Hindawi
Security and Communication Networks
Volume 2018, Article ID 6520258, 11 pages
https://doi.org/10.1155/2018/6520258

Research Article

Bootstrapping of FHE over the Integers with Large Message Space

Zhizhu Lian,[1] **Yupu Hu**,[1] **Hu Chen**,[2] **and Baocang Wang**[1]

[1]*State Key Laboratory of Integrated Service Networks, Xidian University, Xian 710071, China*
[2]*Jiangsu Key Laboratory of Education Big Data Science and Engineering, Jiangsu Normal University, Xuzhou 221116, China*

Correspondence should be addressed to Zhizhu Lian; 594339079@qq.com

Received 4 April 2018; Accepted 27 May 2018; Published 29 July 2018

Academic Editor: Rongxing Lu

For the decryption of the fully homomorphic encryption (FHE) over the integers with the message space $\mathbb{Z}_Q$, Nuida and Kurosawa proposed a $Q^4\lambda$-multiplicative-degree circuit to compute it at Eurocrypt 2015, where λ is the security parameter and the message size Q is a constant. Since the degree of the decryption circuit is polynomial in Q, the range of the message size Q is limited. In this work, we solve this open problem as long as Q is large enough (larger than λ). We represent the decryption circuit as a arithmetic polynomial of multiplicative degree $108 \cdot \lambda \log^3 \lambda$, which is independent of the message size Q except a constraint $Q > \lambda$. Moreover, the bootstrapping process requires only $O(\lambda \cdot \log \lambda)$ number of multiplications to implement the decryption circuit, which is significantly lower than $O(\lambda^4)$ of Nuida and Kurosawa's work. We also show the efficiency of the FHE scheme with message space $\mathbb{Z}_Q$ compared to the FHE scheme with binary message space. As a result, we have that the former is preferable.

1. Introduction

In 1978, Rivest, Adleman, and Dertouzos introduced the notion of fully homomorphic encryption (FHE) which can compute any circuit on encrypted data without decryption [1]. It solves the ciphertext data calculation and the privacy protection of private cloud user in cloud computing environment. Until 2009, Gentry proposed firstly a fully homomorphic encryption scheme based on ideal lattices [2].

Gentry's Blueprint. First, Gentry constructed a somewhat homomorphic encryption (SHE) scheme, whose ciphertexts contain some noises for the security of the scheme. Noises, however, also limit the number of the homomorphic operations, e.g., ciphertexts multiplications. The second step is squashing the decryption circuit associated with an arbitrary ciphertext to obtain a low enough degree polynomial in the ciphertext bits and the secret key bits, which can be homomorphically evaluated by SHE scheme (called bootstrappable scheme). The last step is Gentry's breakthrough, called bootstrapping, which refreshed ciphertexts by homomorphically evaluating this low multiplicative degree decryption circuit on the encryption of those bits, thus resulting in a new encryption of the same plaintext, but with possibly reduced noise. The refreshed ciphertexts can then support subsequent homomorphic operations. By repeatedly refreshing ciphertexts, the number of permissible homomorphic operations becomes unlimited. So a pure FHE scheme is transformed from the bootstrappable SHE scheme.

1.1. FHE over the Integers. At Eurocrypt 2010, van Dijk et al. [3] proposed the first FHE scheme over the integers (called DGHV scheme) following Gentry's blueprint. The security of DGHV relies on the hardness of the Approximate Greatest Common Divisor problem (AGCD) and the Sparse Subset Sum problem (SSSP). Several works have dramatically improved the efficiency and the hardness assumption needed to implement it, including [4–12]. Some of the schemes above are leveled FHE scheme, but they essentially follow Gentry's blueprint.

In DGHV scheme, for the ciphertext c encryption of message m under secret key p (where p is a prime number),

the decryption $m \longleftarrow c \bmod p \bmod 2$ can be turned into the following circuit:

$$(c \bmod p) \bmod 2 = (c \bmod 2) \oplus \left(\left\lfloor \sum_{i=1}^{\Theta} s_i \mathbf{z}_i \right\rceil \bmod 2 \right), \quad (1)$$

where the Θ length vector $\mathbf{s} = (s_1, \ldots, s_\Theta) \in \{0,1\}^\Theta$ is the secret key with Hamming weight θ, and each $\mathbf{z}_i = (z_{i,0}.z_{i,-1}, \ldots, z_{i,-n})_2$ is a real number with $n = \lceil \log \theta \rceil + 3$ bits of precision after the binary point, satisfying $\sum_{i=1}^{\Theta} s_i \mathbf{z}_i \approx c/p$. This decryption circuit is implemented with a binary circuit of degree $64 \cdot \lambda \log^2 \lambda$ on the secret key bits $(s_1, \ldots, s_\Theta)$.

Message Space. Practically, the computation over bitwise encryptions is not efficient. It is important to construct the FHE over lager integers for secure integer arithmetic (see [6, 13]). Fortunately, it is quite straightforward to extend the message space from $\mathbb{Z}_2$ to $\mathbb{Z}_Q$ for SHE scheme [6, 14]. But they cannot convert this extended SHE scheme to an FHE scheme via the bootstrapping procedure. Because computing Q-ary addition seems to need more complex carry computations than binary addition, it seemed technically difficult to obtain a mod-Q arithmetic circuit that performs the decryption circuit

$$m \longleftarrow c \bmod p \bmod Q. \quad (2)$$

At Eurocrypt 2015, Nuida and Kurosawa [8] proposed a Q-ary half adder, yielding the carry a in the procedure $x + y = aQ + b$ for any $x, y \in \mathbb{Z}_Q$. They determined a carry function

$$a = f_{carry,Q}(x, y) = \sum_{i=1}^{Q-1} \binom{x}{i}_Q \binom{y}{Q-i}_Q, \quad (3)$$

where $\binom{x}{i}_Q = x(x-1)\ldots(x-i+1)/(i(i-1)\ldots 1) \bmod Q$. It has the multiplicative degree Q. The squashed decryption in [8] works as

$$(c \bmod p) \bmod Q = (c - p \left(\left\lfloor \sum_{i=1}^{\Theta} s_i \mathbf{z}_i \right\rceil \right) \mod Q, \quad (4)$$

where Q is a constant prime, the secret key $\mathbf{s} = (s_1, \ldots, s_\Theta) \in \{0,1\}^\Theta$ is Θ length vector with Hamming weight $\theta = \lambda$, and $\mathbf{z}_i = (z_{i,0}.z_{i,-1}, \ldots, z_{i,-L})_Q < Q$ is a real number with $L = \lceil \log_Q \lambda \rceil + 2$ bits of precision after the Q-ary point, satisfying $\sum_{i=1}^{\Theta} s_i \mathbf{z}_i \approx c/p$. The decryption circuit is computed by a mod-Q arithmetic circuit of multiplicative degree $Q^4\lambda = O(\lambda)$, where Q is a constant.

In 2017, Cheon et al. [15] presented a faster bootstrapping of FHE over the integers than the previous work in [8]. The degree of the decryption is $O(\lambda^{1+\varepsilon})$, and the number of homomorphic multiplications is $O(\log^2 \lambda)$, where ε is some small constant (being affected by the modulus Q).

However, the modulus Q still needs to be a constant.

For $Q > 8\Theta^2$, Cheon and Kim [16] expressed the decryption circuit as an $\mathscr{L}$-restricted depth-3 $(\sum \prod \sum)$ circuit by the technique in [17]. The $\mathscr{L}$-degree is at most $8\Theta^2$ and the number of product gates is at most $8\Theta^2 + \Theta + 1$. As we know, Θ is $\omega(\lambda^6 \log \lambda)$ in [3] and is reduced to $\widetilde{O}(\lambda^3)$ in [4]. The decryption is too complexity to bootstrap. So, in the FHE scheme, the ciphertext associated with the large prime message space needs a low-degree decryption circuit.

Efficiency. To evaluate homomorphically a mod-Q arithmetic circuit, one can use the FHE scheme with message space $\mathbb{Z}_Q$ directly, or one can firstly convert the arithmetic circuit to a Boolean one and carry out all the computation using an FHE scheme with binary message space. At ACNS 2016, Kim and Tibouchi [18] compared the two approaches for the Nuida-Kurosawa scheme, denoted by NK, and showed that the scheme NK with nonbinary message space is less efficient than its variant with binary message space. Fortunately, the bootstrapping method proposed by Cheon et al. [15] is worthwhile for Q of constant size by comparing both above approaches for CLT scheme. However, the modulus Q still needs to be a constant.

Therefore, it is open for large value of Q to express the decryption circuit of FHE schemes with the form (5) as a low-degree polynomial.

$$m \longleftarrow (c \bmod p) \bmod Q \quad (5)$$

1.2. Contributions. In this paper, we solve this open problem as long as Q is large enough (larger than θ).

The usual technique for squashing the decryption circuit amounts to homomorphically evaluating a large integer sum of the form $\sum_{i=1}^{\Theta} s_i \mathbf{z}_i$, where the s_i are secret bits and the $\mathbf{z}_i$ are public constants computed from the original ciphertexts and public parameters. In [8], Nuida and Kurosawa represented the $\mathbf{z}_i$'s as their Q-ary expansion and applied the mod-Q circuit for iterated addition. And they have also proved that the degree Q of the polynomial $f_{carry,Q}(x, y)$ computing the carry of Q-ary half adder is the lowest degree. In order to obtain a low enough degree (be independent of Q) of decryption circuit, we cannot deal with the carry bit any more. Instead, in this paper we use $\mathbf{z}_i = (z_{i,0}.z_{i,-1}, \ldots, z_{i,-n})_2$ the binary representation of the real number $\mathbf{z}_i$. This means that we have to use mod-Q arithmetic circuit gates to emulate bit operations. Specifically for bits a and b, the XOR operation is computed by $a \oplus b = a + b - 2ab \bmod Q$, and the AND operation is computed by $a \odot b = ab \bmod Q$. So we can use the mod-Q arithmetic circuit to implement the decryption circuit. Usually, emulating binary operations are not that efficient since emulating binary addition needs multiplication. The challenge is how to compute it efficiently.

Note that if using only a three-for-two trick, as mentioned in Section 2.2, the decryption can be implemented with a multiplicative degree $O(\lambda^3)$ of mod-Q arithmetic circuit, which is better than the result of [16]. Our main contribution is reducing the multiplicative degree to $O(\lambda \log^3 \lambda)$ for any large prime Q with a constraint $Q > \lambda$.

Now let us recall the circuit procedure computing $\lfloor \sum_{i=1}^{\Theta} s_i \mathbf{z}_i \rceil \bmod 2$ in DGHV scheme [3].

(1) The first circuit computes the Hamming weight of the vector $\boldsymbol{\sigma}^{(-j)} = (s_1 z_{1,-j}, \ldots, s_\Theta z_{\Theta,-j})$, i.e., $\mathbf{W}_{-j} =$

$\sum_{i=1}^{\Theta} s_i z_{i,-j}$ for $j = 0, 1, \ldots, n$, and denotes the binary representation of $\mathbf{W}_{-j}$ as $(w_{-j,t}, \ldots, w_{-j,0})_2$. Hence, $\sum_{i=1}^{\Theta} s_i \mathbf{z}_i = \mathbf{W}_0 + 2^{-1}\mathbf{W}_{-1} + \ldots + 2^{-n}\mathbf{W}_{-n}$. Specifically, for $0 \le i \le t$, $0 \le j \le n$, the i-th bit $w_{-j,i}$ of $\mathbf{W}_{-j}$ can be obtained by using the elementary symmetric polynomial $e(\boldsymbol{\sigma}^{(-j)})$.

(2) The second circuit computes $\mathbf{a}$ and $\mathbf{b}$, satisfying $\mathbf{W}_0 + 2^{-1}\mathbf{W}_{-1} + \ldots + 2^{-n}\mathbf{W}_{-n} = \mathbf{a} + \mathbf{b}$ by applying the three-for-two trick over $\mathbb{Z}_2$ repeatedly.

(3) The third circuit computes $\lfloor \mathbf{a} + \mathbf{b} \rceil \bmod 2$ by a polynomial of degree 4.

In this work, we use mod-Q arithmetic circuit to simulate those bit operations in the above binary circuit. It is easy to simulate the second circuit by applying the three-for-two trick over $\mathbb{Z}_Q$. It will cost some additional multiplicative degree, since we need an arithmetic polynomial of degree 2 to compute the XOR operation. The third circuit is also easy to be simulated with a polynomial of degree 4.

However, to emulate the elementary symmetric polynomial $e(\boldsymbol{\sigma}^{(-j)})$ in step (1), it will take a polynomial of a high degree (greater than Θ, where Θ is $\omega(\lambda^6 \log \lambda)$ in [3] and is reduced to $\widetilde{O}(\lambda^3)$ in [4]). This cost is unacceptable. So we need to find a new arithmetic function to compute $w_{-j,i}$, the bits in the binary representation of $\mathbf{W}_{-j}$.

Our main idea is as follows.

If we know the value of an integer b, it is easy to obtain each bit in the binary representation of b, but if we only get the range of value of b, namely, $b < \theta$ for some integer θ, it can be a little tricky to get each bit of b. We observe that we can overcome it by applying Lagrange interpolating polynomial, as shown in Section 2.3. Since the Hamming weight of the secret key vector $\mathbf{s} = (s_1, \ldots, s_\Theta)$ is θ, the Hamming weight of the vector $\boldsymbol{\sigma}^{(-j)} = (s_1 z_{1,-j}, \ldots, s_\Theta z_{\Theta,-j})$ is not bigger than θ, namely, $\mathbf{W}_{-j} \le \theta$. So we can get $\mathbf{W}_{-j}$ just by using the mod-Q addition gate to directly add up at the cost of an additional condition that $Q > \theta$. Then, for $0 \le i \le \lceil \log \theta \rceil - 1$, we can obtain all bits $w_{-j,i}$ by applying *Lagrange interpolating polynomial* on $\mathbf{W}_{-j}$.

Conclusion: now we can express the decryption as mod-Q arithmetic polynomial with a constraint $Q > \theta$. The simulation circuit computing step (1) is θ degree of the Lagrange interpolating polynomial. The simulation circuit computing step (2) has the multiplicative degree at most $3^{\lceil \log_{3/2} n \rceil + 2}$. Hence, the multiplicative degree of our decryption circuit is

$$\theta \cdot \left(3^{\lceil \log_{3/2} n \rceil + 2}\right) \cdot 4 \le 108 \cdot \lambda \log^3 \lambda \tag{6}$$

where we set $n = \lceil \log \theta \rceil + 3$, $\theta = \lambda$. Moreover, the number of the multiplications required in our decryption is only $O(\theta \log \theta)$, comparable with $O(\lambda^4)$ in [8].

Efficiency. The arithmetic decryption circuit in NK scheme is not competitive as pointed out by [18], due to the fact that the squashed decryption circuit for NK_Q has a depth polynomial in Q. Fortunately, the degree $108 \cdot \lambda \log^3 \lambda$ of our squashed decryption circuit is independent of Q with a constraint $Q > \lambda$.

We use the leveled FHE scheme over the integer proposed by Coron, Lepoint, and Tibouchi, denoted by $\mathbf{CLT}_2$, and extend its message space to $\mathbb{Z}_Q$, denoted by $\mathbf{CLT}_Q$. To state the efficiency of $\mathbf{CLT}_Q$ with our bootstrapping procedure, we compare it with the scheme **Convert-CLT**$_2$ converting the mod-Q arithmetic circuit to binary and evaluating all the operation using the scheme $\mathbf{CLT}_2$ with binary message space. Here we compare in terms of the ciphertext size and the time complexity of basic operation implemented during homomorphic evaluation.

Then ciphertext size γ_Q of $\mathbf{CLT}_Q$ is a little shorter than that of **Convert-CLT**$_2$, specifically

$$\frac{\log Q \cdot \gamma_2}{\gamma_Q} \sim \frac{\log Q \cdot \left(2\lambda + 2\log\lambda + (\log\lambda + 2)(2\log\lambda + 9)\right)^2}{\left(2\lambda + 2\log\lambda + (\log(\lambda \cdot \log^4 \lambda) + 9)(2\log\lambda + \log Q + 8)\right)^2}. \tag{7}$$

And for some $Q \in [256, 2^{81}]$, we have $1 < \log Q \cdot \gamma_2 / \gamma_Q < 1.4$ when $\lambda = 64$. The ciphertexts for $\mathbf{CLT}_Q$ and **Convert-CLT**$_2$ are of the same size.

Moreover, we denote by T_Q the time complexity of a single ciphertext refresh operation in $\mathbf{CLT}_Q$ and by T_2' the time complexity of carrying out a multiplication mod-Q in **Convert-CLT**$_2$ (by homomorphically evaluating the Boolean circuit for modular multiplication, with a refresh operation after each *AND* gate). Then we show that

$$\frac{T_2'}{T_Q} = \frac{17 \log Q \cdot \lambda}{\log \lambda} \cdot \frac{\log Q \cdot \gamma_2}{\gamma_Q} \sim \frac{17 \log Q \cdot \lambda}{\log \lambda}. \tag{8}$$

For instance, T_Q is faster than T_2' by a factor of more than 930, when $Q \ge \lambda$.

Then, we say that a pure FHE scheme with large message space with our bootstrapping procedure is preferable.

1.3. The Organization. We summarize some notations and tricks in Section 2. In Section 3, we express the decryption circuit as a mod-Q arithmetic circuit of a low enough multiplicative degree. In Section 4, we present an FHE scheme over the integers with bootstrapping for the large prime message space and show its efficiency compared to the FHE scheme with binary message space. Finally, conclusion is given in Section 5.

2. Preliminaries

2.1. Notations. For a real number z, we denote by $\lceil z \rceil, \lfloor z \rfloor, \lfloor z \rceil$ the rounding of a up, down, or the nearest integer. For integers m, n, we denote the integer sets $\{m, m+1, \ldots, n-1, n\}$ and $\{m, m+1, \ldots, n-1\}$ by $[m, n]$, and $[m, n)$, respectively. For a real number r, we use $r = (\ldots, r_1, r_0 . r_{-1}, r_{-2}, \ldots, r_{-n})_Q$ to denote the Q-ary representation of r with n bits of precision

after the Q-ary point. When $Q = 2$, it denotes the binary representation of r. Given $x, p \in \mathbb{R}$, we let $[x]_p$ denote the unique number in $(-p/2, p/2]$ that is congruent to x mod p. All logarithms in the text are base-2 unless stated otherwise.

For a positive integer n, $x \in \mathbb{Z} \cap [0, 2^n)$ and $y \in \mathbb{R}$, define $\text{BitDecomp}_n(x) = (x_0, x_1, \ldots, x_{n-1})_2 \in \{0,1\}^n$ with $x = \sum_{i=0}^{n-1} x_i 2^i$, and $\text{PowersofTwo}_n(y) = (y, 2y, \ldots, 2^{n-1}y) \in \mathbb{R}^n$; then we have

$$\begin{aligned}\left\langle \text{BitDecomp}_n(x), \text{PowersofTwo}_n(y) \right\rangle &= \sum_{i=0}^{n-1} x_i \left(2^i y\right) \\ &= xy.\end{aligned} \tag{9}$$

2.2. Three-for-Two Trick over $\mathbb{Z}_Q$. Three-for-two trick is used to transform three numbers of arbitrary bit length into two numbers that are at most 1 bit longer, such that the sum of the two output numbers is the same as the sum of the three input numbers. And three-for-two trick over $\mathbb{Z}_Q$ has been mentioned in [17]. For $a, b, c \in \{0,1\}^*$, let $a = (\ldots, a_1, a_0)_2$, $b = (\ldots, b_1, b_0)_2$, and $c = (\ldots, c_1, c_0)_2$; then

$$\begin{aligned}a + b + c &= XOR(a,b,c) + 2CARRY(a,b,c) \bmod Q \\ &= (\ldots, XOR(a_0, b_0, c_0)) \\ &\quad + (\ldots, CARRY(a_0, b_0, c_0), 0) \bmod Q,\end{aligned} \tag{10}$$

where

$$\begin{aligned}XOR(a_i, b_i, c_i) &= 4a_ib_ic_i - 2(a_ib_i + a_ic_i + b_ic_i) + a_i \\ &\quad + b_i + c_i \bmod Q,\end{aligned} \tag{11}$$

$$CARRY(a_i, b_i, c_i) = a_ib_i + a_ic_i + b_ic_i - 2a_ib_ic_i \bmod Q.$$

while for $Q = 2$, the bit operation $XOR(a_i, b_i, c_i) = a_i \oplus b_i \oplus c_i$, and $CARRY(a_i, b_i, c_i) = a_ib_i \oplus a_ic_i \oplus b_ic_i$.

2.3. Lagrange Interpolating Polynomial. The Lagrange interpolating polynomial is the polynomial $f(x)$ of degree $n-1$ that passes through the n points $\{(x_0, y_0 = f(x_0)), \ldots, (x_{n-1}, y_{n-1} = f(x_{n-1}))\}$, and given by $f(x) = \sum_{j=0}^{n-1} f_j(x)$, where

$$f_j(x) = y_j \prod_{0 \le k \le n-1, k \ne j} \left(\frac{x - x_k}{x_j - x_k}\right). \tag{12}$$

Our goal of introducing the Lagrange interpolation polynomial is to obtain the mod-Q arithmetic polynomial expression of computing any bit in the binary representation of the integer $x \in [0, \theta]$. For every integer $b \in [0, \theta]$, let $b = (b_{n-1}, \ldots, b_0)_2$, where $n = \lceil \log \theta \rceil$. For each index $t \in [0, n-1]$, we construct a set consisting of integer b and its t-th bit b_t, where $b = 0, 1, \ldots, \theta$, namely, denote the set as $\{(b, b_t)\}_{b \in [0,\theta]}$ for each t. So for each index $t \in [0, n-1]$, the $\theta + 1$ points set is

$$\begin{aligned}\{(x_b = b, y_b = b_t)\}_{b \in [0,\theta]} &= \{(x_0 = 0, y_0 = 0_t), \\ &\quad (x_1 = 1, y_1 = 1_t), \ldots, (x_\theta = \theta, y_\theta = \theta_t)\}.\end{aligned} \tag{13}$$

If the variable x equates to an integer $b \in [0, \theta]$, for the index $t \in [0, \ldots, n-1]$, the output of the Lagrange interpolating polynomial

$$F_t(x) = \sum_{j=0}^{\theta} y_j \prod_{0 \le k \le \theta, k \ne j} \left(\frac{x - x_k}{x_j - x_k}\right) \bmod Q \tag{14}$$

is y_b, which equates to the t-th bit in the binary representation of x. The multiplicative degree of the mod-Q arithmetic circuit is θ.

3. Bootstrapping the Decryption

This section deals mainly with how to implement the decryption $m \longleftarrow (c \bmod p) \bmod Q$ with a mod-Q arithmetic circuit of a low degree.

3.1. Squashing the Decryption with SSSP Assumption. The decryption circuit is

$$m \longleftarrow (c \bmod p) \bmod Q = c - \left\lfloor \frac{c}{p} \right\rceil \bmod Q. \tag{15}$$

Let $\mathbf{y}$ be a vector of Θ rational number in $[0, Q)$ with κ bits of precision after the binary point, and let $\mathbf{sk} = (s_1, s_2, \ldots, s_\Theta)_2$ be the secret key vector of Θ bits with Hamming weight θ such that $1/p = \langle \mathbf{sk}, \mathbf{y} \rangle + \varepsilon \bmod Q$, where $|\varepsilon| < 2^{-\kappa}$. We firstly compute $z_i = [c \cdot y_i]$, keeping only $n = \lceil \log \theta \rceil + 3$ bits of precision after the binary point for $i = 1, 2, \ldots, \Theta$. So $[c \cdot y_i] = z_i - \Delta_i$ for some Δ_i with $|\Delta_i| < 1/16\theta$. We have

$$\begin{aligned}\frac{c}{p} - \sum_{i=1}^{\Theta} s_i z_i &= \frac{c}{p} - \sum_{i=1}^{\Theta} s_i [c \cdot y_i] + \sum_{i=1}^{\Theta} s_i \Delta_i \bmod Q \\ &= \frac{c}{p} - c \cdot \left(\frac{1}{p} - \varepsilon\right) + \sum_{i=1}^{\Theta} s_i \Delta_i \bmod Q \\ &= c \cdot \varepsilon + \sum_{i=1}^{\Theta} s_i \Delta_i \bmod Q.\end{aligned} \tag{16}$$

We set the bit length of ciphertext c is $\gamma < \kappa - 4$; thus, $|c \cdot \varepsilon| < 1/16$. And we observe that $|\sum_{i=1}^{\Theta} s_i \Delta_i| \le \theta/16\theta$. Since c is a valid ciphertext, satisfying that the value of c/p is within $1/8$ of an integer as the definition in [3]; thus, $\sum_{i=1}^{\Theta} s_i z_i$ is within $1/4$ of an integer. Therefore, we have

$$m \longleftarrow (c \bmod p) \bmod Q = c - \left\lfloor \sum_{i=1}^{\Theta} s_i z_i \right\rceil \bmod Q. \tag{17}$$

For $i \in [1, \Theta]$, let $z_i = (\mathbf{z}_i'.\mathbf{z}_i'')_2$, where $\mathbf{z}_i' = (z_{i,\gamma \log Q-1}', \ldots, z_{i,0}')_2$ is the integer part of z_i and $\mathbf{z}_i'' = (z_{i,-1}'', \ldots, z_{i,-n}'')_2$ is the fractional part. Then we have

$$m = c - \sum_{i=1}^{\Theta} \left(s_i \mathbf{z}_i' \bmod Q\right) - \left\lfloor 2^{-n} \sum_{i=1}^{\Theta} s_i \mathbf{z}_i'' \right\rceil \bmod Q. \tag{18}$$

where $2^{-n} \sum_{i=1}^{\Theta} s_i \mathbf{z}_i''$ is within $1/4$ of an integer. (Note that most of the context above in this subsection has been described by van Dijk et al. in [3], which is the procedure of squashing the decryption circuit for the case of $Q = 2$.)

3.2. Bootstrapping. For the integer part, we need to compute $s_i\mathbf{z}'_i \bmod Q$. We can firstly reduce $\mathbf{z}'_i$ with the modulo Q and sum up for all i, namely,

$$\sum_{i=1}^{\Theta}\left(s_i\mathbf{z}'_i \bmod Q\right) = \sum_{i=1}^{\Theta} s_i\left(\mathbf{z}'_i \bmod Q\right) \bmod Q. \tag{19}$$

It only takes Θ multiplication-by-constant gates and Θ mod-Q addition gates.

For the factional part, in order to compute $\lfloor 2^{-n}\sum_{i=1}^{\Theta} s_i\mathbf{z}''_i\rceil$, here we firstly construct a mod-Q circuit that outputs each bit in the binary representation of the sum $\sum_{i=1}^{\Theta} s_i\mathbf{z}''_i = \sum_{i=1}^{\Theta}(s_iz''_{i,-1},\ldots,s_iz''_{i,-n})_2$ in the following step (1).

(1) Generate n integer numbers $\{\mathbf{W}_{-j}\}_{j\in[1,n]}$ such that $\mathbf{W}_{-j} = \sum_{i=1}^{\Theta} s_iz''_{i,-j}$, namely, $\mathbf{W}_{-j}$ is the Hamming weight of the vector $\boldsymbol{\sigma}^{(-j)} = (s_1z''_{1,-j},\ldots,s_\Theta z''_{\Theta,-j})$. Since the Hamming weight of the vector $\mathbf{sk}$ is θ, then $\mathbf{W}_{-j}$ is not bigger than θ, i.e., $\mathbf{W}_{-j} \le \theta$. Firstly, compute the sums $\mathbf{W}_{-j}$ by directly using mod-Q addition gates, this works since $Q > \theta$. Let $\mathbf{W}_{-j} = (w_{-j,n-1},\ldots,w_{-j,0})_2$. Then convert the small integer $\mathbf{W}_{-j}$ into their bit representation by applying the *Lagrange interpolating polynomial* introduced in Section 2.3; namely, for $0 \le t \le n-1$, $1 \le j \le n$, we have $w_{-j,t} = F_t(\mathbf{W}_{-j})$, where the multiplicative degree is θ.

(2) Now $\sum_{i=1}^{\Theta} s_i\mathbf{z}''_i = \sum_{j=1}^{n} 2^{n-j}\mathbf{W}_{-j}$, which is the sum of n $2n$-bit length of numbers. We can compute it by applying the three-for-two trick over $\mathbb{Z}_Q$ mentioned in Section 2.2 repeatedly, resulting in two numbers $\mathbf{t}_1$ and $\mathbf{t}_2$ satisfying $\mathbf{t}_1+\mathbf{t}_2 = \sum_{j=1}^{n} 2^{n-j}\mathbf{W}_{-j}$. Since we need to apply this trick $\lceil\log_{3/2}n\rceil+2$ times, the bit length of $\mathbf{t}_1$ and $\mathbf{t}_2$ becomes $2n+\lceil\log_{3/2}n\rceil+2$.

(3) Let $\mathbf{t}_1 = 2^n\mathbf{t}'_1+\mathbf{t}''_1$, $\mathbf{t}_2 = 2^n\mathbf{t}'_2+\mathbf{t}''_2$, then

$$\begin{aligned}\left\lfloor 2^{-n}\sum_{i=1}^{\Theta} s_i\mathbf{z}''_i\right\rfloor &= \left\lfloor 2^{-n}\left(\mathbf{t}_1+\mathbf{t}_2\right)\right\rceil \\ &= \left(\mathbf{t}'_1+\mathbf{t}'_2\right)+\left\lfloor 2^{-n}\left(\mathbf{t}''_1+\mathbf{t}''_2\right)\right\rceil.\end{aligned} \tag{20}$$

To evaluate $\lfloor 2^{-n}(\mathbf{t}''_1+\mathbf{t}''_2)\rceil$, let

$$\begin{aligned}2^{-n}\mathbf{t}''_1 &= \left(0.a_{-1},\ldots,a_{-n}\right)_2,\\ 2^{-n}\mathbf{t}''_2 &= \left(0.b_{-1},\ldots,b_{-n}\right)_2.\end{aligned} \tag{21}$$

Let $\{c_0,c_{-1},\ldots,c_{-n+1},c_{-n} = 0\}$ be all the carry bits generated in the addition procedure, where $c_{-i+1} = MAJ(a_{-i},b_{-i},c_{-i}) = (a_{-i}\oplus b_{-i})(a_{-i}\oplus c_{-i})\oplus a_{-i}$. Thus, we have

$$\begin{aligned}&2^{-n}\left(\boldsymbol{t}''_1+\boldsymbol{t}''_2\right)\\ &\quad= \left(c_0.a_{-1}\oplus b_{-1}\oplus c_{-1},\ldots,a_{-n}\oplus b_{-n}\oplus 0\right)_2.\end{aligned} \tag{22}$$

Since $2^{-n}\sum_{i=1}^{\Theta} s_i\mathbf{z}''_i$ is within $1/4$ of some integer mentioned in Section 3.1, we have $a_{-1}\oplus b_{-1}\oplus c_{-1} = 0$, $a_{-2}\oplus b_{-2}\oplus c_{-2} = 0$; thus

$$\begin{aligned}c_0 = {}& a_{-1}a_{-2}b_{-2}\oplus a_{-2}b_{-1}b_{-2}\oplus a_{-1}a_{-2}\oplus a_{-1}b_{-1}\oplus a_{-1}b_{-2}\\ &\oplus a_{-2}b_{-1}\oplus b_{-1}b_{-2}.\end{aligned} \tag{23}$$

Using mod-Q gates to compute those bit operations,

$$\begin{aligned}c_0 = {}& a_{-1}\times a_{-2}+a_{-1}\times b_{-1}+a_{-2}\times b_{-1}+b_{-1}\times b_{-2}-2\\ &\times a_{-1}\times a_{-2}\times b_{-1}-a_{-1}\times a_{-2}\times b_{-2}-a_{-2}\times b_{-1}\\ &\times b_{-2}-2\times a_{-1}\times b_{-1}\times b_{-2}+2\times a_{-1}\times a_{-2}\times b_{-1}\\ &\times b_{-2} \mod Q,\end{aligned} \tag{24}$$

which is a polynomial of degree 4. For integer part, to implement $\mathbf{t}'_1+\mathbf{t}'_2$, we can compute $\mathbf{t}'_1 \bmod Q$ and $\mathbf{t}'_2 \bmod Q$ with the stored numbers $a_j \longleftarrow 2^j \bmod Q$ for $j = 1,2,\ldots,n+\lceil\log_{3/2}n\rceil+2$. Since for an integer $b = (b_k,\ldots,b_1,b_0)_2$,

$$\begin{aligned}b \bmod Q &= b_0+2b_1+\ldots+2^kb_k \bmod Q\\ &= a_0b_0+a_1b_1+\ldots+a_{(k \bmod Q)}b_k \bmod Q.\end{aligned} \tag{25}$$

The modified decryption works as

$$\begin{aligned}m = {}& c-\sum_{i=1}^{\Theta}\left(s_i\mathbf{z}'_i \bmod Q\right)\\ &-\left(\left(\mathbf{t}'_1 \bmod Q\right)+\left(\mathbf{t}'_2 \bmod Q\right)+c_0\right) \bmod Q.\end{aligned} \tag{26}$$

We conclude that the degree of the polynomial in the first step is θ, the degree of the polynomial in the second step is at most $3^{\lceil\log_{3/2}n\rceil+2} < 27\cdot\log^{2.71}\theta$, and the degree of the polynomial in the third step is 4. Therefore, the total degree of the decryption circuit over $\mathbb{Z}_Q$ is bounded by $\theta\cdot(27\cdot\log^{2.71}\theta)\cdot 4 < 108\cdot\theta\log^3\theta$. Since we set $\theta = \lambda$ for security, the degree is at most $108\cdot\lambda\log^3\lambda$. So the multiplicative degree of the decryption circuit is $O(\lambda\log^3\lambda)$ for any prime Q with the constraint $Q > \lambda$.

Remark 1. In [4], the authors set $\theta = \lambda/\log\lambda$ ($\theta = 15$ when $\lambda = 72$). It means that we can express the decryption circuit of FHE scheme over the integers as a low-degree polynomial over $\mathbb{Z}_Q$ for any $Q \ge 16$. The multiplicative degree of decryption circuit in [8] is $O(\lambda)$ for the case that Q is a constant prime, and $8\Theta^2$ in [16] for the case $Q > 8\Theta^2$. If Q is bigger than 15, our degree of decryption circuit is smaller than that of [8]. See Table 1.

Moreover, we reduce the number of multiplications in the decryption circuit which is better than almost previous works as shown in Table 2. Here we have to emphasize that we do not count the number of multiplication-by-constant gates in the decryption circuit to the number of multiplications.

Table 1: Multiplicative degree of decryption circuit.

	$Q = 2$	Q is a constant prime	$Q > \lambda$
[3]	$64\lambda \log^2 \lambda$	-	-
[4]	2λ	-	-
[8]	16λ	$Q^4\lambda$	-
[15]	-	$O(\lambda^{1+\epsilon})$	-
[16]	-	-	$8\Theta^2$
our result	-	-	$108\lambda \log^3 \lambda$

Table 2: The number of multiplications in the decryption circuit.

	the number of multiplications
[3]	$O(\theta\Theta)$
[4]	$O(\theta^2)$
[8]	$O(\Theta \log^2 \theta)$
[15]	$O(\log^2 \theta)$
[16]	$\Theta^2 + \Theta + 1$
our result	$O(\theta \log \theta)$

Proposition 2. *The number of multiplications in our squashed decryption circuit is at most* $O(\theta \cdot \log \theta)$.

Proof. For the integer part, we use Θ multiplication-by-constant gates and Θ mod-Q addition gates.

For the factional part, in step (1), we apply the *Lagrange interpolating polynomial* in n variate, which is a polynomial of degree θ. For a variate x, first we compute $y_1 = x, y_2 = x^2, \ldots, y_\theta = x^\theta$ which requires $\theta - 1$ multiplications. So a *Lagrange interpolating polynomial* consists of $\theta - 1$ multiplications, $\theta - 1$ multiplication-by-constant gates, and θ additions gates. Then we need the $n\theta$ multiplications in step (1).

The 3-for-2 trick over $\mathbb{Z}_Q$ for the bits a, b, c needs 4 multiplications gates. Step (2) needs to sum up the n $2n$-bit length of numbers, and for the first time applying this trick, it takes $4 \cdot 2n \cdot (n/3)$ multiplications gates, the second time needs to sum up about $2n/3$ $2n + 1$-bit of numbers, and it takes $4 \cdot (2n + 1) \cdot ((2n/3)/3)$ multiplications gates, and so on. Then it takes about

$$\frac{8n^2/3 \cdot \left(1 - (2/3)^{\lceil \log_{3/2} n \rceil + 3}\right)}{1 - 2/3} + \frac{(4n/3) \cdot \left(1 - (2/3)^{\lceil \log_{3/2} n \rceil + 2}\right)}{1 - 2/3} < 8n^2 + 4n \tag{27}$$

multiplication gates in step (2).

In step (3), we need 9 multiplications to compute c_0.

So the number of multiplications in our squashed circuit is

$$\theta \cdot n + 8n^2 + 4n + 9 = O\left(\theta \cdot \log \theta\right). \tag{28}$$

□

3.3. Removing the Constraint $Q > \lambda$. The constraint $Q > \lambda$ is required because we want to compute the bits of the Hamming weight by directly summing up without regard to the carry bits generated from the addition. We observe that the optimization of the binary decryption circuit proposed in [4] does not counter the Hamming weight, so we can remove the constraint. With the three-for-two trick over $\mathbb{Z}_Q$, we can transform the binary decryption circuit to mod-Q arithmetic circuit, resulting in more complexity of the decryption circuit.

More precisely, we can divide the secret key $\mathbf{sk}$ into θ boxes Box_i of $B = \Theta/\theta$ bits each, such that each box has single 1-bit in it. Then

$$\begin{aligned} m &= c - \left\lfloor \sum_{i=1}^{\Theta} s_i z_i \right\rceil \bmod Q \\ &= c - \left\lfloor \sum_{i=1}^{\theta} \sum_{s_j \in Box_i} z_j \right\rceil \bmod Q. \end{aligned} \tag{29}$$

Let $q_i = \sum_{s_j \in Box_i} z_j$ be obtained by adding B numbers, with only one being nonzero. So

$$q_i = \sum_{s_j \in Box_i} z_j \bmod Q = \sum_{s_j \in Box_i} z_j. \tag{30}$$

It only requires the mod-Q addition gates. Then applying three-for-two trick over $\mathbb{Z}_Q$ to add up the θ numbers and using the rounding computations in step (3) in last subsection, the decryption circuit is implemented by a polynomial of multiplicative degree $3^{\lceil \log_{3/2} \theta \rceil + 2} \cdot 4 < 108\theta^3$, i.e., $O(\lambda^3)$.

The goal we describe in this subsection is to emphasize that our work in the last subsection reduces the multiplicative degree of decryption circuit from $O(\lambda^3)$ to $O(\lambda \log^3 \lambda)$.

4. FHE Scheme CLT_Q with Our Bootstrapping Procedure

To show the usefulness of our squashed decryption circuit, we present a variant of FHE scheme over the integers with our bootstrapping procedure and then compare it with the original scheme in binary setting. By Gentry's bootstrapping theory, we can get the "pure" FHE scheme transformed from the somewhat FHE scheme or the leveled FHE scheme. Here we only describe the latter scheme, since in the former situation the FHE scheme in mod-Q setting is not perforable to the binary setting.

Here we describe an FHE scheme over the integers with bootstrapping for large prime message space just like Cheon, Han, and Kim did in [15], which is a variant of the FHE scheme presented by Coron, Lepoint, and Tibouchi in [7].

Let ρ be a bound on the bit length of the noise, η the bit length of the original secret key, and γ the bit length of the ciphertext. The parameter τ refers to the number of encryptions of zero contained in the public key for encryption, Θ the size of the secret vector, θ the Hamming weight of the vector, and κ the bit length of the rational numbers in the public key.

These parameters must satisfy the following constraints.

(i) $\rho \geq \lambda$, to protect against the brute force attacks on the noise.

(ii) $\eta \geq \rho + O(L(\log\lambda + \log Q))$, where L is the depth of multiplication of the circuits to be evaluated.

(iii) $\gamma > \omega(\eta^2 \cdot \log\lambda)$, to avoid lattice-based attacks [3, 4].

(iv) $\tau = \gamma + 2\lambda + 2$, in order to use the leftover hash lemma in the security proof.

(v) $\Theta^2 \geq \gamma\omega(\log\lambda)$, to avoid known attacks on the sparse subset sum problem [4, 5].

(vi) $\binom{\Theta}{\theta/2} \geq 2^\lambda$ to avoid an attack on the sparse subset sum problem [19].

(vii) $\gamma < \kappa - \log Q - 4$ is required in Section 3.1.

4.1. FHE Scheme with Message Space $\mathbb{Z}_Q$. In this subsection, we describe an FHE scheme over the integers for message space $\mathbb{Z}_Q$ for a prime modulus Q, where Q can be any prime bigger than the security parameter λ, denoted by $\mathbf{CLT}_Q$.

For an η-bit odd integer p, and an integer q_0 in $[0, 2^\gamma/p^2)$, we define the set

$$\mathscr{D}^{\rho}_{p,q_0} = \left\{p^2 \cdot q + r : Chooseq \longleftarrow [0, q_0), r \longleftarrow (-2^\rho, 2^\rho)\right\}. \tag{31}$$

$\mathbf{CLT}_Q$.**KeyGen**(1^λ). Generate a η-bit prime p and a γ-bit integer $x_0 = p^2 \cdot q_0 + r_0$ with $r_0 \longleftarrow (-2^\rho, 2^\rho)$ and $q_0 \longleftarrow [0, 2^\gamma/p^2)$.

(1) Generate the public key for encryption. For $1 \leq i \leq \tau$, sample $x_i \longleftarrow \mathscr{D}^{\rho}_{p,q_0}$, $y' \longleftarrow \mathscr{D}^{\rho}_{p,q_0}$ and $y = y' + \lfloor p/Q \rfloor$.

(2) Generate the public key for multiplication. Let $\mathbf{v}'$ be a vector of Θ numbers with κ bit of precision following the binary point, denoted by $\mathbf{v}' = (v'_1, \ldots, v'_\Theta)$. Choose uniformly a Θ-bit vector $\mathbf{s}' = (s'_1, s'_2, \ldots, s'_\Theta) \in \{0,1\}^\Theta$ at random such that

$$\frac{(Q \cdot 2^\eta)}{p^2} = \left\langle \mathbf{s}', \mathbf{v}' \right\rangle + \epsilon \bmod Q2^\eta \tag{32}$$

with $|\epsilon| < 2^{-\kappa}$. Then, define

$$\boldsymbol{\alpha} = \mathbf{q} \cdot p^2 + \mathbf{r} + \left\lfloor PowersofTwo\left(\mathbf{s}'\right) \cdot \frac{p}{2^{\eta+1}} \right\rceil, \tag{33}$$

where the components of $\mathbf{q}$ are randomly chosen from $[0, q_0) \cap \mathbb{Z}$ and those of $\mathbf{r}$ from $(-2^\rho, 2^\rho) \cap \mathbb{Z}$.

(3) Generate the public key for bootstrapping. Choose uniformly a Θ-bit vector $\mathbf{s} = (s_1, s_2, \ldots, s_\Theta) \in \{0,1\}^\Theta$ at random, with Hamming weight θ.

Choose a random integer $v_i \in [0, Q^{\kappa+1})$ such that $\sum_{i=1}^{\Theta} s_i \cdot v_i = \lfloor Q^{\kappa+1}/p \rceil$ and set $Z = \lfloor 2^\kappa (p \bmod Q)/p \rceil = \lfloor 2^\kappa/p \rceil$.

For $0 \leq i \leq \Theta$, choose $u_i \longleftarrow [0, 2^{\log Q + \kappa + 1}) \cap \mathbb{Z}$ in such a way that $\sum_{i=1}^{\Theta} s_i u_i = Z \bmod 2^{\kappa+1}$. Set $y_i = u_i/2^\kappa$, and denote it by the vector $\mathbf{y} = (y_1, y_2, \ldots, y_\Theta)$. Then $\sum_{i=1}^{\Theta} s_i y_i = 1/p - \varepsilon_p$ for some $|\varepsilon_p| < 2^{-\kappa}$.

For $n = \lceil \log\theta \rceil + 3$, generate the vector $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_\Theta)$:

$$\beta_i = q_i \cdot p^2 + \left\lfloor \frac{p}{Q} \right\rfloor \cdot s_i + r_i, \tag{34}$$

where $q_i \in [0, q_0)$ and $r_i \in (-2^\rho, 2^\rho)$ for $1 \leq i \leq \Theta$.

Output the secret key $sk = (p, s_1, s_2, \ldots, s_\Theta)$, and the public key $pk = \{x_0, y, x_1, \ldots, x_\tau, \mathbf{v}', \boldsymbol{\alpha}, \mathbf{y}, \boldsymbol{\beta}\}$.

$\mathbf{CLT}_Q$.**Enc**(pk, m). Given a message $m \in \mathbb{Z}_Q$, uniformly sample a subset $S \subseteq \{1, 2, \ldots, \tau\}$, and output

$$c = \left[\sum_{i \in S} x_i + y \cdot m\right]_{x_0} \tag{35}$$

$\mathbf{CLT}_Q$.**Dec**(sk, c). Given a ciphertext c, output $m = [\lfloor Qc/p \rceil]_Q$.

$\mathbf{CLT}_Q$.**Add**(pk, c_1, c_2). Given two ciphertexts c_1, c_2, output $c_{add} = [c_1 + c_2]_{x_0}$.

Convert(pk, c). Given a ciphertext c, output $c' = 2\langle \boldsymbol{\alpha}, \mathrm{BitDecomp}_\eta(\mathbf{c}) \rangle$, where $\mathbf{c} = (\lfloor c \cdot v'_i \rceil \bmod 2^\eta)_{1 \leq i \leq \Theta}$.

$\mathbf{CLT}_Q$.**Mult**(pk, c_1, c_2). Given two ciphertexts c_1, c_2, output $c_{mult} = [\mathbf{Convert}(pk, c_1 \cdot c_2)]_{x_0}$.

$\mathbf{CLT}_Q$.**MultCons**(pk, c, a). Given a ciphertext c and a constant $a \in \mathbb{Z}_Q$, output $c_{multcons} = [a \cdot c]_{x_0}$.

$\mathbf{CLT}_Q$.**Recrypt**(pk, c, D_ε). Given a ciphertext c for a message m, for $i = 1, 2, \ldots, \Theta$, compute $z_i = [Qc \cdot y_i]$, keeping only $n = \lceil \log\theta \rceil + 3$ bits of precision after binary point. Denote the binary representation of z_i as $\mathbf{z}_i = (\ldots, z_{i,1}, z_{i,0}.z_{i,-1}, \ldots, z_{i,-n})_2$. And given the decryption D_ε, which is

$$m = \left[\left\lfloor \frac{Qc}{p} \right\rceil\right]_Q = \left[\left\lfloor \sum_{i=1}^{\Theta} s_i \mathbf{z}_i \right\rceil\right]_Q, \tag{36}$$

output a refreshed ciphertext $c' = [\lfloor \sum_{i=1}^{\Theta} \beta_i \mathbf{z}_i \rceil]_{x_0}$. The algorithm homomorphically evaluates the decryption circuit on the encryptions of the secret key bits $(s_1, s_2, \ldots, s_\Theta)$, referring to the bootstrapping process in Section 3 for more details.

4.2. Correctness. In this subjection, we prove the correctness of the homomorphic procedure.

For the scheme $\mathbf{CLT}_Q$, the ciphertext has the form that $c = q \cdot^2 + (Qr^* + m)\lfloor p/Q \rfloor + r$ with two kinds of noise r and r^*. In [7], it is called a ciphertext with noise (ρ, ρ^*) if

$|r| < 2^{\rho}$ and $|r^*| < 2^{\rho^*}$. The authors of [15] proposed a noise growth analysis during the homomorphic addition and multiplication as shown in Lemma 3.

Lemma 3 (Lemma 3 in [15]). *Let c_1 and c_2 be ciphertext with (ρ_1, ρ_1^*) and (ρ_2, ρ_2^*), respectively. Let $\rho = \max(\rho_1, \rho_2)$ and $\rho^* = \max(\rho_1^*, \rho_2^*)$.*

(i) $\mathbf{CLT}_Q.\mathbf{Add}(pk, c_1, c_2)$ *is a ciphertext with noise* $(\rho + 2, \rho^* + 1)$

(ii) $\mathbf{CLT}_Q.\mathbf{Mult}(pk, c_1, c_2)$ *is a ciphertext with noise* $(\rho + \rho^* + \log Q + 8, \log \Theta)$.

Suppose $\rho^* \leq \log \Theta$ during the homomorphic evaluation. By Lemma 3, for the CLT scheme, the noise length in bits has only grown by an additive factor $(\log \Theta + \log Q + 8)$. The noise growth during homomorphic evaluation is linear; then we have the following.

Lemma 4 (**recryption noise**). *Let c be a ciphertext for a message $m \in \mathbb{Z}_Q$; $\mathbf{CLT}_Q.\mathbf{Recrypt}(pk, c, D_\varepsilon)$ is a ciphertext with noise*

$$\left(\rho + \log \Theta + L' \left(\log \lambda + \log Q\right), \log \Theta\right), \tag{37}$$

where the depth of decryption L' is $(2 \cdot \lceil \log_{3/2} n \rceil + 6 + \log \theta)$, less than $(\log(\theta \log^4 \theta) + 8)$.

Proof. The decryption circuit D_ε has been described in Section 3. We get the refreshed ciphertext $c' = [\lfloor \sum_{i=1}^{\Theta} \beta_i \mathbf{z}_i \rfloor]_{x_0}$, where the noise of encryption of the secret key bit β_i is $(\rho, 0)$. We describe the noise increasing with homomorphic evaluations in the bootstrapping procedure. Here we only consider the evaluations for the fraction part in bootstrapping to approximately compute the noise.

In step (1) of Section 3.2, we get the encryption of the Hamming weight of the vector $\boldsymbol{\sigma}^{(-j)}$ with noise $(\rho + \log \Theta, 0)$. The degree of the *Lagrange interpolating polynomial* is θ, which can be implemented by a circuit of depth $\log \theta$. So we obtain the ciphertexts encrypted the bit of the Hamming weight $w_{-j,i}$ with the noise $(\rho + \log \Theta + \log \theta(\log \Theta + \log Q + 8), \log \Theta)$.

In step (2), it needs to apply three-for-two trick $\lceil \log_{3/2} n \rceil + 2$ times. After implementing three-for-two trick one time, we get the ciphertexts with noise $(\rho + \log \Theta + (2 + \log \theta)(\log \Theta + \log Q + 8), \log \Theta)$. Then we get the ciphertexts with noise $(\rho + \log \Theta + (2 \cdot (\lceil \log_{3/2} n \rceil + 2) + \log \theta)(\log \Theta + \log Q + 8), \log \Theta)$ after implementing step (2).

The circuit which computes step (3) has multiplicative degree 4; we get the ciphertexts with noise

$$\begin{aligned} &\left(\rho + \log \Theta + \left(2 \cdot \left(\lceil \log_{3/2} n \rceil + 2\right) + 2 + \log \theta\right)\right. \\ &\quad \left. \cdot \left(\log \Theta + \log Q + 8\right), \log \Theta\right) = \left(\rho + \log \Theta\right. \\ &\quad \left. + \left(2 \lceil \log_{3/2} n \rceil + 6 + \log \theta\right)\right. \\ &\quad \left. \cdot \left(\log \Theta + \log Q + 8\right), \log \Theta\right) \end{aligned} \tag{38}$$

□

Set the depth $L \geq L' + 1$, namely, $L = \log(\theta \cdot \log^4 \theta) + 9$, and $\eta \geq \rho + \log \Theta + L(\log \Theta + \log Q + 8)$; the scheme $\mathbf{CLT}_Q$ is correct and bootstrappable.

According to the conditions for the parameters, one can take $\rho = 2\lambda$, $\eta = \widetilde{\mathcal{O}}(2\lambda + \log \Theta + L(\log \Theta + \log Q + 8)) = \widetilde{\mathcal{O}}(\lambda + L \log Q)$, $\gamma = \widetilde{\mathcal{O}}((2\lambda + \log \Theta + L(\log \Theta + \log Q + 8))^2 \log \lambda) = \widetilde{\mathcal{O}}(\lambda^2 + (L \log Q)^2)$, $\kappa = \gamma\eta/\rho$, $\tau = \gamma + 2\lambda + 2$, and $L = \log(\lambda \cdot \log^4 \lambda) + 9$.

Note that Θ is chosen so that the sparse subset sum problem is hard. We consider that Θ is unaffected by change of Q. We set $\theta = \lambda$, then $\Theta = O(\lambda^2)$ for any Q satisfying all the above conditions.

The following theorem holds by the bootstrapping theorem proposed by Gentry in [2].

Theorem 5. *Our scheme $\mathbf{CLT}_Q$ with the above parameters setting is a pure fully homomorphic encryption.*

4.3. Security. The FHE scheme over the integers is *IND-CPA* secure under the AGCD assumption. Our scheme just extends its message space from $\mathbb{Z}_2$ to $\mathbb{Z}_Q$ and combines a squashing procedure before the bootstrapping. Thus, it is easy to see that the following theorem holds.

Theorem 6. *Under the assumption that both of AGCD and SSSP are hard, our scheme $\mathbf{CLT}_Q$ is IND-CPA secure.*

4.4. Efficiency. As mentioned in [18], to evaluate a mod-Q arithmetic circuit with FHE scheme over the integers, one could either use the FHE scheme with large message space directly or first convert the arithmetic circuit to a Boolean one and then evaluate that converted circuit using an FHE scheme with binary message space.

We denote BAdd_Q and BMult_Q as the Boolean circuits to perform addition and multiplication on two n-bit integers modulus Q as in [18], and we have the following numbers of AND gates for BAdd_Q and BMult_Q.

Proposition 7 (see [18]). *For an n-bit prime Q, $BAdd_Q$ uses $9n$ AND gates, and $BMult_Q$ uses $17n^2$ AND gates.*

We denote **Convert-CLT**$_2$ the FHE scheme obtained from $\mathbf{CLT}_2$ with binary message space using the converting circuit with the BAdd_Q and BMult_Q. Note that, for the scheme $\mathbf{CLT}_2$, the decryption is implemented by a circuit of degree of $2\theta = 2\lambda$ as presented in [4], in which the number of multiplication equals $\theta^2 = \lambda^2$. Just like [18], we compare **Convert-CLT**$_2$ and $\mathbf{CLT}_Q$ in terms of the size of the ciphertexts and the time complexity of basic operations carried out during homomorphic evaluation.

4.4.1. Comparing the Size of the Ciphertexts. The ciphertext size γ_Q of $\mathbf{CLT}_Q$ equals $\widetilde{\mathcal{O}}((\lambda + L(\log \lambda + \log Q))^2 \log \lambda)$ and grows in the $\log Q$ when Q grows. Proposition 8 tells us that the ciphertext size of $\mathbf{CLT}_Q$ is almost the same as that of **Convert-CLT**$_2$ when $\log Q \in [6, 200]$.

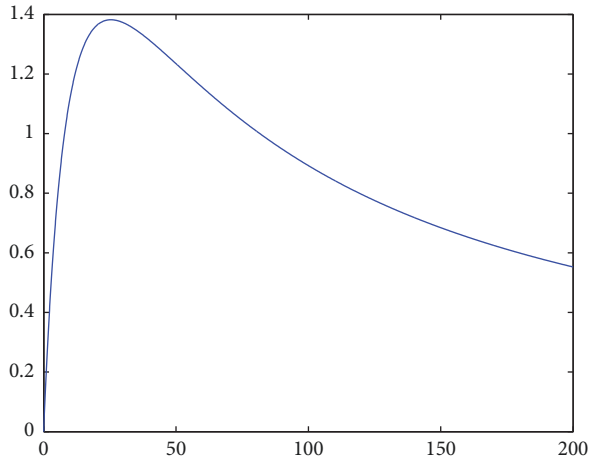

Figure 1: The rate of the ciphertext sizes for $\lambda = 64$.

Proposition 8. *For a given security parameter λ, and any prime Q, let $\gamma_2' = \gamma_2 \cdot \log Q$. Then we have*

$$\frac{\gamma_2'}{\gamma_Q} \sim \frac{\log Q \cdot (2\lambda + 2\log\lambda + (\log\lambda + 2)(2\log\lambda + 9))^2}{(2\lambda + 2\log\lambda + (\log(\lambda \cdot \log^4\lambda) + 9)(2\log\lambda + \log Q + 8))^2}. \tag{39}$$

Proof. We have $\gamma_Q = \widetilde{\mathcal{O}}((2\lambda + \log\Theta + L(\log\Theta + \log Q + 8))^2 \log\lambda)$, where the implied constant does not depend on Q. Since $L_Q = \log(\lambda \cdot \log^4\lambda) + 9$ for odd prime $Q > 2$, and $L_2 = \log\lambda + 2$ for $Q = 2$; thus,

$$\frac{\gamma_2'}{\gamma_Q} \sim \frac{\log Q \cdot \gamma_2}{\gamma_Q} \sim \frac{\log Q \cdot (2\lambda + \log\Theta + L_2(\log\Theta + 9))^2}{(2\lambda + \log\Theta + L_Q(\log\lambda + \log Q + 8))^2} \sim \frac{\log Q \cdot (2\lambda + 2\log\lambda + (\log\lambda + 2)(2\log\lambda + 9))^2}{(2\lambda + 2\log\lambda + (\log(\lambda \cdot \log^4\lambda) + 9)(2\log\lambda + \log Q + 8))^2}. \tag{40}$$

□

In Figure 1, we show that the value of γ_2'/γ_Q as a function of $\log Q$ for the case $\lambda = 64$. It tells us that the ciphertext size of $\mathbf{CLT}_Q$ is a little shorter than **Convert-CLT**$_2$. When $8 \le \log Q \le 81$, namely, $256 \le Q < 2^{81}$, we have $1 < \gamma_2'/\gamma_Q < 1.4$. Roughly speaking, we say that the ciphertexts for the scheme $\mathbf{CLT}_Q$ and **Convert-CLT**$_2$ have almost the same size.

4.4.2. Comparing the Speed of Homomorphic Operations. Now, we would like to compare the speed of homomorphic operations in **Convert-CLT**$_2$ and $\mathbf{CLT}_Q$. That speed is essentially determined by the cost of homomorphic multiplication modulo Q.

Proposition 9. *For a given security parameter λ, and any prime Q, let T_2' be the time complexity of carrying out a multiplication* mod Q *in* **Convert-CLT**$_2$*, and T_Q be the time complexity of a single ciphertext refresh operation in* $\mathbf{CLT}_Q$*. Then we have*

$$\frac{T_2'}{T_Q} \le \frac{17\log Q \cdot \lambda}{\log\lambda} \cdot \frac{\log Q \cdot \gamma_2}{\gamma_Q}. \tag{41}$$

Proof. For any prime Q, let t_Q be the time complexity of ciphertexts multiplication. Then T_Q is t_Q multiplied by the number of multiplications in decryption circuit, and T_2' is T_2 multiplied by the number of AND gates in BMult_Q circuit.

As far as we know, the best time complexity for k-bit multiplication is $k \log k 2^{O(\log^* k)}$, where $\log^* k$ represents the iterated logarithm [20]. In our case, k is γ_Q and

$$\begin{aligned}\log\gamma_Q &= \log\left(\widetilde{\mathcal{O}}\left((2\lambda + \log\Theta + L(\log\Theta + \log Q + 8))^2 \cdot \log\lambda\right)\right) = \log\log\lambda + 2\log(2\lambda + \log\Theta + L(\log\Theta + \log Q + 8)) + \text{constant} \\ &< 2\log(\lambda + \log Q\log\lambda) + \text{constant},\end{aligned} \tag{42}$$

while $\log\gamma_2 > 2\log(\lambda + \log^2\lambda) + \text{constant}$.

Then for $Q = \mathrm{poly}(\lambda)$, even for $Q = \exp(\lambda)$, the effect of $\log Q$ is dominated by that of λ, so we can estimate t_2/t_Q by γ_2/γ_Q, since we can ignore effect of Q in the part $\log\gamma_Q 2^{O(\log^*\gamma_Q)}$.

By Proposition 2, we have that the number of multiplications of the decryption for $\mathbf{CLT}_Q$ scheme is about $\theta\log\theta$, while for $\mathbf{CLT}_2$, it is about θ^2. On the other hand, we need $17\log^2 Q$ AND gates for the Boolean circuit BMult_Q as in Proposition 7. Then we have

$$\frac{T_2'}{T_Q} \le \frac{17\log^2 Q \cdot \theta^2}{\theta\log\theta} \cdot \frac{t_2}{t_Q} \sim \frac{17\log^2 Q \cdot \theta}{\log\theta} \cdot \frac{\gamma_2}{\gamma_Q} \sim \frac{17\log Q \cdot \theta}{\log\theta} \cdot \frac{\log Q \cdot \gamma_2}{\gamma_Q} \tag{43}$$

□

In Figure 2, we show the value of T_2'/T_Q as a function of $\log Q$ for the case $\lambda = 64$, which tells us that homomorphic multiplication for $\mathbf{CLT}_Q$ increase performance as Q grows.

Remark 10. For $\lambda = 64$, the value of γ_2'/γ_Q climbs up and then declines as $\log Q$ grows as shown in Figure 1. If $Q = 64, Q = 256$, it is 0.8548 and 1.0007, respectively. The value of T_2'/T_Q becomes much large as $\log Q$ grows and has an upper bound close to 25447 for $\lambda = 64$ as shown in Figure 2. Table 3 shows the efficiency of $\mathbf{CLT}_2$ measured against $\mathbf{CLT}_Q$ for some primes Q.

5. Conclusion

We propose an FHE scheme over the integers with message space $\mathbb{Z}_Q$ for any prime $Q > \theta$. If we set $\theta = \lambda$, the decryption circuit of this scheme is expressed as a polynomial

Table 3: The value of γ_2'/γ_Q and T_2'/T_Q.

λ	$\log Q$	γ_2'/γ_Q	T_2'/T_Q
64	$\log Q = 6$	0.8548	930.0392
64	$\log Q = 7$	0.9366	1186.9155
64	$\log Q = 8$	1.0073	1461.2334
64	$\log Q = 32$	1.3637	7983.0041
64	$\log Q = 81$	1.004	14751.7915
64	$\log Q = 82$	0.9963	14838.0201

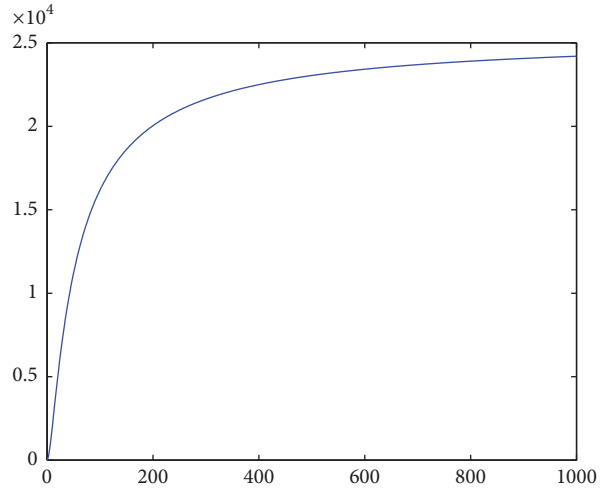

Figure 2: The rate of the time complexity for $\lambda = 64$.

of multiplicative degree $108 \cdot \theta \log^3 \theta = O(\lambda \log^3 \lambda)$, which is independent of the modulus Q except the constraint $Q > \lambda$. And we also reduce the number of multiplications in the decryption circuit which is better than most of previous work.

To explore our squashed decryption circuit is worthwhile for large values of Q, we present a variant of leveled FHE scheme **CLT** that supports arbitrarily homomorphic operations in the message space $\mathbb{Z}_Q$ for $Q > \lambda$. By comparing the two schemes $\mathbf{CLT}_Q$ and $\mathbf{Convert\text{-}CLT}_2$, we have seen that the two schemes have almost the same ciphertext size, but the former is significantly preferable.

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the National Key R&D Program of China under Grant no. 2017YFB0802000, National Natural Science Foundation of China (61472309, 61572390, and 61672412), National Cryptography Development Fund under Grant MMJJ20170104, and the Foundation of Jiangsu Normal University (16XLR031).

References

[1] 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.

[2] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *STOC*, M. Mitzenmacher, Ed., pp. 169–178, ACM, 2009.

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

[4] J.-S. Coron, A. Mandal, D. Naccache et al., "Fully homomorphic encryption over the integers with shorter public keys," IACR Cryptology ePringt Archive 2011:441, 2011.

[5] J.-S. Coron, D. Naccache, and M. Tibouchi, "Public key compression and modulus switching for fully homomorphic encryption over the integers," IACR Cryptology ePringt Archive 2011:440, 2011.

[6] J.-S. Coron, T. Lepoint, M. Tibouchi, J. H. Cheon, J. Kim et al., "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.

[7] J.-S. Coron, T. Lepoint, and M. Tibouchi, "Scale-invariant fully homomorphic encryption over the integers," in *Proceedings of the International Workshop on Public Key Cryptography*, pp. 311–328, Springer, Berlin, Germany, 2014.

[8] K. Nuida and K. Kurosawa, "(Batch) fully homomorphic encryption over integers for non-binary message spaces," IACR Cryptology ePringt Archive 2014:777, 2014.

[9] J. H. Cheon and D. Stehlé, "Fully homomophic encryption over the integers revisited," in *Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques*, vol. 9056, pp. 513–536, Springer, Berlin, Germany, 2015.

[10] J. H. Cheon, J. Kim, M. S. Lee, and A. Yun, "CRT-based fully homomorphic encryption over the integers," *Information Sciences*, vol. 310, pp. 149–162, 2015.

[11] D. Benarroch, Z. Brakerski, and T. Lepoint, "FHE over the integers: decomposed and batched in the post-quantum regime," IACR Cryptology ePringt Archive 2017:065, 2017.

[12] J. Kim, S. Kim, and J. H. Seo, "A new scale-invariant homomorphic encryption scheme," *Information Sciences*, vol. 422, pp. 177–187, 2018.

[13] K. Lauter, M. Naehrig, and V. Vaikuntanathan, "Can homomorphic encryption be practical?" in *Proceedings of the 3rd ACM Workshop on Cloud Computing Security Workshop*, pp. 113–124, New York, NY, USA, 2011.

[14] P. S. Pisa, M. Abdalla, and O. C. M. B. Duarte, "Somewhat homomorphic encryption scheme for arithmetic operations on large integers," in *Proceedings of the 2012 Global Information Infrastructure and Networking Symposium, (GIIS '12)*, pp. 1–8, IEEE, 2012.

[15] H. J. Cheon, K. Han, and D. Kim, "Faster bootstrapping of fhe over the integers," *IACR Cryptology ePrint Archive*, vol. 79, 2017.

[16] J. H. Cheon and J. Kim, "A hybrid scheme of public-key encryption and somewhat homomorphic encryption," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 5, pp. 1052–1063, 2015.

[17] C. Gentry and S. Halevi, "Fully homomorphic encryption without squashing using depth-3 arithmetic circuits," in *Proceedings of the2011 IEEE 52nd Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 107–116, 2011.

[18] E. Kim and M. Tibouchi, "FHE over the integers and modular arithmetic circuits," in *Proceedings of the International Conference on Cryptology and Network Security*, pp. 435–450, Springer International Publishing, 2016.

[19] A. Bhattacharyya, P. Indyk, D. P. Woodruff et al., "The complexity of linear dependence problems in vector spaces," *ICS*, pp. 496–508, 2011.

[20] M. Fürer, "Faster integer multiplication," *SIAM Journal on Computing*, vol. 39, no. 3, pp. 979–1005, 2009.