1

Faster Bootstrapping with Multiple Addends

TanPing ZHOU[1,2], XiaoYuan YANG[1,2*], LongFei LIU[1], Wei ZHANG[1],YiTao DING[1]

[1] Key Laboratory of Network & Information Security under the People's Armed Police, Electronic Department, Engineering University of People's Armed Police, Xi'an, 710086, China
[2] State Key Laboratory Of Cryptology, P.O.Box5159,Beijing,100878,China
850301775@qq.com

Abstract. As an important cryptographic primitive in cloud computing and outsourced computation, fully homomorphic encryption (FHE) is an animated area of modern cryptography. However, the efficiency of FHE has been a bottleneck that impeding its application. According to Gentry's blueprint, bootstrapping, which is used to decrease ciphertext errors, is the most important process in FHE. However, bootstrapping is also the most expensive process that affecting the efficiency of the whole system. Firstly, we notice that, hundreds of serial homomorphic additions take most of the time of bootstrapping. We made use of the properties of Boolean circuit to reduce the number of serial homomorphic additions by two-thirds, and thus constructed an efficient FHE scheme with bootstrapping in 10ms. Secondly, the most expensive parts in our bootstrapping, EHCM and addition operations of multiple matrices, can be accelerated by parallel. This parallel may accelerate the bootstrapping. At last, we found a set of more efficient combination of parameters. As a result, our security parameter level is 128 bits and the correctness is elevated, compared with TFHE scheme in ASIACRYPT 2016. Experiments show that the running time of our bootstrapping is 10ms, which is only 52 percent of TFHE, and is less than CGGI17.

Keywords: fully homomorphic encryption, bootstrapping process, accumulator, TFHE.

1 Introduction

Fully homomorphic encryption (FHE) is a promising cryptographic primitive that allows directly operation on ciphertexts. The output of the operation can be decrypted into a result which matches the output of running the same operation on the corresponding plaintexts. In other words, FHE has commutative property on encryption and operations, namely, the result of first encryption then doing homomorphic operation, is equal to first doing operation and then encrypt. Since the breakthrough work of Gentry in 2009 [1], there have been a variety of works on construction, efficiency improving and security of FHE schemes [2-12]. However, efficiency is the main obstacle that affecting the application of FHE schemes [13-19]. The most important factor that affecting efficiency is due to the intricate decryption algorithm, which

often involves addition, multiplication and rounding operations. As a result, there are many works focus on how to improve the efficiency of bootstrapping.

In 2009, basing on ideal lattice, Gentry [1] constructed the first CPA secure fully homomorphic encryption scheme, which permits arbitrary homomorphic addition and multiplication operations. In this construction, Gentry provided a generic method: Bootstrapping + Squashing. This method is called Gentry's blueprint today. According to this method, we can get to a FHE from any proper somewhat homomorphic encryption scheme (here "proper" means this scheme allows circuits with depth greater than the depth its own decryption circuit). In 2011, BGV scheme is brought forward by Brakerski, Gentry and Vaikuntanathan [5]. On the ground of BV11b [3] scheme appeared in 2011, BGV separates Modulus Switching from dimension-modulus, and thus obtains a linearly growing error rate. To further improve efficiency, BGV scheme also make use of Batching technique and RLWE assumption, and finally reached a computation cost of $\tilde{O}(\lambda)$ for one homomorphic operation. In Crypt 2013, Gentry, Sahai and Waters [6] adopted approximate eigenvector and ciphertext flatten technique to construct a FHE scheme called GSW. In Eurocrypt 2015, Ducas and Micciancio [20] gave a more efficient scheme DM15, bootstrapping of which only takes 1 second. In Asiacrypt 2016, Chillotti et.al. [21] provided a better FHE, called TFHE (Torus Fully Homomorphic Encryption), whose bootstrapping only takes 0.1 second. Considering that multiplication on $\mathbb{T} \in [0,1)$ generates a production no bigger than the multipliers, TFHE is based on a variant GSW on torus $\mathbb{T}$, called TGSW, which can control the noise more efficiently. What's more, they observing that the external product of TGSW ciphertext (a matrix) and TLWE ciphertext (a vector) can be used to substitute the product of TGSW and TGSW, thus gives a faster bootstrapping. However, we noticed that bootstrapping of THFE involves hundreds of serial homomorphic additions, which restricts its speed.

Our work. Bootstrapping is the most expensive process that affecting the efficiency of the whole system. Firstly, we notice that, hundreds of serial homomorphic additions take most of the time of bootstrapping. We made use of the properties of Boolean circuit to reduce the number of addition operations by two-thirds, and thus constructed an efficient FHE scheme with bootstrapping in 10ms. Secondly, the two expensive parts in our bootstrapping, EHCM and serial homomorphic additions, with two different threads can be implemented at the same time. This parallel can accelerate the bootstrapping. At last, we found a set of more efficient combination of parameters for our scheme. Most of our contribution lies in 2.2 (2) (c), just jump to the section, if you are familiar with TFHE,

Recently, we notice that Chillotti et.al present a paper CGGI17[23] in eprint. They can implement bootstrapping in 13ms. Compare to their scheme, our scheme has two advantages. First, our scheme is faster than CGGI17[23] in most presented mode, such as Debug/fftw, release/fftw, release/spqlios mode, detail lines in the table in section 3.4. Secondly, the two main parts, EHCM and serial homomorphic additions, of our bootstrapping, could be paralleled with two different threads. This parallel can accelerate the bootstrapping process. Comparing to TFHE, a flaw of our scheme is the homomorphic evaluation key is increased from 52.6M into 70.9M. Our code is based

on TFHE, we presented it in https://github.com/lonyliu/tfhe-10ms. In fact, we combined the bootstrapping of our scheme and CGGI17 in our code, and get a more efficient one.

2 Mathematical Preliminaries

In the rest of the paper we will use the following notations [21]. The security parameter is denoted as λ. We denote the set $\mathbb{B} \triangleq \{0,1\}$, the real Torus of real numbers modulo 1 $\mathbb{T} \triangleq \mathbb{R}/\mathbb{Z} \in [0,1)$, the ring of polynomials $\mathfrak{R} \triangleq \mathbb{Z}[X]/(X^N+1)$, the ring of polynomials $\mathbb{T}_N[X] \triangleq \mathbb{R}[X]/(X^N+1) \bmod 1$, where X^N+1 is the $(2N)$-th cyclotomic polynomial $\Phi_{2N}(X) = \prod_{i \in \mathbb{Z}_{2N}^*}(X - \omega_{2N}^i) = X^N + 1$. Finally, we denote $\mathcal{M}_{p,q}(E)$ as the set of matrices $p \times q$ with entries in E.

We write $\|\boldsymbol{x}\|_p = \min_{\boldsymbol{u} \in \boldsymbol{x} + \mathbb{Z}^k}(\|\boldsymbol{u}\|_p)$ for all $\boldsymbol{x} \in \mathbb{T}^k$. It is the p-norm of the representative of $\boldsymbol{x}$ with all coefficients in $(-\frac{1}{2}, \frac{1}{2}]$. By extension, the norm $\|P(X)\|_p$ of a real or integer polynomial $P \in \mathbb{R}[X]$ is the norm of its coefficient vector. If the polynomial is modulo (X^N+1), we take the norm of its unique representative of degree $\leq N-1$.

A distribution χ on the torus is concentrated iff. its support is included in a ball of radius 1/4 of $\mathbb{T}$ with high probability. We define the variance $Var(\chi)$ and the expectation $\mathbb{E}(\chi)$ of χ as respectively $Var(\chi) = \min_{\bar{x} \in \mathbb{T}} \sum p(x) |x - \bar{x}|^2$ and $\mathbb{E}(\chi)$ as the position $\bar{x} \in \mathbb{T}$ which minimizes this expression. By extension, we say that a distribution χ' over $\mathbb{T}^n$ or $\mathbb{T}_N[X]^k$ is concentrated iff. each coefficient has an independent concentrated distribution on the torus. Then the expectation $\mathbb{E}(\chi')$ is the vector of expectations of each coefficient, and $Var(\chi')$ denotes the maximum of each coefficient's Variance.

Fact 2.1 [21]. Let χ_1, χ_2 be two independent concentrated distributions on either $\mathbb{T}, \mathbb{T}^n$ or $\mathbb{T}_N[X]^k$, and $e_1, e_2 \in \mathbb{Z}$ such that $\chi = e_1\chi_1 + e_2\chi_2$ remains concentrated, then the means $\mathbb{E}(\chi) = e_1\mathbb{E}(\chi_1) + e_2\mathbb{E}(\chi_2)$ and $Var(\chi) \leq e_1^2 Var(\chi_1) + e_2^2 Var(\chi_2)$.

Definition 2.2 (LWE) [22]. Let $n \geq 1$ be an integer, $\alpha \in \mathbb{R}^+$ be a noise parameter and $\boldsymbol{s}$ be a uniformly distributed secret in some bounded set $S \in \mathbb{Z}^n$. Denote by $\mathcal{D}_{\boldsymbol{s} \in \mathbb{Z}^n, \alpha}^{LWE}$ the distribution over $\mathbb{T}^n \times \mathbb{T}$ obtained by sampling a couple $(\boldsymbol{a}, b)$, where the left member $\boldsymbol{a} \xleftarrow{U} \mathbb{T}^n$ is chosen uniformly random and the right member

$b = \boldsymbol{a} \bullet \boldsymbol{s} + e$. The error e is a sample from a gaussian distribution with parameter $D_{\mathbb{T},\alpha}$.

–Search problem: given access to polynomial LWE samples, find $\boldsymbol{s} \in S$.

–Decision problem: distinguish between LWE samples and uniformly random samples from $\mathbb{T}^n \times \mathbb{T}$.

Definition 2.3 (TLWE samples) [21]. Let $k \geq 1$ be an integer, *N* a power of 2, and $\alpha \geq 0$ be a noise parameter. A TLWE secret key $\boldsymbol{s} \in \mathbb{B}_N[X]^k$ is a vector of *k* polynomials $\mathfrak{R} = \mathbb{Z}[X]/(X^N + 1)$ with binary coefficients. For security purposes, we assume that private keys are uniformly chosen, and that they actually contain $n \approx Nk$ bits of entropy. The message space of TLWE samples is $\mathbb{T}_N[X]$. A fresh TLWE sample of a message $\mu \in \mathbb{T}_N[X]$ with noise parameter $\alpha \geq 0$ under the key $\boldsymbol{s}$ is an element $\boldsymbol{c} = (\boldsymbol{a}, b) \in \mathbb{T}_N[X]^k \times \mathbb{T}_N[X]$, $b \in \mathbb{T}_N[X]$ has Gaussian distribution $D_{\mathbb{T}_N[X],\alpha,\boldsymbol{s}\bullet\boldsymbol{a}+\mu}$ around $\boldsymbol{s} \bullet \boldsymbol{a} + \mu$. The sample is random iff its left member $\boldsymbol{a}$ is uniformly random $\in \mathbb{T}_N[X]^k$, trivial if $\boldsymbol{a}$ is fixed to 0, noiseless if $\alpha = 0$, and homogeneous iff its message μ is 0.

– Search problem: given access to polynomial fresh random homogeneous TLWE samples, find their key $\boldsymbol{s} \in \mathbb{B}_N[X]^k$.

– Decision problem: distinguish between fresh random homogeneous TLWE samples from uniformly random samples from $\mathbb{T}_N[X]^k \times \mathbb{T}_N[X]$.

Definition 2.4 (Phase) [21]. Let $\boldsymbol{c} = (\boldsymbol{a}, b) \in \mathbb{T}_N[X]^k \times \mathbb{T}_N[X]$ and $\boldsymbol{s} \in \mathbb{B}_N[X]^k$, we define the phase of the sample as $\varphi_{\boldsymbol{s}}(\boldsymbol{c}) \triangleq b - \boldsymbol{s} \bullet \boldsymbol{a}$. The phase is linear over $\mathbb{T}_N[X]^{k+1}$ and is $(kN+1)$ -lipschitzian for the l_∞ distance: $\forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{T}_N[X]^{k+1}$, $\|\varphi_{\boldsymbol{s}}(\boldsymbol{x}) - \varphi_{\boldsymbol{s}}(\boldsymbol{y})\|_\infty \leq (kN+1)\|\boldsymbol{x} - \boldsymbol{y}\|_\infty$.

Definition 2.5 [21]. Let $\boldsymbol{c}$ be a random variable $\in \mathbb{T}_N[X]^{k+1}$, which we'll interpret as a TLWE sample. All probabilities are on the **Ω**-space. We say that $\boldsymbol{c}$ is a valid TLWE sample iff there exist a key $\boldsymbol{s} \in \mathbb{B}_N[X]^k$ such that the distribution of the phase $\varphi_{\boldsymbol{s}}(\boldsymbol{c})$ is concentrated. If $\boldsymbol{c}$ is trivial, all keys s are equivalent, else the mask of $\boldsymbol{c}$ is uniformly random, so $\boldsymbol{s}$ is unique. We then define:

the message of $\boldsymbol{c}$, denoted as $msg(\boldsymbol{c}) \in \mathbb{T}_N[X]$ is the expectation of $\varphi_{\boldsymbol{s}}(\boldsymbol{c})$;

the error, denoted $Err(\boldsymbol{c})$, is equal to $\varphi_{\boldsymbol{s}}(\boldsymbol{c}) - msg(\boldsymbol{c})$; $Var(Err(\boldsymbol{c}))$ denotes the variance of $Err(\boldsymbol{c})$, which is by definition also equal to the variance of $\varphi_{\boldsymbol{s}}(\boldsymbol{c})$;

finally, $\|Err(\boldsymbol{c})\|_\infty$ denotes the maximum amplitude of $Err(\boldsymbol{c})$ (possibly with overwhelming probability).

2.1 The General Idea of Bootstrapping

In 2009, Gentry provided a generic method to build FHE: Bootstrapping + Squashing. Starting from a somewhat homomorphic scheme and using bootstrapping, it can get a fully homomorphic one. The bootstrapping can translate a given ciphertext with large noise to a ciphertext with smaller noise. In the following years, researchers have deeper studied on bootstrapping and made some improvements. The principle of bootstrapping is briefly elaborated in Fig1.

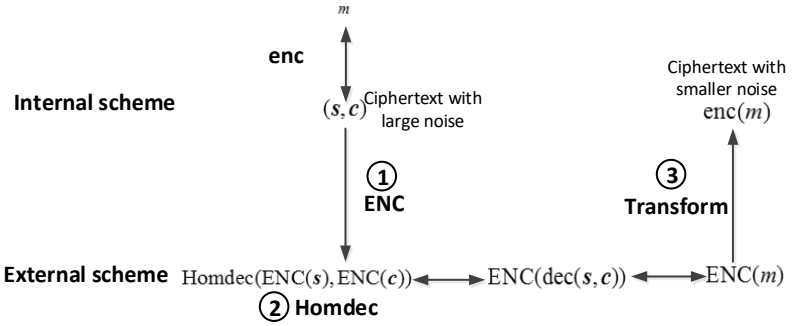

Fig. 1. The general process of bootstrapping

In the process of bootstrapping, decryption of the internal scheme is homomorphically run by the external scheme. Given a "dirty" internal ciphertext $\boldsymbol{c}$ (by "dirty" we mean it has a large noise, also call error, the noise is a random term in encryption, which is often used in lattice based schemes), the external ciphertext $\mathrm{ENC}(\boldsymbol{s})$ of $\boldsymbol{s}$, the transform key(which is used to transform external ciphertext to internal ciphertext). We denote *m* the plaintext of internal ciphertext $\boldsymbol{c}$, $\boldsymbol{s}$ the key of $\boldsymbol{c}$. Bootstrapping works as the following: First, generating the external ciphertext $\mathrm{ENC}(\boldsymbol{c})$. Considering that the ciphertext will not leak the information of the plaintext, so we can directly use $\boldsymbol{c}$ or trivial $\mathrm{ENC}(\boldsymbol{c})$ as the external ciphertext (the trivial $\mathrm{ENC}(\boldsymbol{c})$ means that it has the form of ENC ciphertext, but with no noise). Second, homomorphically run the internal decryption, Homdec, using the external scheme. Notice that, the result $\mathrm{Hom}dec(\ \mathrm{ENC}(\boldsymbol{s}), \mathrm{ENC}(\boldsymbol{c}))$ is equals to $\mathrm{ENC}(dec(\boldsymbol{s}, \boldsymbol{c}))$, by the property of homomorphic encryption. Finally, if the external and internal schemes are different, we should take a step to transform the external ciphertext $\mathrm{ENC}(dec(\boldsymbol{s}, \boldsymbol{c}))$ into a form that consistent with the internal scheme.

The internal ciphertext $\boldsymbol{c}$ is decrypted into plaintext *m* by Homdec, thus the noise will vanished. While the external ciphertext generated initially from $\mathrm{ENC}(\boldsymbol{s})$ and passing through the process of Homdec and ciphertext transformation will contain a large error term. But as long as the final internal ciphertext has less error than the initial internal ciphertext, the aim of error reducing is reached. The final internal ci-

phertext is expected to have a smaller error. To improve efficiency, Homdec, ciphertext transformation and key transformation should also simple and direct in designing.

2.2 Enhanced bootstrapping

TFHE use an enhanced bootstrapping to construct Boolean circuit. The enhanced bootstrapping contains two functions of ciphertext, error descending and plaintext transformation. The former descend ciphertext error through trivial bootstrapping. The latter process can output ciphertext has a plaintext space of $\{\mu_0, \mu_1\} \in \mathbb{T}^2$, which is helpful in Boolean circuit and can be changed in different situation, most of the prior bootstrapping can only output ciphertext with plaintext space {0,1}. The enhanced bootstrapping also has three steps: generate the external ciphertext ENC(***c***), Homomorphic decrypt the internal ciphertext and ciphertext transformation.

(1)Generating the external ciphertext ENC(c):

This process expands the input TLWE ciphertext $\boldsymbol{c} = (\boldsymbol{a}, b) \in \mathbb{T}^{n+1}$ into $\bar{\boldsymbol{c}} = (\bar{\boldsymbol{a}}, \bar{b}) \triangleq (\lfloor 2N\boldsymbol{c} \rceil) \in \mathbb{Z}^{n+1}$, which is the closet integer vector to $2N\boldsymbol{c}$. We use $\bar{\boldsymbol{a}}$ and TLWE ciphertext $trivialTLWE = (\boldsymbol{0}, X^{\bar{b}})$ as the external ciphertext **ENC(c)**, which are the inputs of homomorphic decryption.

(2)Homomorphic decryption:

The Homdec procedure includes round function $round()$, multiply with a constant $\boldsymbol{a}_i \cdot \boldsymbol{s}_i$ and addition $\sum_{i=0}^{n-1} \boldsymbol{a}_i \cdot \boldsymbol{s}_i$, detail lies in following formulas.

$$dec_s(\boldsymbol{a}, b) : \mu' = round(b - \boldsymbol{a} \cdot \boldsymbol{s}) = \begin{cases} 0 & b - \boldsymbol{a} \cdot \boldsymbol{s} \in (0, \frac{1}{4}) \cup (\frac{3}{4}, 1) \\ \frac{1}{2} & -\boldsymbol{a} \cdot \boldsymbol{s} \in (\frac{1}{4}, \frac{3}{4}) \end{cases}$$

$$round(p) = \begin{cases} 0 & p \in (0, \frac{1}{4}) \cup [\frac{3}{4}, 1) \\ \frac{1}{2} & p \in [\frac{1}{4}, \frac{3}{4}) \end{cases}$$

Here round function involves a multi-to-2 mapping, which is hard to implement with normal operations like addition and multiplication. One solution given by FHEW is followed. Mapping the input integer $i \in [0, N-1]$ to the degree of a coefficient bounded polynomial $X^i \bmod (1 + X^{N-1})$, and construct round function through a multi-to-2 mapping between the degree and coefficients of a polynomial, eg, if the coefficient of polynomial is bounded to $(0, \frac{1}{2})$, then the polynomial $\frac{1}{2}X^0 + 0X^1 + \frac{1}{2}X^2 + 0X^3$ can form a multi-to-2 mapping f between the degree and coefficients, where $f(0,2) \to \frac{1}{2}; f(1,3) \to 0$.

Considering that multiplication on $\mathbb{T} \in [0,1)$ generates a production less than the multipliers, we take the internal ciphertext from TLWE to control the growing noise. In other words, we restrict the input ciphertext of bootstrapping to TLWE ciphertext $\boldsymbol{c} \in \mathbb{T}^{n+1}$. In order to map this ciphertext into the degree of a polynomial on $\mathbb{T}_N[X]$, we need to expand the ciphertext $\boldsymbol{c} \in \mathbb{T}^{n+1}$ into $\bar{\boldsymbol{c}} = (\bar{\boldsymbol{a}}, \bar{b}) = (\lfloor 2N\boldsymbol{c} \rceil)$ (where $X^{2N} = 1 \bmod (X^N + 1)$). This is the fundamental reason of expanding the ciphertext $\boldsymbol{c} = (\boldsymbol{a}, b) \in \mathbb{T}^{n+1}$ into $\bar{\boldsymbol{c}} = (\bar{\boldsymbol{a}}, \bar{b}) \triangleq (\lfloor 2N\boldsymbol{c} \rceil) \in \mathbb{Z}^{n+1}$ in 2.2 (1). And because the domain of round function has been expanded, domain of other operations in Homdec should accordingly be changed into $[0, 2N)$.

(a) homomorphic 2Nround function Hom2*Nround*

We introduce a function *2Nround function* and *homomorphic 2Nround function*, which is similar to the round function in $[0, 2N)$. Let $\bar{p} \in [0, 2N)$, $\bar{\mu}' \in \mathbb{T}$, $testv = (1 + \ldots + X^{\frac{N}{2}-1} - X^{\frac{N}{2}} - \ldots - X^{N-1}) \bullet \bar{\mu}'$. We denote $\left(X^{\bar{p}} \bullet testv\right)_0$ as the constant term of the polynomial $X^{\bar{p}} \bullet testv$.

$$2Nround(\bar{p}) = \left(X^{\bar{p}} \bullet testv\right)_0$$

$$= \begin{cases} \left(\bar{\mu}' + \bar{\mu}' X^1 \ldots - \bar{\mu}' X^{N-1}\right)_0 & \bar{p} \in (0, \frac{N}{2}) \cup [\frac{3N}{2}, 2N) \\ \left(-\bar{\mu}' + \bar{\mu}' X^1 \ldots - \bar{\mu}' X^{N-1}\right)_0 & \bar{p} \in [\frac{N}{2}, \frac{3N}{2}) \end{cases} = \begin{cases} \bar{\mu}' & \bar{p} = (0, \frac{N}{2}) \cup [\frac{3N}{2}, 2N) \\ -\bar{\mu}' & \bar{p} = [\frac{N}{2}, \frac{3N}{2}) \end{cases}$$

Constructing homomorphic *2Nround* function. Considering that the *2Nround* function only involves polynomial multiplication $X^{\bar{p}} \bullet testv$ and constant term truncating $\left(X^{\bar{p}} \bullet testv\right)_0$, it can be constructed by homomorphically polynomial multiplication and truncating the constant term. The former operation will be introduced in 3.2 (b), and in the following we describe how to construct a TLWE Extraction function to truncate the constant term of polynomial homomorphically.

Definition 3.1 (TLWE Extraction) [21]. Let $(\boldsymbol{a}'', b'')$ be a $TLWE_{\boldsymbol{s}''}(\mu)$ sample with key $\boldsymbol{s}'' \in \mathfrak{R}^k$, We call $KeyExtract(\boldsymbol{s}'' \in \mathfrak{R}^k)$ the integer vector $\boldsymbol{s}' = (coefs(\boldsymbol{s}_1''(X)), \ldots, coefs(\boldsymbol{s}_k''(X))) \in \mathbb{Z}^{kN}$ and $SampleExtract(\boldsymbol{a}'', b'')$ the LWE sample $(\boldsymbol{a}', b') \in \mathbb{T}^{kN+1}$ where $\boldsymbol{a}' = \{coefs(\boldsymbol{a}_1''(\frac{1}{X})), \ldots, coefs(\boldsymbol{a}_k''(\frac{1}{X}))\}$ and $b' = b_0''$ the constant term of b''. Then $\varphi_{\boldsymbol{s}'}(\boldsymbol{a}', b') = \varphi_{\boldsymbol{s}''}(\boldsymbol{a}'', b'')_0$ (resp. to the constant term of $\mu = msg(\boldsymbol{a}'', b'')_0$, $\|Err(\boldsymbol{a}', b')\|_\infty \le \|Err(\boldsymbol{a}'', b'')\|_\infty$ and $Var(Err(\boldsymbol{a}', b')) \le Var(Err(\boldsymbol{a}'', b''))$.

The TLWE Extraction function can be explained as: Given a private key $\boldsymbol{s}'' \in \mathfrak{R}^k$ and a TLWE ciphertext $(\boldsymbol{a}'', b'') \in \mathbb{T}_N[X]^{k+1}$ with plaintext being a polynomial $\mu \in \mathbb{T}_N[X]$. We can construct a private key related to $\boldsymbol{s}'' \in \mathfrak{R}^k$, and a LWE ciphertext $(\boldsymbol{a}', b') \in \mathbb{T}^{kN+1}$ with plaintext being the constant term μ_0 of μ.

To sum up, we can construct *2Nround* function through polynomial multiplication and homomorphically truncate the constant term of plaintext polynomial. This induces the following formula:

$$Hom2Nround(TLWE(X^{\bar{p}})) = TLWEextraction\left(Hommult(TLWE(X^{\bar{p}}), TLWE(testv))\right)$$

(b) Homomorphic polynomial multiplication

The round function maps the input integer $i \in [0, N-1]$ to the degree of a coefficient bounded polynomial $X^i \bmod(1+X^{N-1})$, this means homomorphic addition should be done on the degree of polynomials, which can be implemented by polynomial multiplication. However, the normal homomorphic multiplication generates very large errors. To control error increasing, we take advantage of two properties of GSW(TGSW) scheme. One property is that, the noise of the homomorphic multiplication is decided by the bound of left multiplier ciphertext mainly. The other is that, multiplication on $\mathbb{T} \in [0,1)$ generate a product less than the multipliers.

1. Preprocessing of low error multiply: ciphertext decomposition on a given basis $\frac{1}{B_g}$.

During practical implementation researchers found that with GSW (TGSW) ciphertext being a matrix, multiplication of two ciphertexts would be very costly. This has been an important reason that affecting multiplication and bootstrapping. However, the error generated by $\boldsymbol{v} \bullet GSW(\mu)$ mainly related with bound of $\boldsymbol{v}$, and there is no need to restrict $\boldsymbol{v}$ having the form of GSW (TGSW) ciphertext. To further reduce computation cost, $\boldsymbol{v}$ can be taken from TLWE ciphertext (a vector), and thus can construct polynomial multiplication through multiplying a vector $\boldsymbol{v}$ and a matrix $GSW(\mu)$.

What's more, researchers want to reduce the noise of the multiply ciphertext, between an TLWE ciphertext $\boldsymbol{v}$ and TGSW ciphertext A. We can preprocessing the TLWE ciphertext $\boldsymbol{v}$, by decomposing the ciphertext $\boldsymbol{v}$ on a given basis $\frac{1}{B_g}$, and output a ciphertext $\boldsymbol{u}$ with higher dimension and smaller bounded coefficients, notice that the noise of the homomorphic multiplication is decided by the bound of left multiplier ciphertext. Thus we can decrease the error during multiplication. Details are shown in Algorithm 1:

Algorithm 1 $Dec_{\boldsymbol{h},\beta,\varepsilon}$ **: Gadget Decomposition of a TLWE Sample**

Input : A TLWE sample $\boldsymbol{v} = (a, b) = (a_1, ..., a_k, b = a_{k+1}) \in \mathbb{T}_N[X]^k \times \mathbb{T}_N[X]$

Output : $\boldsymbol{u} = [e_{1,1}, ..., e_{1,l}, ..., e_{k+1,l}] \in \mathfrak{R}^{(k+1)l}, \mathfrak{R} \triangleq \mathbb{Z}[X]/(X^N + 1)$

1: For each $a_i = \sum_{j=0}^{N-1} a_{i,j} X^j, a_{i,j} \in \mathbb{T}$.set $\overline{a_{i,j}}$ the closest multiple of $\frac{1}{B_g^l}$ to $a_{i,j}$.

2: Decompose $\overline{a_{i,j}} = \sum_{p=1}^{l} \overline{a_{i,j}}_p \frac{1}{B_g^p}$, where $\overline{a_{i,j}}_p \in [-\frac{B_g}{2}, \frac{B_g}{2})$

3: ***for*** $i = 1$ ***to*** $k+1$

4: ***for*** $p = 1$ ***to*** l

5: $e_{i,p} = \sum_{j=0}^{N-1} \overline{a_{i,j}}_p X^j \in \mathfrak{R}$

6: Return $(e_{i,p})_{i,p}$

The following lemma shows that $\boldsymbol{v}$ can be retrieved through multiplying $\boldsymbol{u}$ by the related decomposition basis with error less than $\varepsilon = \frac{1}{2B_g^l}$.

Lemma 3.2 [21]. Let $l \in \mathbb{N}$ and $B_g \in \mathbb{N}$. Then for quality $\beta = \frac{B_g}{2}$ and precision $\varepsilon = \frac{1}{2B_g^l}$, $\boldsymbol{h} \in \mathcal{M}_{(k+1)l,k+1}(\mathbb{T}_N[X])$ as in (1). Algorithm 1 for any TLWE sample $\boldsymbol{v} \in \mathbb{T}_N[X]^{k+1}$, it efficiently and publicly outputs a small vector $\boldsymbol{u} \in \mathfrak{R}^{(k+1)l}$, such that $\|\boldsymbol{u}\|_\infty \le \beta$ and $\|\boldsymbol{uh} - \boldsymbol{v}\|_\infty \le \varepsilon$.

$$\boldsymbol{h} = \begin{Bmatrix} \frac{1}{B_g} & & 0 \\ \vdots & & 0 \\ \frac{1}{B_g^l} & & 0 \\ 0 & \vdots \quad \vdots & 0 \\ 0 & & \frac{1}{B_g} \\ 0 & & \vdots \\ 0 & & \frac{1}{B_g^l} \end{Bmatrix} \in \mathcal{M}_{(k+1)l,k+1}(\mathbb{T}_N[X]) \tag{1}$$

2. Polynomial multiplication with low error——homomorphic addition on the degree of polynomials.

TGSW ciphertext being a matrix, multiplication of two TGSW ciphertexts would be very costly. So TFHE constructed a polynomial multiplication through multiplying a vector $\boldsymbol{b}$ and a matrix A.

Definition 3.3 (External product) [21]. Let A be a valid TGSW sample of message μ_A and let $\boldsymbol{b}$ be a valid TLWE sample of message μ_b.We define the product as

$$\boxdot : \mathrm{TGSW} \times \mathrm{TLWE} \to \mathrm{TLWE} \qquad (A, b) \to A \boxdot \boldsymbol{b} \triangleq Dec_{\mathbf{h},\beta,\varepsilon}(\boldsymbol{b}) \bullet A$$

Theorem 3. 4 (External Product) [21]. Let A be a valid TGSW sample of message μ_A and let $\boldsymbol{b}$ be a valid TLWE sample of message μ_b, let $\beta = \frac{B_g}{2}$ and $\varepsilon = \frac{1}{2B_g^l}$ are the parameters used in the decomposition $Dec_{\boldsymbol{h},\beta,\varepsilon}(\boldsymbol{b})$. Then $A \boxdot \boldsymbol{b}$ is a TLWE sample of message $\mu_A \bullet \mu_b$ and $\left\|Err(A \boxdot \boldsymbol{b})\right\|_\infty \le (k+1)lN\beta \left\|Err(A)\right\|_\infty + \left\|\mu_A\right\|_1 (1+kN)\varepsilon + \left\|\mu_A\right\|_1 \left\|Err(\boldsymbol{b})\right\|_\infty$, $Var(Err(A \boxdot \boldsymbol{b})) \le (k+1)lN\beta^2 Var(Err(A)) + (1+kN)^2 \left\|\mu_A\right\|_2^2 \varepsilon^2 + \left\|\mu_A\right\|_2^2 Var(Err(\boldsymbol{b}))$.

Noticing that addition between the degrees of polynomials has a form of $\mathrm{TGSW} \times \mathrm{TLWE} \to \mathrm{TLWE}$. If there needs to do addition continuously, the new ciphertexts participating in addition should be TGSW ciphertexts. This is shown in Fig 3. These serial homomorphic additions make the scheme hard to do parallel the hundreds of additions. As a result, bootstrapping often needs to do hundreds of serial homomorphic additions, which severely affecting efficiency.

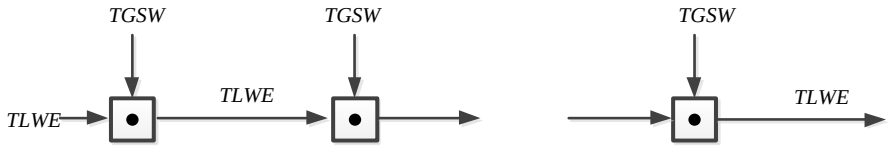

Fig. 3. Accumulate operations in bootstrapping

(c) Enhanced homomorphic constant multiplication on the degree of polynomials, EHCM

To run bootstrapping more efficient, we wish to homomorphic compute $X^{-\bar{\boldsymbol{a}}\cdot\boldsymbol{s}} = X^{-\sum_{i=1}^{n}\bar{\boldsymbol{a}}_i\cdot\boldsymbol{s}_i} = \prod_{i=1}^{n} X^{-\bar{\boldsymbol{a}}_i\cdot\boldsymbol{s}_i}$ quickly. To compute $\prod_{i=1}^{n} X^{-\bar{\boldsymbol{a}}_i\cdot\boldsymbol{s}_i}$ homomorphic, we can first construct TGSW ciphertext $TGSW(X^{-\bar{\boldsymbol{a}}_i\cdot\boldsymbol{s}_i})$. Then using *n* times homomorphic additions on the degree of polynomials, correspond to homomorphic polynomial multiplication given by 2.2 (2) (b), $TGSW(X^{-\bar{\boldsymbol{a}}_i\cdot\boldsymbol{s}_i}) \boxdot TLWE(X^{-\sum_{j=1}^{i}\bar{\boldsymbol{a}}_j\cdot\boldsymbol{s}_j})$. However, homomorphic addition on the degree of polynomials involves multiplying a vector and a matrix, which hundreds of serial homomorphic additions are very expensive and cannot be done paralleled. To solve this problem, we give the following method.

When the private key is taken from {0,1}, instead of computing the ciphertext of $X^{-\bar{\boldsymbol{a}}_i\cdot\boldsymbol{s}_i}$, we can directly generate the ciphertext of $X^{-\overline{\boldsymbol{a}_{i-1}}\bullet\boldsymbol{s}_{i-1}-\bar{\boldsymbol{a}}_i\bullet\boldsymbol{s}_i}$, two addends $-\bar{\boldsymbol{a}}_i\boldsymbol{s}_i$ on a group. This means to construct the ciphertext of $\prod_{i=1}^{n} X^{-\bar{\boldsymbol{a}}_i\cdot\boldsymbol{s}_i}$, it only needs to do addition on the degree of polynomials for *n*/2 times, and in TFHE it is *n* times.

According to the computation properties of Boolean circuit, $X^{-\overline{\boldsymbol{a}_{i-1}}\bullet\boldsymbol{s}_{i-1}-\bar{\boldsymbol{a}}_i\bullet\boldsymbol{s}_i}$ can be expressed in the following formula. This just shows EHCM in plaintext. Similarly, theorem 3.6 presents how to implement EHCM in ciphertext.

$$X^{-\overline{\boldsymbol{a}_{i-1}}\cdot \boldsymbol{s}_{i-1}-\overline{\boldsymbol{a}_i}\cdot \boldsymbol{s}_i} = \begin{cases} 1 & \boldsymbol{s}_{i-1}=0, \boldsymbol{s}_i=0 \\ X^{-\overline{\boldsymbol{a}_i}} & \boldsymbol{s}_{i-1}=0, \boldsymbol{s}_i=1 \\ X^{-\overline{\boldsymbol{a}_{i-1}}} & \boldsymbol{s}_{i-1}=1, \boldsymbol{s}_i=0 \\ X^{-\overline{\boldsymbol{a}_{i-1}}-\overline{\boldsymbol{a}_i}} & \boldsymbol{s}_{i-1}=1, \boldsymbol{s}_i=1 \end{cases}$$

$$= X^{-\overline{\boldsymbol{a}_{i-1}}-\overline{\boldsymbol{a}_i}\cdot \boldsymbol{s}_{i-1}\boldsymbol{s}_i} - X^{-\overline{\boldsymbol{a}_{i-1}}\cdot \boldsymbol{s}_{i-1}(\boldsymbol{s}_i-1)} - X^{-\overline{\boldsymbol{a}_i}\cdot \boldsymbol{s}_i(\boldsymbol{s}_{i-1}-1)} + (\boldsymbol{s}_{i-1}-1)(\boldsymbol{s}_i-1)$$

Definition 3.5(Bootstrapping Key, BK). Let $\boldsymbol{s}\in\mathbb{B}^n, \boldsymbol{s}''\in\mathbb{B}_N[X]^k$ and α be a noise parameter. We define the bootstrapping key $BK_{\boldsymbol{s}\to\boldsymbol{s}'',\alpha}=\{BK_{1,1}, BK_{1,2}, BK_{1,3}, BK_{1,4}, BK_{2,1}, ..., BK_{\frac{n}{2},3}, BK_{\frac{n}{2},4}\}$, where $BK_{i,1}=TGSW_{\boldsymbol{s}'',\alpha}(\boldsymbol{s}_{2i-1}\boldsymbol{s}_{2i}), BK_{i,2}=TGSW_{\boldsymbol{s}'',\alpha}(\boldsymbol{s}_{2i-1}(\boldsymbol{s}_{2i}-1)), BK_{i,3}=TGSW_{\boldsymbol{s}'',\alpha}((\boldsymbol{s}_{2i-1}-1)\boldsymbol{s}_{2i}), \quad BK_{i,4}=TGSW_{\boldsymbol{s}'',\alpha}((\boldsymbol{s}_{2i-1}-1)(\boldsymbol{s}_{2i}-1))$, $i=1,...,\frac{n}{2}$.

Theorem 3.6 (EHCM). Let $\boldsymbol{s}\in\mathbb{B}^n, \boldsymbol{s}''\in\mathbb{B}_N[X]^k$, α be a noise parameter and let $BK=BK_{\boldsymbol{s}\to\boldsymbol{s}'',\alpha}$ be a bootstrapping key, $\vartheta_{BK}=Var(Err(BK_i))=\frac{2}{\pi}\cdot\alpha^2$ and $V_{KS}=Var(Err(KS_i))=\frac{2}{\pi}\cdot\gamma^2$. Then $\text{Keybundle}_i = X^{-\overline{\boldsymbol{a}_{2i-1}}-\overline{\boldsymbol{a}_{2i}}}BK_{i,1} - X^{-\overline{\boldsymbol{a}_{2i-1}}}\cdot BK_{i,2} - X^{-\overline{\boldsymbol{a}_{2i}}}\cdot BK_{i,3} + BK_{i,4}$ is a TGSW sample of message $X^{-\overline{\boldsymbol{a}_{2i-1}}\cdot \boldsymbol{s}_{2i-1}-\overline{\boldsymbol{a}_{2i}}\cdot \boldsymbol{s}_{2i}}$ and $\|Err(\text{Keybundle}_i)\|_\infty \le 4\|Err(BK_{2i})\|_\infty$, $Var(Err(\text{Keybundle}_i)) \le 4\vartheta_{BK}$, $i=1,...,\frac{n}{2}$.

Proof. We analyze the correctness of plaintext, the bound of error and the error variance in ciphertext, in following 3 formulas.

First, we analyze the correctness of plaintext.

$$\begin{aligned}
&\text{Keybundle}_i = X^{-\overline{\boldsymbol{a}_{2i-1}}-\overline{\boldsymbol{a}_{2i}}}BK_{i,1} - X^{-\overline{\boldsymbol{a}_{2i-1}}}\cdot BK_{i,2} - X^{-\overline{\boldsymbol{a}_{2i}}}\cdot BK_{i,3} + BK_{i,4} \\
&= \left(X^{-\overline{\boldsymbol{a}_{2i-1}}-\overline{\boldsymbol{a}_{2i}}}Z_{i,1} - X^{-\overline{\boldsymbol{a}_{2i-1}}}\cdot Z_{i,2} - X^{-\overline{\boldsymbol{a}_{2i}}}\cdot Z_{i,3} + Z_{i,4}\right) + \left(X^{-\overline{\boldsymbol{a}_{2i-1}}\cdot \boldsymbol{s}_{2i-1}-\overline{\boldsymbol{a}_{2i}}\cdot \boldsymbol{s}_{2i}}\right)\boldsymbol{h} \\
&= TGSW_{\boldsymbol{s}'',\alpha}(X^{-\overline{\boldsymbol{a}_{2i-1}}-\overline{\boldsymbol{a}_{2i}}}Z_{i,1} - X^{-\overline{\boldsymbol{a}_{2i-1}}}\cdot Z_{i,2} - X^{-\overline{\boldsymbol{a}_{2i}}}\cdot Z_{i,3} + Z_{i,4}) = TGSW_{\boldsymbol{s}'',\alpha}(X^{-\overline{\boldsymbol{a}_{2i-1}}\cdot \boldsymbol{s}_{2i-1}-\overline{\boldsymbol{a}_{2i}}\cdot \boldsymbol{s}_{2i}})
\end{aligned}$$

Next, we analyze the bound of error in ciphertext.

$$\begin{aligned}
&\|Err(\text{Keybundle}_i)\|_\infty = \left\|Err(X^{-\overline{\boldsymbol{a}_{2i-1}}-\overline{\boldsymbol{a}_{2i}}}BK_{i,1} - X^{-\overline{\boldsymbol{a}_{2i-1}}}\cdot BK_{i,2} - X^{-\overline{\boldsymbol{a}_{2i}}}\cdot BK_{i,3} + BK_{i,4})\right\|_\infty \\
&\le \left\|Err(X^{-\overline{\boldsymbol{a}_{2i-1}}-\overline{\boldsymbol{a}_{2i}}}BK_{i,1}) + Err(X^{-\overline{\boldsymbol{a}_{2i-1}}}\cdot BK_{i,2}) + Err(X^{-\overline{\boldsymbol{a}_{2i}}}\cdot BK_{i,3}) + Err(BK_{i,4})\right\|_\infty \\
&\le \left\|Err(BK_{i,1})\right\|_\infty + \left\|Err(BK_{i,2})\right\|_\infty + \left\|Err(BK_{i,3})\right\|_\infty + \left\|Err(BK_{i,4})\right\|_\infty \\
&\le 4\alpha
\end{aligned}$$

Last, we analyze the error variance in ciphertext.

$$Var(Err(\text{Keybundle}_i)) = Var(Err(X^{-\overline{a_{2i-1}}-\overline{a_{2i}}}BK_{i,1} - X^{-\overline{a_{2i-1}}} \bullet BK_{i,2} - X^{-\overline{a_{2i}}} \bullet BK_{i,3} + BK_{i,4})$$
$$\le Var(Err(BK_{i,1})) + Var(Err(BK_{i,2})) + Var(Err(BK_{i,3})) + Var(Err(BK_{i,4}))$$
$$\le 4\vartheta_{BK}$$

Combining the analysis and construction in (a),(b) and (c), we give the overall process of Homdec in the enhanced bootstrapping. See Fig 3.

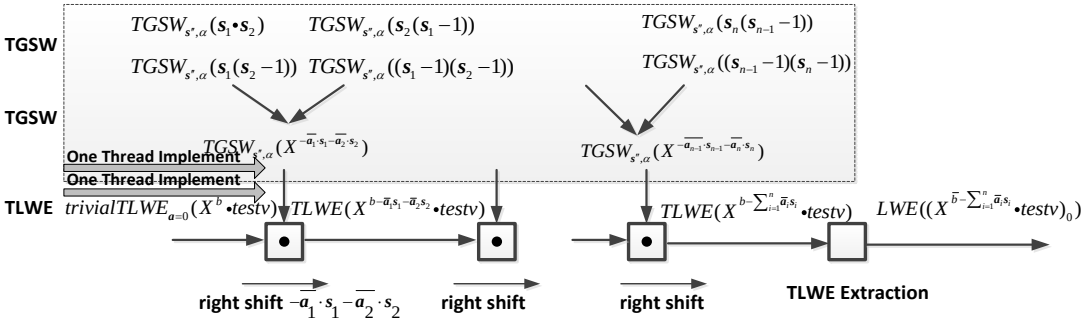

Fig. 3. The overall process of Homdec

Some tricks for implement. (1) The processes of generating $TGSW(X^{-\overline{a_{2i-1}} \bullet s_{2i-1}-\overline{a_{2i}} \bullet s_{2i}})$ ciphertexts are independent with serial homomorphic additions. We can generate the TGSW ciphertexts, at the same time when we operating serial homomorphic additions, with two different threads. This parallel can accelerate the bootstrapping process. (2)The processes of generating $TGSW(X^{-\overline{a_{2i-1}} \bullet s_{2i-1}-\overline{a_{2i}} \bullet s_{2i}})$ ciphertexts $\text{Keybundle}_i = X^{-\overline{a_{2i-1}}-\overline{a_{2i}}}BK_{i,1} - X^{-\overline{a_{2i-1}}} \bullet BK_{i,2} - X^{-\overline{a_{2i}}} \bullet BK_{i,3} + BK_{i,4}$, which only involve rotation and addition on polynomials, can be finished quickly. We recommend generate it before FFT process to accelerate implement.

(3)ciphertext transformation

As shown in Fig 4, the process of Homdec output a LWE ciphertext, with a corresponding key $\boldsymbol{s}' \in \mathbb{Z}^{kN}$. To guarantee the output of bootstrapping can be decrypted by the initial private key $\boldsymbol{s} \in \mathbb{B}^n$, KeySwitch procedure is needed. KeySwitch procedure is designed to transform the ciphertext with one key to another ciphertext with another key, while the plaintext remains the same. KeySwitch procedure is often used in FHE [21]. Due to space limitations, we will not introduce the detail of this process.

Lemma 4.3 (Key switching) [21]. Given $(\boldsymbol{a}', b') \in LWE_{s'}(\mu)$ where $\boldsymbol{s}' \in \{0,1\}^{n'}$ with noise $\eta' \triangleq \left\|Err(\boldsymbol{a}', b')\right\|_\infty$, with noise variance $\eta' \triangleq Var(Err(\boldsymbol{a}', b'))$ and a keyswitching key $KS_{\boldsymbol{s}' \to \boldsymbol{s}, \gamma, t}$, where $\boldsymbol{s} \in \{0,1\}^n$, the key switching procedure outputs a LWE sample $(\boldsymbol{a}, b) \in LWE_{\boldsymbol{s}_n}(\mu)$ where $\left\|Err(\boldsymbol{a}, b)\right\|_\infty \le \eta' + n' t\gamma + n' 2^{-(t+1)}$, $Var(Err(\boldsymbol{a}, b)) \overset{\eta' \triangleq Var(Err(\boldsymbol{a}', b'))}{\le} \eta' + n' t\gamma^2 + n' 2^{-2(t+1)}$.

3 Bootstrapping in 10ms

3.1 The construction of enhanced bootstrapping

Concrete process of enhanced bootstrapping. Combining analysis in (1),(2) ,(3), and the overall process of Homdec, we give the concrete enhanced bootstrapping in the following algorithm.

Algorithm 2: Enhanced Bootstrapping

$Input: (\boldsymbol{a}, b) \in LWE_{\boldsymbol{s}, \eta}(\mu), BK_{\boldsymbol{s} \to \boldsymbol{s}'', \alpha}, KS_{\boldsymbol{s}' \to \boldsymbol{s}, \gamma}, \boldsymbol{s}' = KeyExtract(\boldsymbol{s}'') \in \mathbb{Z}^{kN}, msg = \mu_0, \mu_1 \in \mathbb{T}.$

$Output: LWE_{\boldsymbol{s}, \nu}(\begin{cases} \mu_0 & \varphi_s(\boldsymbol{a}, b) \in (-\frac{1}{4}, \frac{1}{4}] \\ \mu_1 & else \end{cases})$

$1: \bar{\mu} \triangleq \frac{\mu_0 + \mu_1}{2}, \bar{\mu}' \triangleq \frac{\mu_0 - \mu_1}{2}.$

$2: \bar{b} \triangleq \lfloor 2Nb \rceil, \bar{\boldsymbol{a}}_i \triangleq \lfloor 2N\boldsymbol{a}_i \rceil, i \in [1, n]$

$3: testv = (1 + X + ... + X^{N-1}) \times X^{-\frac{2N}{4}} \bullet \bar{\mu}' \in \mathbb{T}_N[X]$

$4: ACC \leftarrow (X^{\bar{b}} \bullet (\boldsymbol{0}, testv)) \in trivialTLWE_{\boldsymbol{a}=0}(\pm\bar{\mu}' + \bar{\mu}' X^1 ... - \bar{\mu}' X^{N-1}) \in \mathbb{T}_N[X]^{k+1}.$

$5:$ *for* $i = 1$ *to* $\frac{n}{2}$

$6:$ $\quad \text{Keybundle}_i = X^{-\overline{\boldsymbol{a}_{2i-1}} - \overline{\boldsymbol{a}_{2i}}} BK_{i,1} - X^{-\overline{\boldsymbol{a}_{2i-1}}} \bullet BK_{i,2} - X^{-\overline{\boldsymbol{a}_{2i}}} \bullet BK_{i,3} + BK_{i,4}$

$7:$ $\quad ACC \leftarrow \text{Keybundle}_i \boxdot ACC.$

$8: \boldsymbol{u} = (\boldsymbol{0}, \bar{\mu}) + SampleExtract(ACC)$

$9:$ *return* $\text{KeySwitch}_{KS_{\boldsymbol{s}' \to \boldsymbol{s}, \gamma}}(\boldsymbol{u})$

Theorem 4.1 (Bootstrapping Theorem). Let $\boldsymbol{h} \in \mathcal{M}_{(k+1)l, k+1}(\mathbb{T}_N[X])$ be the gadget defined in Equation 1 and let $Dec_{\boldsymbol{h}, \beta, \varepsilon}$ be the associated vector gadget decomposition function. Let $\boldsymbol{s} \in \mathbb{B}^n, \boldsymbol{s}'' \in \mathbb{B}_N[X]^k$ and α, γ be noise amplitudes. Let $BK = BK_{\boldsymbol{s} \to \boldsymbol{s}'', \alpha}$ be a bootstrapping key, let $\boldsymbol{s}' = KeyExtract(\boldsymbol{s}'') \in \mathbb{B}^{kN}$ and $KS = KS_{\boldsymbol{s}' \to \boldsymbol{s}, \gamma, t}$ be a keyswitching key. Given $(\boldsymbol{a}, b) \in LWE_{\boldsymbol{s}, \eta}(\mu)$ for $\mu \in \mathbb{T}$, two fixed messages μ_0, μ_1, Algorithm 2 outputs a sample in $LWE_{\boldsymbol{s}}(\mu')$ s.t. $\mu' = \mu_0$ if

$|\varphi_s(\boldsymbol{a},b)| \le -\frac{1}{4}-\delta$ and $\mu' = \mu_1$ if $|\varphi_s(\boldsymbol{a},b)| \ge \frac{1}{4}+\delta$ where δ is the cumulated rounding error equal to $\frac{n+1}{4N}$ in the worst case. Let $\vartheta_{BK} = Var(Err(BK_i)) = \frac{2}{\pi}\bullet\alpha^2$ and $V_{KS} = Var(Err(KS_i)) = \frac{2}{\pi}\bullet\gamma^2$. Let $\boldsymbol{v}$ be the output of Algorithm 2. Then $\|Err(\boldsymbol{v})\|_\infty \le 2n(k+1)l\beta N\alpha + kNt\gamma + \frac{n}{2}(1+kN)\varepsilon + kN2^{-(t+1)}$, $Var(Err(\text{KeySwitch}_{KS_{s'\to s,\gamma}}(\boldsymbol{u}))) \le 2Nn(k+1)l\beta^2\vartheta_{BK} + kNtV_{KS} + \frac{n}{2}(1+kN)\varepsilon^2 + kN2^{-2(t+1)}$.

Proof. According to the above algorithm, we analyze the plaintext, error, as well as error variance in ciphertext.

Line1: It is often required that the output ciphertexts has different plaintext spaces after bootstrapping, for example, in the application of constructing basic gates, plaintext space should be $\{0,\frac{1}{4}\}$. Thus during bootstrapping, we should construct $\bar{\mu} \triangleq \frac{\mu_0+\mu_1}{2}, \bar{\mu}' \triangleq \frac{\mu_0-\mu_1}{2}$ based on concrete requirement and satisfies $\bar{\mu}+\bar{\mu}' \triangleq \mu_0$, $\bar{\mu}-\bar{\mu}' \triangleq \mu_1$.

Line2: Expand ciphertext $\boldsymbol{c}$ into $\bar{\boldsymbol{c}} = (\bar{\boldsymbol{a}},\bar{b}) = (\lfloor 2N\boldsymbol{c} \rceil)$. The object is to make use of the multi-to-2 mapping between degree and coefficients of polynomials, and build round function.

Line3: Construct the multi-to-2 mapping between degree and coefficients of polynomials on $(1+X^N)$. The following formula shows that if the degree of X $\in \{0,\frac{N}{2}-1\} \cup \{\frac{3N}{2},2N-1\}$, then it is mapped to a coefficient $\bar{\mu}'$, and if the degree of X $\in \{\frac{N}{2},\frac{3N}{2}-1\}$, it is mapped to $-\bar{\mu}'$.

$$testv = (1+X+\ldots+X^{N-1})\times X^{-\frac{2N}{4}}\bullet\bar{\mu}' = (-X^N-\ldots-X^{\frac{3}{2}N-1}+X^{\frac{3N}{2}}+\ldots+X^{2N-1})\bullet\bar{\mu}'$$

Line4: Initial assignment for TLWE ciphertext. In this paper, ACC is the accumulator. From the view of plaintext, both of Homadd and *2Nround* function are polynomial multiplication. Considering the associative law of polynomial multiplication, it is equal to first accumulating then computing *2Nround* function, and first do *2Nround* on some of the ciphertexts then accumulating during bootstrapping. We take the second way. The initial TLWE ciphertext generated by $\bar{b}$ is $trivialTLWE_{\boldsymbol{a}=0} = (\boldsymbol{0},X^{\bar{b}})$, which has no error, so first computing *2Nround* function on $\bar{b}$ can better control error expand. This steps output the initial TLWE ciphertext, with a corresponding plaintext $X^{\bar{b}}\bullet testv$, both error and error variance are 0.

$$(X^{\bar{b}}\bullet(\boldsymbol{0},testv)) \in trivialTLWE_{\boldsymbol{a}=0}(X^{\bar{b}}\bullet testv) \in \mathbb{T}_N[X]^{k+1}$$

Line5: Iterate the following step6 and step7 for *n*/2 times.

Line6-9: Let $\bar{\varphi} \triangleq \bar{b} - \sum_{i=1}^{n} \bar{a}_i s_i \bmod 2N$, (2) shows the relationship between $\bar{\varphi}$ and φ , satisfies $2N(\varphi-\delta) \le \bar{\varphi} \le 2N(\varphi+\delta)$. So, when $|\varphi_{\boldsymbol{s}}(\boldsymbol{a},b)| \le -\frac{1}{4}-\delta$, we have $\frac{N}{2} < \bar{\varphi} < \frac{3N}{2}$; when $|\varphi_{\boldsymbol{s}}(\boldsymbol{a},b)| \ge \frac{1}{4}+\delta$, we have $\frac{3N}{2} \le \bar{\varphi} \le 2N$ or $0 \le \bar{\varphi} \le \frac{N}{2}$.

$$\begin{aligned} &\left|\varphi - \tfrac{\bar{\varphi}}{2N}\right| = \left|b - \tfrac{\lfloor 2Nb \rceil}{2N} + \sum_{i=1}^{n}(a_i - \tfrac{\lfloor 2Na_i \rceil}{2N})s_i\right| \overset{s_i \in \mathbb{B}}{\le} \left|\tfrac{1}{4N} + \sum_{i=1}^{n}\tfrac{1}{4N}\right| \le \tfrac{n+1}{4N} < \delta \\ &\to -\delta \le \varphi - \tfrac{\bar{\varphi}}{2N} \le \delta \to 2N(\varphi-\delta) \le \bar{\varphi} \le 2N(\varphi+\delta) \end{aligned} \tag{2}$$

From theorem 3.6, we know that $\text{Keybundle}_i = TGSW_{\boldsymbol{s}'',\alpha}(X^{-\overline{\boldsymbol{a}_{2i-1}}\cdot\boldsymbol{s}_{2i-1}-\overline{\boldsymbol{a}_{2i}}\cdot\boldsymbol{s}_{2i}})$ is TGSW ciphertext, and error term $\|Err(\text{Keybundle}_i)\|_\infty \le 4\|Err(BK_{2i})\|_\infty$, error variance $Var(Err(\text{Keybundle}_i)) \le 4\vartheta_{BK}$. From theorem 3.4, Definition 3.1 and Lemma 4.3, we get following result about plaintext, error and error variance.

Plaintext:

$$msg(\text{KeySwitch}_{KS_{s'\to s,\gamma}}(\boldsymbol{u})) = msg(\boldsymbol{u}) = msg((\boldsymbol{0},\bar{\mu}) + SampleExtract(ACC_{\frac{n}{2}}))$$

$$= \bar{\mu} + msg(ACC_{\frac{n}{2}})_0 = \bar{\mu} + (\prod_{i=1}^{\frac{n}{2}}(X^{-\overline{\boldsymbol{a}_{2i-1}}\cdot\boldsymbol{s}_{2i-1}-\overline{\boldsymbol{a}_{2i}}\cdot\boldsymbol{s}_{2i}})\bullet msg(ACC_0))_0$$

$$= \bar{\mu} + (\prod_{i=1}^{\frac{n}{2}}(X^{-\overline{\boldsymbol{a}_{2i-1}}\cdot\boldsymbol{s}_{2i-1}-\overline{\boldsymbol{a}_{2i}}\cdot\boldsymbol{s}_{2i}})\bullet X^{\bar{b}}testv)_0$$

$$= \begin{cases} \mu_0 & \bar{\varphi} \in (0,\frac{N}{2}) \bigcup [\frac{3N}{2},2N) \Leftarrow |\varphi_{\boldsymbol{s}}(\boldsymbol{a},b)| \le -\frac{1}{4}-\delta \\ \mu_1 & \bar{\varphi} \in [\frac{N}{2},\frac{3N}{2}) \Leftarrow |\varphi_{\boldsymbol{s}}(\boldsymbol{a},b)| \ge \frac{1}{4}+\delta \end{cases} = \begin{cases} \mu_0 & |\varphi_{\boldsymbol{s}}(\boldsymbol{a},b)| \le -\frac{1}{4}-\delta \\ \mu_1 & |\varphi_{\boldsymbol{s}}(\boldsymbol{a},b)| \ge \frac{1}{4}+\delta \end{cases}$$

Error:(The bound of corresponding error in TFHE is $2n(k+1)l\beta N\alpha + kNt\gamma + n(1+kN)\varepsilon + kN2^{-(t+1)}$)

$$\left\|Err(\text{KeySwitch}_{KS_{s'\to s,\gamma}}(\boldsymbol{u}))\right\|_\infty \overset{\eta' \triangleq \|Err(\boldsymbol{a}',b')\|_\infty}{\le} \|Err(\boldsymbol{u})\|_\infty + n't\gamma + n'2^{-(t+1)}$$

$$= \|Err((\boldsymbol{0},\bar{\mu}) + SampleExtract(ACC))\|_\infty + n't\gamma + n'2^{-(t+1)}$$

$$\le \left\|Err(ACC_{\frac{n}{2}})\right\|_\infty + n't\gamma + n'2^{-(t+1)}$$

$$= \sum_{i=1}^{\frac{n}{2}}(4\alpha(k+1)lN\beta + (1+kN)\varepsilon) + \|Err(ACC_0)\|_\infty + n't\gamma + n'2^{-(t+1)}$$

$$\le 2n(k+1)l\beta N\alpha + kNt\gamma + \tfrac{n}{2}(1+kN)\varepsilon + kN2^{-(t+1)}$$

Error variance: (The bound of corresponding error variance in TFHE is $2Nn(k+1)l\beta^2\vartheta_{BK} + kNtV_{KS} + n(1+kN)\varepsilon^2 + kN2^{-2(t+1)}$)

$$
\begin{aligned}
&Var(Err(\text{KeySwitch}_{KS_{s'\to s,\gamma}}(\boldsymbol{u}))) \overset{\eta'\triangleq Var(Err(\boldsymbol{a}',b'))}{\leq} Var(Err(\boldsymbol{u})) + n't\gamma^2 + n'2^{-2(t+1)} \\
&\leq \left\|Var(Err((\boldsymbol{0},\bar{\mu}) + SampleExtract(ACC)))\right\|_\infty + n't\gamma^2 + n'2^{-2(t+1)} \\
&\leq \left\|Var(Err(ACC_{\frac{n}{2}}))\right\|_\infty + n't\gamma^2 + n'2^{-2(t+1)} \\
&\leq \sum_{i=1}^{\frac{n}{2}} (4\vartheta_{BK}\left(k+1\right)lN\beta^2 + \left(1+kN\right)\varepsilon^2) + Var(Err(ACC_0)) + n't\gamma^2 + n'2^{-2(t+1)} \\
&= 2Nn(k+1)l\beta^2\vartheta_{BK} + kNtV_{KS} + \tfrac{n}{2}\left(1+kN\right)\varepsilon^2 + kN2^{-2(t+1)}
\end{aligned}
$$

3.2 Security

Based on TFHE scheme, we made a modification focusing on accelerate the bootstrapping. The main difference between our work and TFHE lies in that, we have built an enhanced homomorphic constant multiplication, EHCM. The input of TFHE constant multiplication is a TGSW ciphertext of key $TGSW(\boldsymbol{s}_i)$ and an integer $\boldsymbol{a}_i$. While in our EHCM process, the input is changed into ciphertexts of $TGSW_{\boldsymbol{s}'',\alpha}(\boldsymbol{s}_{2i-1}\boldsymbol{s}_{2i})$, $TGSW_{\boldsymbol{s}'',\alpha}(\boldsymbol{s}_{2i-1}(\boldsymbol{s}_{2i}-1)), TGSW_{\boldsymbol{s}'',\alpha}((\boldsymbol{s}_{2i-1}-1)\boldsymbol{s}_{2i}), TGSW_{\boldsymbol{s}'',\alpha}((\boldsymbol{s}_{2i-1}-1)(\boldsymbol{s}_{2i}-1))$ and two integers $\boldsymbol{a}_{2i-1}, \boldsymbol{a}_{2i}$. Basing on the circular security assumption, these ciphertexts leak no information about plaintext or key. Thus our scheme is as safe as TFHE.

3.3 Parameters

We find that the parameters of TFHE can be optimized. In detail, the precision $\varepsilon = \frac{1}{2B_g^l}$ of Algorithm 1, which is used to gadget decompose a TLWE Sample, is too tight. We recommend the parameters $l=2, B_g=512, \beta=256$ instead of $l=3, B_g=1024, \beta=512$. Other parameters, corresponding to our scheme, is $n=500$, $N=1024$, $k=1$, $\varepsilon=2^{-31}$, $\alpha=9.0\bullet10^{-9}$, $\gamma=3.05\bullet10^{-5}$, $t=15$.

Notice that Algorithm 1 output a ciphertext $\boldsymbol{u}\in\mathfrak{R}^{(k+1)l}$. Smaller l will lead to a lower dimension ciphertext $\boldsymbol{u}$, and smaller scale of TGSW ciphertext $\mathcal{M}_{(k+1)l,k+1}(\mathbb{T}_N[X])$, and a quicker multiplication between ciphertext $\boldsymbol{u}$ and TGSW ciphertext. Hundreds of multiplications, during the accumulate operations, take up most of the time of bootstrapping. The change of l will affect the efficient seriously.

Correctness of our parameter. Though our parameters will lead to larger precision $\varepsilon = \frac{1}{2B_g^l}$ when Gadget Decomposition, the smaller *l* will lead to lower dimension, which result in smaller errors. Putting all of our recommend parameters into the Theorem 4.1, we find that the final error variance after bootstrapping is 2.41×10^{-5}. It corresponds to a standard deviation of $\sigma=0.0049$, which guarantee that the noise

amplitude after our bootstrapping is less than $\frac{1}{16}$ with very high probability $erf(\frac{1}{16}\bullet 2\delta) \geq 1-2^{-54}$ (this is higher than the probability $1-2^{-33.56}$ in TFHE and $1-2^{-32}$ in FHEW).

Security of our parameter. We changed the parameters in Algorithm 1, which is used to gadget decompose a TLWE Sample. Algorithm 1 operation on the ciphertexts directly, and doesn't involve any other information about the key. So the new parameters will not decrease the security parameter level.

Comparing to TFHE, a flaw of our scheme is that the homomorphic evaluation key is increased from 52.6M into 60.5M, which is the sum of Bootstrapping Key 31.3M and Key Switching Key 29.2M. Bootstrapping Key is the size of $2nl(k+1)$ TLWE ciphertext, and a TLWE samples is almost $(k+1)\bullet N\bullet 32$ bits[21].

3.4 Implementation

Based on the code of TFHE, we implement our scheme with parameters: $n=500$, $N=1024$, $k=1$, $l=2$, $B_g=512$, $\beta=256$, $\varepsilon=2^{-31}$, $\alpha=9.0\bullet 10^{-9}$, $\gamma=3.05\bullet 10^{-5}$, $t=15$. We implemented TFHE, CGGI17 and our scheme on a 64-bit single core (i7-4930MX) at 3.00GHz, with OS Ubuntu Kylin 14.04, GNU version 6.2 and fftw version 3.3.6. We implemented these schemes in four different modes, Debug/fftw, Debug/spqlios, release/fftw, release/spqlios, which using different mathematical library. What's more, we measured the time of key generation, encryption and, bootstrapping during FHE. We didn't analyze the decryption process, since it only taking about 2 microseconds (us). Bootstrapping is the most important part that affecting efficiency of FHE. The result shows that our bootstrapping takes 52 percent of TFHE time in release/spqlios model, which is the most efficient FHE. Detail follows:

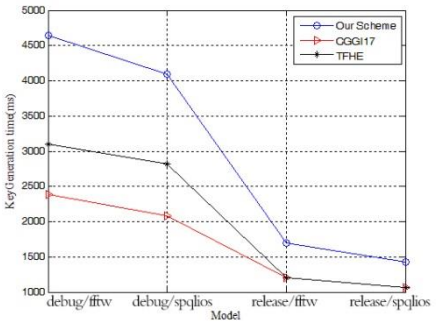

Fig. 4. The key generation time comparison

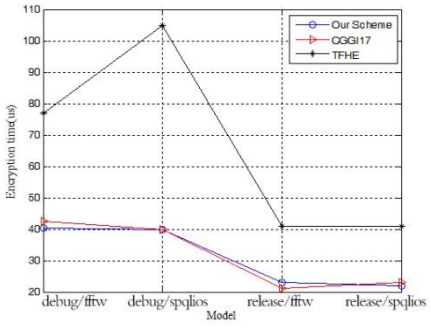

Fig. 5. The encryption time comparison

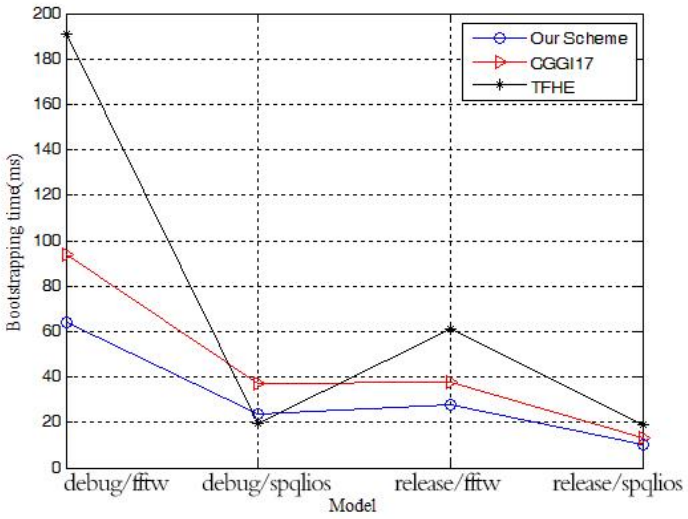

Fig. 6. The bootstrapping time comparison

Table 1. Bootstrapping time of our scheme, TFHE and CGGI17 in different modes

Bootstrapping	debug/fftw	debug/spqlios	release/fftw	release/spqlios
Our scheme	64ms	23ms	28ms	10ms
TFHE [21]	192ms	20ms	61ms	19ms
CGGI17[23]	94ms	38ms	38ms	13ms

Three addends as a group. In this paper, we let two addends $-\bar{\mathbf{a}}_i\mathbf{s}_i$ as a group. In fact, when implement we let three addends $-\bar{\mathbf{a}}_i\mathbf{s}_i$ as a group to get a quicker bootstrapping. In this situation, similar to Theorem 3.6, we construct Keybundle_i as follow, which is a TGSW sample of message $X^{-\overline{\mathbf{a}_{3i-2}}\cdot\mathbf{s}_{3i-2}-\overline{\mathbf{a}_{3i-1}}\cdot\mathbf{s}_{3i-1}-\overline{\mathbf{a}_{3i}}\cdot\mathbf{s}_{3i}}$. The flaw of this setting is that the evaluation key will be 70.9M (which is still reasonable).

$$\text{Keybundle}_i = TGSW(X^{-\overline{a_{3i-2}}\cdot s_{3i-2}-\overline{a_{3i-1}}\cdot s_{3i-1}-\overline{a_{3i}}\cdot s_{3i}}) = X^{-\overline{a_{3i-2}}-\overline{a_{3i-1}}-\overline{a_{3i}}}\cdot TGSW_{s'',\alpha}(s_{3i-2}s_{3i-1}s_{3i})$$
$$-X^{-\overline{a_{3i-2}}-\overline{a_{3i-1}}}\cdot TGSW_{s'',\alpha}(s_{3i-2}s_{3i-1}(s_{3i}-1)) - X^{-\overline{a_{3i-2}}-\overline{a_{3i}}}\cdot TGSW_{s'',\alpha}(s_{3i-2}s_{3i}(s_{3i-1}-1))$$
$$-X^{-\overline{a_{3i-1}}-\overline{a_{3i}}}\cdot TGSW_{s'',\alpha}(s_{3i-1}s_{3i}(s_{3i-2}-1)) + X^{-\overline{a_{3i-2}}}\cdot TGSW_{s'',\alpha}(s_{3i-2}(s_{3i-1}-1)(s_{3i}-1))$$
$$+X^{-\overline{a_{3i-1}}}\cdot TGSW_{s'',\alpha}(s_{3i-1}(s_{3i-2}-1)(s_{3i}-1)) + TGSW_{s'',\alpha}(X^{-\overline{a_{3i}}}\cdot s_{3i}(s_{3i-2}-1)(s_{3i-1}-1))$$
$$-TGSW_{s'',\alpha}((s_{3i-2}-1)(s_{3i-1}-1)(s_{3i}-1))$$

4 Summary and Future Directions

In this paper, an efficient fully homomorphic encryption scheme with bootstrapping in 10ms is presented. Experiment shows that comparing with the scheme in best paper of ASIACRYPT 2016, in the same parameter setting, our scheme has smaller error and error variance, and most important of all, bootstrapping of our scheme cost only 52 percent of TFHE time.

The weakness of our scheme lies in that the process of bootstrapping involves many matrix-vector multiplications which cost more memory. Our future work will focus in optimizing of multiple bootstrappings. Moreover, the EHCM in our scheme can also be done multiple addends $-\bar{\boldsymbol{a}}_i \boldsymbol{s}_i$ on a group, but it leads to error increase. In future research, we want to solve this problem, and thus improve efficiency of FHE.

This work was supported by National Key Research and Development Program of China under Grants No. 2017YFB0802000, National Natural Science Foundation of China (Grant Nos. U1636114,61772550,61572521), Natural Science Basic Research Plan in Shaanxi Province of China (Grant Nos. 2016JQ6037), State Key Laboratory of Information Security (2017-MS-18).

References

1. Gentry C. Fully homomorphic encryption using ideal lattices [C]//Proc of the 41st Annual ACM Symposium on Theory of Computing, STOC. New York: ACM, 2009: 169-178
2. Van Dijk M, Gentry C, Halevi S, et al. Fully homomorphic encryption over the integers [G] // LNCS 6110: Advances in Cryptology–EUROCRYPT 2010. Berlin: Springer, 2010: 24-43
3. Brakerski Z, Vaikuntanathan V. Efficient fully homomorphic encryption from (standard) LWE [C]// Proc of 52nd Annual Symp on Foundations of Computer Science. Los Alamitos,CA:IEEE Computer Society, 2011: 97-106
4. Zhou T, Yang X, Zhang W, et al. Efficient fully homomorphic encryption with circularly secure key switching process[J]. International Journal of High Performance Computing and Networking, 2016, 9(5-6): 417-422.
5. Brakerski Z, Gentry C, Vaikuntanathan V. (Leveled) fully homomorphic encryption without bootstrapping [C]//Proc of the 3rd Innovations in Theoretical Computer Science Conf. New York: ACM, 2012: 309-325
6. Gentry C, Sahai A, Waters B. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based [G] // LNCS 8042: Advances in Cryptology–CRYPTO 2013. Berlin: Springer, 2013: 75-92
7. Brakerski Z, Vaikuntanathan V. Lattice-based FHE as secure as PKE. In ITCS, pages 1,2014.
8. Barrington D. Bounded-width polynomial-size branching programs recognize exactly those languages in NC1. In STOC, pages 1–5. 1986.
9. Alperin-Sheriff J , Peikert C. Practical bootstrapping in quasilinear time. In CRYPTO 2013. 2013: 1–20.
10. Alperin-Sheriff J, Peikert C. Faster bootstrapping with polynomial error[M]//Advances in Cryptology–CRYPTO 2014. Springer Berlin Heidelberg, 2014: 297-314.

11. Halevi S and Shoup V. Bootstrapping for HElib. IACR Cryptology ePrint Archive, 2014. http://eprint.iacr.org/2014/873.
12. Gentry C, Halevi S, and N. P. Smart. Better bootstrapping in fully homomorphic encryption. In M. Fischlin,J. Buchmann, and M. Manulis, editors, PKC 2012, volume 7293 of LNCS, pages 1–16. Springer, May 2012.
13. Cheon J H, Kim M, Song Y S. Secure Searching of Biomarkers Using Hybrid Homomorphic Encryption Scheme[J]. IACR Cryptology ePrint Archive, 2017, 2017: 294.
14. Petrlic R, Sekula S, Sorge C. A privacy-friendly architecture for future cloud computing[J]. International Journal of Grid and Utility Computing 26, 2013, 4(4): 265-277.
15. Zhu X D, Li H, Li F H. Privacy-preserving logistic regression outsourcing in cloud computing[J]. International Journal of Grid and Utility Computing, 2013, 4(2-3): 144-150.
16. Guo S, Xu H. A secure delegation scheme of large polynomial computation in multi–party cloud[J]. International Journal of Grid and Utility Computing, 2014, 6(1): 1-7.
17. Chasaki D, Mansour C. Security challenges in the internet of things[J]. International Journal of Space-Based and Situated Computing, 2015, 5(3): 141-149.
18. Indra G, Taneja R. A time stamp-based elliptic curve cryptosystem for wireless ad-hoc sensor networks[J]. International Journal of Space-Based and Situated Computing 27, 2014, 4(1): 39-54.
19. Meng N, Wang J, Kodama E, et al. Reducing data leakage possibility resulted from eavesdropping in wireless sensor network[J]. International Journal of Space-Based and Situated Computing 6, 2013, 3(1): 55-65.
20. Ducas L, Micciancio D. FHEW: Bootstrapping Homomorphic Encryption in less than a second[M]//Advances in Cryptology--EUROCRYPT 2015. Springer Berlin Heidelberg, 2015: 617-640.
21. Chillotti I, Gama N, Georgieva M, et al. Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds[C]//Advances in Cryptology–ASIACRYPT 2016: 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part I 22. Springer Berlin Heidelberg, 2016: 3-33.
22. Regev O. On lattices, learning with errors, random linear codes, and cryptography. In *STOC*, pages 84–93, 2005.
23. Chillotti I, Gama N, Georgieva M, et al. Improving TFHE: faster packed homomorphic operations and efficient circuit bootstrapping[J].