Faster Bootstrapping of FHE over the Integers

Jung Hee Cheon, Kyoohyung Han, and Duhyeong Kim

Seoul National University, Republic of Korea
jhcheon@snu.ac.kr, satanigh@snu.ac.kr, doodoo1204@snu.ac.kr

Abstract. Bootstrapping in fully homomorphic encryption (FHE) over the integers is a homomorphic evaluation of the squashed decryption function suggested by van Dijk et al. The typical approach for the bootstrapping is representing the decryption function as a binary circuit with a fixed message space. All bootstrapping methods in FHEs over the integers use this approach; however, these methods require too many homomorphic multiplications, slowing down the whole procedure. In this paper, we propose an efficient bootstrapping method using various message spaces. Our bootstrapping method requires only $O(\log^2 \lambda)$ number of homomorphic multiplications, which is significantly lower than $\tilde{O}(\lambda^4)$ of the previous methods. We implement our bootstrapping method on the scale-invariant FHE over the integers; the CLT scheme introduced by Coron, Lepoint and Tibouchi. It takes 6 seconds for a 500-bit message space and a 72-bit security in PC. This is the fastest result among the bootstrapping methods on FHEs over the integers. We also apply our bootstrapping method to evaluate an AES-128 circuit homomorphically. As a result, it takes about 8 seconds per 128-bit block and is faster than the previous result of homomorphic evaluation of AES circuit using FHEs over the integers without bootstrapping.

1 Introduction

Following Gentry's blueprint [9], the essential step from Homomorphic Encryption (HE) to Fully Homomorphic Encryption (FHE) is a homomorphic evaluation of decryption function, which is called *bootstrapping.* The first FHE over the integers was proposed by van Dijk et al. [14], and it was extended to the batch version [4] and the non-binary message space version [13]. Furthermore, following the scale-invariant technique of a lattice-based FHE [2], so called the modulus switching technique, Coron et al. [6] succeeded to construct a scale-invariant FHE over the integers.

In FHEs over the integers with the secret integer $p \in \mathbb{Z}^+$, the message $m \in \mathbb{Z}_t$ is encrypted into an integer $c = pq + te + m$, $pq + \lfloor p/t \rceil m + e$ or $p^2 q + \lfloor p/t \rceil (m + tr) + e$ according to the schemes [5, 6, 14], where q, r, e are uniform randomly chosen integers from some prescribed intervals. Hence the bootstrapping procedure is to homomorphically evaluate the decryption function

$$m = \left\lfloor \frac{t}{p} \cdot c \right\rceil \bmod t \quad \text{or} \quad c - p \cdot \left\lfloor \frac{c}{p} \right\rceil \bmod t.$$

The complicated division by p can be relaxed by using the hardness assumption of the sparse subset sum problem (SSSP): $1/p \approx \sum_{i=1}^{\Theta} s_i y_i \bmod t$ for secret bit $s_i \in \{0, 1\}$, public rational number $y_i \in [0, t)$ of κ-bit precision with $\kappa > \log |c| + \lambda$ and $\Theta = \tilde{O}(\lambda^4)$. In that case, the decryption function is reduced to

$$m = \left\lfloor \sum_{i=1}^{\Theta} s_i \frac{w_i}{t^{n-1}} \right\rceil \bmod t \quad \text{or} \quad c - p \cdot \left\lfloor \sum_{i=1}^{\Theta} s_i \frac{w_i}{t^n} \right\rceil \bmod t,$$

where $n = O(\log_t \lambda)$ and $w_i = \lfloor c \cdot y_i \cdot t^n \rceil \bmod t^{n+1}$.

In the previous methods [13, 14], each s_i is encrypted under a HE with message space $\mathbb{Z}_t$, so we need to expand each w_i t-adically and each digits of $s_i w_i$ are encrypted separately. As a result, each of digits of $s_i w_i$ are encrypted as different ciphertexts, so the homomorphic evaluation of the additions in the decryption circuit should be done digit-wisely. In that case, a large number of carry computations are required, which results in $\tilde{O}(\lambda^4)$ homomorphic multiplication in the bootstrapping. One possible approach to avoid this massive homomorphic multiplication is to encrypt s_i with a HE with plaintext space as large as w_i. However, in that case, $\log t$-bit should be homomorphically extracted and this is regarded as a problem as hard as bootstrapping.

1.1 Overview of our Bootstrapping method

The idea of our new bootstrapping method is to use a homomorphic encryption scheme with *various message spaces.* Let us denote $\mathrm{Enc}_{\mathcal{M}}(m) :=$ a ciphertext of message m under an encryption with plaintext space $\mathcal{M}$. For a given bootstrapping key $(\mathsf{bk}_i)_{1 \le i \le \Theta} =$

$(\mathrm{Enc}_{\mathbb{Z}_{t^{n+1}}}(s_i))_{1\le i\le\Theta}$, the output of the homomorphic additions for bootstrapping is the ciphertext $\hat{c}$ of the form $\hat{c} = \sum_{i=1}^{\Theta} w_i \cdot \mathsf{bk}_i = \mathrm{Enc}_{\mathbb{Z}_{t^{n+1}}}(m \cdot t^n + \sum_{j=0}^{n-1} z_j \cdot t^j)$ for some integers $z_j \in [0, t)$. To complete the bootstrapping process, it is required to compute the ciphertext $c = \mathrm{Enc}_{\mathbb{Z}_t}(m)$ from the ciphertext $\hat{c}$.

For this, we first suggest plaintext space contraction and dilation functions over the ciphertexts of HEs over the integers.

$$\mathsf{PSCon}_i : \mathrm{Enc}_{\mathbb{Z}_{t^k}}(m \cdot t^i) \mapsto \mathrm{Enc}_{\mathbb{Z}_{t^{k-i}}}(m) \text{ for } 1 \le i < k,$$

$$\mathsf{PSDil}_i : \mathrm{Enc}_{\mathbb{Z}_{t^{k-i}}}(m) \mapsto \mathrm{Enc}_{\mathbb{Z}_{t^k}}(m \cdot t^i) \text{ for } 1 \le i < k,$$

which do not affect error growth. In case of lattice-based FHEs, these techniques already exist and bootstrapping can be done efficiently by exploiting them. In this paper, we suggest PSCon and PSDil techniques for HEs over the integers. With these techniques, we can homomorphically extract n-th digit of $(m \cdot t^n + \sum_{i=0}^{n-1} z_i \cdot t^i)$ in HEs over the integers using a gap-increasing polynomial $F_{t,n}(X)$ suggested by Halevi and Shoup [12], which satisfies the following equation:

$$F_{t,n}(x \cdot t^k + a) = y \cdot t^{k+1} + a \text{ for } a \in [0, t) \cap \mathbb{Z}, x, y \in \mathbb{Z}.$$

The overview of homomorphic digit extraction with the functions PSCon and PSDil in HE schemes over the integers are follows. For digit extraction, we need to compute $c_i = \mathrm{Enc}_{\mathbb{Z}_{t^{n-i+1}}}(tx_i + z_i)$ for $0 \le i \le n$ and $x_i \in \mathbb{Z}$. Assume that following ciphertexts are given:

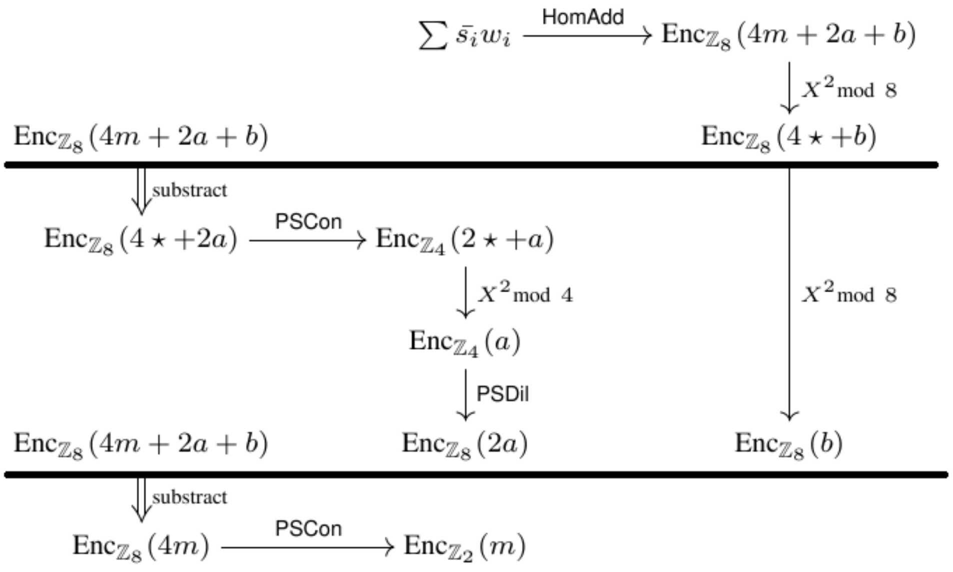
$$c_0 = \mathrm{Enc}_{\mathbb{Z}_{t^{n+1}}}(tx_0 + z_0)$$

$$c_1 = \mathrm{Enc}_{\mathbb{Z}_{t^n}}(tx_1 + z_1)$$

$$\vdots$$

$$c_{i-1} = \mathrm{Enc}_{\mathbb{Z}_{t^{n-i+2}}}(tx_{i-1} + z_{i-1}).$$

By $(i-j)$ time evaluating $F_{t,n}(X)$ for each c_j and using PSDil technique, we can get $c'_j = \mathrm{Enc}_{\mathbb{Z}_{t^{n+1}}}(y_j t^{i+1} + z_j t^j)$ for each $j \in \{0, 1, \cdots, i-1\}$. By subtraction and PSCon technique, we get $c_i = \mathrm{Enc}_{\mathbb{Z}_{t^{n-i+1}}}(tx_i + z_i)$. Now we can get c_i from c_j for $1 \le j \le i-1$, so we can compute $c_n = \mathrm{Enc}_{\mathbb{Z}_t}(m)$ recursively. The figure below is our bootstrapping process with simple case of $n = 2$ and $t = 2$.



$\sum \bar{s}_i w_i$
HomAdd
$\mathrm{Enc}_{\mathbb{Z}_8}(4m + 2a + b)$
$X^2 \bmod 8$
$\mathrm{Enc}_{\mathbb{Z}_8}(4m + 2a + b)$
$\mathrm{Enc}_{\mathbb{Z}_8}(4\star + b)$
substract
$\mathrm{Enc}_{\mathbb{Z}_8}(4\star + 2a)$
PSCon
$\mathrm{Enc}_{\mathbb{Z}_4}(2\star + a)$
$X^2 \bmod 4$
$X^2 \bmod 8$
$\mathrm{Enc}_{\mathbb{Z}_4}(a)$
PSDil
$\mathrm{Enc}_{\mathbb{Z}_8}(4m + 2a + b)$
$\mathrm{Enc}_{\mathbb{Z}_8}(2a)$
$\mathrm{Enc}_{\mathbb{Z}_8}(b)$
substract
$\mathrm{Enc}_{\mathbb{Z}_8}(4m)$
PSCon
$\mathrm{Enc}_{\mathbb{Z}_2}(m)$

Fig. 1: Our bootstrapping process for simple case. Note that $F_{2,n}(X) = X^2$ for all $n \in \mathbb{N}$. Here star shape denotes some integer of which we do not need to consider the exact value.

As you can see in Figure 1, our bootstrapping process is simpler than previous works which are very hard to describe. In our implementations for the security parameter $\lambda = 72$, we set parameters $n = 5$ and $t = 2$, which are very small.

1.2 Our Result

Faster bootstrapping method. We propose a faster bootstrapping method for FHEs over the integers than previous methods [13, 14]. Since the complexity of long integers addition depends on both the length and the number of the integers, the number of multiplications in previous works relies on the large parameter $\varTheta = \tilde{O}(\lambda^4)$. Contrary to that, the complexity of homomorphic digit extraction only depends on the small parameter $n = O(\log \lambda)$, and this difference makes our method efficient. A table below is a comparison with the previous method [13].

	[NK15]	Our Method
Degree	$O(\lambda)$	$O(\lambda^{1+\epsilon})$
#.Hommult	$O(\lambda^4 \log^6 \lambda)$	$O(\log^2 \lambda)$

Table 1: Comparison with [NK15] method.

Note that the small constant ϵ is consequence of using the large message space $\mathbb{Z}_{t^{n+1}}$.

An implementation on the CLT scheme. The CLT scheme has the fastest homomorphic multiplication algorithm among the HEs over the integers, and the ciphertext form of the scheme is appropriate to apply our method. Therefore, we apply our method on the CLT scheme, and provide a precise noise analysis of the bootstrapping procedure. Our implementation on the CLT scheme takes about 6 seconds for a 500-bit message space with a 80-bit security. This result is far superior comparing to the previous result in [4], 13 minutes for a 500-bit message space.

Homomorphic evaluation of AES circuit. Due to the inefficiency of bootstrapping, homomorphic evaluations of AES circuit with leveled or scale-invariant FHEs so far have been implemented without bootstrapping. Contrary to the previous works, we implement a homomorphic evaluation of an AES-128 circuit using our bootstrapping method on the CLT scheme with low depth parameters. In our implementation, the evaluation takes about 8 seconds per block, and this result is faster than the result in [6] without bootstrapping (with large depth parameters), 26 seconds per block. Furthermore, this is the first time that homomorphic evaluation of AES circuit *with bootstrapping is more efficient* than homomorphic evaluation of AES circuit without bootstrapping.

1.3 Notation

- $[n] = \{0, 1, \cdots, n-1\}$, and $\mathbb{Z}_n$ is treated as $[n]$ in this paper.
- $a \bmod p$ for $a \in \mathbb{R}$ is a unique number $\in [0, p)$ such that $a - (a \bmod p)$ is an integer multiple of p.
- $\lfloor a \rceil$ is the nearest integer of a, and $[a]_p$ is a unique integer in $(-p/2, p/2]$ such that $a - [a]_p$ is a multiple of p.
- $a^{(t)}\langle r \rangle := a_k$ for a non-negative integer $a = \sum a_i t^i$ and $a_i \in [t]$. When $t = 2$, we omit the subscript t.

2 Preliminaries

In this section, we introduce a squashing technique of FHEs over the integers and a digit extraction technique in detail, which are used for our bootstrapping method.

2.1 Squashed Decryption Function

In the FHEs over the integers with secret integer p, a ciphertext of message $m \in [t]$ is of the form $c = pq + te + m$ or $pq + \frac{p}{t}m + e$ for q, $e \in \mathbb{Z}$. Each form of ciphertext is decrypted by

$$\left(c - p \cdot \left\lfloor \frac{c}{p} \right\rceil\right) \bmod t \text{ or } \left(\left\lfloor \frac{t}{p} \cdot c \right\rceil\right) \bmod t.$$

The decryption function involves the computation of $\lfloor c/p \rceil$ and this should be evaluated homomorphically for bootstrapping. Since division is very complicated for homomorphic evaluation, decryption functions of FHEs over the integers are *squashed* for efficient bootstrapping.

The *Squashing* is a procedure of expressing secret value $1/p$ as a subset sum of public numbers within very small error, which enable to bootstrap efficiently.

The squashing technique was first introduced in [14], and generalized in [13]. Let κ', Θ', and θ' be additional parameters satisfying $\kappa' > (\gamma+\lambda)/\log t$. The concrete parameter settings of Θ' and θ' are discussed in Section 4.5. The method of squashing is identical to that in [13].

- KeyGen. Generate $\mathsf{sk}^* = p$ and pk as before. Set $x_p = \lfloor t^{\kappa'+1}/p \rceil$, choose a random Θ'-bit vector s with Hamming weight θ', and let $S = \{i : s_i = 1\}$. Choose random integers $u_i \in [0, t^{\kappa'+1})$ such that $\sum_{i \in S} u_i = x_p$. Output secret key $\mathsf{sk} = \boldsymbol{s}$ and $\mathsf{pk} = (\mathsf{pk}^*, \boldsymbol{u})$.
- $\mathsf{Encrypt}$. c^* is a ciphertext of a given FHE over the integers. For $1 \le i \le \Theta'$, let w_i given by an integer nearest to the value of $c^* \cdot u_i / t^{\kappa'-n}$ where $n = \lceil \log_t \theta' \rceil + 3$. Output both c^* and $\boldsymbol{w}$.
- $\mathsf{Decrypt}$. Output $m' \leftarrow \lfloor \sum s_i w_i / t^n \rceil \bmod t$.

The squashing technique can be applied not only to the original scheme in [6], but also to the batch version of the scheme by squashing for each p_j as in [13].

2.2 Digit Extraction Technique

Let $F^k(X)$ be a k-time evaluation of the function F. In general, $F(X) = X^t$ does not satisfy the following property when $t > 2$:

$$F^k(x) \bmod t^{k+1} = x \bmod t \quad \forall k \in \mathbb{N}, x \in [0, t) \cap \mathbb{Z}.$$

In [12], for the prime t and the positive integer e, they constructed the polynomial $F_{t,e}(X)$ satisfying the above equation for any $k \le e$. With this polynomial, we can extract $a\langle e' \rangle_t$ for $1 \le e' \le e$ using similar method in [10]. Following lemmas are about existence and construction of the polynomial $F_{t,e}(X)$, which are introduced in [12].

Digit Extraction Algorithm:

Input: non-negative integers x and r

Compute x_i for $1 \le i \le r$ as following :

$$\begin{aligned}
x_0 &= x \\
x_1 &= \frac{[x - F_{t,r}(x)]_{t^{r+1}}}{t} \\
x_2 &= \frac{[x - F_{t,r}^2(x) - tF_{t,r}(x_1)]_{t^{r+1}}}{t^2} \\
x_3 &= \frac{[x - F_{t,r}^3(x) - tF_{t,r}^2(x_1) - t^2 F_{t,r}(x_2)]_{t^{r+1}}}{t^3} \\
&\vdots \\
x_r &= \frac{[x - F_{t,r}^r(x) - \sum_{i=1}^{r-1} t^i F_{t,r}^{r-i}(x_i)]_{t^{r+1}}}{t^r}
\end{aligned}$$

Output: $x_r = x^{(t)}\langle r \rangle$

Fig. 2: Digit Extraction Algorithm

Lemma 1. *(Corollary 5.4 in [12]) For every prime t, there exists a sequence of integer polynomial $f_1, f_2, \cdots$, all of degree $\le t-1$, such that for every exponent $e \ge 1$ and every integer $z = z_0 + t^e z_1$ ($z_0 \in [t]$, $z_1 \in \mathbb{Z}$), we have*

$$z^t \equiv z_0 + \sum_{i=1}^{e} f_i(z_0) t^i \pmod{t^{e+1}}.$$

Lemma 2. *(Corollary 5.5 in [12]) For every prime t and every $e \ge 1$, there exists a polynomial $F_{t,e}$ of degree p such that the equality $F_{t,e}(z_0 + t^{e'} z_1) \equiv z_0 \pmod{t^{e'+1}}$ holds for every integer z_0, z_1 with $z_0 \in [t]$ and every $1 \le e' \le e$.*

Using a special polynomial $F_{t,r}$, we can extract $x^{(t)}\langle r \rangle$ from x, through a polynomial for any non-negative integers x and r, by digit extraction algorithm in Figure 2. Note that the equality in Lemma 2 implies that recursively defined x_is are integers.

3 Our Bootstrapping Method

3.1 Plaintext Space Contraction and Dilation

Let $\mathcal{E}_k(m)$ is a set of ciphertexts which encrypt m with message space $\mathcal{M} = \mathbb{Z}_{t^k}$. For all HE schemes over the integers, we can construct following two plaintext space switching functions for $1 \leq i < k$:

$$\mathsf{PSCon}_i : \mathcal{E}_k(t^i m) \to \mathcal{E}_{k-i}(m),$$

$$\mathsf{PSDil}_i : \mathcal{E}_{k-i}(m) \to \mathcal{E}_k(t^i m).$$

The definitions of these functions are somewhat different depending on the form of a ciphertext.

$\boldsymbol{c = pq + tr + m}$ In the schemes of [14] and [13], with public exact multiplication $x_0 = pq_0$, plaintext space switching functions are described as follows:

- $\mathsf{PSCon}_i(c) = [t^{-i}]_{x_0} \cdot c \bmod x_0$,
- $\mathsf{PSDil}_i(c) = t^i c \bmod x_0$.

Well-definedness of these functions can be checked easily by following equations:

$$\begin{aligned}\mathsf{PSCon}_i(c) &= [t^{-i}]_{x_0} pq + [t^{-i}]_{x_0} t^k r + [t^{-i}]_{x_0} t^i m \bmod x_0 \\ &= pq' + t^{k-i} r + m, \\ \mathsf{PSDil}_i(c) &= t^i pq + t^{i+1} r + t^i m \bmod x_0 \\ &= pq' + t^{i+1} r + t^i m.\end{aligned}$$

$\boldsymbol{c = pq + \lfloor p/t \rceil \cdot m + r}$ In the schemes of [5] and [6], the functions PSCon_i and PSDil_i are identity functions since

$$\left\lfloor \frac{p}{t^k} \right\rceil (tm + t^k r^*) = \left\lfloor \frac{p}{t^{k-1}} \right\rceil (m + t^{k-1} r^*) + \epsilon,$$

$$\left\lfloor \frac{p}{t^k} \right\rceil \cdot tm = \left\lfloor \frac{p}{t^{k-1}} \right\rceil \cdot m + \epsilon.$$

3.2 Homomorphic digit extraction

The rounding function in the squashed decryption function can be expressed as follows:

$$\begin{aligned}\lfloor \sum s_i w_i / t^n \rceil \bmod t &= \lfloor \sum s_i w_i / t^n + 0.5 \rfloor \bmod t \\ &= (\sum s_i w_i + \lfloor t^n/2 \rfloor)^{(t)} \langle n \rangle.\end{aligned}$$

Thus, the squashed decryption could be expressed as additions and a digit-extraction. The problem is how to homomorphically evaluate the function $(\sum s_i w_i + \lfloor t^n/2 \rfloor)^{(t)} \langle n \rangle$ where each w_i is defined in Section 2.

Let t be a prime integer, n be a positive integer less than $\log \lambda$, and $\mathcal{M}$ be a message space. We follow notations in section 2.1 about squashing. In this Section, we suggest a new bootstrapping method. It works on any HE over the integers which satisfies following conditions:

1. The form of a decryption function is $\lfloor \sum s_i w_i / t^n \rceil \bmod t$ or $c - \lfloor \sum s_i w_i / t^n \rceil \bmod t$ where w_i can be computed by public values c and u_i.
2. It supports homomorphic operations with $\mathcal{M} = \mathbb{Z}_{t^i}$ for $1 \leq i \leq n+1$.
3. There exists a polynomial time algorithm HomExt, a function from $\mathcal{E}_{n+1}(m)$ to $\mathcal{E}_1(m\langle n \rangle_t)$, which is a homomorphic evaluation of digit-extraction algorithm in Figure 2.

Given a HE over the integers satisfying the conditions above, our bootstrapping method works as diagram below. New parameters $s_0 = 1$ and $w_0 = \lfloor t^n/2 \rfloor$ are included in the summation.

$$\begin{tikzcd}
c \arrow[r, "\mathrm{Enc}_{\mathbb{Z}_{t^{n+1}}}(s_i),\, u_i"] \arrow[dr, swap, "\mathsf{Bootstrap}"] & \mathrm{Enc}_{\mathbb{Z}_{t^{n+1}}}(\sum s_i w_i) \arrow[d, "\mathsf{HomExt}"] \\
 & \mathrm{Enc}_{\mathbb{Z}_t}(m)
\end{tikzcd}$$

Actually, all HEs over the integers satisfy the above conditions, which means our method can be applied to all integers-based HEs. In the diagram, our bootstrapping method consists of two steps: addition and extraction. Since w_i can be computed by public values and the set $\{\mathrm{Enc}_{\mathbb{Z}_{t^{n+1}}}(s_i)\}$ is given as bootstrapping key, the addition step in this diagram is composed of homomorphic additions on $\mathcal{M} = \mathbb{Z}_{t^{n+1}}$ and modulus operation. Note that modulus operation mod t^{n+1} is automatically done since the message space is given by $\mathcal{M} = \mathbb{Z}_{t^{n+1}}$.

As a result, the extraction step HomExt is the most important part and this part takes the most of running time in bootstrapping procedure. HomExt is a homomorphic evaluation of digit-extraction algorithm in the Figure 2. The algorithm consists of operations on $\mathbb{Z}_{t^i}$, multiplication and division by t, and modulus operation mod t^i for $1 \le i \le n$. In the $(j+1)-$th stage of **Compute** in digit extraction algorithm (in Figure 2), operations on $\mathbb{Z}_{t^k}$ become operations on $\mathbb{Z}_{t^{k-j}}$ after divided by t^j since the numerator is multiple of t^j. Furthermore, operations on $\mathbb{Z}_{t^k}$ can be regarded as operations on $\mathbb{Z}_{t^{k+j}}$ after multiplied by t^j. Therefore, we can absolutely evaluate digit-extraction with the plaintext space contraction function PSCon and the plaintext space dilation function PSDil.

As mentioned in section 2.2, the digit extraction algorithm can be seen as a polynomial of degree t^n with $n(n+1)/2$ times evaluation of $F_{n,t}$. Since n is an integer less than $\log_t \lambda$, the number of homomorphic multiplications in HomExt is $O(\log^2 \lambda)$. Actually the degree of the algorithm is $t^n \approx \lambda$; however, we use the large message space $\mathbb{Z}_{t^{n+1}}$ in the procedure. Therefore, we write the degree as $O(\lambda^{1+\epsilon})$ and ϵ depends on which scheme is used for our bootstrapping method. We will minutely analyze the degree and error growth of our method applying to scale-invariant homomorphic encryption scheme over the integers, CLT scheme [6], in the next section.

4 Our Method on scale-invariant homomorphic encryption scheme

We apply our method on scale-invariant homomorphic encryption scheme in [6], the CLT scheme, since error growth during homomorphic evaluation is linear so that it is suitable to choose low depth parameter for implementation. Furthermore, as mentioned in remark 1, since PSCon and PSDil are trivial mapping, the description of HomExt is very simple.

As mentioned above, we need three conditions: squashed decryption function with $\mathcal{M} = \mathbb{Z}_t$, homomorphic operations on message spaces $\mathbb{Z}_{t^a}$, and homomorphic digit extraction technique. The scheme below is almost same with scale-invariant homomorphic encryption scheme in [6], and we just extend it to the message space $\mathbb{Z}_t$ for the prime t.

4.1 Scale-invariant homomorphic encryption scheme with $\mathcal{M} = \mathbb{Z}_t$

Scheme Description In this section, we follow the notation in [6], and describe the scheme with the message space $\mathbb{Z}_t$ for the prime t. For an η-bit odd integer p and an integer q_0 in $[0, 2^\gamma/p^2)$, we define the set

$$\mathcal{D}^{\rho}_{p,q_0} = \{p^2 \cdot q + r : \mathsf{Choose}\ q \leftarrow [0, q_0), r \leftarrow (-2^\rho, 2^\rho)\}.$$

- $\mathsf{KeyGen}_t(1^\lambda)$. Generate an odd η-bit integer p and a γ-bit integer $x_0 = q_0 \cdot p^2 + r_0$ with $r_0 \leftarrow (-2^\rho, 2^\rho) \cap \mathbb{Z}$ and $q_0 \leftarrow [0, 2^\gamma/p^2) \cap \mathbb{Z}$. Let $x_i \leftarrow \mathcal{D}^{\rho}_{p,q_0}$ for $1 \le i \le \tau$, $y' \leftarrow \mathcal{D}^{\rho}_{p,q_0}$, and $y = y' + \lfloor p/t \rceil$, which is the encryption of 1. Let $\boldsymbol{z}$ be a vector of length Θ, the components of which have $\kappa = 2\gamma + 2$ bits of precision following the binary point. Let $\boldsymbol{s} \in \{0,1\}^{\Theta}$ such that

$$\frac{t \cdot 2^{\eta}}{p^2} = \langle \boldsymbol{s}, \boldsymbol{z} \rangle + \epsilon \mod (t \cdot 2^{\eta}),$$

with $|\epsilon| \le 2^{-\kappa}$. Now define

$$\boldsymbol{\sigma} = \boldsymbol{q} \cdot p^2 + \boldsymbol{r} + \left\lfloor \mathsf{PowersofTwo}_{\eta}(\boldsymbol{s}) \cdot \frac{p}{2^{\eta+1}} \right\rceil,$$

where the components of $\boldsymbol{q}$ are randomly chosen from $[0, q_0) \cap \mathbb{Z}$ and those of $\boldsymbol{r}$ from $(-2^{\rho}, 2^{\rho}) \cap \mathbb{Z}$. The secret key is $\mathsf{sk} = \{p\}$ and the public key is $\mathsf{pk} = \{x_0, x_1, \cdots, x_{\tau}, y, \boldsymbol{\sigma}, \boldsymbol{z}\}$.

- $\mathsf{Encrypt}_t(\mathsf{pk}, m \in [t])$. Choose a random subset $S \subset \{1, \cdots, \tau\}$ and output

$$c \leftarrow [m \cdot y + \sum_{i \in S} x_i]_{x_0}.$$

- $\mathsf{Decrypt}_t(\mathsf{sk}, c)$. Output $m \leftarrow \left\lfloor t \cdot \frac{c}{p} \right\rceil \mod t$.
- $\mathsf{Add}_t(\mathsf{pk}, c_1, c_2)$. Output $c' \leftarrow c_1 + c_2 \mod x_0$.
- $\mathsf{Convert}_t(\mathsf{pk}, c)$. Output $c' \leftarrow 2 \cdot \langle \boldsymbol{\sigma}, \mathsf{BitDecomp}_{\eta}(\boldsymbol{c}) \rangle$ where $\boldsymbol{c} = (\lfloor c \cdot z_i \rceil \mod 2^{\eta})_{1 \le i \le \Theta}$.
- $\mathsf{Mult}_t(\mathsf{pk}, c_1, c_2)$. Output $c' \leftarrow [\mathsf{Convert}(\mathsf{pk}, c_1 \cdot c_2)]_{x_0}$.

Semantic Security Security for this scheme is from same problem introduced in [6]. The only difference is change of message space from $\mathbb{Z}_2$ to $\mathbb{Z}_t$, so we omit this part.

Conditions on the Parameters The parameters must satisfy the following conditions for security parameter λ and message space $\mathbb{Z}_t$:

- $\rho = \Omega(\lambda)$ to avoid brute force attacks on noise [3,8],
- $\eta \ge \rho + O(L(\log \lambda + \log t))$, where L is the depth of multiplication of the circuits to be evaluated,
- $\gamma \ge \omega((2\eta - \rho)^2 \cdot \log \lambda)$ to avoid lattice-based attacks [7,14],
- $\Theta^2 \ge \gamma \cdot \omega(\log \lambda)$ to avoid lattice attacks on the subset sum problem [7],
- $\tau \ge \gamma + 2\lambda$ to apply the leftover hash lemma.

4.2 Homomorphic Operations with $\mathcal{M} = \mathbb{Z}_{t^a}$

During bootstrapping, we use homomorphic addition and multiplication between ciphertexts on the message space $\mathcal{M} = \mathbb{Z}_{t^a}$ for $1 \le a \le \log_t \lambda$. Homomorphic addition and multiplication are described below. Note that x_0 is defined in the same manner as in the previous section, and the definition of Eval is non-deterministic since the method of the evaluation depends on the formation of a given polynomial.

- $\mathsf{Add}_t^a(\mathsf{pk}, c_1, c_2)$. Output $c_1 + c_2 \mod x_0$.
- $\mathsf{Mult}_t^a(\mathsf{pk}, c_1, c_2)$. Output $\mathsf{Convert}_t(\mathsf{pk}, t^{a-1} \cdot c_1 \cdot c_2)$
- $\mathsf{Eval}_t^a(\mathsf{pk}, f, c)$. Output the homomorphic evaluation of the ciphertext c with the polynomial f by operations defined above.

A ciphertext $c = q \cdot p^2 + (t^a r^* + m) \cdot \lfloor p/t^a \rfloor + r$ has two kinds of errors, r and r^*. We call c a ciphertext with noise (ρ, ρ^*) if $|r| < 2^{\rho}$ and $|r^*| < 2^{\rho^*}$. Lemma 3 shows the correctness of Add_t^a and Mult_t^a as well as analysis on noise growth during the homomorphic operations. We notice that the proof of Lemma 3 is definitely not new one compared to the proof in [6]; we only generalize it from the case of $t = 2$ to the case of arbitrary prime t.

Lemma 3. *(Noise growth analysis) Let c_1 and c_2 be ciphertexts with noise (ρ_1, ρ_1^*) and (ρ_2, ρ_2^*), respectively. Let $\rho = \mathsf{max}(\rho_1, \rho_2)$ and $\rho^* = \mathsf{max}(\rho_1^*, \rho_2^*)$. Then,*

- *$\mathsf{Add}_t^a(\mathsf{pk}, c_1, c_2)$ is a ciphertext with noise $(\rho + 2, \rho^* + 1)$*
- *$\mathsf{Mult}_t^a(\mathsf{pk}, c_1, c_2)$ is a ciphertext with noise $(\rho + \rho^* + a \log t + 8, \log \Theta)$*

Proof. Let c_1, c_2 as below.

$$c_1 = q_1 \cdot p^2 + \lfloor p/t^a \rfloor \cdot (m_1 + t^a r_1^*) + r_1,$$

$$c_2 = q_2 \cdot p^2 + \lfloor p/t^a \rfloor \cdot (m_2 + t^a r_2^*) + r_2.$$

Then addtion of c_1 and c_2 is

$$\begin{aligned} c_1 + c_2 = (q_1 + q_2) \cdot p^2 + \lfloor p/t^a \rfloor \cdot ([m_1 + m_2]_{t^a} \\ + t^a(r_1^* + r_2^* + 1/0)) + r_1 + r_2 \\ = q_3 \cdot p^2 + \lfloor p/t^a \rfloor \cdot (m_3 + t^a r_3^*) + r_3 \end{aligned}$$

for $r_3^* < 2^{\rho_1^*} + 2^{\rho_2^*} + 1$ and $r_3 < 2^{\rho_1} + 2^{\rho_2}$. The ciphertext of $[m_1 + m_2]_{2^a}$ is $c_3 = [c_1 + c_2]_{x_0} = c_1 + c_2 - k \cdot x_0$ for $k \in \{0, 1\}$ since $c_1,\ c_2 < x_0$. Therefore, $c_3 \leftarrow \mathsf{Add}_t^a(\mathsf{pk}, c_1, c_2)$ is a ciphertext $c_3 = q \cdot p^2 + \lfloor p/t^a \rfloor (m + t^a r^*) + r$ satisfying $r^* < 2^{\rho_1^*} + 2^{\rho_2^*} + 1$ and $r < 2^{\rho_1} + 2^{\rho_2} + 2^{\rho_0}$.

Let c_1, c_2 as defined above, and $k,\ l$ be integers such that $\lfloor p/t^a \rfloor = (p-k)/t^a$ and $\lfloor p^2/t^a \rfloor = (p^2 - l)/t^a$. Then the following equation holds,

$$\begin{aligned} c_3 &= q_3 \cdot p^2 + ((p-k)^2/t^a)(m_1 + t^a r_1^*)(m_2 + t^a r_2^*) + R \\ &= q_3 \cdot p^2 + ((p^2 - l)/t^a) \cdot (m_1 m_2 \bmod t^a) + R + R' \\ &= q_3 \cdot p^2 + \lfloor p^2/t^a \rfloor \cdot (m_1 m_2 \bmod t^a) + r_3 \end{aligned}$$

where $|R| < 3 \cdot 2^\eta \cdot t^a \cdot 2^{\rho^* + \rho}$ and $|R'| < 2 \cdot 2^\eta \cdot t^{2a} \cdot 2^{2\rho^*} + t^{2a} \cdot 2^{2\rho^*} < 3 \cdot 2^\eta \cdot t^{2a} \cdot 2^{2\rho^*}$.

Therefore, the inequality $|r_3| < 6 \cdot 2^{\eta + a \log t + \rho + \rho^*}$ holds when assuming $a \log t + \rho^* < \rho$.

Now we will analyze the error of ciphertext after processing $\mathsf{Convert}$ procedure. We followed the proof of lemma 1 in [6].

Let $\lceil \log r_3 \rceil = \rho_3 < \eta + 2a \log t + \rho + \rho^* + 3$ and $c \leftarrow \mathsf{Convert}(c_3/t)$, then from the equation

$$\boldsymbol{\sigma} = p^2 \cdot \boldsymbol{q} + \boldsymbol{r} + \lfloor \boldsymbol{s'} \cdot \frac{p}{2^{\eta+1}} \rceil$$

Let $\boldsymbol{c}' = \mathsf{BitDecomp}_\eta(\boldsymbol{c})$, then we have:

$$c = 2\langle \boldsymbol{\sigma}, \boldsymbol{c}' \rangle = 2p^2 \cdot \langle \boldsymbol{q}, \boldsymbol{c}' \rangle + 2\langle \boldsymbol{r}, \boldsymbol{c}' \rangle + 2\langle \lfloor \boldsymbol{s'} \cdot \frac{p}{2^{\eta+1}} \rceil, \boldsymbol{c}' \rangle.$$

since the components of $\boldsymbol{c'}$ are bits,

$$2\langle \lfloor \boldsymbol{s'} \cdot \frac{p}{2^{\eta+1}} \rceil, \boldsymbol{c'} \rangle = \langle \frac{p}{2^\eta} \cdot \boldsymbol{s'}, \boldsymbol{c'} \rangle + \nu_2 = \frac{p}{2^\eta} \langle \boldsymbol{s'}, \boldsymbol{c'} \rangle + \nu_2,$$

where $|\nu_2| < \Theta \cdot \eta$. From the definition of $\mathsf{BitDecomp}$ and $\mathsf{PowersofTwo}$, we have $\langle \boldsymbol{s'}, \boldsymbol{c'} \rangle = \langle \boldsymbol{s}, \boldsymbol{c} \rangle \bmod 2^\eta = \langle \boldsymbol{s}, \boldsymbol{c} \rangle + q_2 \cdot 2^\eta$. Moreover

$$\langle \boldsymbol{s}, \boldsymbol{c} \rangle = \sum s_i \left\lfloor \frac{c_3}{t} \cdot z_i \right\rceil + \Delta \cdot 2^\eta = \sum \frac{s_i \cdot c_3 \cdot z_i}{t} + \delta_1 + \Delta \cdot 2^\eta$$

$$= \frac{c_3}{t} \cdot \langle \boldsymbol{s}, \boldsymbol{z} \rangle + \delta_1 + \Delta \cdot 2^\eta,$$

for some $\Delta \in \mathbb{Z}$ and $|\delta_1| \leq \Theta/2$. Using $\langle \boldsymbol{s}, \boldsymbol{z} \rangle = 2^\eta \cdot t/p^2 - \epsilon - \mu \cdot 2^\eta \cdot t$ for some $\mu \in \mathbb{Z}$, and $c_3 = r_3 + \lfloor p^2/t^a \rfloor \cdot m + q_3 \cdot p^2$, this gives

$$\langle \boldsymbol{s}, \boldsymbol{c} \rangle = q_3 \cdot 2^\eta + \frac{2^\eta}{t^a} m - \frac{\ell \cdot 2^\eta}{p^2 \cdot t^a} m + \frac{2^\eta}{p^2} r_3 - \frac{c_3}{t} \epsilon + \delta_1 + (\Delta - c_3 \cdot \mu) \cdot 2^\eta.$$

Therefore we can write

$$\langle \boldsymbol{s}, \boldsymbol{c} \rangle = q_1 \cdot 2^\eta + m \cdot \frac{2^\eta}{t^a} + r^*$$

for some $r^* \in \mathbb{Z}$, with $|r^*| \leq 2^{\rho_3 - \eta + 3}$. Now we get an equation below:

$$2 \left\langle \left\lfloor \frac{p}{2^{\eta+1}} \cdot \boldsymbol{s'} \right\rceil, \boldsymbol{c'} \right\rangle = q_4 \cdot p + m \cdot \frac{p}{t^a} + r^* \cdot \frac{p}{2^\eta} + \nu_2$$

with $|q_4| \leq \Theta$; namely the components of $(p/2^{\eta+1}) \cdot \boldsymbol{s'}$ are smaller than p and $\boldsymbol{c'}$ is a binary vector. This gives

$$2 \left\langle \left\lfloor \frac{p}{2^{\eta+1}} \cdot \boldsymbol{s'} \right\rceil, \boldsymbol{c'} \right\rangle = (t^a q_4 + m) \cdot \left\lfloor \frac{p}{t^a} \right\rfloor + r_2^*$$

with $|r_2^*| \le 2^{\rho_3-\eta+4}$. Then we obtain

$$\begin{aligned} c &= 2p^2 \cdot \langle \boldsymbol{q}, \boldsymbol{c'} \rangle + 2\langle \boldsymbol{r}, \boldsymbol{c'} \rangle + (t^a q_4 + m) \cdot \left\lfloor \frac{p}{t^a} \right\rfloor + r_2^* \\ &= 2q'' \cdot p^2 + (t^a q_4 + m) \cdot \left\lfloor \frac{p}{t^a} \right\rfloor + r' \end{aligned}$$

where $|r'| \le |r_2^*| + \eta\Theta 2^{\rho+1} \le 2^{\rho_3-\eta+4} + \eta\Theta 2^{\rho+1} < 2^{a\log t+\rho+\rho^*+7} + \eta\Theta 2^{\lambda+1}$. Therefore, c is ciphertext with noise $(\rho + \rho^* + a\log t + 8, \log\Theta)$ if $a\log t + \rho + \rho^* + 5 > \log\eta + \log\Theta + \lambda$.

4.3 Homomorphic Digit Extraction for Scale-invariant HE over the Integers

During homomorphic digit extraction, we use various message spaces from $\mathbb{Z}_t$ to $\mathbb{Z}_{t^{n+1}}$. Let $\mathrm{Enc}_{\mathbb{Z}_{t^k}}(m)$ be a ciphertext of m with message space $\mathbb{Z}_{t^k}$ in the form of $q \cdot p^2 + \lfloor p/t^k \rceil \cdot (m + t^k \cdot r^*) + r$. The following algorithm represents homomorphic digit extraction with CLT scheme. Note that the polynomial $F_{t,n}$ is explained in the section 2.2.

```
HomExt Algorithm (Homomorphic digit extraction):

    Input: A ciphertext $c$ of message space $\mathbb{Z}_{t^{n+1}}$

    Compute $c_{i,j}$ for $0 \le i \le n$, $0 \le j \le n-i$ :
        $c_{0,0} \leftarrow c$
        For $0 \le i \le n-1$,
            For $0 \le j \le n-i-1$,
            $c_{i,j+1} \leftarrow \mathsf{Eval}_t^{n-i+1}(\mathsf{pk}, F_{t,n}, c_{i,j})$
        $c_{i+1,0} \leftarrow c_{0,0} - c_{0,i+1} - c_{1,i} - \cdots - c_{i,1}$
    Output: $c_{n,0}$
```

Fig. 3: HomExt Algorithm

To understand the above algorithm, we need to check when we can change the message space for a fixed ciphertext. In the scale invariant HE over the integer, since PSDil and PSCon are trivial mapping, $\mathrm{Enc}_{\mathbb{Z}_{t^k}}(m)$ can be treated as $\mathrm{Enc}_{\mathbb{Z}_{t^\ell}}(t^{\ell-k}m)$ for $k < \ell$. Conversely, if m is a multiple of $t^{\ell-k}$, $\mathrm{Enc}_{\mathbb{Z}_{t^\ell}}(m)$ can be treated as $\mathrm{Enc}_{\mathbb{Z}_{t^k}}(m/t^{\ell-k})$.

The following lemma shows the correctness of the proposed homomorphic digit extraction algorithm.

Lemma 4. *(Correctness of HomExt) For given $m = b_{0,0}$, define $b_{i,j}$:*

$$b_{i,0} = (b_{0,0} - \sum_{j=0}^{i-1} t^j \cdot b_{j,i-j} \bmod t^{n+1})/t^i \text{ for } 1 \le i \le n,$$

$$b_{i,j+1} = F_{t,n}(b_{i,j}) \text{ for } 0 \le i < n,\ 0 \le j \le n-i.$$

When we set $c_0 = Enc_{\mathbb{Z}_{t^{n+1}}}(b_{0,0})$ and define $(c_{i,j})$ following the HomExt algorithm, then $c_{i,0} = Enc_{\mathbb{Z}_{t^{n-i+1}}}(b_{i,0})$ for $0 \le i \le n$ so that the equality $c_{n,0} = Enc_{\mathbb{Z}_t}(m^{(t)}\langle n \rangle)$ holds.

Proof. We use induction on i. The statement is clear when $i = 0$. Suppose the proposition is true for $i < m$. Then we have

$$\begin{aligned} c_{m,0} &= c_{0,0} - \sum_{j=0}^{m-1} c_{j,m-j} \\ &= c_0 - \sum_{j=0}^{m-1} Enc_{\mathbb{Z}_{t^{n-j+1}}}\left(F_{t,n}^{m-j}(b_{j,0})\right) \\ &= c_0 - \sum_{j=0}^{m-1} Enc_{\mathbb{Z}_{t^{n+1}}}\left(t^j F_{t,n}^{m-j}(b_{j,0})\right) \\ &= Enc_{\mathbb{Z}_{t^{n+1}}}\left(b_{0,0} - \sum_{j=0}^{m-1} t^j F_{t,n}^{m-j}(b_{j,0})\right) \\ &= Enc_{\mathbb{Z}_{t^{n+1}}}\left(b_{0,0} - \sum_{j=0}^{m-1} t^j b_{j,m-j} \bmod t^{n+1}\right) \\ &= Enc_{\mathbb{Z}_{t^{n+1}}}(t^m b_{m,0}) = Enc_{\mathbb{Z}_{t^{n+1-m}}}(b_{m,0}). \end{aligned}$$

Therefore, this lemma holds for any positive $i \leq n$, and this means $c_{n,0} = Enc_{\mathbb{Z}_t}(b_{n,0}) = Enc_{\mathbb{Z}_t}(m^{(t)}\langle n \rangle)$, so this lemma shows the correctness of our bootstrapping procedure.

To sum up, we can homomorphically evaluate digit-extraction, so CLT scheme satisfies all conditions in section 3; namely, our method can be applied to CLT scheme. Now we introduce the explicit explanation of the application of our method on the scheme.

4.4 Our Method on the CLT scheme

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

$$\mathcal{D}^{\rho}_{p,q_0} = \{q \leftarrow [0, q_0), r \leftarrow (-2^\rho, 2^\rho) : \mathsf{Output}\ p^2 q + r\}.$$

- $\mathsf{KeyGen}^*_t(1^\lambda)$. Generate $\mathsf{pk} = \{x_0, x_1, \cdots, x_\tau, \boldsymbol{\sigma}, \boldsymbol{z}\}$ as in Section 4.1. Choose a random a Θ'-bit vector $\boldsymbol{s}'$ with Hamming weight θ', and let $S' = \{i : s'_i = 1\}$. Choose a random integer $u_i \in [0, t^{\kappa+1})$ such that $\sum_{i \in S'} u_i = \lfloor t^{\kappa+1}/p \rceil$. For $n = \lceil \log_t \theta' \rceil + 3$, generate
$$v_i = q_i \cdot p^2 + \left\lfloor \frac{p}{t^{n+1}} \right\rfloor \cdot s'_i + r_i$$
and $v_0 = q \cdot p^2 + \left\lfloor \frac{p}{t^{n+1}} \right\rfloor \cdot \frac{t^n}{2} + r$, where $q, q_i \in [0, q_0)$ and $r, r_i \in (-2^\rho, 2^\rho)$ for $1 \leq i \leq \Theta'$. The secret key is $\mathsf{sk} = \{p\}$ and the public key is $\mathsf{pk}^* = \{\mathsf{pk}, \boldsymbol{u}, \boldsymbol{v}\}$.
- $\mathsf{HomSum}_t(c, \boldsymbol{u}, \boldsymbol{v})$. Generate $w_0 = 1$, $w_i = \lfloor c \cdot u_i / t^{\kappa - n} \rceil \bmod t^{n+1}$ for $n = \lceil \log_t \theta' \rceil + 3$, and output
$$c' \leftarrow \sum_{i=0}^{\Theta'} v_i \cdot w_i \bmod x_0.$$
- $\mathsf{Bootstrap}_t(c, \boldsymbol{u}, \boldsymbol{v})$. For $c' \leftarrow \mathsf{HomSum}_t(c, \boldsymbol{u}, \boldsymbol{v})$, output the new ciphertext $\mathsf{HomExt}(c')$.

4.5 Conditions on the Parameters

The security of the squashed scheme has been studied in [7, 8, 14]. Here, λ is a security parameter, and γ is as in the previous section.

- $n = \lceil \log_t \theta \rceil + 3$ for the correctness of squashed decryption function,
- $\kappa' > (\gamma + \lambda)/\log t$ for the correctness of squashed decryption function,
- $\Theta'^2 \geq \gamma \cdot \omega(\log \lambda)$ to avoid a lattice-based attack on the subset sum problem [7,8],
- $\binom{\Theta'}{\theta'/2} \geq 2^\lambda$ to avoid an attack on the sparse subset sum problem [1].

5 Analysis of Proposed Bootstrapping Method

Our analysis can be more tight for binary message space, since the evaluation of the polynomial $F_{t,n}(X)$ for $t > 2$ is relatively hard due to its complicated form. In this section, we first check the correctness of our bootstrapping method and analyze the noise growth during bootstrapping procedure. Also, we compute the number of homomorphic multiplications in our method, which directly implies the efficiency of our method.

Theorem 1. *For $c^* \leftarrow \mathsf{Bootstrap}(c, \boldsymbol{u}, \boldsymbol{v})$, c^* is ciphertext with noise*

$$(\rho_2, \rho_2^*) = (\rho + \delta + n \log t (\log t + \log \Theta + 8)(1 + \epsilon),\ \log \Theta + n),$$

and ciphertexts c and c^* have same message if ρ^* and ρ_2^* is smaller than p. Here $\epsilon = \left(\frac{n+1}{2} \cdot \log t + t + \frac{n+2}{\log t}\right) / (\log t + \log \Theta + 8)$ and $\delta = (n+1)\log t + \log(\Theta' + 1)$.

Proof.

1. *HomSum*
Note that $v_i = q_i \cdot p^2 + \lfloor p/t^{n+1} \rfloor \cdot s_i + r_i$ and $v_0 = q \cdot p^2 + \lfloor p/t^{n+1} \rfloor \cdot \lfloor t^n/2 \rfloor + r$ with $q,\ q_i \in [0, q_0)$ and $r,\ r_i \in (-2^\rho, 2^\rho)$ for $1 \leq i \leq \Theta'$. So if $c_{0,0} \leftarrow \mathsf{HomSum}(c, \boldsymbol{u}, \boldsymbol{v})$, then $c_{0,0} = q' \cdot p^2 + \lfloor p/t^{n+1} \rfloor \cdot \left((\sum s_i w_i + \lfloor t^n/2 \rfloor) \bmod t^{n+1} + r^* t^{n+1} \right) + r'$ for $|r'| = |\sum_{i=1}^{\Theta'} w_i r_i + r| < (\Theta' + 1) 2^{\rho + (n+1)\log t}$ and $|r^*| \leq \Theta'$. Therefore, $c_{0,0}$ is a ciphertext with noise $(\rho_1, \rho_1^*) = (\rho + (n+1)\log t + \log(\Theta' + 1),\ \log \Theta')$ whose message space is $\mathbb{Z}_{t^{n+1}}$.

2. HomExt

   Let $c_{i,j}$ is a ciphertext with noise $(\rho_{i,j}, \rho^*_{i,j})$, then the equations $\rho_{0,0} = \rho + (n+1)\log t + \log(\Theta' + 1)$ and $\rho^*_{0,0} = \log\Theta'$ holds by above HomSum *procedure. By applying Lemma 3, we can set*

   $$\rho_{i,0} = \mathsf{max}\{\rho_{0,i}, \cdots, \rho_{i-1,1}\} + 2i, \ \rho^*_{i,0} = \log\Theta + i\log t$$

   for $1 \le i \le n$.
   First, we will show the equality

   $$\mathsf{max}\{\rho_{0,i+1}, \cdots, \rho_{i,1}\} = \rho_{i,1}$$

   holds for $0 \le i \le n-1$. Since $c_{j,i-j+1} = \mathsf{Eval}_t^{n-j+1}(\mathsf{pk}, F_{t,n}, c_{j,i-j})$ for $0 \le j \le i$, it is sufficient to compare noise increase of $c_{j,i-j}$ after Mult^a_t. For $1 \le j \le i-1$, the increase of first noise of $c_{j,i-j}$ is less than or equal to $\log\Theta + (n+1)\log t + 8$, and the increase of noise of $c_{i,0}$ is $\rho^*_{i,0} + (n-i+1)\log t + 8 = \log\Theta + (n+1)\log t + 8$. Therefore, the equality $\mathsf{max}\{\rho_{0,i+1}, \cdots, \rho_{i,1}\} = \rho_{i,1}$ holds and we can get

   $$\rho_{i,0} = \rho_{i-1,1} + 2i.$$

   Second, we will analyze the noise increase in while evaluating $F_{t,n}$. Note that the polynomial $F_{t,n}$ is of degree t and its coefficients are bounded by t^{n+1}. Then, we can regard each term of $F_{t,n}$ is contained by at most t times of multiplications, so we get $\rho_{i-1,1} = \rho_{i-1,0} + \lceil\log t\rceil \cdot (\log t \cdot (n-i+2) + \log\Theta + 8) + t\lceil\log t\rceil$. Now, we obtain a recursion formula:

   $$\begin{aligned}\rho_{i,0} = \rho_{i-1,0} + \lceil\log t\rceil \cdot (\log t \cdot (n-i+2) + \log\Theta + 8) \\ + t\lceil\log t\rceil + 2i.\end{aligned}$$

   The consequence of the recursion formula is

   $$\begin{aligned}\rho_{n,0} &= \rho_{0,0} + \log^2 t \cdot \frac{n^2+3n}{2} + n\log t(\log\Theta + 8) \\ &\quad + nt\log t + n^2 + 2n \\ &= \rho_{0,0} + n\log t(\log t + \log\Theta + 8)(1+\epsilon)\end{aligned}$$

   for $\epsilon = \left(\frac{n+1}{2} \cdot \log t + t + \frac{n+2}{\log t}\right) / (\log t + \log\Theta + 8)$.

3. Correctness

   Let $Enc^{\rho,\rho^*}_{\mathbb{Z}_{t^k}}(m)$ be a set of ciphertext with message $m \in \mathcal{M} = \mathbb{Z}_{t^k}$ and error (ρ, ρ^*). Then our bootstrapping process can be described as below diagram.

   $$\begin{tikzcd}
   Enc^{\rho,\log\Theta}_{\mathbb{Z}_t}(m) \arrow[r, "\mathsf{HomSum}"] \arrow[d, Rightarrow, "\mathsf{Bootstrap}"] & Enc^{\rho_1,\rho^*_1}_{\mathbb{Z}_{t^{n+1}}}(\sum s_i w_i + \lfloor t^n/2 \rfloor \bmod t^{n+1}) \arrow[d, "\mathsf{HomExt}"] \\
   Enc^{\rho_2,\rho^*_2}_{\mathbb{Z}_t}(m) \arrow[r, equal] & Enc^{\rho_2,\rho^*_2}_{\mathbb{Z}_t}((\sum s_i w_i + \lfloor t^n/2 \rfloor \bmod t^{n+1})^{(t)}\langle n \rangle)
   \end{tikzcd}$$

   Top side of the diagram was proved in 1. HomSum*. Also, Lemma 4 and 2.* HomExt *exactly signify the right side of the diagram, and the discussion in Section 4.3 shows the equality $m = (\sum s_i w_i + \lfloor t^n/2 \rfloor \bmod t^{n+1})^{(t)}\langle n \rangle$ holds so that bottom side of the diagram is proved.*

Since the first noise grows approximately $(\log t + \log\Theta + 8)$ per each multiplication, we can think of the degree of Bootstrap function is

$$2^{n\log t(1+\epsilon)+\epsilon_1} = O(\lambda^{1+\epsilon+\frac{\epsilon_1}{n\log t}}) = O(\lambda^{1+\epsilon_2})$$

where $\epsilon_1 = \{(n+1)\log t + \log(\Theta' + 1)\}/(\log t + \log\Theta + 8)$.

Theorem 2. *The number of multiplication operations in our bootstrapping algorithm is $O(n(n+1)/2) = O(\log^2\lambda)$.*

Proof. We will treat t as a constant, so the number of multiplication while evaluating polynomial $F_{t,n}$ is constant. The number of evaluation k is equal to $1 + 2 + \cdots + n = n(n+1)/2$; thus, the number of multiplication operations is $O(n(n+1)/2)$.

As a result, in our bootstrapping method, the number of homomorphic multiplications is $O(\log^2\lambda)$ and multiplicative degree is $O(\lambda^{1+\epsilon})$. Comparing to the previous methods including the result in [13], $\tilde{O}(\lambda^4)$ multiplications, our method shows significantly improved result within the framework of efficiency. In addition to theoretical analysis, we will explain the implementation result of our bootstrapping method applying to the CLT scheme in next section.

6 Implementation

While implementing our bootstrapping method, we use word decomposition and the powers of word instead of BitDecomp and PowersofTwo with word size $w = 32$. Moreover, in order to use a public key of reasonable size, we compress the ciphertext using the same method as in [7]. We implement our bootstrapping method and check the running time of Bootstrap. Furthermore, for precise comparison with other FHEs, we implement the homomorphic evaluation of the AES-128 circuit, which has emerged lately as a standard homomorphic evaluation circuit. We encrypt messages bit-wisely while AES evaultion as in [6].

1. **Parameters ($\ell = 500, \lambda = 72$).**
   - AGCD parameters: $\eta = 192$, $\gamma = 3.8 \times 10^5$, $\rho = 52$
   - Convert parameters: $\Theta = 1500$, $\theta = 100$
   - Bootstrap parameters: $\Theta' = 8000$, $\theta' = 15$
2. **Efficiency.**
   - The number of Add: 8000 + 10
   - The number of Mult: 8
   - Error size after bootstrapping: 122 bit
3. **AES evaluation.**
   - Bootstrap Time : 6.7 × 128 sec (128 ciphertexts)
   - SubByte Time : 128 sec
   - **Total AES** Time : 4020 sec
   - **Relative** Time (**Total AES** Time / ℓ): 8 sec

Remark 1. Implementations of our bootstrapping method and homomorphic evaluation of AES circuit were progressed on a desktop with eight core Intel(R) Core(TM) i7-2600 CPU @ 3.40GHz processors and 16GB RAM using C++ and GMP 6.0.0 [11].

This result shows that bootstrapping process can be done with only 8 number of homomorphic multiplications. Our bootstrapping procedure for one ciphertext takes about 6 seconds. This result is faster than previous results in FHE over the integers [4, 7, 14], and also compatable with the result in [12], 320 seconds for 16000-bit message space. Comparing to the results of homomorphic evaluation of AES circuit in [4, 6], 13 minutes and 23 seconds per block at security level $\lambda = 72$, homomorphic evaluation of AES circuit applying our bootstrapping method takes 8 seconds per block on a 8-core machine at 3.4 GHz for the same security level. This implementation of homomorphic evaluation of AES circuit is the first case that using small depth parameter with bootstrapping can be faster than using large depth without bootstrapping.

References

1. Arnab Bhattacharyya, Piotr Indyk, David P Woodruff, and Ning Xie. The complexity of linear dependence problems in vector spaces. In *ICS*, pages 496–508, 2011.
2. Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, pages 309–325. ACM, 2012.
3. Yuanmi Chen and Phong Q Nguyen. Faster algorithms for approximate common divisors: Breaking fully-homomorphic-encryption challenges over the integers. In *Advances in Cryptology–EUROCRYPT 2012*, pages 502–519. Springer, 2012.
4. Jung Hee Cheon, Jean-Sébastien Coron, Jinsu Kim, Moon Sung Lee, Tancrède Lepoint, Mehdi Tibouchi, and Aaram Yun. Batch fully homomorphic encryption over the integers. In *Advances in Cryptology–EUROCRYPT 2013*, pages 315–335. Springer, 2013.
5. Jung Hee Cheon and Damien Stehlé. Fully homomophic encryption over the integers revisited. In *Advances in Cryptology–EUROCRYPT 2015*, pages 513–536. Springer, 2015.
6. Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. Scale-invariant fully homomorphic encryption over the integers. In *Public-Key Cryptography–PKC 2014*, pages 311–328. Springer, 2014.
7. Jean-Sébastien Coron, Avradip Mandal, David Naccache, and Mehdi Tibouchi. Fully homomorphic encryption over the integers with shorter public keys. In *Advances in Cryptology–CRYPTO 2011*, pages 487–504. Springer, 2011.
8. Jean-Sébastien Coron, David Naccache, and Mehdi Tibouchi. Public key compression and modulus switching for fully homomorphic encryption over the integers. In *Advances in Cryptology–EUROCRYPT 2012*, pages 446–464. Springer, 2012.
9. Craig Gentry. Fully homomorphic encryption using ideal lattices. In *STOC*, volume 9, pages 169–178, 2009.
10. Craig Gentry, Shai Halevi, and Nigel P Smart. Better bootstrapping in fully homomorphic encryption. In *Public Key Cryptography–PKC 2012*, pages 1–16. Springer, 2012.

11. Torbjörn Granlund et al. The gnu multiple precision arithmetic library. *TMG Datakonsult, Boston, MA, USA*, 2(2), 1996.
12. Shai Halevi and Victor Shoup. Bootstrapping for helib. In *Advances in Cryptology–EUROCRYPT 2015*, pages 641–670. Springer, 2015.
13. Koji Nuida and Kaoru Kurosawa. (batch) fully homomorphic encryption over integers for non-binary message spaces. In *Advances in Cryptology–EUROCRYPT 2015*, pages 537–555. Springer, 2015.
14. Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully homomorphic encryption over the integers. In *Advances in cryptology–EUROCRYPT 2010*, pages 24–43. Springer, 2010.