Hindawi
Security and Communication Networks
Volume 2018, Article ID 2363928, 14 pages
https://doi.org/10.1155/2018/2363928

Research Article

Privacy-Preserving Oriented Floating-Point Number Fully Homomorphic Encryption Scheme

Shuangjie Bai,[1] **Geng Yang**,[1,2,3] **Jingqi Shi**,[1] **Guoxiu Liu**,[1] **and Zhaoe Min**[1]

[1]*College of Computer Science, Nanjing University of Posts and Telecommunication, Nanjing 210003, China*
[2]*Key Laboratory of Broadband Wireless Communication & Sensor Networks Technology of Ministry of Education, Nanjing University of Posts and Telecommunications, Nanjing 210003, China*
[3]*Jiangsu Key Laboratory of Big Data Security & Intelligent Processing, Nanjing, Jiangsu 210023, China*

Correspondence should be addressed to Geng Yang; yangg@njupt.edu.cn

Received 29 January 2018; Revised 19 May 2018; Accepted 5 June 2018; Published 24 July 2018

Academic Editor: Roberto Di Pietro

The issue of the privacy-preserving of information has become more prominent, especially regarding the privacy-preserving problem in a cloud environment. Homomorphic encryption can be operated directly on the ciphertext; this encryption provides a new method for privacy-preserving. However, we face a challenge in understanding how to construct a practical fully homomorphic encryption on non-integer data types. This paper proposes a revised floating-point fully homomorphic encryption scheme (FFHE) that achieves the goal of floating-point numbers operation without privacy leakage to unauthorized parties. We encrypt a matrix of plaintext bits as a single ciphertext to reduce the ciphertext expansion ratio and reduce the public key size by encrypting with a quadratic form in three types of public key elements and pseudo-random number generators. Additionally, we make the FFHE scheme more applicable by generalizing the homomorphism of addition and multiplication of floating-point numbers to analytic functions using the Taylor formula. We prove that the FFHE scheme for ciphertext operation may limit an additional loss of accuracy. Specifically, the precision of the ciphertext operation's result is similar to unencrypted floating-point number computation. Compared to other schemes, our FFHE scheme is more practical for privacy-preserving in the cloud environment with its low ciphertext expansion ratio and public key size, supporting multiple operation types and high precision.

1. Introduction

In this age of big data, the amount of data that identifies individuals is increasing. People are enjoying the convenience created by these data; at the same time, leakage of personal data has attracted more attention because these data are easily accessible to third parties, especially in a cloud environment, and because the service provider can easily access users' plaintexts in a cloud server. Additionally, the security and privacy of personal data are threatened as we adopt cloud computing services; this threat is considered the biggest challenge in a cloud environment [1]. To avoid the data being leaked, people usually adopt an encryption algorithm such as AES to encrypt these data and store them on a cloud server. However, AES cannot support the operations over ciphertexts directly on the cloud server. When people need to use these data, they must download ciphertexts from the server and decrypt locally. Such downloads require a large consumption of resources. Only immense storage space in the cloud is used while the availability of ciphertexts is lost. Therefore, people have been seeking a scheme that can utilize the advantages of the cloud, including immense storage space and strong computing power, which can effectively protect the privacy of an individual at the same time. Homomorphic encryption is closely related to the confidentiality of data in the cloud and is the key technology to protect data privacy.

Homomorphic encryption allows third parties to operate over encrypted values without being aware of the content. The idea of homomorphic encryption is as follows. **Enc** represents the encryption algorithm, and **Dec** represents the decryption algorithm. For a function f taking plaintexts $m_0, m_1, \cdots, m_l$

as inputs, there exists a function f' taking the corresponding ciphertexts $c_0, c_1, \cdots, c_l$ as inputs, where $c_i = \mathbf{Enc}(m_i)$, $i = 0, 1, \cdots, l$, such that

$$\mathbf{Dec}\left(f'\left(c_0, c_2, \cdots, c_l\right)\right) = f\left(m_0, m_1, \cdots, m_l\right) \quad (1)$$

According to the property of homomorphic encryption above, to protect a user's privacy in a cloud environment seems to be an excellent method. We can take advantage of the strong computing power in the cloud to operate over the ciphertexts without exposing the plaintexts. There are some applications such as CryptDB [7, 8] and other encrypted databases [9, 10], which apply a homomorphic encryption scheme to protect data privacy.

The present homomorphic encryption schemes have some limitations. Most practical systems apply a partially homomorphic encryption scheme (certain restricted types of computations can be done on ciphertexts), such as Paillier [11], rather than a fully homomorphic encryption scheme. The current fully homomorphic encryption scheme is inefficient. There are also many studies on improving the efficiency of fully homomorphic encryption schemes [3–5, 12, 13], but these schemes cannot be applied to a practical system. Furthermore, most current homomorphic encryption schemes only operate over the integer data type, while most data are not only integers. For example, in the e-healthcare cloud service [14], decision-making models can be used to automatically check a patient's health status, and the parameters of the decision-making model contain real numbers. Additionally, we often need a mathematic modeling method to analyze a patient's health data, which may utilize complex functions taking floating-point numbers as input. However, most existing homomorphic encryption schemes [2, 11, 15–18] only focus on operations on integers. Designing a homomorphic encryption scheme to achieve floating-point number calculation without compromising the privacy of the outsourced data is a challenging issue.

In this paper, we propose a floating-point fully homomorphic encryption scheme (FFHE) to overcome the above problems. The scheme is based on our proposed revised somewhat homomorphic encryption scheme (RSHE), which is more effective than [3–5]. Although FFHE is not appropriate for practical application to a system, we have improved the efficiency. By using five integers to express a floating-point number [19], we convert the operations on a floating-point number to an integer. FFHE can support addition and multiplication operations on floating-point numbers. We then prove that the operation on ciphertexts of a floating-point number does not increase additional precision loss. Using the Taylor series approximation calculation, our scheme can also calculate the result of analysis functions, such as exponential and logistic functions taking floating-point numbers as input without exposing plaintexts.

Our FFHE is more efficient than previous works. It is based on Gentry's original blueprint and supports a more complex function with more diversified input data types. The main contributions of this paper can be summarized as follows.

(i) We construct a more efficient and somewhat homomorphic encryption scheme with a smaller public key size and a secret key size based on [2], which is more effective than Coron's scheme [3–5].

(ii) We follow Gentry's blueprint and construct a floating-point fully homomorphic encryption scheme (FFHE) that can support addition and multiplication operations on floating-point numbers based on our proposed revised somewhat homomorphic encryption (RSHE). Compared to the operation of plaintexts, addition and multiplication of ciphertexts do not increase additional precision loss.

(iii) FFHE can calculate the result of analysis functions taking floating-point numbers as input without exposing plaintexts, and the error of approximate calculation is negligible.

The remainder of this paper is organized as follows: related work is discussed in Section 2. In Section 3, we describe some preliminaries required for understanding of our proposed revised somewhat homomorphic encryption scheme and FFHE, as well as some prior knowledge about floating-point numbers. In Section 4, we review the DGHV scheme. Then, we present the revised somewhat homomorphic encryption scheme in Section 5, followed by our proposed FFHE in Section 6. In Section 7, we explain the type of operations of FFHE. Section 8 concludes this paper.

2. Related Work

In this section, we informally review previous homomorphic encryption schemes. In 1978, Rivest et al. proposed the idea of privacy homomorphism [20]: workers may perform implicit addition and multiplication on plaintext values while exclusively manipulating encrypted data. Homomorphic encryption has attracted significant research attention lately. There are many studies on homomorphic encryption, including a somewhat homomorphic encryption scheme (a crucial component of fully homomorphic encryption which allows many additions and a small number of multiplications on ciphertexts) and a partially homomorphic encryption scheme. For example, RSA [15] is a multiplicatively homomorphic encryption scheme, and Paillier cryptosystem [11] is an additively homomorphic encryption scheme.

However, partially or somewhat homomorphic encryption schemes cannot meet the requirements of processing vast privacy data. In a 2009 breakthrough work, Gentry constructed the first encryption scheme based on ideal lattices that support both addition and multiplication on ciphertexts, i.e., a fully homomorphic encryption scheme [16], and detailed the proposed scheme in his Ph.D. thesis [17], which led to the research on fully homomorphic encryption.

Based on Gentry's research, three types of schemes were constructed.

(1) Gentry's original scheme was based on ideal lattices. The construction follows successive steps: first, construct a somewhat homomorphic scheme that supports limited addition and multiplication on ciphertexts. This step is necessary because ciphertexts contain some noise that becomes larger with successive homomorphic multiplication, and only ciphertexts whose noise size remains below a certain

threshold can be decrypted correctly. Second, Gentry shows how to squash the decryption procedure so that it can be expressed as a low degree polynomial in the bits of the ciphertext and the secret key. Then, Gentry's key idea, called bootstrapping, consists in homomorphically evaluating this decryption polynomial on encryptions of the secret key bits, resulting in a different ciphertext (refreshed ciphertext) associated with the same plaintext with possibly reduced noise. The refreshed ciphertext can then be used in subsequent homomorphic operations. By repeatedly refreshing ciphertexts after every operation, the number of homomorphic operations becomes unlimited, resulting in a fully homomorphic encryption scheme. Gentry and Halevi implemented Gentry's scheme [21] based on algorithmic optimizations proposed by Smart et al. [12].

(2) In 2012, Brakerski et al. proposed a fully homomorphic encryption scheme (BGV scheme) based on Learning with Errors (LWE) and Ring Learning with Errors (RLWE) problems [18], which was a new way of constructing a leveled fully homomorphic encryption scheme (capable of evaluating arbitrary polynomial-size circuits), without Gentry's bootstrapping procedure. They proposed modulus-switching and key switching techniques to reduce noise and the ciphertext expansion ratio. An implementation was described with an efficient homomorphic evaluation of a full AES encryption circuit. At the same time, this branch also drew significant attention, and they proposed a revised scheme based on LWE [13, 22–25].

(3) At Eurocrypt 2010, Dijk, Gentry, Halevi, and Vaikuntanathan (DGHV scheme) described a fully homomorphic encryption scheme over the integers [2]. The main appeal of the scheme (compared to Gentry's original scheme) was its conceptual simplicity; all operations are done over the integers instead of ideal lattices. However, the public key was too large for any practical system. Because of that, Coron et al. [3–5] proposed several optimization techniques to reduce public key size. Homomorphic encryption scheme can also be applied in provable data possession [26–29].

The schemes above address operation over integers, while there are few studies of fully homomorphic encryption schemes for non-integers. Real numbers are the most common numbers used in measuring. Recently, there have been papers focusing on designing a fully homomorphic encryption scheme for real numbers. Seiko, Arita, and Shota Nakasato [30] proposed a fully homomorphic encryption for point numbers based on the FV [31] scheme, which is based on the LWE problem. They construct a first homomorphic encryption scheme that supports operations for fixed-point numbers. Based on this, they constructed a fully homomorphic encryption scheme that can homomorphically compute addition and multiplication of encrypted floating-point numbers. Cheon et al. [32] proposed a floating-point homomorphic encryption scheme based on the BGV scheme [18], which was able to evaluate arbitrary polynomial-size circuits, similar to the BGV scheme. Costache [33] proposed a fixed-point arithmetic in a somewhat homomorphic encryption scheme and investigated an application in homomorphic image processing. However, these papers just identified the feasibility of constructing a homomorphic encryption scheme supporting operations over floating-point numbers. These schemes cannot secretly calculate complex functions with floating-point numbers as input. Additionally, because of the low efficiency of basic schemes over integer, their proposed schemes for non-integers also require a large key space and ciphertext space. Liu [34] realized outsourced calculation on floating-point numbers. They utilized Paillier-based cryptosystem to construct operation protocol over the ciphertexts. However, Paillier is only an additively homomorphic scheme, and they construct a multiplicatively homomorphic scheme by setting up two severs, which requires a higher security model.

Low efficiency, large public key size, and ciphertext expansion ratio are the main reasons for which most fully homomorphic encryption schemes are not practical, and this problem receives the most attention. However, there are some studies on floating-point number homomorphic encryption, and the supported arithmetic types have many limitations; for example, most analytic functions such as exponential functions and logarithmic functions for floating-point numbers cannot be supported. Therefore, it is necessary to construct a fully homomorphic encryption scheme for floating-point numbers.

3. Preliminary

3.1. Notations. For a real number z, we denote using $\lfloor z \rfloor$, $\lceil z \rceil$, and $\lfloor z \rceil$ rounded up, down, or to the nearest integer. For a real number z, and an integer p, we denote the reduction of z modulo p by $[z]_p$ with $-p/2 < [z]_p < p/2$, or $\langle z \rangle_p$ with $0 \leq \langle z \rangle_p < p$. Let x^* be a real number. We use the notation $x = fl(x^*)$ to represent the floating-point value of x^* (the nearest number in the floating-point system). The most useful measures of the accuracy of x are its absolute error $E_{abs}(x) = |x - x^*|$ and its relative error $E_{rel}(x) = |x - x^*|/|x^*|$ [19].

The parameters of the somewhat homomorphic DGHV scheme are shown in Section 4. Given the security parameter λ, the following parameters are used:

γ is the bit-length of the public keys.

η is the bit-length of the secret keys.

ρ is the bit-length of the first noise parameter.

τ is the number of the public keys.

ρ' is the bit-length of the secondary noise parameter.

The other parameters used in our proposed schemes will be described in Sections 5 and 6.

3.2. Floating-Point Number. The floating-point number is the formulaic representation that approximates a real number to support a trade-off between range and precision. We define the floating-point format used in this paper. A floating-point format is characterized by five integers [19]:

$$x = (-1)^{s'} \cdot s \cdot b^{e-k} \tag{2}$$

where $s' \in \{0, 1\}$ is the sign of x. b is the base or radix (b=2 in this paper). e is an integer such that $e_{\min} \leq e \leq e_{\max}$, called

the exponent of the x, where $e_{\min}$ and $e_{\max}$ are two extremal exponents such that $e_{\min} \le 0 \le e_{\max}$. k is the precision (the number of significant digits in the significand), and s is the significand satisfying $b^{k-1} \le s \le b^k - 1$. To determine accuracy, we define the quantity $u = 0.5 \cdot b^{1-k}$. It is the furthest distance relative to unity between a real number and the nearest floating-point number. According to the definition of the relative error above, we represent the relative error as $x = x^*(1+\varepsilon)$ such that $E_{rel}(x) = |\varepsilon| \le u$. According to different demands in a practical system, we may adjust the parameters in the floating-point format we defined above.

3.3. Floating-Point Error Analysis. For real numbers x_1^*, x_2^*, x_1, x_2 represent the floating-point value of x_1^*, x_2^*, which is the nearest number in the floating-point system. For $i = 1, 2$, $x_i = x_i^* + \Delta_i$, where Δ_1, Δ_2 represent the absolute error of the real number and the floating-point number, which are subject to $|\Delta_i| \le 0.5 \cdot b^{e-k}$, $|x_i^*| \ge b^{e-1}$, and the relative error is as follows:

$$E_{rel}(x_i) = \frac{|\Delta_i|}{|x_i^*|} \le 0.5 \cdot b^{1-k} \tag{3}$$

The addition of the two real numbers can be represented as $x_1^* + x_2^* = x_1 + \Delta_1 + x_2 + \Delta_2$, such that

$$E_{rel}(x_1 + x_2) = \frac{|\Delta_1 + \Delta_2|}{|x_1^* + x_2^*|} \le 0.5 \cdot b^{1-k} \tag{4}$$

The addition of floating-point number does not affect the relative error, and the precision is still k.

The multiplication of the two real numbers can be represented as

$$\begin{aligned} x_1^* \cdot x_2^* &= (x_1 + \Delta_1) \cdot (x_2 + \Delta_2) \\ &= x_1 \cdot x_2 + x_1 \cdot \Delta_2 + x_2 \cdot \Delta_1 + \Delta_1 \cdot \Delta_2 \end{aligned} \tag{5}$$

such that

$$\begin{aligned} E_{rel}(x_1 \cdot x_2) &= \frac{|x_1 \cdot \Delta_2 + x_2 \cdot \Delta_1 + \Delta_1 \cdot \Delta_2|}{|x_1^* \cdot x_2^*|} \\ &\le 2 \cdot 0.5 \cdot b^{1-k} \end{aligned} \tag{6}$$

For each multiplication, the relative error approximately doubles. Specifically, the relative error will increase with continuous multiplications [19]. The FFHE we propose below simulates the operations of plaintext in the floating-point system with a Boolean circuit. We prove the relative error of ciphertext is the same as corresponding plaintext in our FFHE.

4. The Somewhat Homomorphic DGHV Scheme

In this section, we recall the somewhat homomorphic encryption scheme over the integers of van Dijk, Gentry, Halevi, and Vaikuntanathan (DGHV) [2]. The notation used in the DGHV scheme is the same as in Section 3.1.

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

$$\begin{aligned} D_{\gamma,\rho}(p) = \{&Choose\ q \longleftarrow \mathbb{Z} \cap [0, 2^\gamma / p), r \longleftarrow \mathbb{Z} \\ &\cap (-2^\rho, 2^\rho) : Output : x = pq + r\} \end{aligned} \tag{7}$$

DGHV.KenGen(λ). Generate an η-bit random prime integer p so that $p \longleftarrow (2\mathbb{Z}+1) \cap [2^{\eta-1}, 2^\eta)$. For $0 \le i \le \tau$, sample $x_i \longleftarrow D_{\gamma,\rho}(p)$. Relabel the x_i's so that x_0 is the largest. Restart unless x_0 is odd and $[x_0]_p$ is even. Let **pk** : $\langle x_0, x_1, \cdots, x_\tau \rangle$ and **sk** = p.

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

DGHV.Evaluate(pk, C, $\mathbf{c_1}$, $\mathbf{c_2}$, $\cdots$, $\mathbf{c_t}$). Given the circuit C with t input bits and t ciphertexts c_i, apply the addition and multiplication gates of C to the ciphertexts, performing all the addition and multiplications over the integers, and return the resulting integer.

DGHV.Decrypt(sk, c). Output $m \longleftarrow (c \bmod p) \bmod 2$.

This completes the description of the scheme as shown in [2], and the scheme is a somewhat homomorphic scheme and it is semantically secure under the approximate-GCD assumption, which is proven in [2].

Definition 1 (approximate GCD). The (ρ, η, γ) – *approximate GCD* problem is as follows: given a random η-bit odd integer p and given many polynomial samples from $D_{\gamma,\rho}(p)$, in outputting p.

According to the scheme parameter constraints, the following parameters are suggested: $\rho = \lambda$, $\rho' = 2\lambda$, $\eta = \widetilde{O}(\lambda^2)$, $\gamma = \widetilde{O}(\lambda^5)$ and $\tau = \gamma + \lambda$. The public key size is $\widetilde{O}(\lambda^{10})$, which is too large for a practical system.

5. Revised Somewhat Homomorphic Encryption Scheme (RSHE)

In this section, we propose our revised somewhat homomorphic encryption scheme (RSHE) with a smaller public key size and ciphertext expansion ratio. As described in Section 4, the public key size of DGHV is $\widetilde{O}(\lambda^{10})$, which is too large for a practical system. References [3–5] proposed some variants of the DGHV scheme. However, our RSHE is more efficient than these schemes, and the detailed performance comparison of our RSHE with these schemes is shown in Section 5.2.

As in the extension in [3], we extend the DGHV scheme for a batch setting. Instead of packing the plaintext vector $m_0, \cdots, m_{l-1}$, we packed plaintext matrix $\{m_{i,j}\}_{1 \le i,j \le \mu}$ with μ^2 elements into a single ciphertext, where $l = \mu^2$. Instead of working with integer $x'_{i,j}$ of the form $x'_{i,j} = x_{i,0} \cdot x_{j,1}$ as in [21], we compressed all of the public keys used in RSHE in the same way, that is $x'_{i,j} = x'_{i,0} \cdot x'_{j,1}$, $\Pi_{i,j} = \Pi_{i,0} \cdot \Pi_{j,1}$ for $1 \le i, j \le \mu$ where $l = \mu^2$, and $x_{i,j} = x_{i,0} \cdot x_{j,1}$ for $1 \le i, j \le \beta$ where $\tau = \beta^2$. Then, only $4\mu + 2\beta$ integers need to be stored in the public key to generate $2\mu^2 + \beta^2$ integers used for encryption in [3]. We considered a linear combination of the $\Pi_{i,j}$ with coefficients

in $(-2^{\alpha'}, 2^{\alpha'})$, and a linear combination of the $x_{i,j}$ in $(-2^{\alpha}, 2^{\alpha})$ to further reduce the public key size. The detailed description of RSHE is as follows.

5.1. Description. **KeyGen(λ)**: Choose η-bit distinct primes $\{p_{u,v}\}_{1\le u,v\le\mu}$, and denote π as their product, that is $\pi = \prod_{1\le u,v\le\mu} p_{u,v}$. Define the error-free public key element $x_0 = q_0 \cdot \pi$, where $q_0 \longleftarrow \mathbb{Z} \cap [0, 2^{\gamma}/\pi)$ is a 2^{λ^2}-rough integer. Initialize two pseudo-random number generators f_1, f_2, with two random seeds se_1, se_2 (the two pseudo-random number generator functions are independent of each other). Use $f_1(se_1)$ and $f_2(se_2)$ to generate a set of integers $\chi_{i,b} \in [0, 2^{\gamma})$, $1 \le i \le \beta, 0 \le b \le 1$, let $\tau = \beta^2$, compute the first set of public keys:

$$\delta'_{i,b} = \langle \chi_{i,b} \rangle_{\pi} + \xi_{i,b} \cdot \pi - r_{i,b} \tag{8}$$

where $r_{i,b} \longleftarrow \mathbb{Z} \cap (-2^{\rho'}, 2^{\rho'})$ and $\xi_{i,b} \longleftarrow \mathbb{Z} \cap [0, 2^{\lambda+\eta l}/\pi)$. For all $1 \le i \le \beta$ and $0 \le b \le 1$, compute $x_{i,b} = \chi_{i,b} - \delta'_{i,b}$. For noise parameters $\omega_{i,b,u,v} \longleftarrow \mathbb{Z} \cap (-2^{\rho}, 2^{\rho})$, $1 \le i \le \mu$, $0 \le b \le 1$, $1 \le u, v \le \mu$, compute the second set of public keys:

$$\Pi_{i,b} \bmod p_{u,v} = 2\omega_{i,b,u,v} + \delta_{i,b,u,v} \cdot 2^{\rho'/2+1} \tag{9}$$

For noise parameters $r'_{i,b,u,v} \longleftarrow \mathbb{Z} \cap (-2^{\rho}, 2^{\rho})$, compute the third set of public keys:

$$x'_{i,b} \bmod p_{u,v} = 2r'_{i,b,u,v} + \delta_{i,b,u,v} \tag{10}$$

We name $\delta_{i,b,u,v}$ as the extended Kronecker function as follows (the definition of the Kronecker function is shown in [3]):

$$\delta_{i,0,u,v} = \begin{cases} 1 & i = u \\ 0 & i \ne u, \end{cases}$$
$$\delta_{i,1,u,v} = \begin{cases} 1 & i = v \\ 0 & i \ne v \end{cases} \tag{11}$$

and using the compression technique used for the first set of public keys, we can also compress x_0, let $\delta_0 = \langle \chi_0 \rangle_{\pi} + \xi_0 \cdot \pi$, and compute $x_0 = \chi_0 - \delta_0$.

Let $sk = \{p_{u,v}\}_{1\le u,v\le\mu}$, $pk = \langle \delta_0, se_1, se_2, (\delta'_{i,b})_{1\le i\le\beta,0\le b\le 1}, (\Pi_{i,b})_{1\le i\le\mu,0\le b\le 1}, (x'_{i,b})_{1\le i\le\mu,0\le b\le 1} \rangle$.

Enc (pk,$\{\mathbf{m_{i,j}}\}_{\mathbf{1\le i,j\le\mu}} \in \{\mathbf{0,1}\}_{\boldsymbol{\mu\times\mu}}$): Generate random integer matrix $\mathbf{b} = (b_{i,j})_{1\le i,j\le\beta} \in (-2^{\alpha}, 2^{\alpha})$, $\mathbf{b'} = (b'_{i,j})_{1\le i,j\le\mu} \in (-2^{\alpha'}, 2^{\alpha'})$. Output the ciphertext:

$$c = \left[\sum_{1\le i,j\le\mu} m_{i,j} x'_{i,0} x'_{j,1} + \sum_{1\le i,j\le\mu} b'_{i,j} \Pi_{i,0} \Pi_{j,1} + 2 \sum_{1\le i,j\le\beta} b_{i,j} x_{i,0} x_{j,1} \right]_{x_0} \tag{12}$$

Dec (sk, c): Output plaintext matrix $\mathbf{m} = \{m_{i,j}\}_{1\le i,j\le\mu}$: $m_{i,j} = [c \bmod p_{i,j}]_2$

Evaluate (pk, C, $\mathbf{c_1}$, …, $\mathbf{c_t}$): as in the original DGHV scheme.

Add (pk, $\mathbf{c_1}$, $\mathbf{c_2}$): Output: $(c_1 + c_2) \bmod x_0$

Mult (pk, $\mathbf{c_1}$, $\mathbf{c_2}$): Output: $(c_1 \cdot c_2) \bmod x_0$

5.2. Parameter Analysis. For the security parameters λ, we have the following constraints on our scheme parameters:

$\rho \ge 2\lambda$: to avoid a brute force attack on the noise, and the value is larger than that in [5] to be secure against an attack proposed in [35].

$\gamma = \omega(\eta^2 \cdot \log\lambda)$: for thwarting lattice-based attacks against the approximate GCD problem [2].

$\eta \ge \rho \cdot \Theta(\lambda \cdot \log^2\lambda)$: for supporting homomorphic operations for evaluating the squashed decryption circuit [2].

$\eta \ge \alpha' + \rho' + 5 + 2\log_2\beta$: for correct decryption of a ciphertext (Section 5.3).

To satisfy the above constraints, we can take $\rho = 2\lambda$, $\eta = \widetilde{O}(\lambda^2)$, $\gamma = \widetilde{O}(\lambda^5)$, $\alpha = \widetilde{O}(\lambda^2)$, $\tau = \widetilde{O}(\lambda^3)$, $\rho' = 5\lambda$, $\alpha' = \widetilde{O}(\lambda^2)$, $l = \widetilde{O}(\lambda^2)$. As the ciphertexts are preserved in the form of a matrix, let $l = \mu^2$, the ciphertext expansion ratio is $\gamma/\mu^2 = \widetilde{O}(\lambda^3)$, the new secret key size is $\eta \cdot l = \widetilde{O}(\lambda^4)$, as in [3]. However, compared to the ciphertext expansion ratio $\gamma = \widetilde{O}(\lambda^5)$ in [2], our scheme has been greatly improved. The new public key for our revised somewhat homomorphic scheme has a size $\widetilde{O}(2\beta\cdot(\lambda+\eta\cdot l)+2\cdot2\cdot\mu\cdot\lambda^5) = \widetilde{O}(\lambda^6)$ instead of $\widetilde{O}(\lambda^8)$ as in [3] and $\widetilde{O}(\lambda^8)$ as in [4]. Though the public key size in [5] is $\widetilde{O}(\lambda^5)$, the ciphertext expansion rate is much larger than that in our paper. We prefer a slightly larger public key size for a smaller ciphertext expansion rate. Compared to a public key, ciphertexts require more storage. Additionally, [6] declares the public key size of their scheme as $\widetilde{O}(\lambda^{5.5})$; however, the value of the public key x'_i in the scheme does not meet the constraints proposed in [2], and it is vulnerable to lattice-based attacks. To satisfy the constraints, the size of public key x'_i is $\widetilde{O}(\lambda^5)$, and the actual public key size is at least $\widetilde{O}(\lambda^7)$. Therefore, our proposed scheme is better in terms of public key size and ciphertext expansion rate.

We also perform computational complexity analysis between these schemes. The computational complexity of [2] is $O(\lambda^5)$ and the computational complexity of [4] is $O(\lambda^4)$. Furthermore, the computational complexity of [3, 5] is all $O(\lambda^3)$. Our scheme's computational complexity is $O(2 \cdot \mu^2 + \beta^2) = O(\lambda^3)$. We have not increased the computational complexity under the premise of reducing the space complexity.

A comprehensive comparison of space complexity and computational complexity between the above schemes is shown in Table 1.

5.3. Correctness. First, define the permitted circuits as follows.

Definition 2 (permitted circuit). For any $i > 1$, and any set of integer inputs all less than $\tau^i \cdot 2^{i(\alpha'+\rho'+2)}$ in absolute value, it holds that the generalized circuit's output has an absolute

Table 1: Performance comparison of somewhat homomorphic encryption scheme.

Scheme	Size of pk	Size of sk	Expansion rate of ciphertext	Computational complexity
DGHV [2]	$\widetilde{O}(\lambda^{10})$	$\widetilde{O}(\lambda^{2})$	$\widetilde{O}(\lambda^{5})$	$O(\lambda^{5})$
[3]	$\widetilde{O}(\lambda^{8})$	$\widetilde{O}(\lambda^{4})$	$\widetilde{O}(\lambda^{3})$	$O(\lambda^{3})$
[4]	$\widetilde{O}(\lambda^{7})$	$\widetilde{O}(\lambda^{2})$	$\widetilde{O}(\lambda^{5})$	$O(\lambda^{4})$
[5]	$\widetilde{O}(\lambda^{5})$	$\widetilde{O}(\lambda^{2})$	$\widetilde{O}(\lambda^{5})$	$O(\lambda^{3})$
[6]	$\widetilde{O}(\lambda^{7})$	$\widetilde{O}(\lambda^{4})$	$\widetilde{O}(\lambda^{3})$	$O(\lambda^{3})$
RSHE	$\widetilde{O}(\lambda^{6})$	$\widetilde{O}(\lambda^{4})$	$\widetilde{O}(\lambda^{3})$	$O(\lambda^{3})$

value not exceeding $2^{i(\eta-3-n)}$, where $n = \lceil \log_2(\lambda+1) \rceil$. Let $\mathscr{C}_\varepsilon$ denote the set of permitted circuits.

According to the above scheme, encrypt the plaintext matrix $\mathbf{m}_1, \cdots, \mathbf{m}_t$, and the ciphertext vector $\mathbf{C} = \{c_1, \cdots, c_t\}$ generate, where $c_i = \text{Encrypt}(pk, \mathbf{m}_i)$. Let C be a circuit with t inputs and an output. We can decrypt properly, if

$$\text{Decrypt}\left(sk, \text{Evaluate}\left(\text{pk}, C, \mathbf{C}\right)\right) = C\left(m_1, \cdots, m_t\right) \quad (13)$$

Lemma 3. *Our somewhat homomorphic encryption scheme is correct for $\mathscr{C}_\varepsilon$.*

Proof. The ciphertext of our scheme is as follows:

$$c = \left[\sum_{1\le i,j\le \mu} m_{i,j} x'_{i,0} x'_{j,1} + \sum_{1\le i,j\le \mu} b'_{i,j} \Pi_{i,0} \Pi_{i,1} + 2 \sum_{1\le i,j\le \beta} b_{i,j} x_{i,0} x_{j,1} \right]_{x_0} \quad (14)$$

For $1 \le i, j \le \mu$, we have:

$$\begin{aligned} \left| c \bmod p_{i,j} \right| &\le \mu^2 \cdot 2^{2\rho+2} + \mu^2 \cdot 2^{\alpha'+2\rho+2} + \beta^2 \cdot 2^{\alpha+2\rho+2} \\ &\le \tau \cdot 2^{\alpha'+\rho'+2} \end{aligned} \quad (15)$$

Let C be a Boolean circuit with t input $m_1, \cdots, m_t$, and let C' be the associated integer circuit where Boolean gates are replaced with integer operations with t ciphertexts $c_1, \cdots, c_t$ of plaintext $m_1, \cdots, m_t$ for input. For $1 \le i, j \le \mu$, we have

$$\begin{aligned} c \bmod p_{i,j} &= C'\left(c_1, \cdots, c_t\right) \bmod p_{i,j} \\ &= C'\left(c_1 \bmod p_{i,j}, \cdots, c_t \bmod p_{i,j}\right) \bmod p_{i,j} \end{aligned} \quad (16)$$

According to the Definition 2,

$$\left| C'\left(c_1 \bmod p_{i,j}, \cdots, c_t \bmod p_{i,j}\right) \right| \le 2^{\eta-4} \le \frac{p_j}{8} \quad (17)$$

so

$$\begin{aligned} &C'\left(c_1 \bmod p_{i,j}, \cdots, c_t \bmod p_{i,j}\right) \bmod p_{i,j} \\ &\quad = C'\left(c_1 \bmod p_{i,j}, \cdots, c_t \bmod p_{i,j}\right) \end{aligned} \quad (18)$$

Then,

$$c \bmod p_{i,j} = C'\left(c_1 \bmod p_{i,j}, \cdots, c_t \bmod p_{i,j}\right) \quad (19)$$

And

$$\begin{aligned} &\left[c \bmod p_{i,j}\right]_2 \\ &\quad = \left[C'\left(\left[c_1 \bmod p_{i,j}\right]_2, \cdots, \left[c_t \bmod p_{i,j}\right]_2\right)\right]_2 \\ &\quad = C\left(m_1, \cdots, m_t\right) \end{aligned} \quad (20)$$

Then, our scheme is correct for the permitted circuit $\mathscr{C}_\varepsilon$. □

In the scheme, addition results in a linear increase of noise, while multiplication results in an exponential increase of noise; therefore, multiplication is dominant in increasing noise. According to Definition 2, ciphertext outputs have noise not exceeding $\tau \cdot 2^{\alpha'+\rho'+2}$; after d multiplication, the new noise does not exceed $(\tau \cdot 2^{\alpha'+\rho'+2})^d$; let d be the degree. Let $f(x_1, \cdots, x_t)$ be the multivariate polynomial computed by C', and let $\|f\|_1$ be the l_1 norm of the coefficient vector of f. Then, $C \in \mathscr{C}_\varepsilon$ provided that $\|f\|_1 \cdot (\tau \cdot 2^{\alpha'+\rho'+2})^d \le 2^{\eta-3-n}$, then,

$$d \le \frac{\eta - 3 - n - \log \|f\|_1}{\alpha' + \rho' + 2 + 2\log\beta} \quad (21)$$

As in [5], we refer to polynomials f as permitted polynomials and denote the set of these polynomials by $\mathscr{P}_\varepsilon$.

5.4. Semantic Security. Our scheme is semantically secure under a new assumption called the error-free-μ^2-approximate GCD assumption. Given integers $\{p_{u,v}\}_{1\le u,v\le\mu}$, we define the following oracle $O_{(p_{u,v})}(\mathbf{m})$ which, given as input a matrix $m \in \{0,1\}^{\mu\times\mu}$, outputs x with

$$x = \text{CRT}_{(p_{u,v})}\left(\begin{pmatrix} m_{1,1} + 4\xi_{1,1} & \cdots & m_{1,\mu} + 4\xi_{1,\mu} \\ m_{2,1} + 4\xi_{2,1} & \cdots & m_{2,\mu} + 4\xi_{2,\mu} \\ \vdots & \ddots & \vdots \\ m_{\mu,1} + 4\xi_{\mu,1} & \cdots & m_{\mu,\mu} + 4\xi_{\mu,\mu} \end{pmatrix} \right) \quad (22)$$

where $\xi_{u,v} \longleftarrow (-2^\rho, 2^\rho)$. We denote by $\text{CRT}_{(p_{u,v})}((a_{u,v})_{1\le u,v\le\mu})$ the unique integer x smaller than $x_0 = q_0 \cdot \prod_{1\le u,v\le\mu} p_{u,v}$ such that $x \equiv a_{u,v} \bmod p_{u,v}$ for $1 \le u, v \le \mu$. Therefore, $O_{(p_{u,v})}(\mathbf{m})$ outputs a ciphertext for the plaintext $\mathbf{m}$.

Definition 4 (**EF-μ^2-AGCD**). The error-free-μ^2-approximate-GCD problem is as follows. Pick random η-bit integers $\{p_{u,v}\}_{1\le u,v\le\mu}$ of product π, a random 2^{λ^2}-rough $q_0 \longleftarrow Z \cap [0, 2^\gamma/\pi)$, integers $m_{u,v} = b$. Given $x_0 = q_0 \cdot \prod_{1\le u,v\le\mu} p_{u,v}$, $x = O_{(p_{u,v})}(\mathbf{m})$ and oracle access to $x = O_{(p_{u,v})}$, guess b.

The decisional problem is therefore to distinguish between an encryption of 0 and an encryption of 1.

Theorem 5. *Our revised somewhat homomorphic encryption scheme is semantically secure under the error-free μ^2-approximate-GCD assumption.*

Proof. An attacker takes as input the public key and outputs two messages m_0 and m_1. The challenger returns an encryption of m_b for a random bit b. The attacker outputs a guess b' and succeeds if $b' = b$.

We introduce a matrix representation of ciphertexts. To any ciphertext $c \bmod x_0$, we associate the matrix:

$$\mathbf{c} = f(c) = \begin{pmatrix} c \bmod p_{1,1} & \cdots & c \bmod p_{1,\mu} \\ c \bmod p_{2,1} & \cdots & c \bmod p_{2,\mu} \\ \vdots & \ddots & \vdots \\ c \bmod p_{\mu,1} & \cdots & c \bmod p_{\mu,\mu} \end{pmatrix} \in \mathbb{Z}^{\mu\times\mu} \tag{23}$$

In the ciphertexts matrix, each ciphertext can be rewritten as a result of matrix operation, for $1 \le u, v \le \mu$,

$$c \bmod p_{u,v} = \left(m_{u,v} + {\mathbf{r}'_{0,u,v}}^T \cdot \mathbf{m} \cdot \mathbf{r}'_{1,u,v} + {\mathbf{r}_0}^T \cdot \mathbf{b} \cdot \mathbf{r}_1 + {\boldsymbol{\omega}_{0,u,v}}^T \cdot \mathbf{b}' \cdot \boldsymbol{\omega}_{1,u,v}\right) \bmod x_0 \tag{24}$$

where $\mathbf{m}$ is the matrix of plaintexts $m = \{m_{u,v}\}_{1\le u,v\le\mu}$, and $\mathbf{r}'_{0,u,v} = (2r'_{1,0,u,v}, 2r'_{2,0,u,v}, \cdots, 2r'_{\mu,0,u,v})^T$, $\mathbf{r}'_{1,u,v} = (2r'_{1,1,u,v}, 2r'_{2,1,u,v}, \cdots, 2r'_{\mu,1,u,v})^T$ are two vectors which contain random noise in the third set of public keys. $\mathbf{b} = (b_{i,j})_{1\le i,j\le\beta}$ and $\mathbf{b}' = (b'_{i,j})_{1\le i,j\le\mu}$ are random integer matrixes. $\mathbf{r}_0 = (r_{1,0}, r_{2,0}, \cdots, r_{\beta,0})^T$, $\mathbf{r}_1 = (r_{1,1}, r_{2,1}, \cdots, r_{\beta,1})^T$ contain random noise in the second set of public keys,

$$\begin{aligned} \boldsymbol{\omega}_{0,u,v} &= \left(2\omega_{1,0,u,v}, 2\omega_{2,0,u,v}, \cdots, 2\omega_{u,0,u,v} + 2^{\rho'/2+1}, \cdots, 2\omega_{\mu,0,u,v}\right)^T \\ \boldsymbol{\omega}_{1,u,v} &= \left(2\omega_{1,1,u,v}, 2\omega_{2,1,u,v}, \cdots, 2\omega_{v,1,u,v} + 2^{\rho'/2+1}, \cdots, 2\omega_{\mu,1,u,v}\right)^T, \end{aligned} \tag{25}$$

and contain random noise in the first set of public keys (with the extended Kronecker function terms). According to the constraints on the parameters in Section 5.1, let random variable $u = {\mathbf{r}'_{0,u,v}}^T \cdot \mathbf{m} \cdot \mathbf{r}'_{1,u,v} + {\mathbf{r}_0}^T \cdot \mathbf{b} \cdot \mathbf{r}_1 + {\boldsymbol{\omega}_{0,u,v}}^T \cdot \mathbf{b}' \cdot \boldsymbol{\omega}_{1,u,v}$ be uniform in

$$\left(-\mu^2 \cdot 2^{2\rho+2}\left(1 + 2^{\alpha'}\right) - \beta^2 \cdot 2^{2\rho'+\alpha}, \mu^2 \cdot 2^{2\rho+2}\left(1 + 2^{\alpha'}\right) + \beta^2 \cdot 2^{2\rho'+\alpha}\right) \tag{26}$$

Then, the statistical distance between the random variables u and $m_{u,v} + u$ does not exceed $1/(\mu^2 \cdot 2^{2\rho+2}(1+2^{\alpha'}) + \beta^2 \cdot 2^{2\rho'+\alpha})$. Therefore,

$$\Pr[m_{u,v} = 0] - \Pr[m_{u,v} = 1] \le \frac{1}{\left(\mu^2 \cdot 2^{2\rho+2}\left(1 + 2^{\alpha'}\right) + \beta^2 \cdot 2^{2\rho'+\alpha}\right)} \tag{27}$$

and $\Pr[m_{u,v} = 0] = \Pr[m_{u,v} = 1] = 1/2$.

This concludes the proof of Theorem 5. □

6. Floating-Point Fully Homomorphic Encryption Scheme (FFHE)

RSHE can only support finite homomorphic operations, so it is necessary to construct a fully homomorphic encryption scheme. In this section, we follow Gentry's approach to transform RSHE into a fully homomorphic encryption scheme (FFHE), and we identify the scheme supporting operations over floating-point numbers.

6.1. Squash the Decryption Circuit. We first need to squash the decryption circuit. Specifically, we must add extra information about the secret key to the public key to "post process" the ciphertext. The ciphertext can be expressed as a low degree polynomial in the bits of the secret key. We add information about the secret key into the public key to construct a lower degree decryption polynomial.

We use the same technique as in [2] and generalize it to a batch setting. Let θ, Θ, κ be three new parameters, concretely, $\theta = \lambda$, $\Theta = \widetilde{O}(\lambda^3)$, $\kappa = \gamma + 2 + \lceil \log_2(\theta + 1) \rceil$. We add to the public key a set $\mathbf{y} = (y_0, \cdots, y_{\Theta-1})$ of rational numbers in $[0, 2)$ with κ bits of precision after the binary point, such that for all $1 \le u, v \le \mu$ there exists a sparse subset $S_{u,v} \subset \{0, \cdots, \Theta - 1\}$ of size θ with $\sum_{i\in S_{u,v}} y_i \approx 1/p_{u,v} \bmod 2$. We also replace the secret key with the indicator vector of the subset $S_{u,v}$. Formally, according to RSHE in Section 5.1, we define FFHE as follows.

(1) KeyGen(λ): Generate $sk = \{p_{u,v}\}_{1\le u,v\le\mu}$ and pk as before. Let $x_{p_{u,v}} \longleftarrow [2^\kappa/p_{u,v}]$, choose at random Θ-bit vectors $\mathbf{s}_{u,v} = \{s_{u,v,0}, \cdots, s_{u,v,\Theta-1}\}$, each of Hamming weight θ, for $1 \le u, v \le \mu$. Choose at random Θ integers $u_i \in [0, 2^{\kappa+1})$ for $0 \le i \le \Theta - 1$, fulfilling the condition that $x_{p_{u,v}} = \sum_{i=0}^{\Theta-1} s_{u,v,i} \cdot u_i \bmod 2^{\kappa+1}$ for $1 \le u, v \le \mu$. Set $y_i = u_i/2^\kappa$ and $\mathbf{y} = (y_0, \cdots, y_{\Theta-1})$. Each y_i is a positive number smaller than two, with κ bits of precision after the binary point, and verifies

$$\frac{1}{p_{u,v}} = \sum_{i=0}^{\Theta-1} s_{u,v,i} \cdot y_i + \varepsilon_{u,v} \bmod 2 \tag{28}$$

for some $|\varepsilon_{u,v}| < 2^{-\kappa}$.

As we set a new public key $\mathbf{y} = (y_0, \cdots, y_{\Theta-1})$ with κ bits of precision after the binary point, the public key size is $\Theta \cdot \kappa = \widetilde{O}(\lambda^8)$, instead of $\widetilde{O}(\lambda^6)$ as in our revised homomorphic encryption scheme. Using the public key compressing method proposed in Section 5.1 , we also generate y_i by using a pseudo-random generator $f(se)$ with seed se. So, only se and y_0 must be stored in a public key.

The new secret key is $sk^* = \{s_{u,v}\}_{1\le u,v\le\mu}$, and the new public key is $pk^* = (pk, y_0, se)$.

(2) Enc (pk, m): the same as Section 5.1 .

(3) Expand (pk*, c): the ciphertext expansion procedure takes as input a ciphertext c and computes an expanded ciphertext: let $z_i = [c \cdot y_i]_2$ with $n = \lceil \log_2(\theta+1) \rceil$ bits of precision after the binary point for $0 \le i \le \Theta - 1$. Output the expanded ciphertext $(c, z_0, \cdots z_{\Theta-1})$.

(4) Dec (sk*, c, $\mathbf{z_0}$, $\cdots$, $\mathbf{z_{\Theta-1}}$): output $\mathbf{m} = \{m_{u,v}\}_{1\le u,v\le\mu}$: $m_{u,v} \longleftarrow [c - \lfloor \sum_{i=0}^{\Theta-1} s_{u,v,i} \cdot z_i \rceil]_2$

Lemma 6. *The scheme after squashing the decryption circuit is correct for the set $\mathscr{C}(\mathscr{P}_\varepsilon)$ of circuits that compute permitted polynomials.*

Proof. According to Lemma 3, the correct decrypted message m is given by $(\lfloor c/p_{i,j} \rceil \bmod 2) \oplus (c \bmod 2)$, so what we need to show is that

$$\left\lfloor \frac{c}{p_{u,v}} \right\rceil = \left\lfloor \sum_{i=0}^{\Theta-1} s_{u,v,i} \cdot z_i \right\rceil \bmod 2 \tag{29}$$

Then,

$$\begin{aligned}
&\left[\left(\frac{c}{p_{u,v}}\right) - \sum_{i=0}^{\Theta-1} s_{u,v,i} \cdot z_i\right]_2 \\
&= \left[\left(\frac{c}{p_{u,v}}\right) - \sum_{i=0}^{\Theta-1} s_{u,v,i} \cdot [c \cdot y_i]_2 + \sum_{i=0}^{\Theta-1} s_{u,v,i} \cdot \Delta_i\right]_2 \\
&= \left[\left(\frac{c}{p_{u,v}}\right) - c \cdot \left[\sum_{i=0}^{\Theta-1} s_{u,v,i} \cdot y_i\right]_2 + \sum_{i=0}^{\Theta-1} s_{u,v,i} \cdot \Delta_i\right]_2 \\
&= \left[\left(\frac{c}{p_{u,v}}\right) - c \cdot \left(\frac{1}{p_{u,v}} - \varepsilon_{p_{u,v}}\right) + \sum_{i=0}^{\Theta-1} s_{u,v,i} \cdot \Delta_i\right]_2 \\
&= \left[c \cdot \varepsilon_{p_{u,v}} + \sum_{i=0}^{\Theta-1} s_{u,v,i} \cdot \Delta_i\right]_2
\end{aligned} \tag{30}$$

To satisfy the constraints on parameters, we have $|\varepsilon_{p_{u,v}}| \le 2^{-\kappa}$, $|\Delta_i| \le 2^{-n}$, so,

$$\begin{aligned}
\left|c \cdot \varepsilon_{p_{u,v}}\right| &\le 2^{\gamma-\gamma-2-\lceil \log_2 \theta+1 \rceil} \le \frac{1}{4(\theta+1)}, \\
\left|\sum_{i=0}^{\Theta-1} s_{u,v,i} \cdot \Delta_i\right| &\le \frac{\theta}{2(\theta+1)}
\end{aligned} \tag{31}$$

then,

$$\left|c \cdot \varepsilon_{p_{i,j}} + \sum_{i=0}^{\Theta-1} s_{u,v,i} \cdot \Delta_i\right| \le \frac{1}{4(\theta+1)} + \frac{\theta}{2(\theta+1)} \le \frac{1}{2} \tag{32}$$

The total distance is strictly less than 1/2. This concludes the proof. □

6.2. Bootstrapping. As in [2], one sees that the scheme is bootstrappable. From Gentry's theorem, we identify homomorphic encryption schemes for circuits of any depth.

Theorem 7. *Let ε be the scheme above and let D_ε be the set of squashed decryption circuits. Then, $D_\varepsilon \subset C(\mathscr{P}_\varepsilon)$.*

For each plaintext, the **Expand** procedures work naturally in parallel. For $1 \le u, v \le \mu$, we consider the decryption equation:

$$m_{u,v} \longleftarrow c^* - \left\lfloor \sum_{i=0}^{\Theta-1} s_{u,v,i} \cdot z_i \right\rceil \bmod 2 \tag{33}$$

Except for the decryption equation, the proof of Theorem 7 is identical to the proof of Theorem 3 in [2].

6.3. Security Analysis. Our squashed scheme is semantically secure under the hardness of subset sum assumption, which is mentioned in [2, 16]. We use the attack analysis in [4] of the sparse subset sum problem. In our scheme, the attacker must solve the following equation:

$$x_{p_{u,v}} = \sum_{i=0}^{\Theta-1} s_{u,v,i} \cdot u_i \bmod 2^{\kappa+1} \tag{34}$$

where $u_i's$ are known and the secret key $\mathbf{s}_{u,v} = (s_{u,v,0}, \cdots, s_{u,v,\Theta-1})$ is of small Hamming weight θ. We assume that the attack knows $p_{u,v}$ and therefore $x_{p_{u,v}}$.

Our squashed scheme makes an additional batch operation based on [4]. Moreover, the **Expand** procedures works naturally parallel over the plaintext bits, which means that the ciphertexts encrypted by different secret keys will not interfere with each other. For each $p_{u,v}$, the attack analysis is similar to [4].

According to the asymptotic analysis in [4, 36], we can take $\gamma = \widetilde{O}(\lambda^5)$, $\Theta = \widetilde{O}(\lambda^3)$ and $\theta = \lambda$ to avoid known attacks on the sparse subset sum problem.

6.4. Floating-Point Calculation. According to Section 3.2, we denote a floating-point number $x = (-1)^{s'} \cdot s \cdot 2^{e-k}$ (let b=2), which can be represented as a triple (s', s, e), and k is the constant precision. To securely store a floating-point number, we use FFHE to encrypt it as $(\llbracket s' \rrbracket, \llbracket s \rrbracket, \llbracket e \rrbracket)$, where $s' \in \{0, 1\}$ is the sign bit of x, s is a k-bits number. The encrypted floating-point number x is constructed as $\llbracket x \rrbracket = (\llbracket s'_x \rrbracket, \llbracket s_x \rrbracket, \llbracket e_x \rrbracket)$. Then, we define two types of circuit operations, "$\oplus$" as addition circuit and "$\otimes$" as multiplication circuit, which take binary bits as input in plaintext operation and take large integer ciphertexts as input in ciphertext operation.

The comparison of magnitude of ciphertexts cannot be avoided in the operation of the ciphertexts. Suppose we have two ciphertexts $\llbracket s_x \rrbracket$ and $\llbracket s_y \rrbracket$, where s_x and s_y are all integers.

Define a bit b to be 1 if $s_x \geq s_y$ and to be 0 otherwise. We propose a Greater-Than Bit (GTB) protocol to compute an encryption $[\![b]\!]$ of the bit b given only ciphertexts $[\![s_x]\!]$ and $[\![s_y]\!]$ without knowing the secret key.

s_x and s_y are signed integers. They can be expressed as binary numbers $s_x = m_x^0 m_x^1 \cdots m_x^k$ and $s_y = m_y^0 m_y^1 \cdots m_y^k$, and m_x^0, m_y^0 are sign bits of s_x, s_y respectively. Through the Boolean circuit, we can calculate $s_x - s_y = m_x^0 m_x^1 \cdots m_x^k - m_y^0 m_y^1 \cdots m_y^k$. We can use two's complement to achieve binary subtraction. For example,

$$\begin{aligned} &7 - 5\,(\textit{in decimal}) = 0111 - 0101\,(\textit{in binary}) \\ &= 0111 \oplus 1101 \\ &= 0111 \\ &\quad \oplus 1011\,\left(1011\ \textit{is the two's complement of}\ 1101\right) \\ &= 0010 \\ &= 2\,(\textit{in decimal}) \end{aligned} \tag{35}$$

Specifically,

$$\begin{aligned} s_x - s_y &= \left(m_x^0 m_x^1 \cdots m_x^k\right) - \left(m_y^0 m_y^1 \cdots m_y^k\right) \\ &= \left(m_x^0 m_x^1 \cdots m_x^k\right) \oplus \left(\left(\left(m_y^0 + 1\right) \bmod 2\right)\right. \\ &\quad \left. \cdot\, m_y^1 m_y^2 \cdots m_y^k\right) \\ &= \left(m_x^0 m_x^1 \cdots m_x^k\right) \oplus \left(\left(\left(m_y^0 + 1\right) \bmod 2\right)\right. \\ &\quad \left. \cdot \left(\left(m_y^1 + 1\right) \bmod 2\right) \cdots \left(\left(m_y^k + 1\right) \bmod 2\right)\right) \oplus 1 \end{aligned} \tag{36}$$

Therefore, using the addition circuit, we define the "$\ominus$" as subtraction circuit as above (we do not need the modulo operation in the subtraction operation of ciphertexts).

Greater-Than Bit Protocol. We use the subtraction circuit to implement our proposed GTB protocol. Given two ciphertexts $[\![s_x]\!]$ and $[\![s_y]\!]$, GTB protocol is to show $s_x \geq s_y$ or $s_x < s_y$. The overall steps of GTB are shown as follows.

Step 1. We execute the subtraction circuit to calculate the ciphertexts of $s_x - s_y$.

$$\begin{aligned} [\![s_z]\!] &= [\![s_x]\!] \ominus [\![s_y]\!] \\ &= \left[\!\left[m_x^0 m_x^1 \cdots m_x^k\right]\!\right] \ominus \left[\!\left[m_y^0 m_y^1 \cdots m_y^k\right]\!\right] \\ &= \left[\!\left[m_z^0 m_z^1 \cdots m_z^k\right]\!\right] \end{aligned} \tag{37}$$

The input of our FFHE scheme is binary bit; therefore, the ciphertext is

$$\left[\!\left[m_z^0 m_z^1 \cdots m_z^k\right]\!\right] = \left[\!\left[m_z^0\right]\!\right] \left[\!\left[m_z^1\right]\!\right] \cdots \left[\!\left[m_z^k\right]\!\right] \tag{38}$$

and m_z^0 is the sign bit of s_z.

Step 2. We need to check whether the signs of s_x and s_y are consistent.

$$\begin{aligned} \textit{same\_sign} = &\left[\!\left[m_x^0\right]\!\right] \otimes \left[\!\left[m_y^0\right]\!\right] \oplus \left([\![1]\!] \ominus \left[\!\left[m_x^0\right]\!\right]\right) \\ &\otimes \left([\![1]\!] \ominus \left[\!\left[m_y^0\right]\!\right]\right) \end{aligned} \tag{39}$$

same_sign is the ciphertext for 1, when the signs of s_x and s_y are the same, and *same_sign* is the ciphertext for 0, when the signs of s_x and s_y are different.

Step 3.

$$\begin{aligned} \textit{Gtb} = &\left([\![1]\!] \ominus \textit{same\_sign}\right) \otimes \left[\!\left[m_x^0\right]\!\right] \oplus \textit{same\_sign} \\ &\otimes \left[\!\left[m_z^0\right]\!\right] \end{aligned} \tag{40}$$

We define GTB protocol as $\mathrm{GTB}([\![s_x]\!], [\![s_y]\!]) = \textit{Gtb}$. *Gtb* is the ciphertext for 0, when $s_x \geq s_y$, and *Gtb* is ciphertext for 1, when $s_x < s_y$.

Equivalent-Bit Protocol. Additionally, it is necessary to check whether two numbers are equivalent or not. Therefore, we propose the Equivalent-Bit (EB) protocol. Given two ciphertexts $[\![s_x]\!]$ and $[\![s_y]\!]$, the EB protocol is to show whether $s_x = s_y$. The overall steps of EB protocol are shown as follows.

Step 1. We execute GTB protocol to check $s_x > s_y$ or $s_x < s_y$.

$$\begin{aligned} [\![g_1]\!] &= \mathrm{GTB}\left([\![s_x]\!], [\![s_y]\!]\right), \\ [\![g_2]\!] &= \mathrm{GTB}\left([\![s_y]\!], [\![s_x]\!]\right) \end{aligned} \tag{41}$$

Step 2.

$$[\![g_0]\!] = [\![g_1]\!] \otimes [\![g_2]\!] \oplus \left([\![1]\!] \ominus [\![g_1]\!]\right) \otimes \left([\![1]\!] \ominus [\![g_2]\!]\right) \tag{42}$$

We define EB protocol as $\mathrm{EB}([\![s_x]\!], [\![s_y]\!]) = [\![g_0]\!]$. Notice that if $g_0 = 1$, then $s_x = s_y$. Otherwise, $s_x \neq s_y$.

Left Shift Protocol. In floating-point number addition, it is necessary to unify the exponents of these floating-point numbers. Therefore, we propose the Left Shift (LS) protocol to adjust the significand of a floating-point number. Given a ciphertext of the significand $[\![s_x]\!]$ and two ciphertexts of the exponents $[\![e_x]\!]$, $[\![e_y]\!]$, we calculate $s_x^* = s_x \cdot 2^{e_x - e_y}$, where $e_x \geq e_y$. The overall steps of LS protocol are shown as follows.

Step 1. For i in $[0, 1, \cdots, e_{\max}]$ ($e_{\max}$ is an extremal exponent as described in Section 3.2),

$$\begin{aligned} [\![a]\!] &= \mathrm{GTB}\left([\![e_x]\!] \ominus \left[\!\left[e_y\right]\!\right], [\![i]\!]\right) \\ [\![b]\!] &= \mathrm{EB}\left([\![e_x]\!] \ominus \left[\!\left[e_y\right]\!\right], [\![i]\!]\right) \\ [\![s_x]\!] &= \left([\![1]\!] \ominus [\![a]\!]\right) \otimes \left([\![1]\!] \ominus [\![b]\!]\right) \otimes [\![2]\!] \otimes [\![s_x]\!] \end{aligned} \tag{43}$$

Return $[\![s_x^*]\!] = [\![s_x]\!]$.

We define LS protocol as

$$\mathrm{LS}\left(\left[\!\left[s_x\right]\!\right], \left[\!\left[e_x\right]\!\right] \ominus \left[\!\left[e_y\right]\!\right]\right) \begin{cases} \left[\!\left[s_x \cdot 2^{e_x - e_y}\right]\!\right] & e_x \geq e_y \\ \left[\!\left[s_x\right]\!\right] & e_x < e_y \end{cases} \tag{44}$$

We also propose floating-point number addition (FAdd) protocol and floating-point number multiplication (FMult) protocol to compute the addition and multiplication of the ciphertexts of the floating-point numbers. Suppose we have two floating-point numbers $x = (-1)^{s'_x} \cdot s_x \cdot 2^{e_x - k}$, $y = (-1)^{s'_y} \cdot s_y \cdot 2^{e_y - k}$. The ciphertexts are $[\![x]\!] = ([\![s'_x]\!], [\![s_x]\!], [\![e_x]\!])$ and $[\![y]\!] = ([\![s'_y]\!], [\![s_y]\!], [\![e_y]\!])$ respectively. The overall steps of FAdd protocol and FMult protocol are shown as follows.

Floating-Point Number Addition Protocol

Step 1. We need to check whether the signs of x and y are consistent.

$$\begin{aligned} same\_sign = {} & \left[\!\left[s'_x\right]\!\right] \otimes \left[\!\left[s'_y\right]\!\right] \oplus \left(\left[\!\left[1\right]\!\right] \ominus \left[\!\left[s'_x\right]\!\right]\right) \\ & \otimes \left(\left[\!\left[1\right]\!\right] \ominus \left[\!\left[s'_y\right]\!\right]\right) \end{aligned} \tag{45}$$

same_sign is the ciphertext for 1, when $s'_x = s'_y$, and *same_sign* is the ciphertext for 0, when $s'_x \neq s'_y$. Supposing that $s'_x = s'_y$, the sign bit of $x + y$ is $s'_x \cdot s'_y$.

Step 2. If $s'_x \neq s'_y$, the sign of $x + y$ is the same as the sign of the larger between $|x|$ and $|y|$, specifically, the sign bit of $x+y$ is s'_d as follows:

$$\begin{aligned} \left[\!\left[s^*_x\right]\!\right] &= \mathrm{LS}\left(\left[\!\left[s_x\right]\!\right], \left[\!\left[e_x\right]\!\right] \ominus \left[\!\left[e_y\right]\!\right]\right), \\ \left[\!\left[s^*_y\right]\!\right] &= \mathrm{LS}\left(\left[\!\left[s_y\right]\!\right], \left[\!\left[e_y\right]\!\right] \ominus \left[\!\left[e_x\right]\!\right]\right), \\ \left[\!\left[G_{s^*}\right]\!\right] &= \mathrm{GTB}\left(\left[\!\left[s^*_x\right]\!\right], \left[\!\left[s^*_y\right]\!\right]\right) \\ \left[\!\left[s'_d\right]\!\right] &= \left(\left[\!\left[1\right]\!\right] \ominus \left[\!\left[G_s\right]\!\right]\right) \otimes \left[\!\left[s'_x\right]\!\right] \oplus \left[\!\left[G_s\right]\!\right] \otimes \left[\!\left[s'_y\right]\!\right] \end{aligned} \tag{46}$$

Step 3. We calculate the sign of the addition of x and y.

$$\begin{aligned} \left[\!\left[s'_+\right]\!\right] = {} & same\_sign \otimes \left[\!\left[s'_x\right]\!\right] \otimes \left[\!\left[s'_y\right]\!\right] \\ & \oplus \left(\left[\!\left[1\right]\!\right] \ominus same\_sign\right) \otimes \left[\!\left[s'_d\right]\!\right] \end{aligned} \tag{47}$$

Step 4. When $s'_x \neq s'_y$, the significand of $x + y$ is as follows:

$$\begin{aligned} \left[\!\left[s_d\right]\!\right] = {} & \left(\left[\!\left[1\right]\!\right] \ominus \left[\!\left[G_s\right]\!\right]\right) \otimes \left(\left[\!\left[s^*_x\right]\!\right] \ominus \left[\!\left[s^*_y\right]\!\right]\right) \oplus \left[\!\left[G_s\right]\!\right] \\ & \otimes \left(\left[\!\left[s^*_y\right]\!\right] \ominus \left[\!\left[s^*_x\right]\!\right]\right) \end{aligned} \tag{48}$$

Therefore, we calculate the final significand of the addition of x and y.

$$\begin{aligned} \left[\!\left[s_+\right]\!\right] = {} & same\_sign \otimes \left(\left[\!\left[s^*_x\right]\!\right] \oplus \left[\!\left[s^*_y\right]\!\right]\right) \\ & \oplus \left(\left[\!\left[1\right]\!\right] \ominus same\_sign\right) \otimes \left[\!\left[s_d\right]\!\right] \\ = {} & \left[\!\left[m^0_+ m^1_+ \cdots m^k_+ \cdots m^{k+l}_+\right]\!\right] \end{aligned} \tag{49}$$

It shows that s_+ is $k+l$ binary bits, and the precision of the result is $k+l$.

Step 5. If $s_+ \geq 2^{k+|e_x - e_y|}$, it shows that the exponent of $x + y$ is $\max\{e_x, e_y\} + 1$. If $s_+ < 2^{k+|e_x - e_y|}$, it shows that the exponent of $x + y$ is $\max\{e_x, e_y\}$. The construction is as follows:

$$\begin{aligned} \left[\!\left[G_e\right]\!\right] &= \mathrm{GTB}\left(\left[\!\left[e_x\right]\!\right], \left[\!\left[e_y\right]\!\right]\right), \\ \left[\!\left[e_{\max}\right]\!\right] &= \left(\left[\!\left[1\right]\!\right] \ominus \left[\!\left[G_e\right]\!\right]\right) \otimes \left[\!\left[e_x\right]\!\right] \oplus \left[\!\left[G_e\right]\!\right] \otimes \left[\!\left[e_y\right]\!\right] \\ \left[\!\left[e_d\right]\!\right] &= \left(\left[\!\left[1\right]\!\right] \ominus \left[\!\left[G_e\right]\!\right]\right) \otimes \left(\left[\!\left[e_x\right]\!\right] \ominus \left[\!\left[e_y\right]\!\right]\right) \oplus \left[\!\left[G_e\right]\!\right] \\ & \quad \otimes \left(\left[\!\left[e_y\right]\!\right] \ominus \left[\!\left[e_x\right]\!\right]\right) \\ \left[\!\left[G_{s_+}\right]\!\right] &= \mathrm{GTB}\left(\left[\!\left[s_+\right]\!\right], \mathrm{LS}\left(\left[\!\left[2^k\right]\!\right], \left[\!\left[e_d\right]\!\right]\right)\right) \end{aligned} \tag{50}$$

We calculate the exponent of the addition of x and y.

$$\begin{aligned} \left[\!\left[e_+\right]\!\right] = {} & \left(\left[\!\left[1\right]\!\right] \ominus \left[\!\left[G_{s_+}\right]\!\right]\right) \otimes \left(\left[\!\left[e_{\max}\right]\!\right] \oplus \left[\!\left[1\right]\!\right]\right) \oplus \left[\!\left[G_{s_+}\right]\!\right] \\ & \otimes \left[\!\left[e_{\max}\right]\!\right] \end{aligned} \tag{51}$$

Step 6. Finally, we need to keep the result significand k bits.

$$\begin{aligned} \left[\!\left[g_+\right]\!\right] &= \mathrm{EB}\left(\left[\!\left[m^{k+1}_+\right]\!\right], \left[\!\left[1\right]\!\right]\right) \\ \left[\!\left[s^*_+\right]\!\right] &= \left[\!\left[g_+\right]\!\right] \otimes \left(\left[\!\left[m^0_+ m^1_+ \cdots m^k_+\right]\!\right] + \left[\!\left[1\right]\!\right]\right) \\ & \quad \oplus \left(\left[\!\left[1\right]\!\right] \ominus \left[\!\left[g_+\right]\!\right]\right) \otimes \left[\!\left[m^0_+ m^1_+ \cdots m^k_+\right]\!\right] \end{aligned} \tag{52}$$

Therefore, s^*_+ is k binary bits. The result of addition can be represented as$([\![s'_+]\!], [\![s^*_+]\!], [\![e_+]\!])$, specifically, $\mathrm{FAdd}(([\![s'_x]\!], [\![s_x]\!], [\![e_x]\!]), ([\![s'_y]\!], [\![s_y]\!], [\![e_y]\!])) = ([\![s'_+]\!], [\![s^*_+]\!], [\![e_+]\!])$.

As above, the precision is still k. Floating-point addition does not add additional relative error. For subsequent operations, the precision is still k, and the relative error $E_{rel}(x+\mathrm{y}) \leq 0.5 \cdot 2^{1-k}$ as in Section 3.2.

Floating-Point Number Multiplication Protocol

Step 1. First, we calculate the sign of the multiplication of x and y.

$$\left[\!\left[s'_\times\right]\!\right] = \left[\!\left[s'_x\right]\!\right] \otimes \left(\left[\!\left[1\right]\!\right] \ominus \left[\!\left[s'_y\right]\!\right]\right) \oplus \left(\left[\!\left[1\right]\!\right] \ominus \left[\!\left[s'_x\right]\!\right]\right) \otimes \left[\!\left[s'_y\right]\!\right] \tag{53}$$

Step 2. Then, we calculate the exponent.

$$\left[\!\left[e_\times\right]\!\right] = \left[\!\left[e_x\right]\!\right] \oplus \left[\!\left[e_y\right]\!\right] \tag{54}$$

Step 3. Finally, we calculate the significand.

$$\begin{aligned} \left[\!\left[s_\times\right]\!\right] &= \left[\!\left[s_x\right]\!\right] \otimes \left[\!\left[s_y\right]\!\right] \\ &= \left[\!\left[m^0_x m^1_x \cdots m^k_x\right]\!\right] \otimes \left[\!\left[m^0_y m^1_y \cdots m^k_y\right]\!\right] \\ &= \left[\!\left[m^0_\times m^1_\times \cdots m^k_\times m^{k+1}_\times \cdots m^{2k}_\times\right]\!\right] \end{aligned} \tag{55}$$

It shows that $s_\times$ is $2k$ binary bits, and the precision of the result is $2k$.

Step 4. Thereafter, we need to keep the result significand k bits.

$$\begin{aligned}\left[\!\left[g_+\right]\!\right] &= EB\left(\left[\!\left[m_\times^{k+1}\right]\!\right], \left[\!\left[1\right]\!\right]\right)\\ \left[\!\left[s_\times^*\right]\!\right] &= \left[\!\left[g_+\right]\!\right] \otimes \left(\left[\!\left[m_\times^0 m_\times^1 \cdots m_\times^k\right]\!\right] + \left[\!\left[1\right]\!\right]\right)\\ &\quad \oplus \left(\left[\!\left[1\right]\!\right] \ominus \left[\!\left[g_+\right]\!\right]\right) \otimes \left[\!\left[m_\times^0 m_\times^1 \cdots m_\times^k\right]\!\right]\end{aligned} \tag{56}$$

Therefore, $s_\times^*$ is k binary bits. The result of multiplication can be represented as $(\left[\!\left[s_\times'\right]\!\right], \left[\!\left[s_\times^*\right]\!\right], \left[\!\left[e_\times\right]\!\right])$, specifically,

$$\begin{aligned}&\mathrm{FMult}\left(\left(\left[\!\left[s_x'\right]\!\right], \left[\!\left[s_x\right]\!\right], \left[\!\left[e_x\right]\!\right]\right), \left(\left[\!\left[s_y'\right]\!\right], \left[\!\left[s_y\right]\!\right], \left[\!\left[e_y\right]\!\right]\right)\right)\\ &\quad = \left(\left[\!\left[s_\times'\right]\!\right], \left[\!\left[s_\times^*\right]\!\right], \left[\!\left[e_\times\right]\!\right]\right).\end{aligned} \tag{57}$$

As above, the precision is still k. According to Section 3.2, the upper bound of the relative error for multiplication will be twice the original, $E_{rel}(x \cdot y) \le 2 \cdot 0.5 \cdot 2^{1-k}$. That is, after the n-layer multiplication operation, the upper bound will be $2^n \cdot 0.5 \cdot 2^{1-k}$, which is the same as the floating-point number operation in plaintext.

The FFHE scheme completely simulates a floating-point operation in plaintext in the form of a circuit so that the relative error of operation result in ciphertext is not increased compared with the result in plaintext.

7. Calculation Types Generalization of FFHE

In Sections 5 and 6, we proposed our revised somewhat homomorphic encryption scheme (RSHE) and our floating-point fully homomorphic encryption scheme (FFHE). As we can see, FFHE only describe five basic protocols to achieve different floating-point numbers operations including addition and multiplication, and most of the current homomorphic encryption schemes also only involve addition and multiplication [2–5, 18]. However, complex functions taking floating-point numbers as input are more common in reality, and we also need to protect the privacy of input and output for these complex functions. For example, in the e-healthcare cloud service [14], we often require a mathematic modeling method to analyze patients' health data, which may utilize some complex functions taking floating-point numbers as input. In this section, supporting calculation types of FFHE are generalized from addition and multiplication to analytic functions, for example, reciprocal function, exponential function, and logarithmic function. The evaluation of reciprocal function also provides a new ideal for computing division without privacy leakage.

Definition 8 (function homomorphism). Let $f(x)$ be an analytic function, homomorphic encryption scheme that satisfies function homomorphism if

$$\mathbf{Dec}\left(sk, f'\left(\mathbf{Enc}\left(pk, x\right)\right)\right) = f\left(x\right) \tag{58}$$

where f' is a corresponding function operating over ciphertexts.

Using Taylor series, our FFHE can homomorphically evaluate analytic functions $f(x)$ taking floating-point numbers as input. According to Section 6.3, FFHE achieves protocols of evaluating addition and multiplication for floating-point numbers, and polynomials consist of addition and multiplication. Therefore, FFHE can homomorphically evaluate polynomial function. The Taylor series establishes a connection between the analytic function and polynomial function. The definition of the Taylor series is as follows.

Definition 9 (Taylor series). The Taylor series of a real or complex-valued function $f(x)$ that is infinitely differentiable at a real or complex number a is the power series:

$$\begin{aligned}&f\left(a\right) + \frac{f'\left(a\right)}{1!}\left(x-a\right) + \frac{f''\left(a\right)}{2!}\left(x-a\right)^2\\ &\quad + \frac{f'''\left(a\right)}{3!}\left(x-a\right)^3 + \cdots\end{aligned} \tag{59}$$

which can be written in the more compact sigma notation as

$$\sum_{n=0}^{\infty} \frac{f^{(n)}\left(a\right)}{n!}\left(x-a\right)^n \tag{60}$$

where $n!$ denotes the factorial of n and $f^{(n)}(a)$ denotes the n-th derivative f evaluated at the point a. The derivative of order zero of f is defined to be f and $(x-a)^0$ and 0! are both defined to be 1. When a=0, the series is also called a Maclaurin series.

Utilizing the Taylor series, analytic functions can be rewritten as a combination of additions and multiplications, which provide a method for evaluating analytic functions without revealing privacy. However, the essential condition must be satisfied that is the independent variable x must be within the convergent domain. We cite reciprocal and exponent which can be rewritten as Taylor series in the example below. Let n is the degree of the polynomial. The larger the n, the smaller the error. Let a=0 below.

Example 10. reciprocal

Description: calculate the reciprocal of x, that is, $1/x$, without revealing the value of x.

Taylor series:

$$\begin{aligned}\frac{1}{x} &= \sum_{n=0}^{\infty}\left(-1\right)^n\left(x-1\right)^n\\ &= 1 - \left(x-1\right) + \cdots + \left(-1\right)^n\left(x-1\right)^n\\ &\quad + \left(-1\right)^{n+1}\frac{\left(x-1\right)^{n+1}}{\left[1+\theta\left(x-1\right)\right]^{n+2}}\end{aligned} \tag{61}$$

Convergent domain: $0 < x < 2$, where $0 < \theta < 1$.

Error: $|\Delta| = \max|(-1)^{n+1}((x-1)^{n+1}/[1+\theta(x-1)]^{n+2})| \le (x-1)^{n+1},\ x \in (0,2),\ 0 < \theta < 1$

Table 2: Performance comparison of fully homomorphic encryption scheme.

Scheme	Size of pk	Size of sk	Expansion rate of ciphertext	Computational complexity	Process floating-point numbers operation	Process analytic function operation
DGHV [2]	$\widetilde{O}(\lambda^{13})$	$\widetilde{O}(\lambda^{7})$	$\widetilde{O}(\lambda^{5})$	$O(\lambda^{5})$	No	No
[3]	$\widetilde{O}(\lambda^{13})$	$\widetilde{O}(\lambda^{9})$	$\widetilde{O}(\lambda^{3})$	$O(\lambda^{3})$	No	No
[4]	$\widetilde{O}(\lambda^{8})$	$\widetilde{O}(\lambda^{3})$	$\widetilde{O}(\lambda^{5})$	$O(\lambda^{4})$	No	No
[6]	$\widetilde{O}(\lambda^{7})$	$\widetilde{O}(\lambda^{3.5})$	$\widetilde{O}(\lambda^{3})$	$O(\lambda^{3})$	No	No
FFHE	$\widetilde{O}(\lambda^{6})$	$\widetilde{O}(\lambda^{5})$	$\widetilde{O}(\lambda^{3})$	$O(\lambda^{3})$	Yes	Yes

The precision of floating-point numbers in our FFHE is k. If $|\Delta| < 2^{-k}$, the Taylor polynomial cannot reduce the precision. Specifically, the degree n must satisfy

$$n > \frac{-k}{\log(x-1)} - 1 \tag{62}$$

Approximate calculation of the reciprocal with the Taylor series provides a method for calculating division.

Example 11. exponent

Description: calculate the exponent of x, that is, e^x.

Taylor series:

$$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!} = 1 + x + \frac{x^2}{2!} + \cdots + \frac{x^n}{n!} + \frac{e^{\theta x} \cdot x^{n+1}}{(n+1)!} \tag{63}$$

Convergent domain: $-\infty < x < +\infty$, where $0 < \theta < 1$,

Error: $|\Delta| = \max|(e^{\theta x} \cdot x^{n+1})/(n+1)!| < (e^{|x|}/(n+1)!)|x|^{n+1}, 0 < \theta < 1$

Let $x = 1$, then $e = 1 + 1 + 1/2! + \cdots + 1/n!$, and the error $|\Delta| < e/(n+1)! < 3/(n+1)!$, let $n = 10$, $e = 2.718282$, the error will not exceed 10^{-6}.

Other analytic functions, such as trigonometric functions, logarithmic functions, and variants of these functions, can also be rewritten as the Taylor series; however, the independent variable must be within the convergent domain. Below, we illustrate how to calculate beyond the convergent domain.

As noted above, when calculating the reciprocal of x, the convergent domain of x is $(0, 2)$. Suppose $x = (-1)^{s'} \cdot s \cdot 2^{e-k}$; we can the transform the formula as

$$\frac{1}{x} = \frac{1}{x \cdot 2^{-t}} = \frac{1}{(-1)^{s'} \cdot s \cdot 2^{e-t-k}} \cdot 2^{-t} \tag{64}$$

Let $x^* = x \cdot 2^{-t} \in (0, 2)$, and the exponent of x^* is $e - t$. Then, $1/(x \cdot 2^{-t})$ can be rewritten as the Taylor series as above within the convergent domain. The product of the Taylor series and 2^{-t} is the approximate value of $1/x$, with the precision loss of t bits. By increasing the value of n and k, the precision can be increased appropriately. If $x < 0$, the approximate value can be calculated in the same way. Calculation of the other analytic functions with the independent variable being out of the convergent domain may be achieved by a similar method. Enlarging or reducing the range of independent variables is an efficient method to satisfy the condition of the convergent domain. Due to the variety and length restrictions of the paper, they are not addressed here.

Precision can be improved by increasing n, but it leads to a higher computation complexity. According to the actual demand, we can determine the size of n to achieve balance between precision and computation complexity.

FFHE scheme is a fully homomorphic encryption scheme that supports analytic function operations based on floating-point numbers. Table 2 compares our FFHE scheme with [2–4, 6].

8. Conclusion

In this paper, we followed Gentry's blueprint to construct a revised fully homomorphic encryption scheme that supports many analytic function operations on floating-point numbers, not just addition and multiplication operations on integers. Through packaging a matrix of plaintext bits as a single ciphertext, we reduce the expansion rate of the ciphertext to $\widetilde{O}(\lambda^3)$. At the same time, with a quadratic form instead of a linear form in three types of public key and multiple pseudo-random number generator, we reduce the public key size to $\widetilde{O}(\lambda^6)$. We constructed a revised floating-point fully homomorphic encryption scheme (FFHE). The scheme remains semantically secure under the error-free approximate-GCD problem and sparse subset sum problem (SSSP). In FFHE, operation on the ciphertexts of floating-point numbers does not increase additional precision loss. That means addition does not raise the upper bound of the relative error, while n multiplication raises the upper bound of the relative error to $n \cdot 0.5 \cdot 2^{1-k}$, as does the operation on plaintext of a floating-point number.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

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

Acknowledgments

This work was supported by the National Natural Science Foundation of China (61572263, 61502251, and 61502243), the University Natural Science Research Project of Jiangsu Province (14KJB520031), and Postdoctoral Science Foundation Project of China (2016M601859).

References

[1] D. G. Feng, M. Zhang, Y. Zhang, and Z. Xu, "Study on cloud computing security," *Journal of Software*, vol. 22, no. 1, pp. 71–83, 2011.

[2] M. Van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan, "Fully homomorphic encryption over the integers," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 24–43, Springer, Berlin, Germany, 2010.

[3] 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.

[4] J.-S. Coron, A. Mandal, D. Naccache, and M. Tibouchi, "Fully homomorphic encryption over the integers with shorter public keys," in *Annual Cryptology Conference*, pp. 487–504, Springer, Heidelberg, Germany, 2011.

[5] J.-S. Coron, D. Naccache, and M. Tibouchi, "Public key compression and modulus switching for fully homomorphic encryption over the integers," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 446–464, Springer, Berlin, Germany, 2012.

[6] W. J. Xiong, Y. Z. Wei, and H. Y. Wang, "An improved fully homomorphic encryption scheme over the integers," *Journal of Cryptologic Research*, vol. 3, no. 1, pp. 67–78, 2016.

[7] R. A. Popa, C. M. S. Redfield, N. Zeldovich, and H. Balakrishnan, "CryptDB: protecting confidentiality with encrypted query processing," in *Proceedings of the 23rd ACM Symposium on Operating Systems Principles (SOSP '11)*, pp. 85–100, October 2011.

[8] R. A. Popa, C. M. S. Redfield, N. Zeldovich, and H. Balakrishnan, "CryptDB: Processing queries on an encrypted database," *Communications of the ACM*, vol. 55, no. 9, pp. 103–111, 2012.

[9] W. K. Wong, B. Kao, D. W. L. Cheung, R. Li, and S. M. Yiu, "Secure query processing with data interoperability in a cloud database environment," in *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD '14)*, pp. 1395–1406, ACM, June 2014.

[10] K. Xue, S. Li, J. Hong, Y. Xue, N. Yu, and P. Hong, "Two-cloud secure database for numeric-related SQL range queries with privacy preserving," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 7, pp. 1596–1608, 2017.

[11] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Proceedings of the 18th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, vol. 547, pp. 223–238, Springer, 1999.

[12] N. P. Smart and F. Vercauteren, "Fully homomorphic encryption with relatively small key and ciphertext sizes," in *International Workshop on Public Key Cryptography*, pp. 420–443, Springer, Berlin, Germany, 2010.

[13] C. Gentry, S. Halevi, and N. P. Smart, "Better bootstrapping in fully homomorphic encryption," in *International Workshop on Public Key Cryptography*, pp. 1–16, Springer, Heidelberg, Germany, 2012.

[14] X. Liu, R. Lu, J. Ma, L. Chen, and B. Qin, "Privacy-preserving patient-centric clinical decision support system on Naïve Bayesian classification," *IEEE Journal of Biomedical and Health Informatics*, vol. 20, no. 2, pp. 655–668, 2016.

[15] R. L. Rivest, A. Shamir, and L. Adleman, *A Method for Obtaining Digital Signatures and Public-Key Cryptosystems*, ACM, 1983.

[16] C. Gentry, "Fully homomorphic encryption using ideal lattices," *Proceedings of the Annual ACM Symposium on Theory of Computing*, vol. 9, no. 4, pp. 169–178, 2009.

[17] C. Gentry, *A Fully Homomorphic Encryption Scheme*, Stanford University, 2009.

[18] 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.

[19] N. J. Higham, *Accuracy and Stability of Numerical Algorithm*, Society for Industrial and Applied Mathematics, Philadelphia, Pa, USA, 2nd edition, 2002.

[20] R. L. Rivest, L. Adleman, and M. L. Dertouzos, "On data banks and privacy homomorphisms," *Foundations of Secure Computation*, vol. 4, no. 11, pp. 169–180, 1978.

[21] C. Gentry and S. Halevi, "Implementing Gentry's fully-homomorphic encryption scheme," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 129–148, Springer, Heidelberg, Germany, 2011.

[22] J. W. Bos, K. Lauter, J. Loftus, and M. Naehrig, "Improved security for a ring-based fully homomorphic encryption scheme," in *IMA International Conference on Cryptography and Coding*, pp. 45–64, Springer, Heidelberg, Germany, 2013.

[23] C. Gentry, A. Sahai, and B. Waters, "Homomorphic encryption from learning with errors: conceptually-simpler, asymptotically-faster, attribute-based," in *Advances in Cryptology–CRYPTO 2013*, pp. 75–92, Springer, Berlin, Germany, 2013.

[24] Z. Li, C. Ma, L. Zhang, and W. Zhang, "Two types LWE-based multi-bit lattice-based encryption schemes," *Netinfo Security*, no. 10, pp. 1–17, 2017.

[25] Z. G. wang, C. G. Ma, and X. Q. Shi, "Research on full homomorphic encryption scheme based on binary LWE," *Netinfo Security*, no. 7, pp. 41–50, 2015.

[26] H. Wang, "Identity-based distributed provable data possession in multi-cloud storage," *IEEE Transactions on Services Computing*, vol. 8, no. 2, pp. 328–340, 2015.

[27] H. Wang, "Proxy provable data possession in public clouds," *IEEE Transactions on Services Computing*, vol. 6, no. 4, pp. 551–559, 2013.

[28] H. Wang, D. He, and J. Han, "VOD-ADAC: anonymous distributed fine-grained access control protocol with verifiable outsourced decryption in public cloud," *IEEE Transactions on Services Computing*, 2017.

[29] H. Wang, D. He, J. Yu, and Z. Wang, "Incentive and unconditionally anonymous identity-based public provable data possession," *IEEE Transactions on Services Computing*, 2016.

[30] S. Arita and S. Nakasato, "Fully homomorphic encryption for point numbers," Cryptology ePrint Archive Report 2016/402, 2016, http://eprint.iacr.org/2016/402.

[31] J. Fan and F. Vercauteren, "Somewhat practical fully homomorphic encryption," Cryptology ePrint Archive Report 2012/144, 2012, http://eprint.iacr.org/2012/144.

[32] J. H. Cheon, A. Kim, M. Kim et al., "Floating-point homomorphic encryption," Cryptology ePrint Archive Report 2016/421, Cryptology ePrint Archive, 2016, http://eprint.iacr.org/2016/421.

[33] A. Costache, N. P. Smart, S. Vivek et al., "Fixed point arithmetic in she schemes," Cryptology ePrint Archive Report 2016/250, 2016, http://eprint.iacr.org/2016/250.

[34] X. Liu, R. H. Deng, W. Ding, R. Lu, and B. Qin, "Privacy-preserving outsourced calculation on floating point numbers," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 11, pp. 2513–2527, 2016.

[35] Y. Chen and P. Q. Nguyen, "Faster algorithms for approximate common divisors: breaking fully-homomorphic-encryption challenges over the integers," in *International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 502–519, Springer, Berlin, Germany, 2012.

[36] M. J. Coster, A. Joux, B. A. LaMacchia, A. M. Odlyzko, C.-P. Schnorr, and J. Stern, "Improved low-density subset sum algorithms," *Computational Complexity*, vol. 2, no. 2, pp. 111–128, 1992.