Hindawi
Wireless Communications and Mobile Computing
Volume 2018, Article ID 8142102, 13 pages
https://doi.org/10.1155/2018/8142102

Research Article

Homomorphic Evaluation of the Integer Arithmetic Operations for Mobile Edge Computing

Changqing Gong,[1] **Mengfei Li**,[1] **Liang Zhao**,[1] **Zhenzhou Guo**,[1] **and Guangjie Han**[2]

[1]*School of Computer Science and Technology, Shenyang Aerospace University, Shenyang 110136, China*
[2]*Laboratory for Ubiquitous Network and Service Software of Liaoning Province, School of Software, Dalian University of Technology, Dalian 116024, China*

Correspondence should be addressed to Liang Zhao; lzhao@sau.edu.cn

Received 26 September 2018; Accepted 31 October 2018; Published 15 November 2018

Guest Editor: Mianxiong Dong

With the rapid development of the 5G network and Internet of Things (IoT), lots of mobile and IoT devices generate massive amounts of multisource heterogeneous data. Effective processing of such data becomes an urgent problem. However, traditional centralised models of cloud computing are challenging to process multisource heterogeneous data effectively. Mobile edge computing (MEC) emerges as a new technology to optimise applications or cloud computing systems. However, the features of MEC such as content perception, real-time computing, and parallel processing make the data security and privacy issues that exist in the cloud computing environment more prominent. Protecting sensitive data through traditional encryption is a very secure method, but this will make it impossible for the MEC to calculate the encrypted data. The fully homomorphic encryption (FHE) overcomes this limitation. FHE can be used to compute ciphertext directly. Therefore, we propose a ciphertext arithmetic operation that implements data with integer homomorphic encryption to ensure data privacy and computability. Our scheme refers to the integer operation rules of complement, addition, subtraction, multiplication, and division. First, we use Boolean polynomials (BP) of containing logical AND, XOR operations to represent the rulers. Second, we convert the BP into homomorphic polynomials (HP) to perform ciphertext operations. Then, we optimise our scheme. We divide the ciphertext vector of integer encryption into subvectors of length 2 and increase the length of private key of FHE to support the 3-multiplication level additional. We test our optimised scheme in DGHV and CMNT. In the number of ciphertext refreshes, the optimised scheme is reduced by 2/3 compared to the original scheme, and the time overhead of our scheme is reduced by 1/3. We also examine our scheme in CNT of without bootstrapping. The time overhead of optimised scheme over DGHV and CMNT is close to the original scheme over CNT.

1. Introduction

With the rapid development of the 5G network and Internet of Things (IoT), mobile devices and IoT devices are more convenient to access the internet and generate massive amounts of data. Since these data come from edge network devices, traditional centralised cloud computing models are difficult to process these multisource heterogeneous data quickly and efficiently. If we migrate some of the features of cloud computing to an edge network as [1–3], it will be beneficial to data collection and calculation. Therefore, mobile edge computing (MEC) emerges as the above requires. Edge computing [4] is a method of optimizing applications or cloud computing systems by taking some portion of an application, its data, or services away from one or more central nodes (the "core") to the other logical extreme (the "edge") of the internet which contacts with the physical world or end users. In one vision of this architecture, specifically for IoT devices, data comes in from the physical world via various sensors, and actions are taken to change physical state via various forms of output and actuators; by performing analytics and knowledge generation at the edge, communications bandwidth between systems under control and the central data centre is reduced. The MEC is to put any computer program that needs low latency nearer to the requests in particular for mobile networks such as 5G. MEC allows terminal devices to migrate storage and computing tasks to network edge nodes. The architecture of edge computing is shown in Figure 1.

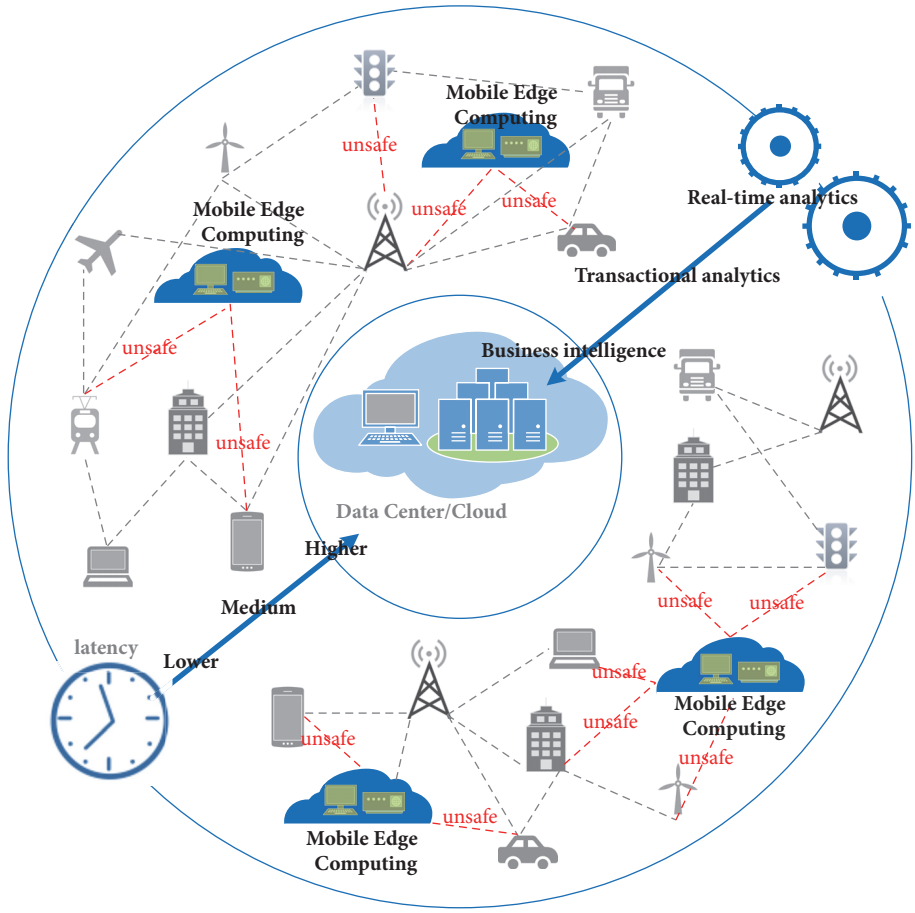

Figure 1: The architecture of mobile edge computing.

MEC covers a wide range of technologies including wireless sensor networks, mobile data acquisition, mobile signature analysis, cooperative distributed peer-to-peer ad hoc networking, and processing. The foundation of MEC is traditional network, wireless network, mobile network, and Internet of Vehicles (IoV), in which a large number of network infrastructure technologies and services are applied. Include energy efficiency and spectral efficiency tradeoff in device-to-device [5], QoS-aware interdomain multicast [6], efficient cross-layer relay node selection model [7], mobile anchors assisted localization [8], and vehicular communications [9]. Mobile edge computing (see Figure 1) is a service for multiple entities (mobile devices and IoT devices). Many entities put their real-time data (outsourced data in cloud computing) on mobile edge computing for analysis or storage. The extensive features of mobile edge computing such as data collection, real-time analytics, and parallel processing make the privacy issues that exist in the cloud computing environment become more prominent. Therefore, the security of outsourced data remains a fundamental issue for data security of MEC. The red dotted lines (see Figure 1) represent safety risk between the different entities and mobile edge computing, such as IoV and smart home services provided by mobile edge computing. In IoV, the location information of the vehicle and the online browsing records generated by the owner are collected by the mobile edge calculation, so as to feed back the precise navigation information of the vehicle, and the push information for the owner's preferences. In this process, the data of the owner and the car are exposed to the mobile edge computing, which is extremely unsafe. In smart home, many sensors monitor environment changes in the room, such as temperature, humidity, surveillance video, etc., through mobile edge calculation storage and analysis, giving the best home environment settings. This process is also unsafe. Traditional methods, including intrusion detection, access control, and virtual isolation, can only protect data from being stolen by external attackers. For internal attacks (honest and curious mobile edge computing), data security is still not guaranteed. The encrypted data are relatively considered a safe storage status. However, it is unable to satisfy the computability of ciphertext data. If we want the edge computing to be able to do nontrivial computations with the ciphertext data, and therefore the problem is severe to solve. The nontrivial computations include deep learning for the IoV with Edge Computing [10], offloading computation for MEC [11]. The fully homomorphic encryption (FHE) overcomes this limitation. Gentry described the first encryption scheme that supports both addition and multiplication base on ciphertexts, i.e., FHE scheme [12]. The FHE scheme allows anyone to perform arbitrary computations

on encrypted data, despite not having the secret decryption key.

The development of FHE can be partitioned into three generations [13]. The first generation includes Gentry's original scheme using ideal lattices [12], the somewhat simpler scheme of van Dijk et al. [14], and some optimisations for first generation in public key size [15–17]. All these schemes have a problem of rapidly growing noise, which affected both efficiency and security. The second generation begin with Brakerski-Vaikuntanathan [18, 19] and Brakerski et al. [20] and is characterized by modulus-switching and key-switching techniques for controlling the noise, resulting in improved efficiency, including LWE [18], Ring-LWE [19], and NTRU [21, 22] hardness assumption. The third generation begins with the scheme of Gentry et al. [23]. Third-generation schemes are usually slightly less efficient than second-generation ones, but they can be based on somewhat weaker hardness assumptions.

In the homomorphic evaluation of FHE, different circuits in the real world are realized by multiplication homomorphic, addition homomorphic, and decryption homomorphic (ciphertext refresh). The application of FHE is ciphertext arithmetic operation and ciphertext retrieval. Gentry, Halevi, and Smart propose the first evaluation of a complex circuit, i.e., a full AES-128 block evaluation [24] by using a BGV [20] style scheme. The scheme makes use of batching [25, 26], key switching, and modulus switching techniques to obtain an efficient levelled implementation. Chen Y. et al. [27] propose the integer arithmetic over ciphertext and homomorphic data aggregation by using a BGV style scheme. The scheme uses the HElib library to implement homomorphic evaluation for addition, subtraction, multiplication, and division of unsigned integrins. Gai K. et al. [28] propose the blend arithmetic operations on tensor-based FHE over real numbers and proposes a novel tensor-based FHE solution [29]. Yang J. et al. [30] propose the secure tensor decomposition using FHE scheme.

In [24], AES-128 block evaluation cannot implement carry operation. In [27], the integer arithmetic over ciphertext just implement 2-4 bits arithmetic operation of unsigned integer. In [28], the blend arithmetic operations do not implement division operation over tensor-based FHE over real numbers. Therefore, the [24, 27, 28] cannot be used as a complete homomorphic evaluation of a signed integer arithmetic operations scheme by using addition and multiplication homomorphism in MEC. The homomorphic evaluation of integer arithmetic operations is an important foundation for nontrivial computations with the ciphertext data. Therefore, it is significant to construct homomorphic evaluation of integer arithmetic operations scheme.

Contribution. We propose the homomorphic evaluations of integer arithmetic operations base on DGHV [14] and its variants [15, 31, 32] in mobile edge computing. And we use the features of homomorphic encryption to prevent sensitive data from being stolen. At the same time, we can also calculate ciphertext data in mobile edge computing.

(i) Our scheme refers to the integer operation rules which are expressed in Boolean polynomials (BP) that only contains logical AND, XOR operations. Then, we convert BP into homomorphic polynomials (HP) by using addition and multiplication on ciphertexts.

(ii) We propose judgment choose Boolean polynomials (JCBP) to solve the constantly choose problem in the multiplication and division. Then we convert JCBP into judgment choose homomorphic polynomials (JCHP) by using addition and multiplication on ciphertexts.

(iii) We optimise the process of homomorphic evaluation of integer arithmetic operations. We divide the ciphertext vector of integer encryption into subvectors of length 2 and add the length of the private key of FHE to support the 3-multiplication level additional, except for the 15-multiplication level required by bootstrapping.

(iv) We test the optimized scheme in DGHV [14] and CMNT [17]. In the number of ciphertext refreshes, the optimized scheme is reduced by 2/3 compared to the original scheme, and the homomorphic evaluation time overhead of integer arithmetic operations is reduced by 1/3. We also test our scheme in CNT [31] of without bootstrapping. The time overhead of optimized scheme over DGHV [14] and CMNT [17] is close to the original scheme over CNT [31].

Organization. In Section 2, we will introduce DGHV [14], the variants of DGHV [17, 31, 32] and some homomorphic evaluation in detail. In Section 3, we modify the polynomials of integer arithmetic operation according to the computation process of complement, addition, subtraction, multiplication, and division of integer in the computer. We also propose the BP of integer arithmetic operation and convert the BP into the HP of integer arithmetic operation. In Section 4, we analyse the noise ceiling and optimize the process of homomorphic evaluation of integer arithmetic operations. In Section 5, we show our implementation and experimental result. We show the efficiency and conclusion of our scheme.

2. Related Work

Fully homomorphic encryption scheme over the integers and its variants are an essential branch of homomorphic encryption research. Also, we propose the homomorphic evaluations of integer arithmetic operations base on DGHV and its variants. We use DGHV and its variants to encrypt data and upload ciphertext data to a MEC data center. The server of MEC can process ciphertext data by using features of homomorphic encryption. Our scheme involves homomorphic encryption and ciphertext computing. Therefore, we will introduce the original homomorphic encryption scheme DGHV [14] and its variants on integer, including shorter public keys CMNT [17], public keys compress and modulus switching CNT [31], batch encryption CCKL+

[32]. Below we will replace [14, 17, 31, 32] with DGHV, CMNT, CNT, and CCKL+. At the same time, we will also introduce some ciphertext computing techniques related to our scheme, including full AES-128 block evaluation [24] by using a BGV style scheme, integer arithmetic over ciphertext and homomorphic data aggregation [27], the blend arithmetic operations on tensor-based FHE over real numbers [28].

2.1. DGHV and Variants Scheme. The DGHV scheme is described by van Dijk et al. base on integer and to simplify [12]. The advantages of DGHV include simple encryption process and being easy to understand. However, it has a weakness that the noise in ciphertext increases rapidly with the multiplicative level increases. The scheme is based on a set of public integers: $x_i = q_i p + r_i$, $0 \le i \le \tau$, where the integer p is secret. We use the same notation as in DGHV. The DGHV use the following parameters (all polynomial in the security parameter λ):

(i) γ is the bit-length of the x_i's.

(ii) η is the bit-length of private key p.

(iii) ρ is the bit-length of the noise r_i.

(iv) τ is the number of x_i's in the public key.

(v) ρ' is used for encryption.

For a specific η-bit odd integer p, DGHV use the following distribution over γ-bit integers:

$$\mathscr{D}_{\gamma,\rho(p)} = \left\{ Choose\ q \longleftarrow \mathbb{Z} \cap \left[0, \frac{2^\gamma}{p}\right),\ r \longleftarrow \mathbb{Z} \cap \left(-2^\rho, 2^\rho\right) : Output\ x = q \cdot p + r \right\} \tag{1}$$

***KeyGen*(λ).** Generate a random odd integer p of size η bits as a sk. For the public key, sample $x_i \longleftarrow \mathscr{D}_{\gamma,\rho(p)}$ for $0 \le i \le \tau$. Relabel so that x_0 is the largest. Let x_0 be odd and $[x_0]_p$ even. Let $pk = \langle x_0, x_1, \ldots, x_\tau \rangle$ and $sk = p$.

***Encrypt*($pk, m \in \{0,1\}$).** Choose a random subset $S \subseteq \{1, 2, \ldots, \tau\}$ and a random integer $r \in (-2^\rho, 2^\rho)$, and output $c \longleftarrow [m + 2r + 2\sum_{i \in S} x_i]_{x_0}$.

***Decrypt*(sk, c).** Output $m' = ((c) \bmod p) \bmod 2 = (c \bmod 2)(\lfloor c/p \rceil \bmod 2)$.

***Evaluate*($pk, C, c_1, c_2, \ldots, c_t$).** Given the function F with t input, and t ciphertexts c_i, convert logic AND and logic XOR of F into addition and multiplication, performing all the addition and multiplication, and return the resulting integer.

The following parameter set is suggested in DGHV: $\rho = \lambda$, $\rho' = 2\lambda, \eta = \widetilde{\mathcal{O}}(\lambda^2), \gamma = \widetilde{\mathcal{O}}(\lambda^5), \tau = \gamma + \lambda$. The public key size is then $\widetilde{\mathcal{O}}(\lambda^{10})$.

The CMNT scheme is described by Coron et al. to reduce the public key size of the DGHV scheme from $\widetilde{\mathcal{O}}(\lambda^{10})$ down to $\widetilde{\mathcal{O}}(\lambda^7)$. CMNT scheme applies a new parameter β by the form $x'_{ij} = x_{i,0} \bullet x_{j,1} \bmod x_0$, $1 \le i, j \le \beta$ to generate the $\tau = \beta^2$ integers x'_{ij} used for encryption. CMNT scheme enables reducing the public key size from τ down to roughly $2\sqrt{\tau}$ integers of bits. CMNT scheme uses an error-free x_0, that is, $x_0 = q_0 p$, since otherwise, the error would grow too large. Additionally, for encryption CMNT scheme consider a linear combination of the x'_{ij} with a coefficient vector $\boldsymbol{b} = (b_{i,j})$ instead of bits; this enables reducing the public key size further. The vector $\boldsymbol{b}$ with components in $[0, 2^\alpha)$.

The CMNT scheme takes $\rho = \lambda$, $\eta = \widetilde{\mathcal{O}}(\lambda^2)$, and $\gamma = \widetilde{\mathcal{O}}(\lambda^5)$ as in the DGHV scheme. However, it takes $\alpha = \lambda$, $\beta^2 = \widetilde{\mathcal{O}}(\lambda^2)$, and $\rho' = 4\lambda$. The main difference is that instead of having $\tau = \widetilde{\mathcal{O}}(\lambda^5)$ integers x_i's, CMNT scheme has only $2\beta = \widetilde{\mathcal{O}}(\lambda^2)$ integers x_i. Hence the public key size becomes $\widetilde{\mathcal{O}}(\lambda^7)$ instead of $\widetilde{\mathcal{O}}(\lambda^{10})$.

Coron and Naccache et al. describe the CNT scheme. The CMT describes a method that can compress the public key size of the DGHV scheme and optimise the noise management technique by modifying the modulus switching technology [20]. The noise ceiling of CNT scheme increases only linearly with the multiplicative level instead of exponentially. So, a levelled DGHV variant was implemented. This scheme gives two optimisations for homomorphic encryption on DGHV as below.

The first optimisation is public keys compression. First generate the secret key p of size η bits and use a pseudo-random function f with random seed se to generate a set of $\chi_i \in [0, 2^\gamma)$, $1 \le i \le \tau$. Finally, compute δ_i that $x_i = \chi_i - \delta_i$ is small modulo p and store δ_i in the public key, instead of the full x_i's.

The second optimisation is Modulus-Switching Technique. The CMNT show how to adapt Brakerski, Gentry, and Vaikuntanathan's (BGV) FHE framework [20] to the DGHV scheme over the integers. Under the [20] framework, the noise ceiling increases only linearly with multiplicative depth, instead of exponentially.

The CNT scheme takes $\rho = \lambda$, $\eta = \widetilde{\mathcal{O}}(\lambda^2)$, $\gamma = \widetilde{\mathcal{O}}(\lambda^5)$, $\alpha = \widetilde{\mathcal{O}}(\lambda^2)$, $\tau = \widetilde{\mathcal{O}}(\lambda^3)$, and $\rho' = \widetilde{\mathcal{O}}(\lambda^2)$. The new public key of CNT scheme has size $\gamma + \tau \bullet (\eta + \lambda) = \widetilde{\mathcal{O}}(\lambda^5)$ instead of $\widetilde{\mathcal{O}}(\lambda^{10})$ of DGHV.

J.H. Cheon et al. describe the CCKL+ scheme. It extends DGHV to support the same batching capability as in RLWE-based schemes [20, 25], and to homomorphically evaluate a full AES circuit with roughly the same level of efficiency as [24]. The CCKL+ scheme is a merger of two independent works [33, 34] built on the same basic idea but with different contributions. Its security under the (stronger) Error-Free Approximate-GCD assumption already DGHV, CMNT, and CNT.

The CCKL+ scheme extends the DGHV scheme by packing ℓ plaintexts $m_0, \cdots m_{\ell-1}$ into a single ciphertext, using the Chinese Remainder Theorem (CRT). For somewhat homomorphic encryption, this allows us to encrypt not only bits but elements from rings of form $\mathbb{Z}_Q$. The CKLL+ scheme takes $\rho = 2\lambda$, $\eta = \widetilde{\mathcal{O}}(\lambda^2)$, $\gamma = \widetilde{\mathcal{O}}(\lambda^5)$, $\alpha = \widetilde{\mathcal{O}}(\lambda^2)$, and $\tau = \widetilde{\mathcal{O}}(\lambda^3)$ as in CNT scheme, with $\rho' = \widetilde{\mathcal{O}}(\lambda)$, $\alpha' = \widetilde{\mathcal{O}}(\lambda^2)$, and $\ell = \widetilde{\mathcal{O}}(\lambda^2)$.

Table 1: The parameters of DGHV, CMNT, CNT and CCKL+, and public key storage size.

FHE scheme	λ	ρ'	η	γ	pk length	pk size
DGHV	λ	2λ	$\widetilde{\mathcal{O}}(\lambda^2)$	$\widetilde{\mathcal{O}}(\lambda^5)$	$\widetilde{\mathcal{O}}(\lambda^{10})$	41 GB
CMNT	λ	4λ	$\widetilde{\mathcal{O}}(\lambda^2)$	$\widetilde{\mathcal{O}}(\lambda^5)$	$\widetilde{\mathcal{O}}(\lambda^7)$	800 MB
CNT	λ	$\widetilde{\mathcal{O}}(\lambda^2)$	$\widetilde{\mathcal{O}}(\lambda^2)$	$\widetilde{\mathcal{O}}(\lambda^5)$	$\widetilde{\mathcal{O}}(\lambda^5)$	10.1 MB
CCKL+	λ	3λ	$\widetilde{\mathcal{O}}(\lambda^2)$	$\widetilde{\mathcal{O}}(\lambda^5)$	$\widetilde{\mathcal{O}}(\lambda^8)$	5.6 GB

We can conclude (see Table 1) that the CNT scheme performs best in public key storage and only 10.1 MB. We test public key size of DGHV by using "large" parameters of CMNT, and up to 41 GB. The CNT scheme has better noise management technique, in which the noise ceiling increases only linearly with the multiplicative level. Therefore, the CNT scheme supports more multiplication level than other schemes. The CCKL+ scheme implements batch FHE scheme to encrypt a plaintext vector to a ciphertext by using the CRT. However, each one of the components in the plaintext vector is required to be independent. If we encrypt a plaintext vector, the encryption result is a plaintext vector using DGHV, CMNT, and CNT scheme, and the encryption result is a ciphertext by using CCKL+. Therefore, when we do arithmetic operations on the ciphertext of a plaintext vector, we can perform carry operation by using DGHV, CMNT, and CNT scheme, instead of using CCKL+.

2.2. Homomorphic Evaluation. The advantage of homomorphic evaluation is implementing various operations in the real world on ciphertext and is not only multiplication homomorphic, addition homomorphic, and decryption homomorphic (ciphertext refresh). The FHE abstracts various operations of the real world as a collection of circuits consisting of logical XOR and logical AND. The homomorphic evaluation is to implement different circuits in this collection of circuits. Below we will introduce some relevant schemes for homomorphic evaluation of arithmetic operations.

Gentry, Halevi, and Smart propose the first evaluation of a complex circuit, i.e., a full AES-128 block evaluation [24] by using a BGV [20] style scheme. The scheme makes use of batching [25, 26], key switching, and modulus switching techniques to obtain an efficient levelled implementation. After that, Gentry, Smart, and Halevi publish significantly improved runtime results. Compared to the earlier implementation [21], Gentry et al. use the latest version of the HElib library [35]. Two variations of the implementation are reported: one with bootstrapping and one without bootstrapping.

Chen Y. et al. [27] propose the integer arithmetic over ciphertext and homomorphic data aggregation by using a BGV [20] style scheme. The scheme uses the HElib library [35] to implement homomorphic evaluation for addition, subtraction, multiplication, and division of unsigned integrins. However, the scheme report time overhead of integer arithmetic over ciphertext without bootstrapping and modulus switching (somewhat homomorphic encryption). The length of integer ciphertext only sets 2, 3, 4 bits and sets 128 security level to guarantee right result. The scheme does not optimise the operations of integer arithmetic over ciphertext with bootstrapping and modulus switching.

Gai K. et al. [28] propose the blend arithmetic operations on tensor-based FHE over real numbers and propose a novel tensor-based FHE solution [29]. The scheme uses tensor laws to carry the computations of blend arithmetic operations over real numbers. However, blend arithmetic operations only include addition and multiplication. The scheme does not implement blend arithmetic operations with the division.

3. Homomorphic Evaluation of the Integer Arithmetic Operations

Integer addition, subtraction, multiplication, and division are operated by complement addition and shift. One bit full-adder uses XOR ($\bigoplus$) gate to get sum and uses AND ($\wedge$) gate to get carry. Below we will explain our notation. The integer arithmetic operations will take $\mathscr{A} = a_{n-1} \cdots a_0$, $\mathscr{B} = b_{n-1} \cdots b_0$ and $\mathscr{A}^* = a^*_{n-1} \cdots a^*_0$, $\mathscr{B}^* = b^*_{n-1} \cdots b^*_0$ as inputs. $\mathscr{A}$ and $\mathscr{B}$ are the complements. $\mathscr{A}^*$ and $\mathscr{B}^*$ are two's complement of $\mathscr{A}$ and $\mathscr{B}$. However, in complement operation, $\mathscr{A}$ is original code, $\mathscr{A}^*$ is the complement of the $\mathscr{A}$. The $\mathfrak{A} = \langle \mathfrak{a}_{n-1}, \ldots, \mathfrak{a}_0 \rangle$ represents ciphertext vector of $\mathscr{A}$. Let $\mathfrak{a}_i = Enc(a_i)$, $0 \le i \le n-1$. The $\mathfrak{A}^* = \langle \mathfrak{a}^*_{n-1}, \ldots, \mathfrak{a}^*_0 \rangle$ represents ciphertext vector of $\mathscr{A}^*$, $\mathfrak{a}^*_i = Enc(a^*_i)$, $0 \le i \le n-1$. The $\mathfrak{B}$ represents the ciphertext vector of $\mathscr{B}$, $\mathfrak{b}_i = Enc(b_i)$, $0 \le i \le n-1$. The $\mathfrak{B}^* = \langle \mathfrak{b}^*_{n-1}, \ldots, \mathfrak{b}^*_0 \rangle$ represents ciphertext vector of $\mathscr{B}^*$, $\mathfrak{b}^*_i = Enc(b^*_i)$, $0 \le i \le n-1$. The $\mathfrak{A}$, $\mathfrak{B}$, $\mathfrak{A}^*$, and $\mathfrak{B}^*$ are inputs of homomorphic evaluation of the integer arithmetic operations. The n is big enough. We do not consider the overflow about integer arithmetic operations.

3.1. Homomorphic Evaluation of the Complement Operations. Fixed-point number use the complement to finish arithmetic operations. The rules of converting original code into complement:

(i) Positive number: a positive complement and the same original code.

(ii) Negative number: negative complement is the symbol for the numerical bit reverse and then at the bottom (LSB) plus 1.

Given the original code $\mathscr{A}$, apply the XOR and AND to get complement $\mathscr{A}^*$. The MSB a_{n-1} is the signed bit of $\mathscr{A}$, and the size of the remaining bits $a_{n-2} \cdots a_0$ represent the value.

Given an initialised carry $c_{-1} = 0$, we output the $\mathcal{A}^*$. The BP of complement:

$$\begin{aligned} c_{-1} &= 0 \\ c_i &= a_i \vee c_{i-1} \\ a_i^* &= \left(a_i \oplus a_n c_{i-1}\right) \\ & \qquad 0 \le i \le n-2 \end{aligned} \tag{2}$$

We convert c_i into a BP by using XOR gate and AND gate. The BP: $c_i = a_i c_{i-1} \oplus a_i \oplus c_{i-1}$, $0 \le i \le n-2$. Due to initialised carry $c_{-1} = 0$, we can convert c_i into a polynomial without c_{-1}:

$$c_i = \sum_{k=1}^{i+1} \sum_{|\mathcal{S}|=i+1} \prod_{j \in S} a_j \bmod 2 \tag{3}$$

where $\mathcal{S} = \{a_i, \cdots a_0\}$, $0 \le i \le n-3$, $|\mathcal{S}|$ is the Hamming weight of the $\mathcal{S}$. We can convert above polynomials into HP of complement by using addition and multiplication on ciphertexts. The HP of complement:

$$\begin{aligned} \mathfrak{c}_i &= \left(\sum_{k=1}^{i+1} \sum_{|\mathcal{S}|=i+1} \prod_{j \in S} \mathfrak{a}_j \right) \bmod x_0 \\ \mathfrak{a}_i^* &= \left(\mathfrak{a}_i + \mathfrak{a}_n^* \bullet \mathfrak{c}_{i-1}\right) \bmod x_0 \\ & \qquad 0 \le i \le n-2 \end{aligned} \tag{4}$$

where $\mathfrak{c}_i$ represents the ciphertext result of c_i, and $Dec(\mathfrak{c}_i) = c_i$. Let $\mathfrak{a}_{n-1}^* = \mathfrak{a}_{n-1}$. The x_0 is the largest odd public key in FHE. We can get ciphertext complement $\mathfrak{A}^*$ by using above HP.

3.2. Homomorphic Evaluation of the Addition and Subtraction Operations. Integer complement addition operation needs to calculate results in order from low to high. Every result bit requires an addend bit, an augend bit, and a carry bit from the low. Every carry bit requires an addend bit, an augend bit, and a carry bit from low as well. By iterating above operations, we can get the result of complement addition. We set integer complement $\mathcal{A}$ and $\mathcal{B}$ as inputs to calculate $\mathcal{S} = \mathcal{A} + \mathcal{B}$, where $\mathcal{S} = s_{n-1} \cdots s_0$. The n is big enough. We do not consider the overflow of the $\mathcal{S}$. The BP of addition:

$$\begin{aligned} c_{-1} &= 0 \\ s_i &= a_i \oplus b_i \oplus c_{i-1} \\ c_i &= a_i b_i \oplus c_{i-1}\left(a_i \oplus b_i\right) \\ & \qquad 0 \le i \le n-1 \end{aligned} \tag{5}$$

We can convert above BP into HP of addition by using addition and multiplication on ciphertexts. The HP of addition:

$$\begin{aligned} \mathfrak{c}_{-1} &= Enc(0) \\ \mathfrak{s}_i &= \left(\mathfrak{a}_i + \mathfrak{b}_i + \mathfrak{c}_{i-1}\right) \bmod x_0 \\ \mathfrak{c}_i &= \left(\mathfrak{a}_i \mathfrak{b}_i + \mathfrak{c}_{i-1}\mathfrak{a}_i + \mathfrak{c}_{i-1}\mathfrak{b}_i\right) \bmod x_0 \\ & \qquad 0 \le i \le n-1 \end{aligned} \tag{6}$$

where the $\mathfrak{s}_i$ is i-th ciphertext of result vector $\mathfrak{S} = \langle \mathfrak{s}_{n-1}, \cdots \mathfrak{s}_0 \rangle$, and $Dec(\mathfrak{A}) + Dec(\mathfrak{B}) = Dec(\mathfrak{S})$.

The addition operation can do integer subtraction operation. If we calculate $\mathcal{A} - \mathcal{B}$, we can convert $\mathcal{B}$ to $\mathcal{B}^*$, and calculate $\mathcal{A} + \mathcal{B}^*$ by using integer addition operation. In order to get $\mathcal{B}^*$, we need to use the complement operation to get $\mathcal{B}' = b'_{n-1} \cdots b'_0$, and then let $b^*_{n-1} = b'_{n-1} \oplus 1$, $b_i^* = b_i'$, $0 \le i \le n-2$. The HP of subtraction contains two parts, including HP of complement and HP of addition.

3.3. Homomorphic Evaluation of the Multiplication Operations. Integer multiplication operation is based on Booth's multiplication algorithm [36]. It is a multiplication algorithm that multiplies two signed numbers in two's complement notation. We set multiplicand $\mathcal{A}$, and multiplier $\mathcal{B}$. Booth's algorithm examines adjacent pairs of bits of the multiplier $\mathcal{B}$ in signed two's complement representation, including an implicit bit below the least significant bit, $b_{-1} = 0$. Also, we denote by $\mathcal{P}$ the product accumulator. The steps of the basic algorithm for multiplication operations:

(1) We reinitialise the value of $\mathcal{A}$, $\mathcal{A}^*$, and $\mathcal{P}$.

- (i) $\mathcal{A}$: $\mathcal{A} = \mathcal{A} \ll n$, arithmetic left shift $(n+1)$ bits. $\mathcal{A} = a_n a_{n-1} \cdots a_0 0 \cdots 0$.
- (ii) $\mathcal{A}^*$: $\mathcal{A}^* = \mathcal{A}^* \ll n$, arithmetic left shift $(n+1)$ bits. $\mathcal{A}^* = a^*_{n-1} \cdots a^*_0 0 \cdots 0$.
- (iii) $\mathcal{P}$: fill the most significant n bits with 0. To the right of this, append the value of $\mathcal{B}$. Fill the LSB with a 0. $\mathcal{P} = 0 \cdots 0 b_{n-1} \cdots b_0 0$.

(2) Determine the two least significant (rightmost) bits of $\mathcal{P}$.

- (i) If $b_{-1} = b_0$, do nothing. Use $\mathcal{P}$ directly in the next step. Arithmetic right shift 1 bit.
- (ii) If $b_0 b_{-1} = 01$, find the value of $\mathcal{P} = \mathcal{P} + \mathcal{A}$. Ignore any overflow. Arithmetic right shift 1 bit.
- (iii) If $b_0 b_{-1} = 10$, find the value of $\mathcal{P} = \mathcal{P} + \mathcal{A}^*$. Ignore any overflow. Arithmetic right shift 1 bit.

Repeat above second steps until they have been done $n-1$ times. Drop the LSB from $\mathcal{P}$. According to second steps mentioned technique, we can summarise a judgment choice Boolean polynomial (JCBP):

$$\begin{aligned} &\mathrm{JCBP}_{mult}\left(b_0, b_{-1}, \mathcal{A}^*, \mathcal{A}\right) \\ &\quad = \left(b_0 \oplus b_{-1}\right)\left[b_0 \mathcal{A}^* + b_{-1}\mathcal{A}\right] \end{aligned} \tag{7}$$

We use $\mathcal{P}_i$ to represent i-th times iteration. The BP of multiplication operation:

$$\begin{aligned} \mathcal{P}_i &= \mathcal{P}_{i-1} + \mathrm{JCBP}_{mult}\left(b_0, b_{-1}, \mathcal{A}^*, \mathcal{A}\right) \\ \mathcal{P}_i &= \mathcal{P}_{i-1} \gg 1 \\ & \qquad 0 \le i < n-1 \end{aligned} \tag{8}$$

where $\gg$ represent the arithmetic right shift. We can convert formula (7) into HP of addition by using addition and multiplication on ciphertexts. The HP of $\mathrm{JCBP}_{mult}(\mathscr{b}_0, \mathscr{b}_{-1}, \mathscr{A}^*, \mathscr{A})$:

$$\begin{aligned}&\mathrm{JCHP}_{mult,x_0,\rho'}\left(\mathfrak{b}_0, \mathfrak{b}_{-1}, \mathfrak{A}^*, \mathfrak{A}, \boldsymbol{r}, switch\,(\oplus, \wedge)\right)\\&\quad = \left[\left(\mathfrak{b}_0+\mathfrak{b}_{-1}\right)\left(\mathfrak{b}_0\mathfrak{A}^* + \mathfrak{b}_{-1}\mathfrak{A}\right) + 2\boldsymbol{r}\right] \bmod x_0\end{aligned} \tag{9}$$

The $\mathfrak{A}$ and $\mathfrak{A}^*$ represent ciphertext vector of reinitialising $\mathscr{A}$ and $\mathscr{A}^*$. The $\boldsymbol{r} = \langle r_0, \ldots, r_{n-1}\rangle$ is a noise vector, and $r_i \longleftarrow \mathbb{Z} \cap (-2^{\rho'}, 2^{\rho'})$, $0 \le i \le n-1$. The HP of multiplication:

$$\begin{aligned}&\mathfrak{P}_i\\&\quad = \mathfrak{P}_{i-1}\\&\qquad + \mathrm{JCHP}_{mult,x_0,\rho'}\left(\mathfrak{b}_0, \mathfrak{b}_{-1}, \mathfrak{A}^*, \mathfrak{A}, r, switch\,(\oplus, \wedge)\right)\end{aligned} \tag{10}$$

$$\mathfrak{P}_i = \mathfrak{P}_{i-1} \sim\!\gg 1 \qquad 0 \le i < n-1$$

where $\sim\!\gg$ represent the right shift of ciphertext vector $\mathfrak{P}_{n-1}$. When the right shift of $\mathfrak{P}_{n-1}$ by 1 ciphertext slot, the most significant component of $\mathfrak{P}_{n-1}$ is filled with a copy of the original most significant component. The final value of $\mathfrak{P}_{n-1}$ is the signed ciphertext product.

3.4. Homomorphic Evaluation of the Division Operation. Division is the most complex of the basic arithmetic operations. For a simple computer that operate with an adder circuit for its arithmetic operations, a variant using traditional long division, called nonrestoring division, provides a simpler and faster speed. This method only needs one decision and addition/subtraction per quotient bit, and need not restoring step after the subtraction. We set dividend $\mathscr{A}$ and divisor $\mathscr{B}$. The $\mathscr{B}^*$ is two's complement of $\mathscr{B}$. The $\mathscr{R}$ is the partial remainder, and the $\mathscr{Q}$ is quotient. The basic algorithm for binary (radix 2) nonrestoring division is as follows:

(1) Reinitialize value of $\mathscr{B}$, $\mathscr{B}^*$, $\mathscr{R}$, and $\mathscr{Q}$.

(i) $\mathscr{B}$: $\mathscr{B} = \mathscr{B} \gg n$, do arithmetic left shift n bits. $\mathscr{B} = \mathscr{b}_{n-1} \cdots \mathscr{b}_0 0 \cdots 0$.

(ii) $\mathscr{B}^*$: $\mathscr{B}^* = \mathscr{B}^* \gg n$, do arithmetic left shift n bits. $\mathscr{B}^* = \mathscr{b}^*_{n-1} \cdots \mathscr{b}^*_0 0 \cdots 0$.

(iii) $\mathscr{R}$: $\mathscr{R} = \mathscr{A} \gg n$, do arithmetic right shift n bits. $\mathscr{R} = \mathscr{r}_{2n-1} \cdots \mathscr{r}_0$.

(iv) $\mathscr{Q}$: $\mathscr{Q} = \mathscr{q}_{n-1} \cdots \mathscr{q}_0$, fill it n bits with 0.

(2) Determine the one most significant (a signed bit) bit of $\mathscr{R}$.

(i) If $\mathscr{r}_{2n-1} = 0$, fill the LSB of $\mathscr{Q}$ with 1 digit, do logical left shift 1 bit. Find the value of $\mathscr{R} = 2 * \mathscr{R} + \mathscr{B}^*$.

(ii) If $\mathscr{r}_{2n-1} = 1$, fill the LSB of $\mathscr{Q}$ with 0 digit, do logical left shift 1 bit. Find the value of $\mathscr{R} = 2 * \mathscr{R} + \mathscr{B}$.

(3) Repeat above second steps until they have been done $n-1$ times.

(4) Convert the quotient $\mathscr{Q}$. We suppose original $\mathscr{Q} =$ 11101010.

(i) Start: $\mathscr{Q} = 11101010$.

(ii) Mask the zero term (Signed binary notation with one's complement): $\overline{\mathscr{Q}} = 00010101$.

(iii) Subtract $\mathscr{Q} = \mathscr{Q} - \overline{\mathscr{Q}}$: $\mathscr{Q} = 11010101$.

(5) The actual remainder is $\mathscr{R} = \mathscr{R} \gg n$. Final result of quotient is always odd, and the remainder $\mathscr{R}$ is in the range $-\mathscr{B} < \mathscr{R} < \mathscr{B}$. To convert to a positive remainder, do a single restoring step after $\mathscr{Q}$ is converted from a nonstandard form to standard form. If $\mathscr{R} < 0$, find the value of $\mathscr{Q} = \mathscr{Q} - 1$ and $\mathscr{R} = \mathscr{R} + \mathscr{B}$.

According to the second step mentioned technique, we can summarise a judgment choice Boolean polynomial (JCBP):

$$\mathrm{JCBP}_{div}\left(\mathscr{r}_{2n-1}, \mathscr{B}, \mathscr{B}^*\right) = \mathscr{r}_{2n-1}\mathscr{B} + \left(\mathscr{r}_{2n-1} \oplus 1\right)\mathscr{B}^* \tag{11}$$

We use $\mathscr{R}_i$ to represent i-th times iteration. The BP of division operation:

$$\begin{aligned}&\mathscr{R}_i = 2\mathscr{R}_{i-1} + \mathrm{JCBP}_{div}\left(\mathscr{r}_{2n-1}, \mathscr{B}, \mathscr{B}^*\right) \qquad 0 \le i < n-1\\&\mathscr{q}_{n-1-i} = \mathscr{r}_{i,2n-1} \oplus 1 \quad 0 \le i < n-1\\&\mathscr{Q} = \mathscr{Q} - \overline{\mathscr{Q}}\\&\mathscr{R}_{n-1} = \mathscr{R}_{n-1} \gg n\end{aligned} \tag{12}$$

where $\mathscr{r}_{i,2n-1}$ represent the MSB of $\mathscr{R}_i$. Finally, doing the fifth step corrects $\mathscr{Q}$ and $\mathscr{R}_{n-1}$. We can convert above BP into HP of addition by using addition and multiplication on ciphertexts. The HP of $\mathrm{JCBP}_{div}(\mathscr{r}_{2n-1}, \mathscr{B}, \mathscr{B}^*)$:

$$\begin{aligned}&\mathrm{JCBP}_{div,,x_0,\rho'}\left(\mathfrak{r}_{2n-1}, \mathfrak{B}^*, \mathfrak{B}, \boldsymbol{r}, switch\,(\oplus, \wedge)\right)\\&\quad = \left[\mathfrak{r}_{2n-1}\mathfrak{B} + \left(\mathfrak{r}_{2n-1} + Enc\,(1)\right)\mathfrak{B} + 2\boldsymbol{r}\right] \bmod x_0\end{aligned} \tag{13}$$

where $\mathfrak{B}$ and $\mathfrak{B}^*$ represent ciphertext vector of reinitialising $\mathscr{B}$ and $\mathscr{B}^*$, respectively. The HP of division:

$$\begin{aligned}&\mathfrak{R}_i\\&\quad = 2\mathfrak{R}_{i-1}\\&\qquad + \mathrm{JCBP}_{div,,x_0,\rho'}\left(\mathfrak{r}_{2n-1}, \mathfrak{B}^*, \mathfrak{B}, r, switch\,(\oplus,\wedge)\right)\\&\qquad\qquad 0 \le i < n-1\\&\mathfrak{q}_{n-1-i} = \left(\mathfrak{r}_{i,2n-1} + Enc\,(1)\right) \bmod x_0 \quad 0 \le i < n-1\\&\mathfrak{Q} = \mathfrak{Q} - \overline{\overline{\mathfrak{Q}}}\\&\mathfrak{R}_{n-1} = \mathfrak{R}_{n-1} \sim\!\gg n\end{aligned} \tag{14}$$

where $\mathfrak{R}_i = \langle \mathfrak{r}_{i,2n-1}, \cdots \mathfrak{r}_{i,0}\rangle$ represents ciphertext vector of $\mathscr{R}_i$ and $\mathfrak{Q}$=$\langle \mathfrak{q}_{n-1} \cdots \mathfrak{q}_0\rangle$ represents ciphertext vector of $\mathscr{Q}$. $\overline{\overline{\mathfrak{Q}}}$

represent $\mathfrak{q}_i = (\mathfrak{q}_i + Enc(1)) \bmod x_0$, $0 \le i < n$. Finally, we need to correct $\mathfrak{Q}$ and $\mathfrak{R}_{n-1}$ by the following operations:

$$\begin{aligned} \mathfrak{Q} &= \left[\mathfrak{r}_{n-1,n-1}\left(\mathfrak{Q} - Enc\left(1\right)\right)\right] \bmod x_0 \\ \mathfrak{R} &= \left[\mathfrak{r}_{n-1,n-1}\left(\mathfrak{R}_{n-1} + \mathfrak{B}\right)\right] \bmod x_0 \end{aligned} \tag{15}$$

The final value of $\mathfrak{Q}$ and $\mathfrak{R}$ is the result of HP of division.

4. Noise Analysis and Optimization

In Section 3, we describe the homomorphic evaluation of the integer arithmetic operations including complement, addition, subtraction, multiplication, and division. In this section, we will analyse the noise ceiling and optimisation for our scheme. Under the DGHV scheme, the noise ceiling increases exponentially with the multiplicative degree. When ciphertexts have noise at most $2^{\eta-2} < p/2$, the ciphertexts cannot be decrypted correctly. Therefore, we need bootstrapping to control noise of ciphertexts, but using bootstrapping to refresh ciphertext will reduce the efficiency of DGHV. We will show the noise ceiling about the homomorphic evaluation of the integer arithmetic operations in this section. Moreover, we will describe an optimisation for the process of integer arithmetic operations to reduce time overhead in FHE with bootstrapping.

4.1. Noise Analysis of Our Scheme. According to Section 3, we show the noise ceiling and items of homomorphic evaluation of the n bits signed integer arithmetic operations. We denote by HE-IAO the homomorphic evaluation of the integer arithmetic operations and use HE-com, HE-add, HE-sub, HE-mul, HE-div to represent five operations of HE-IAO. The details are shown in Table 2.

Proof. According to homomorphic evaluation of the complement operations formula (3),

$$c_i = \sum_{k=1}^{i+1} \sum_{|\mathcal{S}|=i+1} \prod_{j\in S} a_j \bmod 2, \quad 0 \le i \le n-2 \tag{16}$$

$\sum_{|\mathcal{S}|=i} \prod_{j\in S} a_j$ has $\binom{n-2}{i+1}$ terms, and degree amounts to $i+1$. According to binomial theorem, the term of formula (3) can be represented as $(1+x)^{n-2}-1 = \sum_{i=0}^{n-2} \binom{n-2}{i} x^i - 1$, when $x = 1$. Therefore, the polynomial c_i of term is up to $2^{n-2} - 1$, and degree amounts to $n-2$. Because of $a_i^* = (a_i \oplus a_n c_{i-1})$ $0 \le i \le n-1$, a_i^* of term amounts to 2^{n-2}, and degree amounts to $n-1$.

According to homomorphic evaluation of the addition operations formula (5), the degree of carry formula $c_i = a_i b_i \oplus c_{i-1}(a_i \oplus b_i)$ is higher 1 than c_{i-1}, and the term of c_i is higher $2 * term(c_{i-1}) + 1$ than c_{i-1}, where $term(c_{i-1})$ represents the term of c_{i-1}. The degree of $c_0 = a_0 b_0$ amounts to 2, and the term of $c_0 = a_0 b_0$ amounts to 1. The degree of c_{n-2} amounts to n, and terms of c_{n-2} amount to $2^{n-1} - 1$. The MSB of $\mathcal{S}s_{n-1} = a_{n-1} \oplus b_{n-1} \oplus c_{n-2}$ where the degree of s_{n-1} amounts to n, and the term of s_{n-1} amounts to $2^{n-1} + 1$. If we do not consider complement operation, subtraction and addition have the same process. Therefore, the degree and term are the same as the addition. Multiplication and division require iteration $n-1$ times addition and shift. The processes of multiplication and division are particularly complicated, we can't find the formula to express the degree. According to our calculations, the degree of multiplication and division is close to the 2 to the power of $2n-4$. Therefore, the degree of multiplication is not more than $\psi \bullet 2^{2n-4}$, and the degree of division is not more than $\varphi \bullet 2^{2n-4}$. The term of multiplication and division is too high, and the degree has made noise more than the limitation of correct decryption. □

Table 2: The degree ceiling and items of homomorphic evaluation of the n bits signed integer arithmetic operations. Above showing the HP of integer arithmetic operations n-th iterations ciphertext results' degree and term.

HE-IAO	degree	term
HE-com	$n-1$	2^{n-2}
HE-add	n	$2^{n-1}+1$
HE-sub	n	$2^{n-1}+1$
HE-mul	$\psi \bullet 2^{2n-4}$	–
HE-div	$\varphi \bullet 2^{2n-4}$	–

Table 3: The noise ceiling of homomorphic evaluation of the n bits signed integer arithmetic operations. The base of the log is 2.

HE-IAO	noise ceiling
HE-com	$\rho'^{\log(n-1)} + \log 2^{n-3}$
HE-add	$\rho'^{\log(n)} + \log\left(2^{n-1}+1\right)$
HE-sub	$\rho'^{\log(n)} + \log\left(2^{n-1}+1\right)$
HE-mul	$\rho'^{\log\psi \bullet 2^{2n-4}} + -$
HE-div	$\rho'^{\log\varphi \bullet 2^{2n-4}} + -$

We can conclude (see Table 2) the degree of homomorphic evaluation of addition and subtraction is $\widetilde{\mathcal{O}}(n)$, and the depth of the homomorphic evaluation of multiplication and division is $\widetilde{\mathcal{O}}(\psi \bullet 2^{2n-4})$. We use items that represent the l_1 norm of HP (the coefficient vector of HP). The noise ceiling of homomorphic evaluation of the n bits integer arithmetic operations can be calculated by the following polynomial:

$$Noise = \rho'^{\log d} + \log\left|\overrightarrow{f}\right| \tag{17}$$

where d represents the degree of HP and $\log d$ represents multiplication level of HP. $|\overrightarrow{f}|$ is the l_1 norm of HP. ρ' is the noise of length in every ciphertext. The noise ceiling of homomorphic evaluation of the n bits integer arithmetic operations is shown in Table 3.

We can conclude (see Table 3) the noise of homomorphic evaluation of the n bits signed integer arithmetic operations. The noise increases very rapidly. In the homomorphic evaluation of the complement, addition, and subtraction, noise increases up to $\widetilde{\mathcal{O}}(\rho' n)$. In the homomorphic evaluation of the multiplication and division, noise is more than $\widetilde{\mathcal{O}}(\rho' \psi \bullet 2^{2n-4})$.

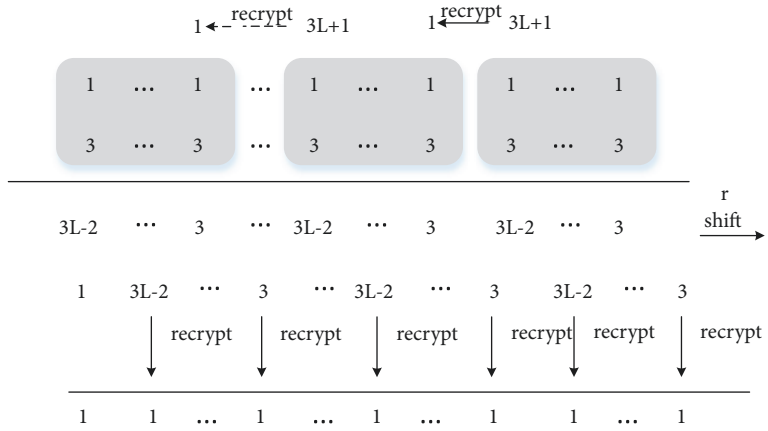

Figure 2: The optimised addition operations; each colour block represents same position subvectors of $\mathfrak{P}_i$ and $\mathfrak{F}$, the length of each colour block is L, "r shift" represents ciphertext vector right shift one position, "recrypt" represents ciphertext refresh. This figure shows the degree change of each component of $\mathfrak{P}_i$ and $\mathfrak{F}$.

4.2. Optimization of Our Scheme. From the analysis of noise ceiling in Section 4.1, we need to control the noise increases by using ciphertext refresh, or modulus-switching technique in each multiplication level of homomorphic evaluation of the n bits signed integer arithmetic operations. However, the ciphertext refresh or modulus-switching technique will reduce our scheme efficiency. Therefore, we will describe an optimisation for our scheme in this section. The optimisation can reduce the number of ciphertext refresh and improve efficiency.

Because the integer arithmetic operation is completed based on the addition operation, we optimise the homomorphic evaluation of the integer addition. We divide the ciphertext vector of integer encryption into subvectors of length L. The homomorphic evaluation of arithmetic operations between the subvectors in the same position do not need ciphertext refresh, and the subvectors in different positions need to refresh ciphertext-carry once. Therefore, the optimised scheme need not ciphertext refresh in each multiplication level and reduces the number of ciphertext refresh. We take addition operations of the product accumulator to explain our optimisation. In the homomorphic evaluation of the multiplication operations, the product accumulator $\mathfrak{P}_i$ is as formula (9). The formula (9) is a polynomial of degree three, and $\mathfrak{P}_i$ is a ciphertext vector of degree one. We denote by $\mathfrak{F}$ vector of $\mathrm{JCHP}_{mult,x_0,\rho'}$. We divide the ciphertext vector of $\mathfrak{P}_i$ and $\mathfrak{F}$ into ciphertext subvectors of length L. The ciphertext operations of subvectors of the same position are the same as homomorphic evaluation of the addition operations (without ciphertext refresh). The ciphertext operations of subvectors of different position need to refresh ciphertext-carry once (see Figure 2).

The subvectors of $\mathfrak{P}_i$ and $\mathfrak{F}$ generate ciphertext-result block and ciphertext-carry. In every ciphertext-result block, the maximum degree is 3L − 2 (leftmost), and the minimum degree is 3 (rightmost). The ciphertext-carry degree is 3L + 1 in each subvector of the different position. Each ciphertext of ciphertext-result block and ciphertext-carry can use ciphertext refresh (recrypt) to reduce degree up to one. The number of ciphertext refresh of once product accumulator ($\mathfrak{P}_i + \mathfrak{F}$) for ciphertext-carry:

(i) If n is divisible by L, it will generate $n/\mathrm{L} - 1$ ciphertext-carry, and the number of ciphertext refresh is $cnt = n/\mathrm{L} - 1$ times.

(ii) If n is inalienable by L, it will generate $\lfloor n/\mathrm{L} \rfloor$ ciphertext-carry, and the number of ciphertext refresh is $cnt = \lfloor n/\mathrm{L} \rfloor$ times.

Homomorphic evaluation of the multiplication operations needs $n-1$ times product accumulator, and whole operations need $(n-1) \bullet cnt$ times ciphertext refresh for ciphertext-carry, and $(n-1) \bullet n$ times ciphertext refresh for ciphertext-result block. Total times of ciphertext refresh for homomorphic evaluation of the multiplication operations:

$$CNT = (n-1) \bullet n + (n-1) \bullet cnt \tag{18}$$

In the original homomorphic evaluation of the multiplication operations, every multiplication level needs once ciphertext refresh. Whole operations need $(n-1)(3n-3) + n(n-1) + 2n - 3$ times ciphertext refresh. We apply above optimisation to the homomorphic evaluation of the complement, addition, subtraction, multiplication, and division operations, and set $n = 16$, $\mathrm{L} = 1, 2, 3, 4$. We show the number of ciphertext refresh of the original scheme ($\mathrm{L} = 0$) and an optimised scheme ($\mathrm{L} = 1, 2, 3, 4$) as Table 4.

In order to implement our optimisation, we need to adjust the parameters of homomorphic encryption and make FHE support the $\log(3\mathrm{L}+1)$ multiplication level additional, except for the 15-multiplication level required by bootstrapping. Therefore, we reset the length of the private key: $\eta \geq \rho' \bullet \Theta(\lambda \log^2 \lambda) + \rho' \bullet 2^{\log(3\mathrm{L}+1)} + \log(2^{\mathrm{L}-1} + 1)$. We set security parameter $\lambda = 52$, the length of noise $\rho' = 24$, and $\theta = 15$, $\Theta = 500$. Parameters are set as Table 5.

Table 4: The number of ciphertext refresh of homomorphic evaluation of the integer arithmetic operations.

HE-IAO	L = 0	L = 1	L = 2	L = 3	L = 4
HE-com	29	29	22	20	18
HE-add	45	31	23	21	19
HE-sub	74	60	45	41	37
HE-mul	944	494	367	335	303
HE-div	1095	585	437	397	359

Table 5: Concrete parameters, based on the "small" parameters of CMNT scheme, and reset the length of public key η and γ. Fixed λ, ρ' and the number of public keys τ, only changed η and γ according to L.

parameters	λ	ρ'	η	γ	τ
L = 0	52	24	1632	$2.0 \bullet 10^6$	1000
L = 1	52	24	1728	$2.5 \bullet 10^6$	1000
L = 2	52	24	1801	$3.0 \bullet 10^6$	1000
L = 3	52	24	1874	$3.5 \bullet 10^6$	1000
L = 4	52	24	1993	$4.2 \bullet 10^6$	1000

5. Experimental Result

Our optimisations described in our scheme were incorporated in our code, which is built on top of GnuMP. We tested our implementation on a desktop computer with Intel Core i5-3470 running at 3.2 GHz, on which we run an Ubuntu 18.04 with 8 GB of RAM and with the gcc compiler version 6.2. We regard this desktop computer as an edge data centre to test our scheme efficiency in edge computing. We choose the ciphertext vector of 16 and 8 bits signed integer as input. Due to our optimisations for the FHE with bootstrapping, we test our original scheme (L = 0) and optimised scheme (L = 1, 2, 3, 4) in DGHV and CMNT scheme (see Figures 3(a), 3(b), 3(c), 3(d), and 3(e)). In CNT scheme, we only use modulus-switching technique to control noise. So, we test the original scheme in CNT scheme (see Figures 4(a) and 4(b)). In our figures, "HE-com (CMNT 16)" represent the homomorphic evaluation of the complement operations and calculate the ciphertext vector of 16 bits signed integer in the CMNT scheme. "HE-com (DGHV 8)" represent the homomorphic evaluation of the complement operations and calculate the ciphertext vector of 8 bits signed integer in DGHV scheme. The explanation of other legends is the same as above. We show the following experimental results based on the Section 4.2 parameter settings.

We can conclude (see Figures 3(b), 3(c), 3(d), and 3(e)) that L = 2 is best parameters setting for homomorphic evaluation of the addition, subtraction, multiplication, and division. However, the time overhead of homomorphic evaluation of the complement operations is monotone increasing at L (see Figure 3(a)). When L = 0, the time overhead is the absolute minimum. When L = 0 and L = 1, the number of ciphertext refresh is the same. Also, the degree of complement operations is $2n - 3$. When $n = 16$, the degree is 29 in L = 0. It is the same as L = 1. The reduced time overhead of the optimised complement operations cannot offset the computation cost caused by the increase in ciphertext length. However, the relative minimum value appears at L = 2 (see Figures 3(a), 3(b), 3(c), 3(d), and 3(e)). It shows that our optimisation is useful and can reduce the time overhead for our scheme, except homomorphic evaluation of the complement operations. Namely, we divide the ciphertext vector of integer encryption into subvectors of length 2 and make DGHV and CMNT scheme to support the $\log(3 \bullet 2 + 1) \approx 3$ multiplication level additional. The optimisation of our scheme can achieve the best results. In the number of ciphertext refreshes, the optimised scheme is reduced by 2/3 compared to the original scheme over DGHV and CMNT, and the homomorphic evaluation time overhead of integer arithmetic operation is reduced by 1/3.

The DGHV and CMNT schemes are different with CNT scheme in noise management technology. The DGHV and CMNT schemes use bootstrapping to control noise, and the CNT scheme uses the modulus switching technology to control noise. Our optimization is used for homomorphic evaluation of integer arithmetic operations which are implemented by DGHV and CMNT schemes (with bootstrapping), rather than by CNT scheme (modulus-switching). Therefore, we just compare the beat result (L = 2) of our scheme based on DGHV and CMNT schemes with CNT scheme (L = 0). We can draw a conclusion (see Figures 4(a) and 4(b)) that the time overhead of our optimised scheme based on DGHV and CMNT schemes (bootstrapping) with L = 2 close to the time overhead of basing on CNT (modulus-switching) with L = 0. It also shows that our optimisation is effective. And our optimized scheme can be applied to mobile edge computing to solve privacy data computing problems in ciphertext.

6. Conclusion

We implement the homomorphic evaluation scheme of integer arithmetic operations under DGHV and its variants in edge computing. We use the features of homomorphic encryption to prevent sensitive data from being stolen in edge computing. At the same time, we can also calculate ciphertext data in edge computing and improve the QoS

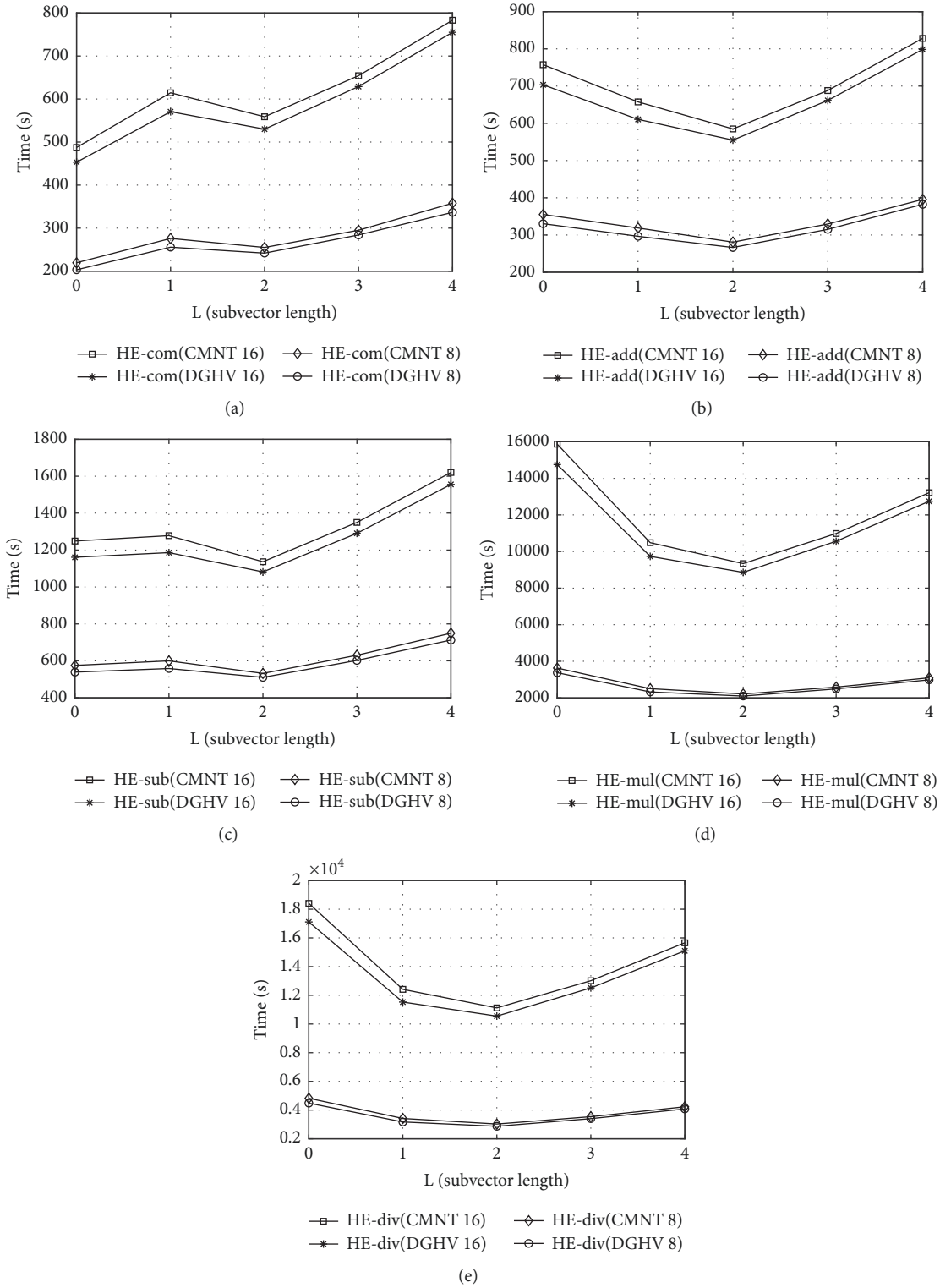

Figure 3: Tested (a) HE-com, (b) HE-add, (c) HE-sub, (d) HE-mul, (e) HE-div CPU time.

of edge computing. Through scheme design, noise analysis, scheme optimisation, and experimental comparison, the different performance of homomorphic evaluation of the integer arithmetic operations is obtained under different FHE schemes. Although we optimise our scheme in Section 4.2, the time overhead of our scheme is still high. The primary open problem is to improve the efficiency of the FHE scheme. This requires us to work together to find a natural FHE scheme or to continue to optimise the FHE architecture to achieve more efficient noise management techniques.

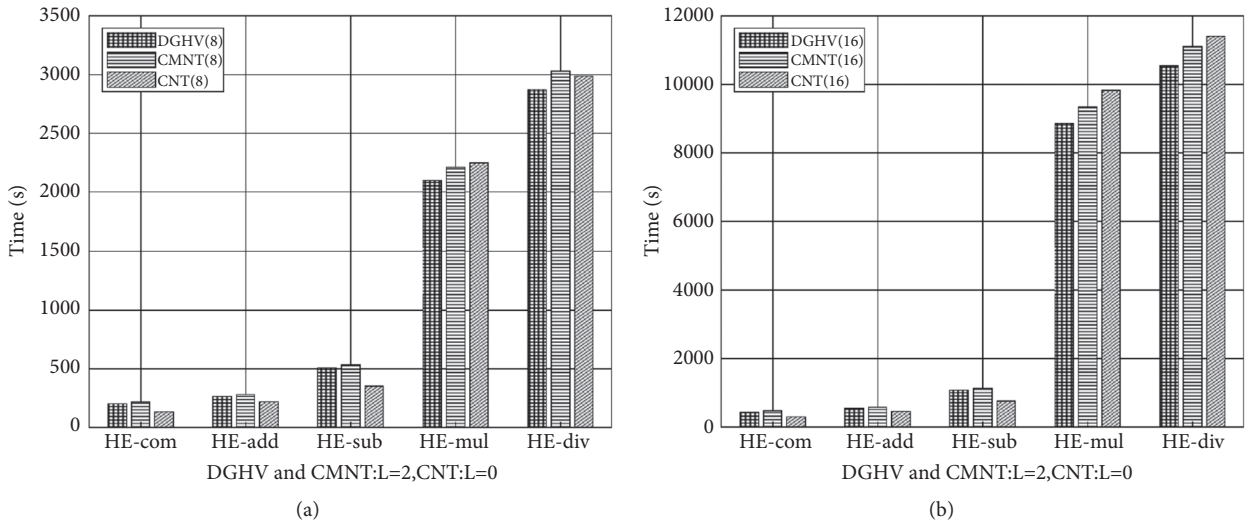

Figure 4: (a) Tested HE-IAO CPU time of ciphertext vector of length 8. (b) Tested HE-IAO CPU time of ciphertext vector of length 16.

Data Availability

The data and code used to support the findings of this study have been deposited in the GitHub repository (https://github.com/limengfei1187/Homomorphic-Encryption.git).

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work is partly supported by the National Science Foundation of China (61701322), the Key Projects of Liaoning Natural Science Foundation (20170540700), and the Liaoning Provincial Department of Education Science Foundation (L201630).

References

[1] L. He, O. Kaoru, and D. Mianxiong, "ECCN: Orchestration of Edge-Centric Computing and Content-Centric Networking in the 5G Radio Access Network," *IEEE Wireless Commun*, vol. 25, no. 3, pp. 88–93, 2018.

[2] M. Tao, K. Ota, and M. Dong, "Foud: Integrating Fog and Cloud for 5G-Enabled V2G Networks," *IEEE Network*, vol. 31, no. 2, pp. 8–13, 2017.

[3] J. Xu, K. Ota, and M. Dong, "Real-Time Awareness Scheduling for Multimedia Big Data Oriented In-Memory Computing," *IEEE Internet of Things Journal*, 2018.

[4] P. G. Lopez, A. Montresor, D. Epema et al., "Edge-centric computing: vision and challenges," *Computer Communication Review*, vol. 45, no. 5, pp. 37–42, 2015.

[5] Z. Zhou, M. Dong, K. Ota, J. Wu, and T. Sato, "Energy efficiency and spectral efficiency tradeoff in device-to-device (D2D) communications," *IEEE Wireless Communications Letters*, vol. 3, no. 5, pp. 485–488, 2014.

[6] A. Y. Al-Dubai, L. Zhao, A. Y. Zomaya, and G. Min, "QoS-Aware Inter-Domain Multicast for Scalable Wireless Community Networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 11, pp. 3136–3148, 2015.

[7] L. Zhao, A. Al-Dubai, X. Li, and G. Chen, "A new efficient cross-layer relay node selection model for Wireless Community Mesh Networks," *Computers Electrical Engineering*, vol. 61, pp. 361–372, 2017.

[8] G. Han, J. Jiang, C. Zhang, T. Q. Duong, M. Guizani, and G. K. Karagiannidis, "A Survey on Mobile Anchor Node Assisted Localization in Wireless Sensor Networks," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 3, pp. 2220–2243, 2016.

[9] L. Zhao et al., "Vehicular Communications: Standardization and Open Issues," *IEEE Communications Standards Magazine*, 2019.

[10] H. Li, K. Ota, and M. Dong, "Learning IoT in edge: deep learning for the internet of things with edge computing," *IEEE Network*, vol. 32, no. 1, pp. 96–101, 2018.

[11] X. Tao, K. Ota, M. Dong, H. Qi, and K. Li, "Performance guaranteed computation offloading for mobile-edge cloud computing," *IEEE Wireless Communications Letters*, vol. 6, no. 6, pp. 774–777, 2017.

[12] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proceedings of the 41st annual ACM symposium on Theory of Computing (STOC '09)*, pp. 169–178, ACM, Bethesda, Md, USA, 2009.

[13] S. Halevi, "Homomorphic Encryption," in *Tutorials on the Foundations of Cryptography*, Information Security and Cryptography, pp. 219–276, Springer International Publishing, Cham, 2017.

[14] M. van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan, "Fully homomorphic encryption over the integers," in *Advances in cryptology—EUROCRYPT 2010*, vol. 6110, pp. 24–43, Springer, Berlin, Germany, 2010.

[15] C. Gentry and S. Halevi, "Implementing Gentry's fully—homomorphic encryption scheme," in *Advances in cryptology—EUROCRYPT 2011*, vol. 6632 of *Lecture Notes in Computer Science*, pp. 129–148, Springer, Heidelberg, Germany, 2011.

[16] N. P. Smart and F. Vercauteren, "Fully homomorphic encryption with relatively small key and ciphertext sizes," *Lecture Notes in*

Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics): Preface, vol. 6056, pp. 420–443, 2010.

[17] J.-S. Coron, A. Mandal, D. Naccache, and M. Tibouchi, "Fully homomorphic encryption over the integers with shorter public keys," in *Proceedings of the 31st Annual International Cryptology Conference (CRYPTO '11)*, vol. 6841 of *Lecture Notes in Computer Science*, pp. 487–504, Springer, Santa Barbara, Calif, USA.

[18] Z. Brakerski and V. Vaikuntanathan, "Efficient fully homomorphic encryption from (standard) LWE," in *Proceedings of the IEEE 52nd Annual Symposium on Foundations of Computer Science (FOCS '11)*, pp. 97–106, Palm Springs, Calif, USA, October 2011.

[19] Z. Brakerski and V. Vaikuntanathan, "Fully homomorphic encryption from ring-LWE and security for key dependent messages," in *Advances in Cryptology—CRYPTO 2011*, R. Phillip, Ed., vol. 6841, pp. 505–524, Springer, Berlin, Germany, 2011.

[20] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(Leveled) fully homomorphic encryption without bootstrapping," in *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, pp. 309–325, ACM, 2012.

[21] A. López-Alt, E. Tromer, and V. Vaikuntanathan, "On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption," in *Proceedings of the 44th Annual ACM Symposium on Theory of Computing (STOC '12)*, pp. 1219–1234, ACM, May 2012.

[22] J. W. Bos, K. Lauter, J. Loftus, and M. Naehrig, "Improved security for a ring-based fully homomorphic encryption scheme," in *Cryptography and coding*, vol. 8308 of *Lecture Notes in Comput. Sci.*, pp. 45–64, Springer, Heidelberg, 2013.

[23] C. Gentry, A. Sahai, and B. Waters, "Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based," *Proceedings of CRYPTO 2013*, vol. 8042, no. 1, pp. 75–92, 2013.

[24] C. Gentry, S. Halevi, and N. P. Smart, "Homomorphic evaluation of the AES circuit," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics): Preface*, vol. 7417, pp. 850–867, 2012.

[25] C. Gentry, S. Halevi, and N. P. Smart, "Fully homomorphic encryption with polylog overhead," in *Advances in cryptology—EUROCRYPT 2012*, vol. 7237 of *Lecture Notes in Comput. Sci.*, pp. 465–482, Springer, Heidelberg, 2012.

[26] N. P. Smart and F. Vercauteren, "Fully homomorphic SIMD operations," *Designs, Codes and Cryptography*, vol. 71, no. 1, pp. 57–81, 2014.

[27] Y. Chen and G. Gong, "Integer arithmetic over ciphertext and homomorphic data aggregation," in *Proceedings of the 3rd IEEE International Conference on Communications and Network Security, CNS 2015*, pp. 628–632, Italy, September 2015.

[28] K. Gai and M. Qiu, "Blend Arithmetic Operations on Tensor-Based Fully Homomorphic Encryption Over Real Numbers," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 8, pp. 3590–3598, 2018.

[29] K. Gai, M. Qiu, Y. Li, and X.-Y. Liu, "Advanced Fully Homomorphic Encryption Scheme over Real Numbers," in *Proceedings of the 4th IEEE International Conference on Cyber Security and Cloud Computing, CSCloud 2017 and 3rd IEEE International Conference of Scalable and Smart Cloud, SSC 2017*, pp. 64–69, USA, June 2017.

[30] L. Kuang, L. T. Yang, J. Feng, and M. Dong, "Secure Tensor Decomposition Using Fully Homomorphic Encryption Scheme," *IEEE Transactions on Cloud Computing*, vol. 6, no. 3, pp. 868–878, 2018.

[31] J.-S. Coron, D. Naccache, and M. Tibouchi, "Public key compression and modulus switching for fully homomorphic encryption over the integers," in *Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT 2012)*, vol. 7237 of *Lecture Notes in Comput. Sci.*, pp. 446–464, Springer.

[32] J. H. Cheon, J. S. Coron, J. Kim et al., "Batch fully homomorphic encryption over the integers," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, vol. 7881 of *Lecture Notes in Computer Science*, pp. 315–335, Springer, Berlin, Germany, 2013.

[33] J.-S. Coron, T. Lepoint, M. Tibouchi, J. H. Cheon, and J. Kim, "Batch fully homomorphic encryption over the integers," in *Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 315–335, Springer, Berlin, Germany, 2013.

[34] J. Kim, M. S. Lee, A. Yun et al., CRT-based fully homomorphic encryption over the integers , Cryptology ePrint Archive, https://eprint.iacr.org//057.pdf.

[35] S. Halevi and V. Shoup, "Algorithms in helib," in *Lecture Notes in Computer Science*, vol. 8616 of *Lecture Notes in Comput. Sci.*, pp. 554–571, Springer, Heidelberg, 2014.

[36] A. D. Booth, "A signed binary multiplication technique," *The Quarterly Journal of Mechanics and Applied Mathematics*, vol. 4, pp. 236–240, 1951.