Optimizing Bootstrapping and Evaluating Large FHE Gates in the LWE-based GSW-FHE

Chao Liu[1], Anyu Wang[2], and Zhongxiang Zheng[2]

[1] School of Cyber Science and Technology, Shandong University
[2] Institute for Advanced Study/BNRist, Tsinghua University
liu_chao@mail.sdu.edu.cn,
anyuwang@mail.tsinghua.edu.cn,zhengzx13@tsinghua.org.cn

Abstract. *Fully homomorphic encryption* (FHE) allows us to perform computations directly over encrypted data and can be widely used in some highly regulated industries. Gentry's bootstrapping procedure is used to refresh noisy ciphertexts and is the only way to achieve the goal of FHE up to now. In this paper, we optimize the LWE-based GSW-type bootstrapping procedure. Our optimization decreases the lattice approximation factor for the underlying worst-case lattice assumption from $\tilde{O}(N^{2.5})$ to $\tilde{O}(N^2)$, and is time-efficient by a $O(\lambda)$ factor. Our scheme can also achieve the best factor in prior works on bootstrapping of standard lattice-based FHE by taking a larger lattice dimension, which makes our scheme as secure as the standard lattice-based PKE. Furthermore, in this work we present a technique to perform more operations per bootstrapping in the LWE-based FHE scheme. Although there have been studies to evaluate large FHE gates using schemes over ideal lattices, (i.e. using FHEW or TFHE), we are the first to study how to perform complex functions homomorphically over standard lattices.

Keywords: Fully homomorphic encryption · GSW-FHE · LWE-based · Large FHE gates

1 Introduction

Fully homomorphic encryption (FHE) allows us to evaluate arbitrary computations over encrypted data by only using public information. In 2009, Gentry [20] proposed the first construction for a FHE scheme. A lot of effort has been made (e.g. [8,9,5,7,21,10,1,18,13], etc.) to push FHE toward practicality following Gentrys blueprint. Among those FHE schemes, there are LWE-based schemes, e.g. the scheme in [8,5,21,10,1,22]. One advantage of such schemes is the high-security strength. LWE can be reduced to some worst-case lattice problems on general lattices (algebraically unstructured lattices), and the research focus of this kind of schemes is not only on the efficiency improvement but also on the security strength of the scheme, that is, the improvement of the approximation parameters of the underlying worst-case lattice assumption. For example, in the existing LWE-based schemes, some schemes can achieve the same security strength as the standard PKE schemes (i.e. the approximate factor can be *small* polynomial), e.g. the scheme in [10,1,22]. Meanwhile, RLWE can be reduced to some worst-case lattice problems on ideal lattices (algebraically structured lattices), and RLWE-based FHE such as [7,19,12,18,13] has been widely studied because of its advantages in terms of efficiency.

Compared with the Boolean gates, some complex operations (referred to as large FHE gates) such as the Look Up Table (LUT) function or max/min functions are harder to perform in FHE. In order to efficiently evaluate those large FHE gates, some special algebraic structures are needed. In the RLWE setting, the technologies to evaluate large FHE gates is gradually mature[14,3,11,15], but there are no similar technical researches on the LWE-based bootstrapping scheme. Without a doubt, the LWE-based FHE scheme is difficult to implement in the real-world (with enormous storage consumption and slow efficiency), and it is often used as a frontier theoretical research. But the research on LWE-based FHE scheme is essential, as the researches on LWE-based schemes often stimulate follow-up research. For example, some LWE-based FHE scheme, like Brakerski et al.'s schemes [8,7] and Gentry et al.'s scheme [21] are very important works in the field of FHE. Furthermore, the algebraically unstructured lattice seems to be essentially different from the structured lattice in quantum computing. Some recent works [16,17] has given a quantum polynomial-time algorithm for very large but subexponential $2^{\tilde{O}(\sqrt{n})}$ approximations to the worst-case Shortest Vector Problem on ideal lattices, (in contrast to just slightly subexponential $2^{O(nloglogn/logn)}$ factors obtainable for algebraically unstructured lattice [23]). So the motivation of our work is to optimize the LWE-based FHE scheme and to study how to evaluate large FHE gates in the LWE setting.

Up to now, one of the fastest and simplest LWE-based FHE arose from the GSW scheme by Gentry, Sahai and Water [21] (referred to as GSW-FHE). Gentry, Sahai and Water's construction avoids the expensive "relinearization" step in homomorphic multiplication [8,7], which makes the GSW scheme supports a different class of functions. Brakerski and Vaikuntanathan [10] showed that the GSW scheme supports branching programs and it is sufficient to bootstrap the GSW to FHE by using Barringtons theorem [2]. The approximation factor of Brakerski and Vaikuntanathans FHE decreases from super-polynomial to polynomial (i.e. $\tilde{O}(N^{1.5+\epsilon})$ for $\epsilon > 0$, but at a great cost in runtime and space), hence obtained an FHE scheme as secure as the standard lattice-based PKE. Alperin-Sheriff and Peikert [1] introduced a new method of constructing FHE that can avoid the costly use of Barrington's transformation in Brakerski and Vaikuntanathan's construction. They found that one can view the decryption as an arithmetic circuit and the inner product in the decryption can be computed using a group of cyclic permutations. By this property, Alperin-Sheriff and Peikert constructed a bootstrapping procedure that can refresh ciphertexts faster than Brakerski and Vaikuntanathan's scheme, with a slightly stronger underlying security assumption (the approximate factor is $\tilde{O}(N^3)$, but a great improvement of the runtime). Hiromasa, Abe and Okamoto [22] presented a technique to encrypt matrices in GSW encryption and showed how to homomorphically operate matrices addition and multiplication. They used this technique to optimize Alperin-Sheriff and Peikert's bootstrapping scheme. Their optimization scheme is time and space-efficient and the lattice approximation factor is decreased to $\tilde{O}(N^{2.5})$. Then the latter works about the GSW-FHE are mainly RLWE-based schemes, including Ducas and Micciancio's scheme FHEW [18] and Chillotti et al.'s scheme TFHE [13,14]. In this paper, we aim to optimize the GSW-type LWE-based bootstrapping scheme. In terms of safety and efficiency, the optimal LWE-based GSW-FHE scheme is Hiromasa, Abe and Okamoto's scheme. Their scheme supports homomorphic matrix multiplication, and this property can be used to evaluate the linear operation in the homomorphic decryption. But homomorphic matrix multiplication is not optimal for bootstrapping.

1.1 Our Works

We have two contributions in this work:

- We propose a new homomorphic matrix-vector multiplication operation. Although the GSW encryption packing technology for matrix[22] and LWE encryption packing technology for vector [27,6] have been proposed before, no one has done further researches about the relation of these two encryption structures. Here we find that these two kinds of encryption can be combined to construct a homomorphic matrix-vector multiplication operation, which is more efficient than the homomorphic matrix multiplication by Hiromasa, Abe and Okamoto. We use this operation to construct the linear operation in the bootstrapping technique and proposed a new LWE-based GSW-type bootstrapping scheme that performs better than Hiromasa, Abe and Okamoto's work in safety and efficiency.
- We are the first to study how to perform more operates per bootstrapping for the LWE-based bootstrapping scheme. Bootstrapping technology originally was used for homomorphic decryption[20], but later it was found that bootstrapping can be used to perform some Boolean gates in the RLWE-based schems[18,14]. The key to the realization of this technology is to use the *nega-cyclic* function property on the ring structure. Furthermore, in [14,3,11,15], there are works to use the special structure of the ring to evaluate some complex operations, such as LUT functions and max/min operations. But there are no similar researches on LWE-based bootstrapping schemes before, and it is unknown whether similar functions (i.e. Boolean gates and large FHE gates) can be realized in the LWE setting. In this work, we give an exact answer. By using the matrix-vector multiplication and a "cyclic rotation" property of the vector, our scheme can evaluate Boolean gates and some large FHE gates. We say "functional bootstrapping"[4], it means that to evaluate a function during bootstrapping. We show how to homomorphically compute a r-to-v LUT function with v functional bootstrapping; homomorphically compute a max/min function or a comparison function for two small integers with only one functional bootstrapping.

Finally, we propose an LWE-based GSW-type bootstrapping scheme that can evaluate large FHE gates, at the same time our scheme is secure assuming the hardness of approximating the standard lattice problem to within the factor $\tilde{O}(N\lambda)$ on any N dimensional lattices. When choosing $N = \Theta(\lambda)$ for 2^{λ} hardness, this yields an approximation factor of $\tilde{O}(N^2)$ for the underlying worst-case lattice assumption. Compared to Hiromasa et al's work [22], our scheme decreases the lattice approximation factor from $\tilde{O}(N^{2.5})$ to $\tilde{O}(N^2)$, and is time-efficient by a $O(\lambda)$. By choosing the dimension to be $N = \lambda^{1/\epsilon}$ for $\epsilon > 0$, we obtain a factor as small as $\tilde{O}(N^{1.5+\epsilon/2})$ (i.e. the same factor as in Brakerski and Vaikuntanathan's scheme, but with a much smaller runtime and space). Since the standard lattice-based public-key encryption can be based on the hardness of approximating the problem to $\tilde{O}(N^{1.5})$ [28], our bootstrapping scheme can be as secure as the standard lattice-based PKE.

1.2 Our Techniques

The goal of bootstrapping is to decrypt an LWE ciphertext $(\boldsymbol{a}, b) \in \mathbb{Z}_q^{n+1}$ homomorphically. There are two processes for decryption. One is the linear operation, i.e. $b - \langle \boldsymbol{a}, \boldsymbol{s} \rangle \in \mathbb{Z}_q$, where $\boldsymbol{s}$ is the secret key (usually sampled from Gauss distribution), the other is the non-linear operation, i.e. the rounding operation $\lfloor \cdot \rceil_2$, which output 1 if the input is close to $q/2$ and 0 otherwise. For the linear operation, we need to compute additions in $\mathbb{Z}_q$ homomorphically. The additive group $\mathbb{Z}_q$ is isomorphic to a group of cyclic permutation. For any x in $\mathbb{Z}_q$, it corresponds to a cyclic permutation which can be represented by an indicator vector with 1 in the $x+1$-th position. The permutation matrix can be obtained from the cyclic rotation of the indicator vector, and the addition in $\mathbb{Z}_q$ leads to the multiplication of the corresponding permutation matrices. Note that there is an efficient way to multiply two permutation matrices by multiplying one permutation matrix with the first column of the other matrix, and our first technique is an efficient method to homomorphically compute the matrix-vector product. We show that the GSW-type matrix packing ciphertext [22] and the LWE-type vector packing ciphertext [27,6] can fit together to construct a homomorphic matrix-vector multiplication.

Homomorphic matrix-vector multiplication. We first recall the matrix packing techniques by Hiromasa, Abe and Okamoto [22] and vector packing techniques by Peikert et al.[27,6].

- GSW-type Matrix Packing[22]. Given a secret key matrix $\mathbf{S} \in \mathbb{Z}_Q^{r \times N}$ and a fixed "gadget" matrix $\mathbf{G} \in \mathbb{Z}_Q^{(N+r)\times(N+r)\cdot l}$ where $l = \lceil \log_2 Q \rceil$, a matrix packing GSW encryption for message matrix $\mathbf{M} \in \{0,1\}^{r\times r}$ is:

$$MatGSW_{\mathbf{S}}(\mathbf{M}) = \left(\frac{\mathbf{A}}{\mathbf{SA}+\mathbf{E}}\right) + \left(\frac{\mathbf{0}}{-\mathbf{MS}||\mathbf{M}}\right) \cdot \mathbf{G} \in \mathbb{Z}_Q^{(N+r)\times(N+r)\cdot l}$$

  where $\mathbf{A} \in \mathbb{Z}_Q^{N\times(N+r)\cdot l}$ is uniformly sampled and $\mathbf{E} \in \mathbb{Z}^{r\times(N+r)\cdot l}$ is a noise matrix. Let $\mathbf{SK} = [-\mathbf{S}||\mathbf{I}_r] \in \mathbb{Z}_Q^{r\times(N+r)}$, where $\mathbf{I}_r$ is the $r \times r$ identity matrix. For any $\mathbf{C} = MatGSW_{\mathbf{S}}(\mathbf{M})$, there is $\mathbf{SK}\cdot\mathbf{C} = \mathbf{E} + \mathbf{M}\cdot\mathbf{SK}\cdot\mathbf{G}$.

- LWE-type vector packing [27,6]. Given a secret key matrix $\mathbf{S} \in \mathbb{Z}_Q^{r \times N}$, for a message vector $\boldsymbol{m} \in \mathbb{Z}_Q^r$, a vector packing LWE encryption:

  $$VecLWE_{\mathbf{S}}(\boldsymbol{m}) = \left(\frac{\boldsymbol{a}}{\mathbf{S}\boldsymbol{a} + \boldsymbol{e} + \boldsymbol{m}} \right) \in \mathbb{Z}_Q^{N+r}$$

  where $\boldsymbol{a} \in \mathbb{Z}_Q^N$ is uniformly sampled and $\boldsymbol{e} \in \mathbb{Z}^r$ is a small noise vector. Let $\mathbf{SK} = [-\mathbf{S}||\mathbf{I}_r] \in \mathbb{Z}_Q^{r\times(N+r)}$. For any $\boldsymbol{c} = VecLWE_{\mathbf{S}}(\boldsymbol{m})$, there is $\mathbf{SK} \cdot \boldsymbol{c} = \boldsymbol{e} + \boldsymbol{m}$.

In this paper we show an operation that combine above two packing techniques. For a given vector $\boldsymbol{c} \in \mathbb{Z}_q^{N+r}$, let $\mathbf{G}^{-1}(\boldsymbol{c})$ be the "decomposition" function that output an "entries small" vector $\mathbf{x} \in \mathbb{Z}_Q^{(N+r)l}$ such that $\mathbf{Gx} \equiv \mathbf{c}(\text{mod } Q)$. For a $\mathbf{C} = MatGSW_{\mathbf{S}}(\mathbf{M}_0 \in \{0,1\}^{r\times r})$ with small noise matrix $\mathbf{E}$, and a $\boldsymbol{c} = VecLWE_{\mathbf{S}}(\boldsymbol{m}_1 \in \mathbb{Z}_Q^r)$ with small noise vector $\boldsymbol{e}$, by above definitions about MatGSW encryption and VecLWE encryption, a ciphertext $\boldsymbol{c}_{mult} = \mathbf{C} \cdot \mathbf{G}^{-1}(\boldsymbol{c})$ satisfies

$$\begin{aligned}
\mathbf{SK} \cdot \boldsymbol{c}_{mult} &= \mathbf{SK} \cdot \mathbf{C} \cdot \mathbf{G}^{-1}(\boldsymbol{c}) \\
&= (\mathbf{E} + \mathbf{M}_0 \cdot \mathbf{SK} \cdot \mathbf{G}) \cdot \mathbf{G}^{-1}(\boldsymbol{c}) \\
&= \mathbf{E} \cdot \mathbf{G}^{-1}(\boldsymbol{c}) + \mathbf{M}_0 \cdot \mathbf{SK} \cdot \boldsymbol{c} \\
&= (\mathbf{E} \cdot \mathbf{G}^{-1}(\boldsymbol{c}) + \mathbf{M}_0 \cdot \boldsymbol{e}) + \mathbf{M}_0 \cdot \boldsymbol{m}_1
\end{aligned}$$

where $(\mathbf{E} \cdot \mathbf{G}^{-1}(\boldsymbol{c}) + \mathbf{M}_0 \cdot \boldsymbol{e})$ is small, and this means that the $\boldsymbol{c}_{mult}$ is a VecLWE encryption of message vector $\mathbf{M}_0 \cdot \boldsymbol{m}_1 \in \mathbb{Z}_Q^r$. Therefore we have a homomorphic matrix-vector multiplication operation:

$$MatGSW(\mathbf{M}_0) \times VecLWE(\boldsymbol{m}_1) \to VecLWE(\mathbf{M}_0 \cdot \boldsymbol{m}_1). \tag{1}$$

We will use operation (1) to construct our bootstrapping procedure, which speeds up the homomorphic matrix multiplication by a factor $(N + r) \cdot l$ compared with using the operation (a homomorphic matrix-matrix multiplication operation) in scheme [22].

Computing non-linear function. Our second technique is a new way to homomorphically compute the non-linear function. In previous work [1,22], one can compute the rounding function by summing the entries of the indicator vector corresponding to those values in $\mathbb{Z}_q$. In this work we compute the non-linear function in a completely different way.

Our work is inspired by the calculation of nonlinear operation in FHEW and TFHE scheme. In their schemes, the underly ring is $R_q = \mathbb{Z}_q[X]/\langle X^N + 1\rangle$, where N is a power of 2. First, notice that the roots of unity $\langle X\rangle = \{1, X, \ldots, X^{N-1}, -1, \ldots, -X^{N-1}\}$ form a cyclic group, and when setting $q = 2N$, the message space $\mathbb{Z}_q \simeq \langle X\rangle$. So to evaluate a non-linear function $F : \mathbb{Z}_q \to \mathbb{Z}_t$, one can initialize a polynomial

$$acc = \Delta \cdot (F(b) + F(b-1)X + \ldots + F(b-N+1) \cdot X^{N-1}),$$

where $\varDelta$ is an encoding constant. To compute the linear operation, for a sample example, to compute $F(b+2)$, one can homomorphically compute

$$\begin{aligned} acc\cdot X^2 =&\varDelta\cdot(F(b)\cdot X^2+F(b-1)\cdot X^3+\ldots+F(b-N+1)\cdot X^{N+1})\\ =&\varDelta\cdot(-F(b-N+2)-F(b-N+1)\cdot X+F(b)\cdot X^2\\ &+\ldots+F(b-N+3)\cdot X^{N-1}). \end{aligned}$$

(X^2 is encrypted, so this step is executed homomorphically). Then if F satisfy $F(x+N)=-F(x)$, i.e. a *negacyclic* property, one can derive the first coefficient to obtain result $F(b+2)$ in their schemes.

We found there are similar property in the LWE setting. In our scheme, note that in operation (1), if $\mathbf{M}_0$ is a cyclic permutation matrix, $\mathbf{M}_0\cdot\boldsymbol{m}_1$ is a "cyclic rotation" of $\boldsymbol{m}_1$, so we can set $\boldsymbol{m}_1$ as a special vector and use the rotation property to compute the the non-linear function. More detailed, we initialize $\boldsymbol{m}_1$ to be

$$\boldsymbol{m}_1 := \varDelta\cdot(F([b]_q),F([b-1]_q),\ldots,F([b-q+1]_q))$$

where F is a known function and $\varDelta$ is an encoding constant. Assume that $\mathbf{M}_0$ is the permutation corresponds to $\phi(-a_i\cdot s_i)\in S_q$ where $-a_i\cdot s_i\in\mathbb{Z}_q$ and ϕ is the isomorphism of an element in $\mathbb{Z}_q$ into the cyclic permutation (see Section 2.3 for a better understanding of ϕ), then after operation (1), the result is a VecLWE ciphertext that encrypts

$$\mathbf{M}_0\cdot\boldsymbol{m_1}=\varDelta\cdot(F([b-a_i\cdot s_i]_q),\ldots,F([b-q+1-a_i\cdot s_i]_q)).$$

Then for every $i\in\{1,\ldots,n\}$, by iteratively computing operation (1) for every permutation matrix corresponding to $-a_i\cdot s_i$, we can obtain an LWE ciphertext that decrypts to the message $F([b-\langle\boldsymbol{a},\boldsymbol{s}\rangle]_q)$, which is a decryption for $(\boldsymbol{a},b)$ when we set F as the rounding function (this LWE ciphertext can be extracted from the first LWE element of the final VecLWE ciphertext).

Note that the function that can be evaluated in our scheme didn't need to be "negacyclic" (i.e. $F(x+N)=-F(x)$), so we can set $F:=func\circ f$ where $f:\mathbb{Z}_q\to\mathbb{Z}_t$ is the rounding function (f is $\lfloor\cdot\rceil_2$ when $t=2$) and $func:\mathbb{Z}_t\to\mathbb{Z}_h$ is an arbitrarily given function to evaluate large FHE gates. Except for some Boolean gates, we also show how to evaluate LUT function, max/min function and comparison in this paper[3].

1.3 Related Works

Some studies focus on evaluating large FHE gates in the existed works. A Look Up Table (LUT) is an array that replaces runtime computation with a simpler

[3] The correctness can be verified at https://github.com/LiuChaoCrypto/MatGSWscheme. This implementation can perform decryption and some Boolean gates homomorphically. Because of the huge storage and time consumption of this kind of LWE-based FHE, we use a very small parameter to verify the correctness. Also for this reason, it is only for the correctness verification, but not for the performance testing.

array indexing operation. A boolean LUT is defined as $f_{LUT} : \mathbb{Z}_2^r \to \mathbb{Z}_2^v$, and for the i-bit output function we can define it to be $f_i : \mathbb{Z}_2^r \to \mathbb{Z}_2$. Chillotti et al. [14] applied a special packing technique to construct the CMux tree for the LUT function f_i, and they also constructed a weighted automata to evaluate arithmetic operations such as max function and multiplication. Compared with Chillotti et al. [14]'s strategy, our method to evaluate LUT function is more sample. Since the $func$ in our scheme can be set as an arbitrary function, we can set the LUT function f_i as the bootstrapping function $func$, and for max/min, it just needs to set $func$ specifically. Bonnoron et al. [3] improved the FHEW scheme [18] and introduced to perform the linear-step in a CRT fashion to evaluate large FHE gates. Thanks to the special structure of ring $\mathbb{Z}[x]/\langle X^N - 1\rangle$, the function $func$ be bootstrapped in Bonnoron et al.'s scheme can also be arbitrary. This is the same with the function $func$ in our scheme, but since we construct the bootstrapping in the LWE setting, the underly algebraic structure is totally different (no special structure of rings $\mathbb{Z}[x]/\langle X^N - 1\rangle$). Carpov et al. [11] optimized the TFHE scheme [13] and showed how to homomorphically perform operations on multi-value inputs. Carpov et al.'s strategy is to set a special test polynomial in the TFHE scheme for a given operation like LUT, so this strategy is also different from the method in our scheme. In [15], Chillotti et al. presented a new technique called programmable bootstrapping, which enables the homomorphic evaluation of any function of a ciphertext. Compared with Carpov et al.'s work, Chillotti et al. encoded a LUT function in a test polynomial in a different way. The above existed works rely heavily on the ring structure in the RLWE-based scheme, but for the LWE setting, there are no works before.

1.4 Organization

In Section 2, we describe some preliminaries about subgaussian distribution and the symmetric groups. In Section 3, we present the matrix/vector packing techniques and then describe how to homomorphically operate matrix-vector multiplication. We present our optimized FHE scheme in Section 4, and then give the analysis of our scheme. In the final we conclude and discuss in Section 6.

2 Preliminaries

Let $[N] = \{1, \dots, N\}$, where N is a nonnegative integer. We denote $\mathbb{Z}_Q = \mathbb{Z}/Q\mathbb{Z}$ as the quotient ring of integers modulo Q, and $(\mathbb{Z}_Q, +)$ its additive group. Sometimes we write $x \bmod Q$ as $[x]_Q$.

In this paper, we assume that vectors are in lower-case letters and matrices are in bold capital letters, unless otherwise noted. Usually, We assume that the vector $\mathbf{v} = (v_1, v_2, \dots, v_N)$ is in column form, and denote its transpose as $\mathbf{v}^T = [v_1, v_2, \dots, v_N]$. For vectors (matrices) $\mathbf{m}_1, \mathbf{m}_2, \dots, \mathbf{m}_N$, we denote the horizon concatenation of those vectors as $\mathbf{M} = [\mathbf{m}_1, \mathbf{m}_2, \dots, \mathbf{m}_N]$, and the vertical concatenation as $\mathbf{M}^T = (\mathbf{m}_1^T, \mathbf{m}_2^T, \dots, \mathbf{m}_N^T)$. We denote the l_2 norm of vector $\boldsymbol{v}$ by $||\boldsymbol{v}||_2$ and the l_∞ by $||\boldsymbol{v}||_\infty$. We denote $\mathbf{I}_N$ as the $N \times N$ identity matrix. Suppose χ is a probability distribution, $x \overset{\$}{\leftarrow} \chi$ means the sampling of x according to χ, and $x \overset{\$}{\leftarrow} U(\mathbb{Z}_Q)$ means that sample x from $\mathbb{Z}_Q$ uniformly.

2.1 Learning with Errors

The learning with errors (LWE) assumption was introduced by Regev [28], and we state its definition (decision version) in the following:

Definition 1 (DLWE). *For a security parameter λ, let $N = N(\lambda)$ be an integer dimension, $Q = Q(\lambda) \geq 2$ be an integer modulus, and $\chi = \chi(\lambda)$ be an error distribution over $\mathbb{Z}$. Given two distribution: In the first distribution, one draws $\boldsymbol{s} \overset{\$}{\leftarrow} U(\mathbb{Z}_Q^N)$, samples $\boldsymbol{a} \overset{\$}{\leftarrow} U(\mathbb{Z}_Q^N)$ and $e_i \overset{\$}{\leftarrow} \chi$, then a tuple $(\boldsymbol{a}_i, b_i)$ is sampled, where $b_i = \langle \boldsymbol{a}_i, \boldsymbol{s} \rangle + e_i$. In the second distribution, one samples $(\boldsymbol{a}_i, b_i)$ uniformly from $\mathbb{Z}_Q^{N+1}$. The $DLWE_{N,Q,\chi}$ problem is to distinguish those two distribution, and the $DLWE_{N,Q,\chi}$ assumption is that $DLWE_{N,Q,\chi}$ problem is infeasible.*

Given a lattice dimension parameter N and a number b, the $\mathbf{GapSVP}_\gamma$ problem is that to distinguish whether a N-dimensional lattice has a vector shorter than b or no vector shorter than $\gamma(N) \cdot b$. The $\mathbf{SIVP}_\gamma$ problem is to find the set of short linearly independent vectors in a lattice.

The $DLWE_{N,Q,\chi}$ problem has reductions to standard lattice assumptions as follows. The reductions take χ as a discrete Gaussian distribution $D_{\mathbb{Z},\alpha Q}$, which is centered around 0 and has parameter αQ for some $\alpha < 1$.

Theorem 1 ([24,25,26,28]). *Let $Q = Q(N) \geq 2$ be a power of prime $Q = p^r$ or a product of distinct prime numbers $Q = \Pi_i q_i (q_i = poly(N))$, and let $\alpha \geq \sqrt{N}/Q$. If there exists an efficient algorithm that solves (average-case) $DLWE_{N,Q,D_{\mathbb{Z},\alpha Q}}$, then:*

- *there exists an efficient quantum algorithm that can solve $\boldsymbol{GapSVP}_{\tilde{O}(N/\alpha)}$ and $\boldsymbol{SIVP}_{\tilde{O}(N/\alpha)}$ in the worst-case for any N-dimensional lattices.*
- *if $Q \geq \tilde{O}(2^{N/2})$, there exists an efficient classical algorithm that can solve $\boldsymbol{GapSVP}_{\tilde{O}(N/\alpha)}$ in the worst-case for any N-dimensional lattices.*

2.2 Subgaussian Random Variables

A real random variable X is subgaussian with parameter s if for all $x \in \mathbb{R}$, its (scaled) moment-generating function satisfies $\mathbb{E}[exp(2\pi x X)] \leq exp(\pi s^2 x^2)$. Any B-bounded centered random variable X is subgaussian with parameter $B \cdot \sqrt{2\pi}$.

There are two useful properties for subgaussian random variables:

- Homogeneity: if X is subgaussian with parameter s, then $t \cdot X$ is subgaussian with parameter $t \cdot s$.
- Pythagorean additivity: if X_1 is subgaussian with parameter s_1, and X_2 is subgaussian with parameter s_2, then $X_1 + X_2$ is subgaussian with parameter $\sqrt{s_1^2 + s_2^2}$.

For a real random vector $\boldsymbol{v}$, we say it is subgaussian with parameter s if for all real unit vectors $\boldsymbol{u}$, their marginal $\langle \boldsymbol{u}, \boldsymbol{v} \rangle$ is subgaussian with parameter s. If one vector is the concatenation of subgaussian variables or vectors, each of which has a parameter s and is independent of the prior one, then it is also subgaussian with parameter s. The two properties homogeneity and Pythagorean additivity also hold from the linearity of vectors. There is also a useful lemma for the Euclidean norm of the subgaussian random vector.

Lemma 1 ([29]). *Let $\boldsymbol{v} \in \mathbb{R}^N$ be a random vector with independent coordinates which are subgaussian with parameter s. Then we have $Pr[||\boldsymbol{v}||_2 > C \cdot s\sqrt{N}] \leq 2^{-\Omega(N)}$ where C is some universal constant.*

Alperin-Sheriff and Peikert [1] introduced to apply the randomized "decomposition" function $\mathbf{G}^{-1}$ instead of the decomposition procedure and we make a sample description here. For a module Q, let $\boldsymbol{g} = (1, 2, \ldots, 2^{l-1})$ where $l = \lceil log_2 Q \rceil$, and $\mathbf{G} = \boldsymbol{g}^T \otimes \mathbf{I}_N$ is the block matrix with N copies of $(1, 2, \ldots, 2^{l-1})^T$ as diagonal blocks, and zeros elsewhere. Define a randomized "decomposition" function $\boldsymbol{g}^{-1} : \mathbb{Z}_Q \to \mathbb{Z}_2^l$ for $c \in \mathbb{Z}_Q$ such that $\boldsymbol{g}^{-1}(c)$ is subgaussian with parameter $O(1)$ and $\langle \boldsymbol{g}^{-1}(c), \boldsymbol{g} \rangle = c$. Note that for $c = \sum_{i \in [l]} c_i 2^{i-1}$, $\boldsymbol{g}^{-1}(c)$ can be $(c_1, \ldots, c_l)$. Similarly, for vectors and matrices, we can by applying $\boldsymbol{g}$ independently to each entry and define the randomized function $\mathbf{G}^{-1} : \mathbb{Z}_Q^{N \times m} \to \mathbb{Z}_2^{N \cdot l \times m}$ such that $\mathbf{G} \cdot \mathbf{G}^{-1}(\mathbf{A}) = \mathbf{A}$ where $\mathbf{A} \in \mathbb{Z}_Q^{N \times m}$.

Lemma 2 ([1]). *There is a randomized efficiently computable function $\mathbf{G}^{-1} : \mathbb{Z}_Q^N \to \mathbb{Z}_2^{N \cdot \lceil \log Q \rceil}$ such that for any $\boldsymbol{v} \in \mathbb{Z}_Q^N$, $\boldsymbol{x} \leftarrow \mathbf{G}^{-1}(\boldsymbol{v})$ is subgaussian with parameter $O(1)$ and $\mathbf{G}\boldsymbol{x} = \boldsymbol{v}$.*

2.3 Symmetric Groups and $\mathbb{Z}_q$-Embeddings

Alperin-Sheriff and Peikert [1] observed that the additive group $\mathbb{Z}_q$ can embed (i.e., has an injective homomorphism) into the symmetric group S_q, and they use this property to introduce their efficient bootstrapping algorithm. We describe this property here. Denote S_q as the symmetric group of order q, i.e., the group of permutations (bijections) $\pi : \{1, \ldots, q\} \to \{1, \ldots, q\}$ with function composition as the group operation. By the injective homomorphism that sends the generator $1 \in \mathbb{Z}_q$ to the "cyclic shift" permutation π in S_q, where $\pi(i) = i + 1$ for $0 < i < q$ and $\pi(q) = 1$, the additive cyclic group $(\mathbb{Z}_q, +)$ can embed into the symmetric group S_q. Besides, for the multiplicative group of q-by-q permutation matrix, there is a map that associates the element π in S_q with the permutation matrix $\mathbf{P}_\pi = [\boldsymbol{u}_{\pi(1)}, \ldots, \boldsymbol{u}_{\pi(q)}]$, where $\boldsymbol{u}_i$ is the i-th standard basis vector, and this means that S_q is isomorphic to the multiplicative group of q-by-q permutation matrices. In the final, the addition in $\mathbb{Z}_q$ leads to the multiplication of the corresponding permutation matrices.

$\mathbb{Z}_q$	0	1	$\cdots$	$q-1$
Corresponding permutation matrices	$\begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix}$	$\begin{pmatrix} 0 & \cdots & 0 & 1 \\ 1 & \cdots & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 1 & 0 \end{pmatrix}$	$\cdots$	$\begin{pmatrix} 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \\ 1 & 0 & \cdots & 0 \end{pmatrix}$

3 Homomorphic Matrix-vector Multiplication

In this section, we give some definitions for LWE encryption and the vector/matrix packing encryption. Then we introduce the homomorphic matrix-vector multiplication which will be used in our bootstrapping scheme.

3.1 Definitions

We give definitions for LWE, VecLWE and MatGSW encryptions.

- **LWE type encryption [28].**

  Encryption. Define

  $$LWE_{\boldsymbol{s}}(m) = (\boldsymbol{a}, [\langle \boldsymbol{a}, \boldsymbol{s} \rangle + e + m]_Q) \in \mathbb{Z}_Q^{N+1}$$

  as an LWE encryption of a message encoding $m = \Delta \cdot \tilde{m} \in \mathbb{Z}_Q$ under key $\boldsymbol{s} \in \mathbb{Z}_Q^N$, where explicit random vector $\boldsymbol{a} \xleftarrow{\$} U(\mathbb{Z}_Q^N)$, error $e \xleftarrow{\$} \chi$, $\Delta = \lfloor Q/t \rfloor$ and $\tilde{m} \in \mathbb{Z}_t$. When we want to emphasize the error term we write $LWE_{\boldsymbol{s}}(m; e)$.

  Decryption. A ciphertext $(\boldsymbol{a}, b) \in \mathbb{Z}_Q^{N+1}$ is decrypted by computing

  $$LWE_{\boldsymbol{s}}^{-1}(\boldsymbol{a}, b) = f([b - \langle \boldsymbol{a}, \boldsymbol{s} \rangle]_Q)$$

  where $f : \mathbb{Z}_Q \to \mathbb{Z}_t$ is an appropriate decoding function to correct the error e and recover the message. Then assuming the error distribution χ is concentrated with bounded by $\Delta/2$ in absolute value, message $\tilde{m}$ can be decoded using function

  $$f(x) = \lfloor \frac{t}{Q} \cdot x \rceil \mod t.$$

  We can easily check the correctness. Assume $b - \langle \boldsymbol{a}, \boldsymbol{s} \rangle = \Delta \cdot \tilde{m} + e + Q \cdot r$ for some integer r, then if we divide by Q and multiply by t, we have $\tilde{m} + \frac{t}{Q} \cdot (e - (\frac{Q}{t} - \Delta) \cdot \tilde{m}) + t \cdot r$. We also have $\frac{t}{Q} \cdot (e - (\frac{Q}{t} - \Delta) \cdot \tilde{m}) < 1/2$ by the fact $e < \Delta/2$, which means that the decryption is correct.
- **VecLWE type encryption [27,6].**

Encryption. Define

$$VecLWE_{\mathbf{S}}(\boldsymbol{m}) = (\boldsymbol{a}, \boldsymbol{b}) \in \mathbb{Z}_Q^{N+r}$$

as a VecLWE encryption of a message $\boldsymbol{m} = (m_1, \dots, m_r) = \Delta \cdot (\tilde{m}_1, \cdots, \tilde{m}_r) \in \mathbb{Z}_Q^r$ under key $\mathbf{S} \in \mathbb{Z}_Q^{r \times N}$, where $\boldsymbol{a} \overset{\$}{\leftarrow} U(\mathbb{Z}_Q^N)$, $\boldsymbol{b} = [\mathbf{S}\mathbf{a} + \boldsymbol{e} + \boldsymbol{m}]_Q = (b_1, b_2, \dots, b_r) \in \mathbb{Z}_Q^r$, the small noise vector term $\boldsymbol{e} \overset{\$}{\leftarrow} \chi^r$, $\Delta = \lfloor \frac{Q}{t} \rfloor$ and $(\tilde{m}_1, \dots, \tilde{m}_r) \in \mathbb{Z}_t^r$. When we want to emphasize the error term we write $VecLWE_{\mathbf{S}}(\boldsymbol{m}; \boldsymbol{e})$.

Decryption. The decryption is same with LWE encryption. For $(\boldsymbol{a}, \boldsymbol{b}) \in \mathbb{Z}_Q^{N+r}$, one needs to compute

$$VecLWE_{\mathbf{S}}^{-1}(\boldsymbol{a}, \boldsymbol{b}) = f([\boldsymbol{b} - \mathbf{S} \cdot \boldsymbol{a}]_Q)$$

where for a vector $\boldsymbol{x} = (x_1, \dots, x_r)$, $f(\boldsymbol{x}) := (f(x_1), \dots, f(x_r))$ and function f is defined in above LWE decryption. The decryption is correct if $||\boldsymbol{e}||_\infty < \Delta/2$.

- **MatGSW type encryption [22].**

Encryption. Define

$$MatGSW_{\mathbf{S}}(\mathbf{M}) = \left[\left(\frac{\mathbf{A}}{\mathbf{SA} + \mathbf{E}} \right) + \left(\frac{\mathbf{0}}{-\mathbf{MS}||\mathbf{M}} \right) \cdot \mathbf{G} \right]_Q \in \mathbb{Z}_Q^{(N+r) \times (N+r) \cdot l}$$

as a MatGSW encryption of message matrix $\mathbf{M} \in \{0,1\}^{r \times r}$ under key $\mathbf{S} \in \mathbb{Z}_Q^{r \times N}$, where $l = \lceil log_2 Q \rceil$, $\mathbf{A} \overset{\$}{\leftarrow} U(\mathbb{Z}_Q^{N \times (N+r) \cdot l})$, small noise matrix $\mathbf{E} \overset{\$}{\leftarrow} \chi^{r \times (N+r) \cdot l}$ and $\mathbf{G} = \boldsymbol{g}^T \otimes \mathbf{I}_{N+r} \in \mathbb{Z}_Q^{(N+r) \times (N+r) \cdot l}$. Since this is an encryption of secret key information, the security of this scheme is based on the circular security [20,22] of the LWE encryption. When we want to emphasize the error term we write $MatGSW_{\mathbf{S}}(\mathbf{M}; \mathbf{E})$.

Decryption. For a ciphertext $\mathbf{C} = MatGSW_{\mathbf{S}}(\mathbf{M})$, note that the $(Nl+jl-1)$-th column of $\mathbf{C}$ is

$$\boldsymbol{c}_{Nl+jl-1} = (\boldsymbol{a}_{Nl+jl-1}, \mathbf{S} \cdot \boldsymbol{a}_{Nl+jl-1} + \boldsymbol{e}_{Nl+jl-1} + \boldsymbol{m}_j \cdot 2^{l-2}) \in \mathbb{Z}_Q^{N+r}$$

where $\boldsymbol{a}_{Nl+jl-1}$ and $\boldsymbol{e}_{Nl+jl-1}$ is the $(Nl+jl-1)$-th column of $\mathbf{A}$ and $\mathbf{E}$, respectively, and $\boldsymbol{m}_j$ is the j-th column of $\mathbf{M}$. One can obtain the i,j-th element of $\mathbf{M}$ by computing

$$\mathbf{M}_{i,j} = \lfloor \langle (-\boldsymbol{s}_i, \boldsymbol{u}_i), \boldsymbol{c}_{Nl+jl-1} \rangle \rceil_2 \in \{0,1\},$$

where $i,j \in [r]$, $\boldsymbol{s}_i$ is the i-th row of $\mathbf{S}$, $\boldsymbol{u}_i$ is the i-th standard basis vector and $\lfloor x \rceil_2$ outputs 1 if x is close to $Q/4$, and 0 otherwise. If $||\boldsymbol{e}_{Nl+jl-1}||_\infty < Q/8$, the decryption is correct.

3.2 Operations

We first show the general homomorphic matrix-vector multiplication by a lemma. Then we give a special homomorphic matrix-vector multiplication when the matrix is a cyclic permutation matrix (described in Section 2.3).

Our general homomorphic matrix-vector multiplication is stated by the following lemma.

Lemma 3. *For any* $\mathbf{C} = MatGSW_{\mathbf{S}}(\mathbf{M_0} \in \{0,1\}^{r\times r}; \mathbf{E}) \in \mathbb{Z}_Q^{(N+r)\times(N+r)\cdot l}$ *and any* $(\boldsymbol{a}, \boldsymbol{b}) = VecLWE_{\mathbf{S}}(\boldsymbol{m_1} \in \mathbb{Z}_Q^r; \boldsymbol{ve}) \in \mathbb{Z}_Q^{N+r}$*, if* $\boldsymbol{e}_i$ *is the i-th row of* $\mathbf{E}$*, the computation result of operation*

$$\begin{aligned}(\diamond) : MatGSW \times VecLWE &\to VecLWE \\ (\mathbf{C}, (\boldsymbol{a}, \boldsymbol{b})) &\mapsto \mathbf{C} \diamond (\boldsymbol{a}, \boldsymbol{b}) = [\mathbf{C} \cdot \mathbf{G}^{-1}(\boldsymbol{a}, \boldsymbol{b})]_Q\end{aligned} \tag{2}$$

is a VecLWE encryption of message $[\mathbf{M_0} \cdot \boldsymbol{m_1}]_Q \in \mathbb{Z}_Q^r$ *with small noise vector* $\boldsymbol{e} = (e_1, \ldots, e_r)$*, where* e_i *is subgaussian with parameter* $O(\sqrt{||\boldsymbol{e}_i||_2^2 + ||\boldsymbol{ve}||_2^2})$*.*

To proof the correctness of Lemma 3, we first introduce a new type encryption. Define

$$\widehat{MatLWE}_{\mathbf{S}}(\mathbf{M}) = \left[\begin{pmatrix}\mathbf{A} \\ \hline \mathbf{SA}+\mathbf{E}\end{pmatrix} + \begin{pmatrix}\mathbf{0} \\ \hline \mathbf{M}\end{pmatrix} \cdot \mathbf{G}\right]_Q \in \mathbb{Z}_Q^{(N+r)\times v\cdot l}$$

as a $\widehat{MatLWE}$ encryption of $\mathbf{M} \in \mathbb{Z}_Q^{r\times v}$ under key $\mathbf{S} \in \mathbb{Z}_Q^{r\times N}$, where $l = \lceil log_2 Q \rceil$, $\mathbf{A} \overset{\$}{\leftarrow} U(\mathbb{Z}_Q^{N\times v\cdot l})$, small noise matrix $\mathbf{E} \overset{\$}{\leftarrow} \chi^{r\times v\cdot l}$ and $\mathbf{G} = \boldsymbol{g}^T \otimes \mathbf{I}_v \in \mathbb{Z}_Q^{v\times vl}$.

Note that for MatGSW and $\widehat{MatLWE}$, we have

$$MatGSW_{\mathbf{S}}(\mathbf{M}) = \widehat{MatLWE}_{\mathbf{S}}([-\mathbf{MS}, \mathbf{M}]).$$

In order to simplify the proof of Lemma 3, we introduce the following lemma for the $\widehat{MatLWE}$ encryption.

Lemma 4. *For any* $\mathbf{C} = \widehat{MatLWE}_{\mathbf{S}}(\mathbf{M} \in \mathbb{Z}_Q^{r\times v}; \mathbf{E}) \in \mathbb{Z}_Q^{(N+r)\times v\cdot l}$ *and* $\boldsymbol{d} \in \mathbb{Z}_Q^v$*, if* $\boldsymbol{e}_i$ *is the i-th row of* $\mathbf{E}$*, the computation result of operation*

$$\widehat{MatLWE}_{\mathbf{S}}(\mathbf{M}) \odot \boldsymbol{d} = [\mathbf{C} \cdot \mathbf{G}^{-1}(\boldsymbol{d})]_Q, \tag{3}$$

is a VecLWE encryption of message $[\mathbf{M} \cdot \boldsymbol{d}]_Q \in \mathbb{Z}_Q^r$ *with small noise vector* $\boldsymbol{e} = (e_1, \ldots, e_r)$*, where* e_i *is subgaussian with parameter* $O(||\boldsymbol{e}_i||_2)$*.*

Proof. Let $\boldsymbol{x} = \mathbf{G}^{-1}(\boldsymbol{d}) \in \mathbb{Z}_2^{v\cdot l}$, and assume $\mathbf{C} = [(\mathbf{A}, \mathbf{B}) + (\mathbf{0}, \mathbf{M}) \cdot \mathbf{G}]_Q$ where $\mathbf{B} = \mathbf{SA} + \mathbf{E}$, then one can compute

$$\begin{aligned}\widehat{MatLWE}_{\mathbf{S}}(\mathbf{M}) \odot \boldsymbol{d} =& [((\mathbf{A}, \mathbf{B}) + (\mathbf{0}, \mathbf{M}) \cdot \mathbf{G}) \cdot \boldsymbol{x}]_Q \\ =& [(\mathbf{A}, \mathbf{SA} + \mathbf{E}) \cdot \boldsymbol{x} + (\mathbf{0}, \mathbf{M} \cdot \boldsymbol{d})]_Q \\ =& [(\mathbf{A}\boldsymbol{x}, \mathbf{S} \cdot \mathbf{A}\boldsymbol{x} + \mathbf{E}\boldsymbol{x} + \mathbf{M} \cdot \boldsymbol{d})]_Q.\end{aligned}$$

Since $\boldsymbol{x}$ is the subgaussian with parameter $O(1)$, $\mathbf{E}\boldsymbol{x}$ is a small vector. So the final result is a VecLWE encryption

$$[(\boldsymbol{a}, \mathbf{S} \cdot \boldsymbol{a} + \boldsymbol{e} + \boldsymbol{m})]_Q \in \mathbb{Z}_Q^{N+r}$$

where $\boldsymbol{a} = \mathbf{A} \cdot \boldsymbol{x}$, $\boldsymbol{e} = \mathbf{E} \cdot \boldsymbol{x}$ and $\boldsymbol{m} = \mathbf{M} \cdot \boldsymbol{d}$.

Assume $\boldsymbol{e} = (e_1, \ldots, e_r)$, then there is $e_i = \langle \mathbf{e}_i, \boldsymbol{x} \rangle$ where $\boldsymbol{e_i}$ is the i-th row of $\mathbf{E}$. By the Pythagorean additivity, the error $e_i = \langle \boldsymbol{e}_i, \boldsymbol{x} \rangle$ is subgaussian with parameter $O(||\boldsymbol{e}_i||_2)$ □

Specially, in operation (2), when $\mathbf{M}_0$ is a permutation matrix for a cyclic permutation $\pi \in S_r$, i.e. $\mathbf{M}_0 = [\boldsymbol{u}_{\pi(1)}, \boldsymbol{u}_{\pi(2)}, \ldots, \boldsymbol{u}_{\pi(r)}] \in \{0,1\}^{r \times r}$, where $\boldsymbol{u}_i$ is the i-th standard basis vector, we have lemma:

Lemma 5. *For any $\mathbf{C} = MatGSW_{\mathbf{S}}(\mathbf{M_0}; \mathbf{E}) \in \mathbb{Z}_Q^{(N+r) \times (N+r) \cdot l}$ where the message matrix is a permutation matrix $\mathbf{M_0} \in \{0,1\}^{r \times r}$ for a cyclic permutation $\pi \in S_r$, and any $(\boldsymbol{a}, \boldsymbol{b}) = VecLWE_{\mathbf{S}}(\boldsymbol{m_1} \in \mathbb{Z}_q^r; \boldsymbol{ve}) \in \mathbb{Z}_Q^{N+r}$ where $\boldsymbol{ve} = (ve_1, \ldots, ve_r)$, if $\boldsymbol{e}_i$ is the i-th row of $\mathbf{E}$, the computation result of operation*

$$\mathbf{C} \diamond (\boldsymbol{a}, \boldsymbol{b}) = [\mathbf{C} \cdot \mathbf{G}^{-1}(\boldsymbol{a}, \boldsymbol{b})]_Q$$

is a VecLWE encryption of message $[\mathbf{M_0} \cdot \boldsymbol{m_1}]_Q \in \mathbb{Z}_Q^r$ with small noise vector $\boldsymbol{e} = (e_1, \ldots, e_r)$, where e_i is subgaussian with parameter $O(\sqrt{||\mathbf{e}_i||_2^2 + ve_{\pi(r-2+i)}^2})$.

Proof. The proof of Lemma 5 is similar with Lemma 3. The only different in this case is that $\mathbf{M}_0 \cdot \boldsymbol{ve}$ is a cyclic permutation of $\boldsymbol{ve}$, i.e. a "cyclic rotation" vector of $\boldsymbol{ve}$. We can rewrite

$$\mathbf{M}_0 = [\boldsymbol{u}_{\pi(1)}, \boldsymbol{u}_{\pi(2)}, \ldots, \boldsymbol{u}_{\pi(r)}] = (\boldsymbol{u}_{\pi(r-1)}^T, \boldsymbol{u}_{\pi(r)}^T, \ldots, \boldsymbol{u}_{\pi(r-2)}^T) \in \{0,1\}^{r \times r}$$

where $\boldsymbol{u}_i$ is the i-th standard basis vector. Then the i-th element of $\mathbf{M}_0 \cdot \boldsymbol{ve}$ is $ve_i = \langle \boldsymbol{u}_{\pi(r-2+i)}, \boldsymbol{ve} \rangle = ve_{\pi(r-2+i)}$. So the i-th element of the total error is subgaussian with parameter $O(\sqrt{||\mathbf{e}_i||_2^2 + ve_{\pi(r-2+i)}^2})$. □

So given a permutation matrix and a vector, we present an operation to homomorphically compute the matrix-vector product of them of two. In scheme [22], they present a homomorphic matrix-matrix multiplication and use it to construct the bootstrapping scheme, and in the following section we show how to use homomorphic matrix-vector multiplication to construct the bootstrapping procedure.

4 Our Bootstrapping Procedure

We describe our bootstrapping procedure in this section. In the first part, we present some background about bootstrapping. We give the details of our bootstrapping scheme in the second part. In the final, we analyze our scheme.

4.1 Bootstrapping

The goal of bootstrapping is to decrypt a ciphertext homomorphically. An LWE ciphertext $(\boldsymbol{a}, b) \in \mathbb{Z}_q^{n+1}$ under key $\boldsymbol{s} \in \mathbb{Z}_q^n$ is decrypted. by computing

$$\tilde{m} = LWE_{\boldsymbol{s}}^{-1}(\boldsymbol{a}, b) = f([b - \langle \boldsymbol{a}, \boldsymbol{s} \rangle]_q) = [\lfloor \frac{t}{q} \cdot [b - \langle \boldsymbol{a}, \boldsymbol{s} \rangle]_q \rceil]_t.$$

A new ciphertext with smaller noise can be obtained by homomorphically decrypting a ciphertext with large noise. Since there needs the information of secret key $\boldsymbol{s}$ to decrypt the ciphertext, a bootstrapping key $Enc(\boldsymbol{s})$ needs to be generated using an encryption $Enc()$. In the final, the noise of the output ciphertext depends on the noise of $Enc(\boldsymbol{s})$, but not on the noise of the ciphertext $(\boldsymbol{a}, b)$. In our scheme, such a scheme $Enc()$ is the MatGSW encryption.

There are two processes for homomorphic decryption. One is linear operation, i.e. $b - \langle \boldsymbol{a}, \boldsymbol{s} \rangle = b - \sum_i a_i s_i$, the other is non-linear operation, i.e. the rounding operation $f : \mathbb{Z}_q \to \mathbb{Z}_t$. For the linear operation, as mentioned before in section 2.3, the addition in $\mathbb{Z}_q$ leads to the multiplication of the corresponding permutation matrices. Since we can multiply two permutation matrices by multiplying one permutation matrix with the first column of the other matrix, the linear operation can be computed by iteratively operating the homomorphic matrix-vector multiplication described by Lemma 5. For the non-linear operation, it is automatically executed by the "cyclic rotation" property of the message vector as described in Section 1.2. Actually, we can further evaluate a known arbitrary function (mapping) $func : \mathbb{Z}_t \to \mathbb{Z}_h$ on $\tilde{m} \in \mathbb{Z}_t$, so in generally, we can define $F = func \circ f$ as the final non-linear step. In Section 5, we present how to set $func$ to evaluate some complex functions.

So the bootstrapping procedure includes two steps:

- **BootKeyGen**$(\mathbf{SK}, \boldsymbol{s})$: takes as input a secret key $\mathbf{SK}$ for MatGSW encryption, and a secret key vector $\boldsymbol{s} \in \mathbb{Z}_q^n$ of the ciphertext to be bootstrapped. It outputs a bootstrapping key **BootKey** that appropriately encrypts $\boldsymbol{s}$ under **SK**.
- **Bootstrap**(**BootKey**, $\boldsymbol{c}$): takes as input the bootstrapping key **BootKey** and a ciphertext vector $\boldsymbol{c} = (\boldsymbol{a}, b) \in \mathbb{Z}_q^{n+1}$, which is decrypted to $\tilde{m} \in \mathbb{Z}_t$ under key $\boldsymbol{s}$. It outputs an LWE ciphertext which decrypts to $F(\tilde{m}) \in \mathbb{Z}_h$ under key $\boldsymbol{sk}_1$ (with a smaller noise), where $\boldsymbol{sk}_1$ is the first row of **SK**.

For the ciphertext $(\boldsymbol{a}, b = \langle \boldsymbol{a}, \boldsymbol{s} \rangle + e + \Delta \cdot \tilde{m})$, where $\boldsymbol{a} = (a_1, \ldots, a_n)$ and $\boldsymbol{s} = (s_1, \ldots, s_n)$, let $a_i = \sum_{k \in [w]} a_{i,k} 2^{k-1}$, $w = \lceil log_2 q \rceil$ and $a_{i,k} \in \{0, 1\}$ is an integer. To decrypt the ciphertext, the linear term $b - \langle \boldsymbol{a}, \boldsymbol{s} \rangle$ can be write as

$$b - \sum_{i \in [n]} a_i s_i = b - \sum_{i \in [n]} (\sum_{k \in [w]} a_{i,k} 2^{k-1} s_i).$$

So in the bootstrapping key generation algorithm, the secret key information $[2^{k-1} s_i]_q$ will be embedded into a matrix $\mathbf{M}_{\phi([2^{k-1} s_i]_q)} \in \{0, 1\}^{q \times q}$ and then encrypted into a $MatGSW$ ciphertexts. By the relationship between $(\mathbb{Z}_q, +)$ and matrix in Section 2.3, addition operations $-\sum_{i\in[n]}(\sum_{k\in[w]} a_{i,k}2^{k-1}s_i)$ can be computed using the homomorphic matrix-vector multiplication, i.e. the operation given in Lemma 5. So we only need to initialize a VecLWE ciphertext, and then iteratively operate the homomorphic matrix-vector multiplication on the result VecLWE ciphertext.

4.2 Procedures

In our scheme, Q is a module and $l = \lceil log_2 Q \rceil$. N is the dimension of the explicit random vector of the MatGSW encryption and the message dimension is $q \times q$, i.e., a ciphertext $MatGSW \in \mathbb{Z}_Q^{(N+q)\times(N+q)\cdot l}$. Let $w = \lceil log_2 q \rceil$ and $\phi : \mathbb{Z}_q \to S_q$ be the isomorphism of an element in $\mathbb{Z}_q$ into the cyclic permutation that corresponds to this element. We follows a procedure structure of RLWE-based scheme FHEW [18] and THFE[13], i.e., a bootstrapping scheme includes two algorithms: **BootKeyGen** and **Bootstrap**; and in **Bootstrap** there are three steps: in **Initialize**, b is set into a message vector $\boldsymbol{m}$; in **Increment**, the linear operation $b - \langle \boldsymbol{a}, \boldsymbol{s} \rangle$ is executed; in the final step, an LWE ciphertext is derived.

- **BootKeyGen**($\mathbf{SK}, \boldsymbol{s}$): given the secret key $\boldsymbol{s} \in \mathbb{Z}_q^n$ for ciphertext to be bootsrapped and a secret key $\mathbf{SK} \in \mathbb{Z}_Q^{q\times N}$ for MatGSW encryption, outputs a bootstrapping key.
  For every $i \in [n]$, $k \in [w]$, let $\mathbf{M}_{\phi([2^{k-1}s_i]_q)} \in \{0,1\}^{q\times q}$ be the matrix corresponding to $\phi([2^{k-1}s_i]_q)$, and compute
  $$\mathbf{BK}_{i,k} = MatGSW_{\mathbf{SK}}(\mathbf{M}_{\phi([2^{k-1}s_i]_q)}) \in \mathbb{Z}_Q^{(N+q)\times(N+q)\cdot l}.$$
  Let $\mathbf{BootKey} = \{\mathbf{BK}_{i,k}\}_{i\in[n],k\in[w]}$ and return $\mathbf{BootKey}$.
- **Bootstrap**($\mathbf{BootKey}, \boldsymbol{c}$): given a ciphertext $\boldsymbol{c} = (\boldsymbol{a}, b) \in \mathbb{Z}_q^{n+1}$ and a bootstrapping key $\mathbf{BootKey}$, outputs the refreshed LWE ciphertext $\mathbf{c}' \in \mathbb{Z}_Q^{N+1}$.
  - **Initialize**: For every $i \in [q]$, set
    $$m_i = \Delta' \cdot F([b-i+1]_q) = \Delta' \cdot func(f([b-i+1]_q)) \in \mathbb{Z}_Q$$
    where $f : \mathbb{Z}_q \to \mathbb{Z}_t$ is the rounding function, $func : \mathbb{Z}_t \to \mathbb{Z}_h$ is a known arbitrary function and $\Delta' = \lfloor \frac{Q}{h} \rfloor$. Set $\boldsymbol{acc} := (\mathbf{0}, \boldsymbol{m}) \in \mathbb{Z}_Q^{N+q}$ where $\boldsymbol{m} = (m_1, \ldots, m_q)$.
  - **Increment**: For every $i \in [n]$ and $k \in [w]$, let $a_i' = -a_i \mod q$ and set $z_{i,k} = \lfloor \frac{a_i'}{2^{k-1}} \rfloor \mod 2$. Then for every $i \in [n], k \in [w]$, if $z_{i,k} > 0$, iteratively compute
    $$\boldsymbol{acc} \leftarrow \mathbf{BK}_{i,k} \diamond \boldsymbol{acc}.$$
  - **Extract**: If the final ciphertext is $\boldsymbol{acc} = (\boldsymbol{a}', \boldsymbol{b}' = (b_1', \ldots, b_q'))$, return $(\boldsymbol{a}', b_1')$.

For the final ciphertext, one can use Module-Switch to reduce the module from Q back to q and use Key-Switch to turn the output into an LWE encryption under $\boldsymbol{s}$ [8,7]. Then one can perform additional operations on this ciphertext.

4.3 Correctness

For the correctness of our procedure, we have the following lemma.

Lemma 6 (Correctness). *Let* $\mathbf{SK}$ *be the secret key for our scheme and* $\boldsymbol{sk}_1$ *be the first row of* $\mathbf{SK}$*. Let* $\boldsymbol{c}$ *and* $\boldsymbol{s}$ *be a ciphertext and secret key described in our scheme. Assume* $\boldsymbol{c}$ *decrypts to* $\tilde{m} \in \mathbb{Z}_t$ *under key* $\boldsymbol{s}$*. For* $\mathbf{BootKey} \leftarrow \boldsymbol{BootKeyGen}(\mathbf{SK}, \boldsymbol{s})$*, the refreshed ciphertext* $\boldsymbol{c}' \leftarrow \boldsymbol{Bootstrap}(\mathbf{BootKey}, \boldsymbol{c})$ *decrypts to* $func(\tilde{m}) \in \mathbb{Z}_h$ *under secret key* $\boldsymbol{sk}_1$*, where* $func : \mathbb{Z}_t \to \mathbb{Z}_h$ *is a known arbitrary function.*

Proof. Note that $(\mathbf{0}, \boldsymbol{m}) \in \mathbb{Z}_Q^{N+q}$ can be seen as a VecLWE encryption of message $\boldsymbol{m}$ under key $\mathbf{SK}$, i.e., $(\mathbf{0}, \boldsymbol{m} = \mathbf{SK} \cdot \mathbf{0} + \boldsymbol{m}) = VecLWE_{\mathbf{SK}}(\boldsymbol{m}; \mathbf{0})$. In addition, $\mathbf{BK}_{i,k}$ is a MatGSW encryption of $\mathbf{M}_{\phi([2^{k-1}s_i]_q)}$. By Lemma 5, $\boldsymbol{acc} \leftarrow \mathbf{BK}_{i,k} \diamond VecLWE_{\mathbf{SK}}(\boldsymbol{m})$ is a VecLWE encryption of message $\mathbf{M}_{\phi([2^{k-1}s_i]_q)} \cdot \boldsymbol{m}$. Then in our scheme, by iteratively computing $\boldsymbol{acc} \leftarrow \mathbf{BK}_{i,k} \diamond \boldsymbol{acc}$ for every $i \in [n]$ and $k \in [w]$, the final VecLWE ciphertext $\boldsymbol{acc}$ encrypts message vector

$$\mathbf{M}_{\phi([z_{n,w}2^{w-1}s_n]_q)} \cdot (\cdots(\mathbf{M}_{\phi([z_{1,1}2^0 s_1]_q)} \cdot \boldsymbol{m})). \tag{4}$$

Besides, if $\mathbf{M}_{\phi(p)} \in \{0,1\}^{q \times q}$ is the permutation matrix corresponding to $\phi(p)$, for vector

$$\boldsymbol{m} = \Delta' \cdot (F([b]_q), F([b-1]_q), \ldots, F([b-q+1]_q))$$

which is the message vector in the **Initialize** step, we have that

$$\mathbf{M}_{\phi(p)} \cdot \boldsymbol{m} = \Delta' \cdot (F[b+p]_q), \ldots, F([b-q+1+p]_q)). \tag{5}$$

So applies equation (5) for the vector (4), the final ciphertext $\boldsymbol{acc}$ is a VecLWE encryption of message vector

$$\begin{aligned}\boldsymbol{m} =& \mathbf{M}_{\phi([z_{n,w}2^{w-1}s_n]_q)} \cdot (\cdots(\mathbf{M}_{\phi([z_{1,1}2^0 s_1]_q)} \cdot \boldsymbol{m}))\\ =& \Delta' \cdot (F([b + \sum_{i\in[n],k\in[w]} z_{i,k}2^{k-1}s_i]_q), \ldots, F([b-q+1+\sum_{i\in[n],k\in[w]} z_{i,k}2^{k-1}s_i]_q))\\ =& \Delta' \cdot (F([b - \langle \boldsymbol{a}, \boldsymbol{s}\rangle]_q), \ldots, F([b-q+1-\langle \boldsymbol{a}, \boldsymbol{s}\rangle]_q)).\end{aligned}$$

Assume the final ciphertext is $\boldsymbol{acc} = (\boldsymbol{a}', \boldsymbol{b}' = (b'_1, \ldots, b'_q))$, then the returned ciphertext $(\boldsymbol{a}', b'_1)$ is an LWE encryption of message encoding

$$\Delta' \cdot F([b - \langle \boldsymbol{a}, \boldsymbol{s}\rangle]_q) = \Delta' \cdot func(f([b - \langle \boldsymbol{a}, \boldsymbol{s}\rangle]_q)) = \Delta' \cdot func(\tilde{m}),$$

e.g., $(\boldsymbol{a}', b'_1 = \langle \boldsymbol{a}', \boldsymbol{sk}_1\rangle + e + \Delta' \cdot func(\tilde{m}))$, where $\boldsymbol{sk}_1$ is the first row of secret $\mathbf{SK}$ and e is the error. Hence the refreshed ciphertext $\boldsymbol{c}'$ decrypts to $func(\tilde{m}) \in \mathbb{Z}_h$ under secret key $\boldsymbol{sk}_1$. □

We further quantify the error in the ciphertext output by **Bootstrap**. We assume the error distribution χ over $\mathbb{Z}$ of MatGSW in our scheme is subgaussian with parameter s.

Lemma 7. *For any $\boldsymbol{c} \in \mathbb{Z}_q^{n+1}$, the error of the refreshed ciphertext $\boldsymbol{c}' \leftarrow$ **Bootstrap**$(\mathbf{BootKey}, \boldsymbol{c})$ is subgaussian with parameter $O(s\sqrt{(N+q)\cdot nwl})$, except with probability $2^{-\Omega((N+q)\cdot nwl)}$ over the random choices of* **BootKey** *and* ***Bootstrap****.*

Proof. In our scheme, $\boldsymbol{acc}$ is initialized to be a VecLWE ciphertext $(\mathbf{0}, \boldsymbol{m})$ with noise vector $\mathbf{0}$. Then if the noise matrix of $\mathbf{BK}_{i,k}$ is $\mathbf{E}_{i,k} \overset{\$}{\leftarrow} \chi^{q\times(N+q)\cdot l}$ and $\boldsymbol{e}_{i,k,j} \in \mathbb{Z}^{(N+q)\cdot l}$ is the j-th row of $\mathbf{E}_{i,k}$, by Lemma 5, the ciphertext after operation $\mathbf{BK}_{i,k} \diamond \boldsymbol{acc} = \mathbf{BK}_{i,k} \diamond (\mathbf{0}, \boldsymbol{m})$ has a noise vector $\boldsymbol{e}' = (e_1', \ldots, e_q') \in \mathbb{Z}^q$, where e_j' is subgaussian with parameter $O(||\boldsymbol{e}_{i,k,j}||_2)$ (note that the noise vector of $(\mathbf{0}, \boldsymbol{m})$ is $\mathbf{0}$).

Then by iteratively computing $\boldsymbol{acc} \leftarrow \mathbf{BK}_{i,k} \diamond \boldsymbol{acc}$ for every $i \in [n]$ and $k \in [w]$, for the noise vector $(e_1, \ldots, e_q)$ of the final VecLWE ciphertext, its entry e_j is subgaussian with parameter $\sqrt{\sum_{i\in[n],k\in[w]} ||\boldsymbol{e}_{i,k,c_{i,k,j}}||_2^2}$ by Pythagorean additivity and Lemma 5, where $j \in [q]$ and $\boldsymbol{e}_{i,k,c_{i,k,j}}$ is the $c_{i,k,j}$-th row of $\mathbf{E}_{i,k}$ (here the value of $c_{i,k,j}$ depends on the the permutation $\phi([2^{k-1}s_i]_q)$ and j by Lemma 5). More concisely, let

$$\mathbf{er}_j = (\boldsymbol{e}_{1,1,c_{1,1,j}}, \ldots, \boldsymbol{e}_{i,k,c_{i,k,j}}, \ldots, \boldsymbol{e}_{n,w,c_{n,k,j}}) \in \mathbb{Z}^{(N+q)\cdot nwl}$$

to be the concatenation of the individual noise vectors $\boldsymbol{e}_{i,k,c_{i,k,j}}$, then the final result $\boldsymbol{acc}$ has a noise vector $(e_1, \ldots, e_q)$ whose entry e_j is subgaussian with parameter $O(||\boldsymbol{er}_j||_2)$.

By Lemma 1, the l_2 norm of $\boldsymbol{er}_j$ is within $O(s\sqrt{(N+q)\cdot nwl})$ except with probability $2^{-\Omega((N+q)\cdot nwl)}$, which means that the final ciphertext error is subgaussian with parameter $O(||\boldsymbol{er}_1||_2) = O(s\sqrt{(N+q)\cdot nwl})$, except with probability $2^{-\Omega((N+q)\cdot nwl)}$. □

By above lemma, we can see that the error growth factor is $O(\sqrt{(N+q)\cdot nwl})$. By setting the modulus such that $\Delta'/2$ is larger than the final noise, we can evaluate a function $func \circ LWE_{\boldsymbol{s}}^{-1}()$ on the ciphertext $\boldsymbol{c}$, where $func$ is a known arbitrary function (mapping) and $LWE_{\boldsymbol{s}}^{-1}()$ is the decryption function.

4.4 Security

Given a security parameter λ, we analyze the security of our scheme. Firstly, it is easy to see that our bootstrapping procedure can be secure under the security of the DLWE assumption and circular security. Recall that for the MatGSW encryption ciphertext, $MatGSW_{\mathbf{SK}}(\mathbf{M} \in \{0,1\}^{q\times q}) \in \mathbb{Z}_Q^{(N+q)\times(N+q)\cdot l}$ where $l = \lceil log_2 Q\rceil$, and the error distribution χ over $\mathbb{Z}$ is subgaussian with parameter s. For the LWE ciphertext $\boldsymbol{c} \in \mathbb{Z}_q^{n+1}$ to be bootstrapped, by [8], we can set $q = \tilde{O}(\lambda)$ and $d = n \cdot w = \tilde{O}(\lambda)$, where $w = \lceil log_2 q\rceil$. For the output message space parameter h, we set $h = O(1)$ (this parameter can be set larger at the expense of security strength).

Theorem 2. *Our bootstrapping scheme can be instantiated to be correct and secure assuming the quantum worst-case hardness of approximating* $\boldsymbol{GapSVP}_{\tilde{O}(N\lambda)}$ *and* $\boldsymbol{SIVP}_{\tilde{O}(N\lambda)}$*, or the classical worst-case hardness of approximating* $\boldsymbol{GapSVP}_{\tilde{O}(N^{1.5}\lambda)}$ *on any* N *dimensional lattice.*

Proof. To rely on the quantum worst-case hardness of LWE, we need to set $s = \Theta(\sqrt{N})$ by [28]. If we choose $N < q$, by Lemma 7, for the correct of the scheme we need to take a large $Q = \tilde{\Omega}(\lambda\sqrt{N \log Q})$, and some $Q = \tilde{O}(\lambda\sqrt{N})$ suffices. Therefore the LWE inverse error rate is $1/\alpha = Q/s = \tilde{O}(\lambda)$, and by Theorem 1 the security of our scheme is reduced to $\mathbf{GapSVP}_{\tilde{O}(N\lambda)}$ and $\mathbf{SIVP}_{\tilde{O}(N\lambda)}$. For the classical security, recall that $Q = \tilde{\Omega}(\lambda\sqrt{NlogQ})$, and we need to set $Q = 2^{N/2}$, then the inverse error rate is $1/\alpha = Q/s = \tilde{O}(\lambda\sqrt{N})$. So by Theorem 1 the security of our scheme is reduced to the classical hardness of $\mathbf{GapSVP}_{\tilde{O}(N^{1.5}\lambda)}$. □

For $poly(N)$-factor approximations to **GapSVP** and **SIVP** on N-dimensional lattices, it take $2^{\Omega(N)}$ times for all known algorithms. We need to choose $N = \Theta(\lambda)$ for 2^{λ} hardness, and this yields a approximation factor of $\tilde{O}(N^2)$ in the quantum case and $\tilde{O}(N^{2.5})$ in the classical case. Those approximation factor are smaller than the result given by Hiromasa, Abe and Okamoto [22], which are $\tilde{O}(N^{2.5})$ in the quantum case and $\tilde{O}(N^3)$ in the classical case.

At the expense of efficiency, we can further set $N > q$ to optimize the approximation factor. In the case $N > q$, by Lemma 7, for the correctness of the scheme we need to select $Q = \tilde{\Omega}(N\sqrt{\lambda logQ})$; some $Q = \tilde{O}(N\sqrt{\lambda})$ suffices. Similar with above analysis, for any const $\epsilon > 0$, by choosing the dimension to be $N = \lambda^{1/\epsilon}$, we obtain a factor as small as $\tilde{O}(N^{1.5+\epsilon/2})$ in the quantum case, and $\tilde{O}(N^{2+\epsilon/2})$ in the classical case. Note that this result achieves the best factor in prior works on bootstrapping of standard lattice-based FHE, i.e. Brakerski and Vaikuntanathan's work [10]. Since the standard lattice-based public-key encryption can be based on the hardness of approximating the problem to $\tilde{O}(N^{1.5})$ using the quantum reduction [28] and $\tilde{O}(N^2)$ using the classical reduction [26], our bootstrapping scheme can be as secure as the standard lattice-based PKE.

4.5 Time and Space Complexity

For the time and space complexity, let $d = nw$, then the time complexity of our scheme is $O(dl \cdot (N+q)^2)$ and the space complexity for the bootstrapping keys is about $dl^2 \cdot (N+q)^2$. We can make a comparison with the bootstrapping scheme of Hiromasa, Abe and Okamoto [22]. The time complexity of their scheme is about $O(tl^2 \cdot (d+q)(N+r)^3)$ and the space complexity for the bootstrapping keys is about $(3td + qt + 1)l^2 \cdot (N+r)^2$, where parameters N, q, l, d is same with our scheme, and $t = O(\log\lambda/\log\log\lambda), r = O(\log\lambda)$ are parameters for the Chinese Reminder Theorem. When setting $q = \tilde{O}(\lambda)$, $d = \tilde{O}(\lambda)$ (by [9]), $N = \Theta(\lambda)$, $Q = \tilde{O}(\lambda\sqrt{N})$ for our scheme and $Q = \tilde{O}(\lambda N)$ for Hiromasa, Abe and Okamoto's scheme, our scheme is time-efficient by about a $O(\lambda \log Q/\log\lambda \log\log\lambda) = O(\lambda)$ factor and a slightly space growth with a factor $O(\log\lambda \log\log\lambda)$. For a stronger assumption parameter $N = \tilde{O}(\lambda)$, our scheme is time-efficient by about a $\tilde{O}(\lambda)$ factor and space-reduced by a $O(t) = O(\log \lambda / \log \log \lambda)$ factor. A detailed comparison for $N = \Theta(\lambda)$ is given in Table 1.

Table 1. Comparison among LWE-based GSW-type bootstrapping schemes, including Alperin-Sheriff and Peikert's work [1], Hiromasa, Abe and Okamoto's work [22] and this work. For the parameters N, q, l, t, d, Q, see Section 4.5. λ is the security parameter. Here N is set to be $\Theta(\lambda)$. In the "Approximation Factor" column, it is the lattice approximation factor in the quantum security, and in the "Large Gates?" column, it means that whether the scheme is allowed to evaluate large FHE gates within once bootstrapping.

Scheme	Time Complexity	Storage	Approximation Factor	Large Gates?
[1]	$O(trN^3l^2 \cdot (dr+q))$ $= O(\lambda^4 \log^6 \lambda \log\log\lambda)$	$dtrl^2 \cdot (N+1)^2$ $= O(\lambda^3 \log^5 \lambda \log\log\lambda)$	$\tilde{O}(N^3)$	×
[22]	$O(tl^2 \cdot (d+q)(N+r)^3)$ $= O(\lambda^4 \log^4 \lambda \log\log\lambda)$	$l^2 \cdot (3td+qt+1)(N+r)^2$ $= O(\lambda^3 \log^4 \lambda \log\log\lambda)$	$\tilde{O}(N^{2.5})$	×
This work	$O(dl \cdot (N+q)^2)$ $= O(\lambda^3 \log^4 \lambda \log\log\lambda)$	$dl^2 \cdot (N+q)^2$ $= O(\lambda^3 \log^5 \lambda \log\log^2 \lambda)$	$\tilde{O}(N^2)$	✓

5 Determining the Function *func*

Note that by Lemma 6, using an LWE ciphertext $\boldsymbol{c}$ which decrypts to a message $\tilde{m} \in \mathbb{Z}_t$, we can evaluate a known function $func : \mathbb{Z}_t \to \mathbb{Z}_h$ on $\tilde{m}$. So for a certain gate, like Boolean gates, LUT function, max/min function or comparison, we just need to make clear $func : \mathbb{Z}_t \to \mathbb{Z}_h$. Note that similar functions has studied in related works [18,3,11,15], so the related technology here is a simple promotion. We present how to evaluate Boolean gates and complex functions in our scheme in the following.

5.1 Boolean Gates

Here we describe how to homomorphically compute Boolean gates for ciphertexts $\boldsymbol{c}_0 = LWE_{\boldsymbol{s}}(\tilde{m}_0 \cdot \lfloor \frac{q}{4} \rfloor)$ and $\boldsymbol{c}_1 = LWE_{\boldsymbol{s}}(\tilde{m}_1 \cdot \lfloor \frac{q}{4} \rfloor)$, where $\tilde{m}_0, \tilde{m}_1 \in \{0, 1\}$. The input message space is $\mathbb{Z}_t = \mathbb{Z}_4$, and we set the output message space to be $\mathbb{Z}_h = \mathbb{Z}_4$. We specify the function (mapping) $func : \mathbb{Z}_t \to \mathbb{Z}_h$ for some Boolean gates such as OR, AND, XOR, etc, in Table 2.

We take the OR gate as an example. For our scheme, first set the $func$ in our **Bootstrap** scheme as follow: $func$ assigns $\{0\} \to 0$ and $\{1, 2\} \to 1$. Then given tow ciphertexts $\boldsymbol{c}_0 = LWE_{\boldsymbol{s}}(\tilde{m}_0 \cdot \lfloor \frac{q}{4} \rfloor)$ and $\boldsymbol{c}_1 = LWE_{\boldsymbol{s}}(\tilde{m}_1 \cdot \lfloor \frac{q}{4} \rfloor)$, where $\tilde{m}_0, \tilde{m}_1 \in \{0, 1\}$, after generating the bootstrapping keys $\mathbf{BootKey} \leftarrow \mathbf{BootKeyGen}(\mathbf{SK}, \mathbf{s})$, computes

$$\mathbf{c}' \leftarrow \mathbf{Bootstrap}(\mathbf{BootKey}, \mathbf{c_1} + \mathbf{c_2}).$$

Note that the addition induces only a small error, so the input ciphertext of our scheme is set to be $\boldsymbol{c}_0 + \boldsymbol{c}_1$, which is an encryption of $(\tilde{m}_0 + \tilde{m}_1) \cdot \lfloor \frac{q}{4} \rfloor$. So if $\tilde{m}_0 = 0$ and $\tilde{m}_1 = 0$, $\boldsymbol{c}_0 + \boldsymbol{c}_1$ encrypts 0, and by $func$ definition (i.e. $func$ assigns $\{0\} \rightarrow 0$) and Lemma 6, the final output of our scheme $\mathbf{c}'$ is an encryption of $func(0) = 0$. For other cases (i.e. $\tilde{m}_0 + \tilde{m}_1 > 0$), the output is an encryption of 1. By this way, we realize to evaluate the OR gate. The correctness of other operations can also be easily checked. For the NOT gate, by the definition of LWE encryption, i.e.

$$\mathbf{c}_0 = (\mathbf{a}, [\langle \mathbf{a}, \mathbf{s} \rangle + e + \tilde{m}_0 \cdot q/4]_q) \in \mathbb{Z}_q^{n+1},$$

it is only need to set $(\mathbf{0}, q/4) - \boldsymbol{c}_0$ to evaluate NOT gate, and this operation don't induce additional errors.

The majority gate can also be easily evaluated in our scheme. On input 3 bits $\tilde{m}_0, \tilde{m}_1, \tilde{m}_2 \in \{0, 1\}$, a majority gate outputs 1 if at least two of the inputs are 1, and 0 if at least two of the inputs are zero. For the majority gate, the function $func$ just needs to assign $\{0, 1\} \rightarrow 0$, $\{2, 3\} \rightarrow 1$.

Table 2. Function $func$ for some Boolean gates

Gate	Input $\boldsymbol{c}$	$func$	$func(LWE_{\boldsymbol{s}}^{-1}(\boldsymbol{c}))$
OR	$\boldsymbol{c}_0 + \boldsymbol{c}_1$	$\{0\} \rightarrow 0$; $\{1, 2\} \rightarrow 1$	$OR(\tilde{m}_0, \tilde{m}_1)$
XOR	$\boldsymbol{c}_0 + \boldsymbol{c}_1$	$\{0, 2\} \rightarrow 0$; $\{1\} \rightarrow 1$	$XOR(\tilde{m}_0, \tilde{m}_1)$
AND	$\boldsymbol{c}_0 + \boldsymbol{c}_1$	$\{0, 1\} \rightarrow 0$; $\{2\} \rightarrow 1$	$AND(\tilde{m}_0, \tilde{m}_1)$
NOR	$\boldsymbol{c}_0 + \boldsymbol{c}_1$	$\{1, 2\} \rightarrow 0$; $\{0\} \rightarrow 1$	$NOR(\tilde{m}_0, \tilde{m}_1)$
XNOR	$\boldsymbol{c}_0 + \boldsymbol{c}_1$	$\{1\} \rightarrow 0$; $\{0, 2\} \rightarrow 1$	$XNOR(\tilde{m}_0, \tilde{m}_1)$
NAND	$\boldsymbol{c}_0 + \boldsymbol{c}_1$	$\{2\} \rightarrow 0$; $\{0, 1\} \rightarrow 1$	$NAND(\tilde{m}_0, \tilde{m}_1)$
NOT	$(\mathbf{0}, q/4) - \boldsymbol{c}_0$	no bootstrapping	$NOT(\tilde{m}_0)$

5.2 Large FHE gates

Our scheme can be used to compute more complex operations on large module plaintexts efficiently. Firstly note that since we need set $\Delta'/2 = \lfloor Q/h \rfloor$ to be bigger than the final error, for a large output message space $\mathbb{Z}_h$, the security strength of our scheme is reduced. For example if we set $h = \Theta(\sqrt{N})$, the approximate factor in Theorem 2 will amplifies to $\tilde{O}(N^{1.5}\lambda)$ in the quantum case and $\tilde{O}(N^2\lambda)$ in the classical case.

LUT function. A Look Up Table (LUT) is an array that replaces runtime computation with a simpler array indexing operation. We show how to evaluate LUT functions over encrypted data. A boolean LUT is defined as $f_{LUT} : \mathbb{Z}_2^r \rightarrow \mathbb{Z}_2^v$. For ciphertexts $\{\boldsymbol{c}_1, \ldots, \boldsymbol{c}_r\}$ where $\boldsymbol{c}_i = LWE_{\boldsymbol{s}}(\Delta \cdot \tilde{m}_i)$, $\tilde{m}_i \in \{0, 1\}$ and

$\varDelta = \lfloor \frac{q}{2^r} \rfloor$, the goal is to compute the encryptions of a set of bits $\{z_1, \ldots, z_v\} \leftarrow f_{LUT}(\{\tilde{m}_1, \ldots, \tilde{m}_r\})$, where $z_1, \ldots, z_v \in \{0, 1\}$.

We can follow the strategy in [11]. One can first compute $\sum_{i \in [r]} 2^{i-1} \cdot \boldsymbol{c}_i$, which is an encryption of $\tilde{m} = \sum_{i \in [r]} 2^{i-1} \cdot \tilde{m}_i$. Obviously for every $i \in [v]$, there is a mapping $func_i(\tilde{m}) \to z_i$ that can compute every bit z_i from $\tilde{m}$, so we can evaluate every $func_i$ separately to obtain the encryption for every z_i. For every output bit of LUT functions, there needs one bootstrapping operation, and in every bootstrapping, set the mapping $func$ as $func_i : \mathbb{Z}_{2^r} \to \mathbb{Z}_2$, which outputs the i-th bit of the LUT function. Then after generating the bootstrapping keys **BootKey** $\leftarrow$ **BootKeyGen**(**SK**, **s**), computes

$$\mathbf{c}' \leftarrow \mathbf{Bootstrap}(\mathbf{BootKey}, \sum_{\mathbf{i} \in [\mathbf{r}]} \mathbf{2^{i-1}} \cdot \boldsymbol{c}_\mathbf{i}).$$

By Lemma 6, $\mathbf{c}'$ is an encryption of $func_i(\sum_{i \in [r]} 2^{i-1} \cdot \tilde{m}_i)$. To evaluate a LUT function with v output bits, it costs v functional bootstrappings.

Our scheme can evaluate some special functions more efficiently. Firstly note that one can evaluate the max/min function and comparison using the LUT function described above. To homomorphically compute the max/min function of two r bits numbers, it needs to evaluate a $2r$-to-r LUT function, which costs r functional bootstrappings and the input message space needs set to be $\mathbb{Z}_{2^{2r}}$. At the same time, to homomorphically compare two r bits numbers, it needs to evaluate a $2r$-to-1 LUT function, which costs one functional bootstrapping, and the input message space needs to be $\mathbb{Z}_{2^{2r}}$. Here we described a method that can homomorphically compute max/min function and comparison using just one functional bootstrapping and the input message space just needs set to be $\mathbb{Z}_{2^{r+1}}$.

Max/Min function. We show how to homomorphically compute max/min function for two numbers. Firstly, we give an operation $MaxCon(\boldsymbol{c}, x, y)$ that can homomorphically compute an encryption of $\tilde{m}$ if $\tilde{m} > x$, and an encryption of y otherwise, where $\boldsymbol{c} = LWE_{\boldsymbol{s}}(\varDelta \cdot \tilde{m})$, $\tilde{m} \in \mathbb{Z}_t$, $\varDelta = \lfloor \frac{q}{t} \rfloor$ and $x, y \in \mathbb{Z}_t$ are two const numbers. In our **Bootstrap** procedure, the input and output message space are both $\mathbb{Z}_t$; the input ciphertext is $\boldsymbol{c}$ and for the function $func$, we set it to be $\tilde{m} \leftarrow func(\tilde{m})$ if $\tilde{m} > x$, and $y \leftarrow func(\tilde{m})$ otherwise. Note that if we set $x = y$, we can homomorphically compute the smaller value between elements $\tilde{m}$ and y. Similarly, we can define $MinCon(\boldsymbol{c}, x, y)$ to homomorphically compute an encryption of $\tilde{m}$ if $\tilde{m} \le x$, and an encryption of y otherwise.

Then for $\boldsymbol{c}_0 = LWE_{\boldsymbol{s}}(\varDelta \cdot \tilde{m}_0)$ and $\boldsymbol{c}_1 = LWE_{\boldsymbol{s}}(\varDelta \cdot \tilde{m}_0)$ where $\tilde{m}_0, \tilde{m}_1 \in \mathbb{Z}_{\lfloor \frac{t}{2} \rfloor}$ and $\varDelta = \lfloor \frac{q}{t} \rfloor$, we can obtain an encryption for the smaller one between $\tilde{m}_0$ and $\tilde{m}_1$ by computing

$$MIN(\boldsymbol{c}_0, \boldsymbol{c}_1) = MaxCon(\boldsymbol{c}_0 - \boldsymbol{c}_1, \lfloor \frac{t}{2} \rfloor, 0) + \boldsymbol{c}_1.$$

If $\tilde{m}_0 < \tilde{m}_1$, we have $[\tilde{m}_0 - \tilde{m}_1]_t > \lfloor \frac{t}{2} \rfloor$, then $MaxCon(\boldsymbol{c}_0 - \boldsymbol{c}_1, \lfloor \frac{t}{2} \rfloor, 0) + \boldsymbol{c}_1$ is an encryption of $[\tilde{m}_0 - \tilde{m}_1 + \tilde{m}_1]_t = \tilde{m}_0$; if $\tilde{m}_0 \ge \tilde{m}_1$, we have $[\tilde{m}_0 - \tilde{m}_1]_t < \lfloor \frac{t}{2} \rfloor$, then the result is an encryption of $\tilde{m}_1$. Similarly, we can define

$$MAX(\boldsymbol{c}_0, \boldsymbol{c}_1) = MinCon(\boldsymbol{c}_0 - \boldsymbol{c}_1, \lfloor \frac{t}{2} \rfloor, 0) + \boldsymbol{c}_1$$

to homomorphically compute the bigger one between elements $\tilde{m}_0 \in \mathbb{Z}_{\lfloor \frac{t}{2} \rfloor}$ and $\tilde{m}_1 \in \mathbb{Z}_{\lfloor \frac{t}{2} \rfloor}$.

Comparison. Next we show how to homomorphically compare two numbers. For $\boldsymbol{c}_0 = LWE_{\boldsymbol{s}}(\Delta \cdot \tilde{m}_0)$ and $\boldsymbol{c}_1 = LWE_{\boldsymbol{s}}(\Delta \cdot \tilde{m}_0)$ where $\tilde{m}_0, \tilde{m}_1 \in \mathbb{Z}_{\lfloor \frac{t}{2} \rfloor}$ and $\Delta = \lfloor \frac{q}{t} \rfloor$, we define the comparison function as $Compare(\boldsymbol{c}_0, \boldsymbol{c}_1)$, which outputs an encryption of 1 if $\tilde{m}_0 \geq \tilde{m}_1$ and an encryption of 0 otherwise. In the **Bootstrap** procedure, the input ciphertext of our bootstrapping scheme is set to be $\boldsymbol{c}_0 - \boldsymbol{c}_1$; the output message space is set to be $\mathbb{Z}_2$. For function $func$, set $1 \leftarrow func(\tilde{m})$ if $\tilde{m} \leq \lfloor \frac{t}{2} \rfloor$, and $0 \leftarrow func(\tilde{m})$ otherwise. Note that the input of our bootstrapping function is an encryption of $[\tilde{m}_0 - \tilde{m}_1]_t$. Then if $\tilde{m}_0 \geq \tilde{m}_1$, we have that $[\tilde{m}_0 - \tilde{m}_1]_t \leq \lfloor \frac{t}{2} \rfloor$ and the output ciphertext is an encryption of message 1; if $\tilde{m}_0 < \tilde{m}_1$, there is $[\tilde{m}_0 - \tilde{m}_1]_t > \lfloor \frac{t}{2} \rfloor$ and the output ciphertext decrypts to 0.

Note that to evaluate max/min function and comparison, we just need one functional bootstrapping, and the input message space just needs set to be $\mathbb{Z}_{2^{r+1}}$ where r is the bit size of the input numbers.

Arbitrary Functions. Another sample example is to homomorphically compute an function

$$y_i \leftarrow map(x) \text{ when } x \in A_i$$

where $x \in \mathbb{Z}_t = \bigcup A_i$ and $y_i \in \mathbb{Z}_h$. In the **Bootstrap** procedure, we just need to set the input message space and the output message space as $\mathbb{Z}_t$ and $\mathbb{Z}_h$, respectively, and then let $y_i \leftarrow func(x)$ if $x \in A_i$. We need just one functional bootstrapping to evaluate this function, compared with $\lceil log_2 h \rceil$ functional bootstrappings if applying the LUT to realized this function.

6 Discussion and Conclusion

In this paper, we propose an optimized bootstrapping homomorphic encryption from LWE. We decrease the approximation factor of the underlying worst-case lattice factor assumption from $\tilde{O}(N^{2.5})$ to $\tilde{O}(N^2)$. We are the first to present how to perform more operations per bootstrapping in the LWE setting.

Discussions. It is an inherent problem for the implementation of an LWE-based bootstrapping scheme (with a huge storage space consumption). Although we have implemented the scheme for a small parameter, it is only used to verify the correctness of the scheme. There is still a long way to realize the practical application of the LWE-based bootstrapping scheme.

Note that our scheme uses the homomorphic matrix-vector multiplication instead of homomorphic matrix-matrix multiplication, which improves the efficiency of the scheme, but with a large dimension of the matrix/vector, the ciphertext expansion ratio is also very large. When the calculation of complex functions is not considered, similar to the scheme in [1,22], we can further use the Chinese Remainder Theorem (CRT) to improve the efficiency. That is, set q as the product of some primes, and then encrypt the matrix under each small prime dimension separately, which can reduce the ciphertext expansion ratio and improve the efficiency. However, to evaluate large FHE gates, our scheme requires that the values in the vector are some special encoding values, and we need the vector to have the "cyclic rotation" property when the permutation matrix is multiplied with the vector. After using the CRT, these special properties of the vector will be destroyed. So there is a trade-off, that is, if we want to further use the CRT to improve the efficiency and storage, we can not evaluate large FHE gates, and if we want our scheme to have a function of computing complex functions, we can not use the CRT. In Table 1, the reason why our scheme is relatively poor in storage is that we don't use CRT for optimization. However, it is because we don't use CRT, we can apply our new method to operate the nonlinear part of the decryption and hence the extra error accumulation is avoided (compared with [1,22]), which leads to the best lattice SVP approximate factor in Table 1. An open problem is how to use the Chinese Remainder Theorem to further optimizing the scheme while making the scheme can evaluate large FHE gates.

References

1. Alperin-Sheriff, J., Peikert, C.: Faster bootstrapping with polynomial error. In: Garay, J.A., Gennaro, R. (eds.) Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I. Lecture Notes in Computer Science, vol. 8616, pp. 297–314. Springer (2014)
2. Barrington, D.A.M.: Bounded-width polynomial-size branching programs recognize exactly those languages in nc^1. In: Hartmanis, J. (ed.) Proceedings of the 18th Annual ACM Symposium on Theory of Computing, May 28-30, 1986, Berkeley, California, USA. pp. 1–5. ACM (1986)
3. Bonnoron, G., Ducas, L., Fillinger, M.: Large FHE gates from tensored homomorphic accumulator. In: Joux, A., Nitaj, A., Rachidi, T. (eds.) Progress in Cryptology - AFRICACRYPT 2018 - 10th International Conference on Cryptology in Africa, Marrakesh, Morocco, May 7-9, 2018, Proceedings. Lecture Notes in Computer Science, vol. 10831, pp. 217–251. Springer (2018)
4. Boura, C., Gama, N., Georgieva, M., Jetchev, D.: CHIMERA: combining ring-lwe-based fully homomorphic encryption schemes. J. Math. Cryptol. **14**(1), 316–338 (2020)
5. Brakerski, Z.: Fully homomorphic encryption without modulus switching from classical gapsvp. In: Safavi-Naini, R., Canetti, R. (eds.) Advances in Cryptology - CRYPTO 2012 - 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings. Lecture Notes in Computer Science, vol. 7417, pp. 868–886. Springer (2012)
6. Brakerski, Z., Gentry, C., Halevi, S.: Packed ciphertexts in lwe-based homomorphic encryption. In: Kurosawa, K., Hanaoka, G. (eds.) Public-Key Cryptography -

PKC 2013 - 16th International Conference on Practice and Theory in Public-Key Cryptography, Nara, Japan, February 26 - March 1, 2013. Proceedings. Lecture Notes in Computer Science, vol. 7778, pp. 1–13. Springer (2013)

7. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (leveled) fully homomorphic encryption without bootstrapping. In: Goldwasser, S. (ed.) Innovations in Theoretical Computer Science 2012, Cambridge, MA, USA, January 8-10, 2012. pp. 309–325. ACM (2012)
8. Brakerski, Z., Vaikuntanathan, V.: Efficient fully homomorphic encryption from (standard) LWE. In: Ostrovsky, R. (ed.) IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011. pp. 97–106. IEEE Computer Society (2011)
9. Brakerski, Z., Vaikuntanathan, V.: Fully homomorphic encryption from ring-lwe and security for key dependent messages. In: Rogaway, P. (ed.) Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings. Lecture Notes in Computer Science, vol. 6841, pp. 505–524. Springer (2011)
10. Brakerski, Z., Vaikuntanathan, V.: Lattice-based FHE as secure as PKE. In: Naor, M. (ed.) Innovations in Theoretical Computer Science, ITCS'14, Princeton, NJ, USA, January 12-14, 2014. pp. 1–12. ACM (2014)
11. Carpov, S., Izabachène, M., Mollimard, V.: New techniques for multi-value input homomorphic evaluation and applications. In: Matsui, M. (ed.) Topics in Cryptology - CT-RSA 2019 - The Cryptographers' Track at the RSA Conference 2019, San Francisco, CA, USA, March 4-8, 2019, Proceedings. Lecture Notes in Computer Science, vol. 11405, pp. 106–126. Springer (2019)
12. Cheon, J.H., Kim, A., Kim, M., Song, Y.S.: Homomorphic encryption for arithmetic of approximate numbers. In: Takagi, T., Peyrin, T. (eds.) Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I. Lecture Notes in Computer Science, vol. 10624, pp. 409–437. Springer (2017)
13. Chillotti, I., Gama, N., Georgieva, M., Izabachène, M.: Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. In: Cheon, J.H., Takagi, T. (eds.) 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. Lecture Notes in Computer Science, vol. 10031, pp. 3–33 (2016)
14. Chillotti, I., Gama, N., Georgieva, M., Izabachène, M.: Faster packed homomorphic operations and efficient circuit bootstrapping for TFHE. In: Takagi, T., Peyrin, T. (eds.) Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I. Lecture Notes in Computer Science, vol. 10624, pp. 377–408. Springer (2017)
15. Chillotti, I., Joye, M., Paillier, P.: Programmable bootstrapping enables efficient homomorphic inference of deep neural networks. IACR Cryptol. ePrint Arch. **2021**, 91 (2021), `https://eprint.iacr.org/2021/091`
16. Cramer, R., Ducas, L., Peikert, C., Regev, O.: Recovering short generators of principal ideals in cyclotomic rings. In: Fischlin, M., Coron, J. (eds.) Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-

12, 2016, Proceedings, Part II. Lecture Notes in Computer Science, vol. 9666, pp. 559–585. Springer (2016)

17. Cramer, R., Ducas, L., Wesolowski, B.: Short stickelberger class relations and application to ideal-svp. In: Coron, J., Nielsen, J.B. (eds.) Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part I. Lecture Notes in Computer Science, vol. 10210, pp. 324–348 (2017)
18. Ducas, L., Micciancio, D.: FHEW: bootstrapping homomorphic encryption in less than a second. In: Oswald, E., Fischlin, M. (eds.) Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I. Lecture Notes in Computer Science, vol. 9056, pp. 617–640. Springer (2015)
19. Fan, J., Vercauteren, F.: Somewhat practical fully homomorphic encryption. IACR Cryptol. ePrint Arch. **2012**, 144 (2012), `http://eprint.iacr.org/2012/144`
20. Gentry, C.: A fully homomorphic encryption scheme. In: PhD thesis, Stanford University (2009), `http:crypto.stanford.edu/craig.`
21. Gentry, C., Sahai, A., Waters, B.: Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In: Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I. pp. 75–92 (2013)
22. Hiromasa, R., Abe, M., Okamoto, T.: Packing messages and optimizing bootstrapping in GSW-FHE. In: Public-Key Cryptography - PKC 2015 - 18th IACR International Conference on Practice and Theory in Public-Key Cryptography, Gaithersburg, MD, USA, March 30 - April 1, 2015, Proceedings. pp. 699–715 (2015)
23. Lenstra, A.K., Lenstra, H.W., Lovsz, L.: Factoring polynomials with rational coefficients. Mathematische Annalen **261**(4) (1982)
24. Micciancio, D., Mol, P.: Pseudorandom knapsacks and the sample complexity of LWE search-to-decision reductions. In: Rogaway, P. (ed.) Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings. Lecture Notes in Computer Science, vol. 6841, pp. 465–484. Springer (2011)
25. Micciancio, D., Peikert, C.: Trapdoors for lattices: Simpler, tighter, faster, smaller. In: Pointcheval, D., Johansson, T. (eds.) Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings. Lecture Notes in Computer Science, vol. 7237, pp. 700–718. Springer (2012)
26. Peikert, C.: Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In: Mitzenmacher, M. (ed.) Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009. pp. 333–342. ACM (2009)
27. Peikert, C., Vaikuntanathan, V., Waters, B.: A framework for efficient and composable oblivious transfer. In: Wagner, D.A. (ed.) Advances in Cryptology - CRYPTO 2008, 28th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2008. Proceedings. Lecture Notes in Computer Science, vol. 5157, pp. 554–571. Springer (2008)
28. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: Gabow, H.N., Fagin, R. (eds.) Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, May 22-24, 2005. pp. 84–93. ACM (2005)

29. Vershynin, R.: Introduction to the non-asymptotic analysis of random matrices. In: Eldar, Y.C., Kutyniok, G. (eds.) Compressed Sensing, pp. 210–268. Cambridge University Press (2012)