)$, then compute

$$m=\sum_{i=1}^{k}v_i\cdot\pi_i(\mathrm{mod}\,n) \quad (4)$$

Proof: For $m=\pi_i \bmod n_i$, we can compute $\pi_i=m-\lfloor m/n_i\rceil n_i$, and produce a sequence of integers $\pi_1,\pi_2,...,\pi_k$. Assume that $l_i=n/n_i$, for $j\ne i$, then we can conclude that $l_j=0(\mathrm{mod}\,n_i)$. After that, assume that $v_i=l_i(l_i^{-1}\bmod n_i)$, then we can conclude that $v_i=1(\mathrm{mod}\,n_i)$,

$v_j = l_j = 0(\mathrm{mod}\, n_i)$. As a result, a mapping $v_i \leftrightarrow (0,0,...,1,...0)$ from $v_i = 1(\mathrm{mod}\, n_i)$ and $v_j = l_j = 0(\mathrm{mod}\, n_i)$ is obtained. For i, $l_i = n / n_i$, $v_i = l_i(l_i^{-1} \bmod n_i)$, $m = v_i \cdot \pi_i (\mathrm{mod}\, n_i)$, it is easy to verify that $m = \pi_i \bmod n_i$. Hence, we can conclude that Eq.(3) and Eq.(4) are correct. □

2.4 DGHV scheme

We informally review the DGHV Somewhat Homomorphic Encryption [17]. The construction consists of four algorithms:

KeyGen(1^λ). Choose a random e-bit odd integer from the right open interval $[2^{e-1}, 2^e)$ as the secret key p. For $i = 0,1,...,\tau$, choose a random integer q_i from the interval $[0, 2^g / p)$, choose another integer r_i from the open interval $(-2^r, 2^r)$, and compute $x_i = pq_i + r_i$. Relabel so that x_0 is the largest. Restart unless x_0 is odd and $(x_0 \bmod p) \bmod 2 = 0$. Generate the public key $pk = (x_0, x_1, ..., x_\tau)$ and the secret key $sk = p$.

Enc($pk, m \in \{0,1\}$) . Choose a random subset $S \subseteq \{1,2,...,\tau\}$, a random integer r in $(-2^{\rho'}, 2^{\rho'})$, and compute $c = [m + 2r + 2\sum_{i \in S} x_i] \bmod x_0$.

Eval($pk, C, c_1, c_2, ..., c_\tau$) . Given the (binary) circuit C with τ inputs, and τ ciphertexts c_i, apply the (integer) addition and multiplication gates of C to the ciphertexts, and generate the evaluated ciphertexts.

Dec(sk, c) . Output $m = (c \bmod p) \bmod 2$.

This completes the description of the scheme. It is noted that the scheme described above is a somewhat homomorphic scheme. The security is reduced to the AGCD assumption.

3 Our construction

In this section, we propose a somewhat homomorphic encryption that supports integer arithmetic instead of bit-operation. The message space is $\mathbf{Z}_{\min(n_i)}$.

3.1 Parameters

In addition to the various parameters used in the DGHV scheme, some more parameters are used in our construction. It is worth noting that the bit lengths of various parameters used in the scheme are designated by some parameters, which are polynomial in the security parameter λ. The values for these parameters are described as follows:

λ : security parameter; the same as Gentry suggested [6], $\lambda = 80$;

e : the bit length of secret key p; $e = O(\lambda^3)$, in order to resist brute force attack;

e' : the bit length of second secret key u; $e' = O(\lambda^2)$, to support sufficiently homomorphic evaluation on the intermediate message;

k : the number of message space , for $n = 2^{O(\beta)}$, $k \in (0, 2^{O(\sqrt{\lambda})-1})$;

β : the bit length of intermediate message; $\beta = O(\lambda)$, to support sufficiently homomorphic evaluation;

g : the bit length of public key x_0 and x_1; $g = O(\lambda^4)$, to be secure against DGHV's lattice attack [17];

θ : the bit length of public key n_i, $\theta = O(\sqrt{\lambda})$;

s : the bit length of random number r, which is used in the encryption procedure to enhance the security of the scheme; $s = O(\sqrt{\lambda})$, to support sufficiently homomorphic evaluation on the initial message;

t : the bit length of random number h, $t = O(\sqrt{\lambda})$, to support sufficiently homomorphic evaluation on the intermediate message.

3.2 The construction

Idea. There are k initial messages $\pi_1, \pi_2, ..., \pi_k$, $\pi_i \in \mathbf{Z}_{n_i}$, for $i \in \{1,2,...,k\}$. According to $\sigma : m \leftrightarrow (\pi_1, \pi_2, ..., \pi_k)$, we compute an intermediate message $m \in \mathbf{Z}_n$. Proposition 1 shows that each

operation on intermediate message m is equivalent to the operation on k initial messages $\pi_1, \pi_2, ..., \pi_k$. It means that we can encrypt the intermediate message m instead of k initial messages $\pi_1, \pi_2, ..., \pi_k$.

KeyGen(1^λ). Choose two random odd integers p of size e and u of size e', the secret key is $sk = (p, u)$. Choose two random integers q_0 and q_1 from the interval $\left[0, 2^g / p\right)$, and choose a random integer h of size t. Compute $x_0 = pq_0$ and $x_1 = pq_1 + uh$. Restart unless x_0 and x_1 are co-prime, x_0 is larger than x_1. Select k primes n_i as a portion of public key, all of which are pair-wise co-prime. And then compute $n = \prod_{i=1}^{k} n_i$. The public key is $pk = (x_0, x_1, n, (n_1, n_2, ..., n_k))$.

Enc($pk, (\pi_1, \pi_2, ..., \pi_k)$). The encryption algorithm consists of two steps.

Step1: According to China Remainder Theorem, compute an intermediate message $m \in \mathbf{Z}_n$ from k initial messages $\pi_1, \pi_2, ..., \pi_k$. Assume that $l_i = n / n_i$, $v_i = l_i(l_i^{-1} \bmod n_i)$, then compute $m = \sum_{i=1}^{k} v_i \cdot \pi_i \pmod n$.

Step2: Given $m \in \mathbf{Z}_n$, choose a s-bit random integers r, compute the ciphertext $c = (m + rx_1) \bmod x_0$.

Dec(sk, c). The decryption algorithm consists of two steps.

Step1: Given a ciphertext c, and the secret key $sk = (p, u)$, compute the intermediate message $m = (c \bmod p) \bmod u$;

Step2: For $m = \pi_i \bmod n_i$, the output is $\pi_i = m - \lfloor m / n_i \rceil n_i$.

Eval($f, (c_1, ..., c_\tau)$). Given a τ-variable function f, and a τ-tuple ciphertext $\mathbf{c} = (c_1, c_2, ..., c_\tau)$, where $c_i = \text{Enc}(sk, m_i)$, performs homomorphic operations over c_i, and outputs the evaluated ciphertext c.

For two ciphertexts c_1 and c_2, additive and multiplicative homomorphic operations are as follows: Additive homomorphic: $c_{\text{add}} = (c_1 + c_2) \bmod x_0$, multiplicative homomorphic: $c_{\text{mult}} = (c_1 \cdot c_2) \bmod x_0$. The resulting ciphertext after homomorphic evaluation is decrypted by the decryption algorithm Dec.

4 Analyses

4.1 Correctness

The proof of correctness is consists of two parts: the correctness of the decryption algorithm Dec, and the correctness of the evaluation algorithm Eval.

Theorem 2. For an initial ciphertext c, and k initial messages $\pi_1, \pi_2, ..., \pi_k$, the decryption algorithm is correct.

Proof: For k initial messages $\pi_1, \pi_2, ..., \pi_k$, we assume that $l_i = n / n_i$, $v_i = l_i(l_i^{-1} \bmod n_i)$, and compute $m = \sum_{i=1}^{k} v_i \cdot \pi_i \pmod n$, where $m \in \mathbf{Z}_n$. As $c = (m + rx_1) \bmod x_0$, we have $c = (m + rx_1) - a_1 \cdot x_0$ for an integer a_1. Furthermore, $x_0 = pq_0$, $x_1 = pq_1 + uh$, we have

$$c = (m + rpq_1 + ruh) - a_1 pq_0 = p(rq_1 - a_1 q_0) + urh + m \quad (5)$$

From the parameter setting, $urh + m << p$ and $m << u$. Compute $m = (c \bmod p) \bmod u$, $\pi_i = m - \lfloor m / n_i \rceil n_i$, generate the k initial messages $\pi_1, \pi_2, ..., \pi_k$. □

The notion of permitted circuit [17], which is defined as follows, is helpful to prove the correctness and homomorphism.

Definition 4 (Permitted Circuit). For a homomorphic encryption scheme, and a real number α. Similar to the DGHV scheme, we define a permitted circuit as one where for any $\alpha \geq 1$ and for any set of integer inputs each $\leq 2^{(\beta + \alpha s)}$ in absolute value, it holds that the circuit's output has absolute

value at most 2^{e-2}. Let $C_{\text{per-c}}$ denote the set of permitted circuits.

Theorem 3. For a permitted circuit C which takes $2k$ initial messages $\pi_{1_1}, \pi_{2_1}, ..., \pi_{k_1}$ and $\pi_{1_2}, \pi_{2_2}, ..., \pi_{k_2}$ as input, the decryption algorithm is correct.

Proof: for $2k$ initial messages $\pi_{1_1}, \pi_{2_1}, ..., \pi_{k_1}$ and $\pi_{1_2}, \pi_{2_2}, ..., \pi_{k_2}$, assume that $l_i = n / n_i$, $v_i = l_i(l_i^{-1} \bmod n_i)$, compute two intermediate messages $m_1 = \sum_{i=1}^{k} v_i \cdot \pi_{i_1} (\bmod n)$ and $m_2 = \sum_{i=1}^{k} v_i \cdot \pi_{i_2} (\bmod n)$. For two ciphertexts $c_1 = \text{Enc}(pk, m_1)$ and $c_2 = \text{Enc}(pk, m_2)$, the addition of two ciphertexts is $c_{\text{add}} = (c_1 + c_2) \bmod x_0$. According to the permitted circuit, we have $c_{\text{add}} = (m_1 + m_2) + a_2 \cdot u + a_3 \cdot p$ for two random integers a_2 and a_3, where $a_2 \in [2^{\alpha s-1}, 2^{\alpha s})$. Then, we can conclude that $(m_1 + m_2) = c_{\text{add}} \bmod p \bmod u$. Similarly, the multiplication of two ciphertexts is $c_{\text{mult}} = (c_1 \cdot c_2) \bmod x_0$. According to the permitted circuit, we have $c_{\text{mult}} = (m_1 \cdot m_2) + a_4 \cdot u + a_5 \cdot p$ for two random integers a_4 and a_5, where $a_4 \in [2^{e'+\alpha s-1}, 2^{e'+\alpha s})$. We can conclude that $(m_1 \cdot m_2) = c_{\text{mult}} \bmod p \bmod u$. As we know, an arithmetic circuit consists of addition and multiplication modulo- x_0 gates. Combine the permitted circuit with the homomorphic mapping $\sigma : \mathbf{Z}_n \leftrightarrow \mathbf{Z}_{n_1} \times \mathbf{Z}_{n_2} \times \cdots \times \mathbf{Z}_{n_k}$, the evaluated ciphertext can be decrypted correctly. □

4.2 Homomorphic

Additively Homomorphic. According to triangle inequality, each noise of $c_1 + c_2$ is increased at most 1-bit. As described in section 3.1, the bit length of the second secret key u is $O(\lambda^2)$, and the bit length of the random number r is $s = O(\sqrt{\lambda})$. Clearly, the proposed scheme can supports approximately $O(\lambda^2)$ additions on ciphertexts.

Multiplicatively homomorphic. One multiplicative operation on ciphertexts may square the noise - i.e., double their bit-lengths. That's to say, the noise expansion through multiplication is more significant than addition. The homomorphic evaluation capacity of our scheme is mainly influenced by the number of multiplications, which is defined as the degree of the evaluated polynomial.

Lemma 1 (DGHV, lemma 3.5). Let C be an arithmetic circuit and f be the multivariate polynomial computed by C. It is easy to give a sufficient condition on a multivariate polynomial f for the associated arithmetic circuit C to be permitted. If $\|f\|_1 \cdot (2^{\beta+\alpha s})^d \le 2^{e-2}$, then $C \in C_{\text{per-c}}$, where $\|f\|_1$ is the 1- norm of the coefficient vector of f and $d = \deg(f)$.

From the above conditions and parameters, we have $d \le \dfrac{e - 2 - \log_2(\|f\|_1)}{\beta + \alpha s}$, which is similar to the DGHV scheme. Clearly, the proposed scheme can support approximately $O(\lambda^2)$ multiplications on ciphertexts. In this work, we can assume $\log_2(\|f\|_1)$ is relatively small to e and β.

4.3 Security

Informally speaking, we consider a game with a solver Ψ and an attacker atk. The game can be described as follows: initially, atk receives the public key. And then, atk sends two different messages to the Ψ, who chooses one to encrypt. After receiving the ciphertext, atk guesses which message generates the ciphertext and wins the game if he gets it right. The scheme is secure if the probability of attacker wins the game is at most $1/2 + \varepsilon$, where ε is a negligible value. In this work, we propose a new hardness problem, which is called the RAGCD problem. Note that, RAGCD problem is a stronger version of AGCD problem [23]. Crucially, the security of the scheme can be reduced to the RAGCD problem.

Definition 5 (RAGCD). The RAGCD problem is: given two integers x_0 and x_1, for $h \xleftarrow{R} \mathbf{H}$, $q_i \xleftarrow{R} \mathbf{Q}$, $i = 0,1$, $x_0 = q_0 p$ and $x_1 = q_1 p + uh$, output p and u.

Definition 6 (AGCD). The AGCD problem is: given two integers x_0 and x_1, for $h \xleftarrow{R} \mathbf{H}$, $q_i \xleftarrow{R} \mathbf{Q}$, $i = 0,1$, $x_0 = q_0 p$ and $x_1 = q_1 p + h$, output p.

Theorem 4. If there is a solver Ψ that solves the AGCD problem with advantage ς, then the solver Ψ can solve the RAGCD problem with advantage $2^{-\sqrt{\lambda}}\varsigma$.

Proof: given two integers x_0 and x_1, a sovler Ψ that solves the AGCD problem with advantage ς, output p. For p, we can compute $uh = x_1 \bmod p$. In the worst case, the sovler Ψ must try each element h in the set $\mathbf{H}$ of size $2^{\sqrt{\lambda}}$. As a result, we can conclude that the sovler Ψ can solve the RAGCD problem with advantage $2^{-\sqrt{\lambda}}\varsigma$. □

In this work, the secret key is $sk = (p,u)$, and the public key is $pk = (x_0, x_1, n, (n_1, n_2, ..., n_k))$. For the k primes $(n_1, n_2, ..., n_k)$, atk can break the security of the scheme without accessing them. The public key is an instance of RAGCD problem, especially; x_0 is an exact multiply of p, x_1 is an approximately multiply of p. Theorem 5 shows that the security of the scheme can be reduced to RAGCD problem. The proof of Theorem 5 is similar to the proof of theorem 4.2 in DGHV scheme. Without loss of validity, we directly reference a subroutine Binary-GCD in DGHV scheme.

Theorem 5. Let atk be an attacker with advantage ε against our encryption scheme with parameters (e, e', g, t) polynomial in the security parameter λ. There exists a solver Ψ for solving the RAGCD problem that succeeds with at least a probability of $\varepsilon / 2$.

Proof: Let atk be an attacker against the scheme. Namely, atk takes a public key and a ciphertext (as produced by our scheme) as input, and outputs the correct plaintext with probability $1/2 + \varepsilon$ for some noticeable ε. After that, atk is used to construct a solver Ψ for RAGCD problem. For two randomly chosen odd integers p of size e and u of size e', the solver Ψ can access to a portion of the public key $x_0 = q_0 p$ and $x_1 = q_1 p + uh$, and the goal is to find p and u. Next, Ψ produces a sequence of integers, and attempts to recover p by utilizing atk to learn the least significant bit (LSB) of the quotients of these integers with respect to p. The subroutine Learn-LSB is as follows.

Table 2. Learn-LSB Algorithm

Learn-LSB Algorithm Learn the LSB of the quotients of these integers with respect to p

Input: $z \in [0, 2^g)$ with $|r_p(z)| < 2^{e'-2}$, $pk = (x_0, x_1, n, (n_1, n_2, ..., n_k))$

Output: The LSB of $q_p(z)$

Method:

1. For $j = 1$ to $\text{poly}(\lambda)/\omega$ do:
2. Choose noise $r_j \xleftarrow{R} (-2^s, 2^s)$, a set of random integer $\pi_{i_j} \in \mathbf{Z}_{n_i}$, $i = 1,2,...,k$;
3. Compute $m_j \in \mathbf{Z}_n$, $m_j = \sum_{i=1}^{k} v_i \cdot \pi_{i_j} (\bmod n)$, $l_i = n / n_i$, $v_i = l_i (l_i^{-1} \bmod n_i)$;
4. Compute $c_j = (z + m_j + r_j x_1) \bmod x_0$;
5. Call *atk* to get a prediction $a_j = \text{atk}(pk, c_j)$;
6. Compute $b_j = [a_j]_2 \oplus [z]_2 \oplus [m_j]_2$;
7. Output the majority vote among the b_j's

According to lemma 2, we show that for all but a negligible fraction of the public keys generated by the scheme, the "ciphertext" c_j in step 4 of the Subroutine is distributed almost identically to a valid encryption of the $r_p(z) + m_j$. Note also that

since p is odd, we always have $[q_p(z)]_2 = [r_p(z)]_2 \oplus [z]_2$, and b_j should be the parity bit of the $q_p(z)$. We can conclude that if atk has a noticeable advantage in guessing the encrypted bit under $pk = (x_0, x_1, n, (n_1, n_2, ..., n_k))$, then Learn-LSB(z, pk) will return $b_j = [q_p(z)]_2$ with overwhelming probability. Once we turned atk into an oracle for the LSB of $q_p(z)$, recovering p is rather straightforward. The simplest way of doing it is using the Binary-GCD algorithm [17]: to recover p, the solver Ψ draws two integers z_1, z_2 with $|r_p(z_i)| < 2^{e'-2}$, $i = 0,1$, and applies the Binary-GCD algorithm to them. With a probability of at least 0.6 [17], the odd part of $\mathrm{GCD}(q_p(z_1), q_p(z_2))$ is one. As a result, the algorithm should generate an element $\tilde{z} = 1 \cdot p + uh$ with $|r_p(z)| < 2^{e'-2}$. Thus, we should choose the appropriate integer to ensure that $q_p(z_1)$ is one.

Lastly, Ψ repeats the Binary-GCD algorithm from above using z_1 and $z_2 = \tilde{z}$, and the sequence of parity bits of the $q_p(z_1)$ s in all the iterations spell out the binary representation of $q_p(z_1)$. Now Ψ can recover $p = \lfloor z_1 / q_p(z_1) \rceil$ and $q_p(z_2) = \lfloor z_2 / p \rceil$. Next, Ψ computes $z_1^* = z_1 - (q_p(z_1) \cdot p) = uh_1$ and $z_2^* = z_2 - (q_p(z_2) \cdot p) = uh_2$. In the end, Ψ can easily computes the general common divisor of the z_1^* and z_2^*, and recovers u. We conclude that a polynomial time solver Ψ can solves the RAGCD problem with an advantage of at least $\varepsilon / 2$. □

Lemma 2. Fix the parameters (e, e', g, t), fix any $sk = (p, u)$, and let $pk = (x_0, x_1, n, (n_1, n_2, ..., n_k))$ be chosen as in the KeyGen of our scheme. For every integer $z \in [0, 2^g)$ which is at most $2^{e'-2}$ away from a multiple of p, and a random integer $r_j \xleftarrow{R} (-2^s, 2^s)$, an intermediate message $m_j \in \mathbf{Z}_n$, consider the following distribution $c = (z + m_j + r_j x_1) \bmod x_0$, every distribution c is statistically close to the distribution $\mathrm{Enc}(pk, (r_p(z) + m_j))$.

Proof. According to encryption algorithm, $c = q'p + r_j uh + r_p(z) + m_j$. Regarding q', we claim that in the scheme the value $q_p(c)$ of a ciphertext is uniform in $(-q_0/2, q_0/2)$. According to the parameter setting, it implies that $r_p(z) + m_j = (r_j uh + r_p(z) + m_j) \bmod u$. As a result, c is distributed almost identically to a valid encryption of $r_p(z) + m_j$. □

4.4 Complexity Analysis

The advantage of our scheme lies in a short key size and low computation complexity, and we give a detailed analysis below. Compared with the previous schemes, which involve generating a big public key that consists of a large set of $O(\lambda^5)$ integers [17] or $O(\lambda^2)$ integers [18] each having a size of $O(\lambda^5)$, the public key of the proposed scheme consists of two integers of size $O(\lambda^4)$ and k small primes of size $O(\lambda)$. The size of the public key is $O(\lambda^4)$. The encryption algorithm Enc consists of two steps: the computation cost of intermediate message is $O(\sqrt{\lambda})$; Enc involves a multiplication of complexity $O(\lambda^5)$ resulting in an ciphertext of size $O(\lambda^4)$. The modular reduction of this ciphertext with $O(\lambda^4)$ bit x_0 takes $O(\lambda^8)$ computations. In the decryption algorithm Dec, the modular reduction of the $O(\lambda^4)$ bit ciphertext with the $O(\lambda^3)$ bit secret integer p takes $O(\lambda^8)$ computations, resulting in an integer of size $O(\lambda^3)$. The modular reduction of this $O(\lambda^3)$ bit integer with $O(\lambda^2)$ bit secret integer u takes $O(\lambda^6)$ computations, and results in an intermediate message of size less than or equal to $O(\lambda^2)$. The computation cost of the initial message $\pi_1, \pi_2, ..., \pi_k$ is $O(\sqrt{\lambda})$. The total computation cost of decryption algorithm is $O(\lambda^8)$.

Homomorphic addition: The addition of two ciphertexts is simply integer addition, and the computation cost is $O(\lambda^4)$. After an addition, the length of ciphertext is not increased, and accordingly the computation cost of decryption remains the same. Homomorphic multiplication: When multiply two ciphertexts, it needs to compute integer multiplication, the computing cost is $O(\lambda^8)$.

It can be conclude that the overall computation cost of the scheme is $O(\lambda^8)$.

5 Two Known Attacks for the AGCD Problem

In this section, we informally review two known attacks on the AGCD problem, including brute-forcing the remainders, and the GCD Attack [24].

5.1 Brute-Force Attack

The easiest attack is the brute force attack on the noise in the public key. Given the public key (x_0, x_1), the brute-force attack can be described as follows: choosing two random integers u and h from the interval $[2^{e'-1}, 2^{e'})$ and $[2^{t-1}, 2^t)$ respectively, subtracting uh from x_1, and computing $\mathrm{GCD}(x_0, x_1 - uh)$. In a worst case, this process may need to be repeated for all the product of the integers u and h. As h is chosen uniformly from the set $\mathbf{H}$ of size $2^{\sqrt{\lambda}}$, and u is chosen uniformly from the set $\mathbf{U}$ of size 2^{λ^2}, and the state of the art algorithm for computing GCD problem is the Stehle-Zimmermann algorithm [25] with time complexity $O(g)$ for integers of g bits. As a result, the complexity of this attack is $2^{\lambda^4+\lambda^2+\sqrt{\lambda}}$.

For the chosen parameter values, the size of x_0 is big enough so that, even the best known integer factoring algorithms such as the General Number Field Sieve [26] will not be able to factor x_0 in a reasonable time. Meanwhile, the algorithms such as Lenstra's elliptic curve factoring [27] generates p with time complexity 2^{λ^3}. Furthermore, the secret key p will not be recovered directly as it is not prime.

5.2 The GCD Attack

In [27], the author declared that the GCD of the public key (x_0, x_1) is the smallest positive element in the set $\{ax_0 + bx_1 : a, b \in \mathbf{Z}\}$. As a result, the common divisor of the public key (x_0, x_1) will divide all the possible linear combinations of (x_0, x_1). Modular reduction of a ciphertext with such common divisor results in the plaintext, because a ciphertext contains a linear combination of (x_0, x_1). As a result, taking the pair of integers (x_0, x_1) as co-prime can defends this attack.

6 Experimental

Our experiment is conducted in a laptop computer (Intel Core i3 at 2.53 GHz, 2GB RAM). NTL-5.2.2 is used as the C++ library for writing the program. Note that we select the average run-time, and the number of iterations is 20. We take the average values except the maximum and minimum for each item. We use five security levels inspired by the levels from [6]: "toy", "small", "medium" and "large", corresponding to 42, 52, 62 and 72 bits of security respectively. Moreover, we use one additional security level to improve our scheme, corresponding to 80 bits of security. Note that, our "suggested" level of security can improve the security without sacrificing the performance. To obtain more accurate results, we compare the methods under the same experimental environments. We provide in Table 3 timings for our batch DGHV scheme, and provide in Table 4 concrete key sizes for our batch DGHV scheme.

Table 3. The timings of our scheme

	λ	KeyGen	Enc	Dec	Additively Homomorphic	Multiplicatively homomorphic
Toy	42	2.15s	0s	0s	0s	0.8s
Small	52	43.3s	0.03s	0s	0s	0.8s
Medium	62	758s	0.08s	0.04s	0s	21.4s
Large	72	10742s	6.86s	0.15s	0.03s	141s
Suggested	80	69451s	192.5s	45.6s	0.41s	1793s

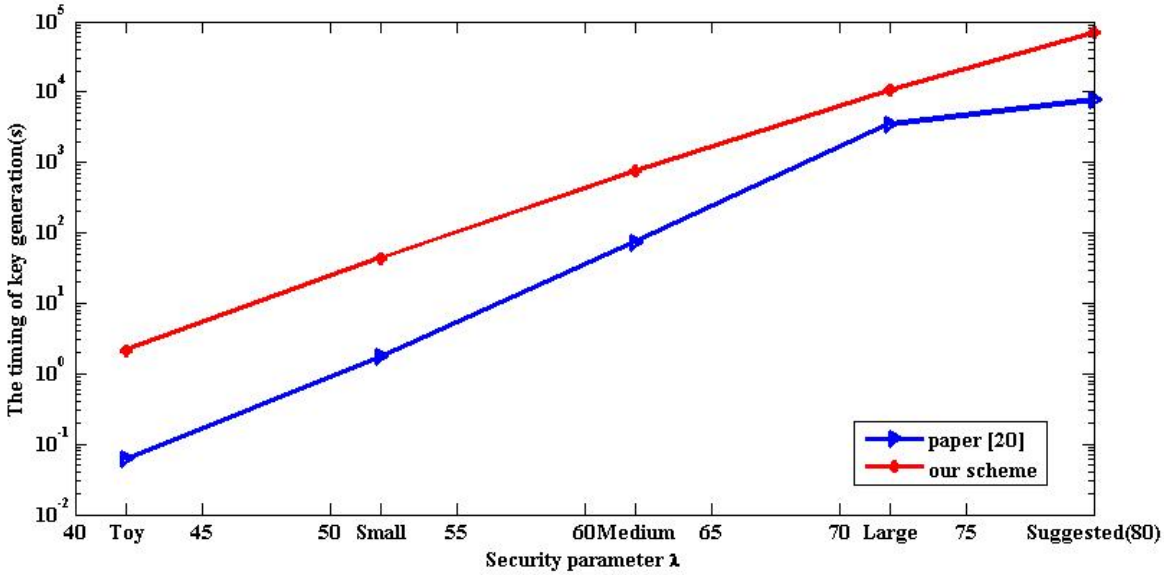

Fig.2: The comparison of timing of key generation

Timings. In order to generate the public key, primality testing algorithms is needed. Figure 2 shows that the timing of key generation is increased than before. However, the timings of encryption procedure and decryption procedure are equal to previous work [20]. In the following work, how to improve the efficiency of key generation is an interesting problem.

Table 4. The concrete key sizes of our scheme

	λ	k	e	e'	β	θ	s	t
Toy	42	27	$1.3 \cdot 10^4$	$1.5 \cdot 10^3$	51	22	16	29
Small	52	30	$1.4 \cdot 10^5$	$2.8 \cdot 10^3$	77	19	8	21
Medium	62	29	$3.6 \cdot 10^5$	$7.1 \cdot 10^3$	103	7	13	12
Large	72	55	$1.7 \cdot 10^6$	$3.9 \cdot 10^4$	158	11	30	31
Suggested	80	106	$8.7 \cdot 10^6$	$2.1 \cdot 10^5$	261	23	46	73

Table 5. The comparison of public key size

	λ	Pk size of our scheme(MB)	Pk size of Coron's scheme[20](MB)
Toy	42	0.04	0.63
Small	52	0.95	13.3
Medium	62	51	304
Large	72	467	5734
Suggested	80	2889	49533

Key sizes. As described in Table 5, the public key size is shorter than previous scheme [20]. Moreover, we can encrypt integers (for suggested level) instead of a single bit. Our estimates are backed up with experimental data.

Comparing the experimental results with previous work [20], we note that the proposed algorithm is more efficient, especially; the public key size is improved. Our estimates are backed up with experimental data.

7 Conclusions

In this work, an efficient, parallel multi-integer homomorphic encryption scheme over the integers is proposed. It can be regarded as an extension of Coron's scheme [20] with larger message space. The ciphertext expansion rate is smaller than previous works. We put forward a new hardness problem, called the RAGCD problem. Theorem 4 shows that RAGCD problem is a stronger version of AGCD problem. More importantly, the security of this scheme can be reduced to RAGCD problem. As a consequence, we obtain a shorter public key without sacrificing the security of the scheme. It is expected that, the proposed scheme makes the encrypted data processing practical for suitable applications. However, the proposed scheme is a potential somewhat homomorphic encryption scheme. It is an open problem to construct a fully homomorphic encryption scheme, while preserving the hardness of the RAGCD assumption. How to improve the efficiency of the scheme is also an interesting problem. Moreover, a concrete, not just asymptotic condition for the parameters of our scheme is needed.

Acknowledgments

I would like to express my thanks and appreciation to my Doc advisor Yixian Yang, for his encouragement and guidance in completing this work. In particular, I would like to thank Prof. Shoushan Luo for many helpful guidance and constructive comments that helped present this work in a more coherent way. This work is supported by the National Natural Science Foundation of China (Grant No. 61121061, 61161140320).

Reference:

[1] R. L. Rivest, L. Adleman, D. L. Michael, On data banks and privacy homomorphisms, *In Foundations of Secure computation*, 1978, pp. 169-180.

[2] D. Boneh, J. Goh, K. Nissim, Evaluating 2-DNF formulas on ciphertexts, *Proceedings of the second Theory of Cryptography Conference*, Cambridge, MA, USA, February, 2005, pp. 325-341.

[3] C. Gentry, Fully homomorphic encryption using ideal lattices, *Proceedings of the 41st annual ACM symposium on Theory of computing*, Bethesda, MD, USA, May 31-June 02, 2009, pp. 168-179.

[4] C. Gentry. *A fully homomorphic encryption scheme*, Stanford: Stanford University, 2009.

[5] N. P. Smart, F. Vercauteren, Fully homomorphic encryption with relatively small key and ciphertext sizes, *Proceedings of the 13th International Conference on Practice and Theory in Public Key Cryptography*, Paris, France, May, 2010, pp. 420-443.

[6] C. Gentry, S. Halevi, Implementing Gentry's Fully-Homomorphic Encryption Scheme, *Proceedings of the 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, Tallinn, Estonia, May, 2011, pp. 129-148.

[7] D. Stehlé D, R. Steinfeld, Faster fully homomorphic encryption, *Proceedings of the 16th International Conference on the Theory and Application of Cryptology and Information Security*, Singapore, December, 2010, pp. 377-394.

[8] N. Ogura, G. Yamamoto, T. Kobayashi, et al, An improvement of key generation algorithm for Gentry's homomorphic encryption scheme, *Proceedings of the 5th International Workshop on Security*, Kobe, Japan, November, 2010, pp. 70-83.

[9] S. Garg, C. Gentry, S. Halevi, Candidate multilinear maps from ideal lattices, *Proceedings of the 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques*, Athens, Greece, May, 2013, pp. 1-17.

[10] Z. Brakerski, V. Vaikuntanathan, Efficient fully homomorphic encryption from (standard) LWE, *Proceedings of the 2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, Palm Springs, CA, USA, October, 2011, pp. 97-106.

[11] Z. Brakerski, C. Gentry, V. Vaikuntanathan, (Leveled) Fully homomorphic encryption without bootstrapping, *Proceedings of the 3rd Innovations in Theoretical Computer Science*

Conference, Cambridge, Massachusetts. New York, USA, Jan, 2012, pp. 309-325.

[12] C. Gentry, S. Halevi, N. P. Smart, Fully homomorphic encryption with polylog overhead, *Proceedings of the 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques*, Cambridge, UK , April, 2012, pp. 465-482.

[13] C. Gentry, S. Halevi, N. P. Smart, Homomorphic evaluation of the AES circuit, *Proceedings of the 32nd Annual Cryptology Conference*, Santa Barbara, CA, USA, August, 2012, pp. 850-867.

[14] C. Gentry, A. Sahai, B. Watersz, Homomorphic Encryption from Learning with Errors:Conceptually-Simpler,Asymptotically-Faster, Attribute-Base, *Proceedings of the 33rd Annual Cryptology Conference*, Santa Barbara, CA, USA, August, 2013, pp. 75-92.

[15] Z. Brakerski, C. Gentry, S. Halevi, Packed Ciphertexts in LWE-Based Homomorphic Encryption, *Proceedings of the 16th International Conference on Practice and Theory in Public-Key Cryptography*, Nara, Japan, February 26 - March 1, 2013, pp. 1-13.

[16] A. Lopez-Alt, E. Tromer, V. Vaikuntanathan, On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption, *Proceedings of the 44th symposium on Theory of Computing*, New York, USA, May, 2012, pp. 1219-1234.

[17] M. V. Dijk, C. Gentry, S. Halevi, et al, Fully homomorphic encryption over the integers, *Proceedings of the 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, French Riviera, May 30 – June 3, 2010, pp. 24-43.

[18] J. S. Coron, A. Mandal, D. Naccache, et al, Fully homomorphic encryption over the integers with shorter public keys, *Proceedings of the 31st Annual Cryptology Conference*, Santa Barbara, CA, USA, August, 2011, pp. 487-504.

[19] J. Kim J, M. S. Lee, A. Yun, et al, *CRT-based Fully Homomorphic Encryption over the Integers*, IACR Cryptology ePrint Archive, 2013:057.

[20] J. S. Coron, T. Lepoint, M. Tibouchi, Batch Fully Homomorphic Encryption over the Integers, *Proceedings of the 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques*, Athens, Greece, May 2013, pp. 315-335.

[21] J. S. Coron, T. Lepoint, M. Tibouchi, Scale-Invariant Fully Homomorphic Encryption over the Integers, *Proceedings of the 17th International Conference on Practice and Theory in Public-Key Cryptography*, Buenos Aires, Argentina, March, 2014, pp. 311-328.

[22] C. S. Ding, D. Y. Pei, A. Salomaa, *Chinese Remainder Theorem: Applications in Computing, Coding, Cryptography*, World Scientific Publishing, 1996.

[23] N. Howgrave-Graham, Approximate integer common divisors, *Proceedings of the international Conference*, Providence, RI, USA, March, 2001, pp. 51-66.

[24] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, *Introduction to Algorithms, 2nd edn*. MIT Press, 2002.

[25] D. Stehle, P. Zimmermann, A binary recursive GCD algorithm, *Proceedings of the 6th International Symposium*, Burlington, VT, USA, June, 2004, pp. 411-425.

[26] M. Briggs, *An Introduction to the General Number Field Sieve*. Virginia Tech, 1998.

[27] H. Lenstra, Factoring Integers with Elliptic Curves, *Annals of Mathematics*, Vol.126, No.3, 1987, pp. 649–673.