Efficient Lattice Gadget Decomposition Algorithm with Bounded Uniform Distribution*

Sohyun Jeon, Hyang-Sook Lee, Jeongeun Park

Department of Mathematics, Ewha Womans University, Republic of Korea
jeonsh099@ewhain.net, hsl@ewha.ac.kr, jungeun7430@ewhain.net

Abstract. A gadget decomposition algorithm is commonly used in many advanced lattice cryptography applications which support homomorphic operation over ciphertexts to control the noise growth. For a special structure of a gadget, the algorithm is digit decomposition. If such algorithm samples from a subgaussian distribution, that is, the output is randomized, it gives more benefits on output quality. One of important advantages is Pythagorean additivity which makes resulting noise contained in a ciphertext grow much less than naive digit decomposition. Therefore, the error analysis becomes cleaner and tighter than the use of other measures like ℓ_2 and ℓ_∞. Even though such advantage can also be achieved by the use of discrete Gaussian sampling, it is not preferable for practical performance due to large factor in resulting noise and the complex computation of exponential function, whereas more relaxed probability condition is required for subgaussian distribution. Nevertheless, subgaussian sampling has barely received an attention so far, thus no practical algorithms was implemented before an efficient algorithm is presented by Genis et al., recently.
In this paper, we present a practically efficient gadget decomposition algorithm where output follows a subgaussian distribution. We parallelize the existing practical subgaussian gadget decomposition algorithm, using bounded uniform distribution. Our algorithm is divided into two independent subalgorithms and only one algorithm depends on input. Therefore, the other algorithm can be considered as pre-computation. As an experimental result, our algorithm performs over 50% better than the existing algorithm.

Keywords: Subgaussian distribution, Gadget decomposition, Bounded uniform distribution, Lattice gadget

1 Introduction

A cryptosystem requires an efficiently computable function of which inverse function is hard to be computed without secret information called trapdoor. In lattice based cryptography, a function $f_{\mathbf{A}}(\mathbf{x}) = \mathbf{Ax}$ is used for Short Integer Solution (SIS) or Inhomogeneous Short Integer Solution (ISIS) and $g_{\mathbf{A}}(\mathbf{x}, \mathbf{e}) =$

* This is the full paper version of the journal paper which will appear in IEEE Access

$\mathbf{x}^t\mathbf{A} + \mathbf{e}^t$ is used for Learning With Error (LWE) with a proper random matrix $\mathbf{A}$. Computing its inverse function of each is known as computationally hard without trapdoor, in other words, one who holds a trapdoor can easily compute the inverse. The problem to compute an inverse of $f_{\mathbf{A}}$ and $g_{\mathbf{A}}$ in a lattice is called SIS problem or ISIS problem (depending on a function value) and LWE problem, respectively. If a trapdoor of [22] is known, the inverting problems of $f_{\mathbf{A}}$, $g_{\mathbf{A}}$ are transformed to much easier problems to compute inverse functions of $f_{\mathbf{G}}$ and $g_{\mathbf{G}}$ for a special matrix $\mathbf{G}$ called a gadget matrix of a given lattice. The inversion problem of $f_{\mathbf{G}}$ is particularly called gadget decomposition problem. A gadget matrix is a matrix that defines a lattice which is a discrete additive subgroup of $\mathbb{R}^n$ for a positive integer n.

Our main focus is a subgaussian (gadget) decomposition algorithm (or subgaussian sampling for short) out of several types of gadget decomposition algorithm. It is a simply randomized digit decomposition. It samples a short solution of the inversion problem following a subgaussian distribution. Informally, any distribution where its tails are bounded by a Gaussian distribution is a subgaussian distribution, hence it has more relaxed condition in terms of the probability function than a Gaussian distribution. The subgaussian sampling has a significant property, Pythagorean additivity, which mainly has an important role in lattice based cryptosystems that support homomorphic operations over ciphertexts. In particular, homomorphic encryption (HE) is a notable example. In lattice based HE schemes where all ciphertexts contain noisy terms, a ciphertext gets larger noise when an homomorphic evaluation is performed. In fact, it is important to keep the noise growth small as much as possible for further operations. For the purpose of that, a gadget decomposition algorithm (denoted by $\mathbf{G}^{-1}(\cdot)$) is used in an evaluation algorithm or encryption algorithm, where an error vector $\mathbf{e}$ contained in an input ciphertext becomes $\mathbf{G}^{-1}(\mathbf{e})$ in an output ciphertext. Even though $\mathbf{G}^{-1}(\cdot)$ is a deterministic digit decomposition, it can control the noise growth by keeping each output component less than a certain digit. If every component of $\mathbf{G}^{-1}(\mathbf{e})$ independently follows a subgaussian distribution, Pythagorean additivity allows resulting noise smaller only as $O(\sqrt{k})$ than digit decomposition which scales the noise linearly with k, where k is a dimension of an output vector. Moreover, it also enables tighter analysis about error growth (average case) as discussed in [2].

Discrete Gaussian sampling also achieves Pythagorean additivity property since a Gaussian distribution is also a subgaussian distribution by definition. However, it is not attractive in practice due to the complex computation of probability and its output quality. The coefficient of output is larger than binary decomposition with an additional factor $\Omega(\sqrt{\log k})$. Therefore, subgaussian sampling is regarded as a potential alternative [2, 14] since it can achieve not only significant properties of the Gaussian distribution such as Pythagorean additivity without the factor $\Omega(\sqrt{\log k})$ but also somewhat faster implementation performance due to its relaxed probability condition. Even though subgaussian sampling has much more attractive advantages in many advanced cryptosystems, the importance of the subgaussian distribution has mainly been studied in theoretical way [2].

In fact, the practical HE schemes [9,10,13] and their variants [7,8,20] use a naive digit decomposition algorithm and heuristically assume that each output coefficient of such algorithm independently follows a subgaussian distribution for practical purpose. Then they analyze the output quality by using the benefits of subgaussian sampling such as Pythagorean additivity. However, thanks to the first practical implementation of subgaussian sampling with a triangular distribution recently proposed by Genise et al. [14], the deterministic algorithms used in many HE schemes can be replaced by their algorithm with small computation time overhead and without heuristic independence assumption. Furthermore, such digit decomposition algorithm can be used for many of other lattice based cryptographic primitives which support homomorphic operations such as GSW-style HE [5,16,19,24] , homomorphic signature scheme [17], Attribute-Based Encryption (ABE) schemes [3,12], homomorphic identity-based encryption [11] and more. Therefore, any efficient subgaussian sampling can replace the naive (deterministic) digit decomposition algorithm in applications providing better aforementioned advantages.

1.1 Subgaussian Gadget Decomposition Algorithm

The gadget decomposition algorithm finds a solution $\mathbf{x}$ such that $\mathbf{Gx} = \mathbf{u} \mod q$, given a gadget matrix $\mathbf{G}$ and a vector $\mathbf{u}$. If the output follows a subgaussian distribution, we call it subgaussian gadget decomposition algorithm (or subgaussian decomposition for short). In fact, the set of solution $\mathbf{x}$ is also a lattice denoted by $\Lambda^{\perp}_{\mathbf{u}}(\mathbf{G})$. And there is a lattice which is the set of the solutions such that $\mathbf{Gx} = \mathbf{0} \mod q$ denoted by $\Lambda^{\perp}_{q}(\mathbf{G})$. Then the pre-image sampling is carried out over the lattice $\Lambda^{\perp}_{\mathbf{u}}(\mathbf{G})$ which is a coset of $\Lambda^{\perp}_{q}(\mathbf{G})$. The existing such algorithm [14] finds a solution in the following way:

(1) Compute $\mathbf{z}$ such that $\mathbf{Gz} = \mathbf{u} \mod q$. Each component of $\mathbf{z}$ is in range from 0 to b.
(2) Sample $\mathbf{e} \leftarrow \mathcal{SG}_{\Lambda^{\perp}_{q}(\mathbf{G}),-\mathbf{z},s}$, where $\mathcal{SG}_{\Lambda^{\perp}_{q}(\mathbf{G}),-\mathbf{z},s}$ is a subgaussian distribution with parameter s over the lattice $\Lambda^{\perp}_{q}(\mathbf{G})$ centered at $-\mathbf{z}$.
(3) Output $\mathbf{x} = \mathbf{z} + \mathbf{e} \in \Lambda^{\perp}_{\mathbf{u}}(\mathbf{G})$

More precisely, the first step (1) gives a deterministic solution and it is a b-ary decomposition. Then the second step (2) randomly samples a lattice vector over $\Lambda^{\perp}_{\mathbf{u}}(\mathbf{G})$ among the points near to the vector $-\mathbf{z}$ which is outside the lattice. The probability depends on $-\mathbf{z}$ and it follows a triangular distribution. Since the distribution is triangular in some bounded area and the probability for others outside the bound is 0, where its tails are bounded by a Gaussian distribution with parameter s, hence it is a subgaussian distribution. The Euclidean norm of the output vector is bounded by $b\sqrt{kn}$, where n is the dimension of the lattice, $k = \lceil \log_b q \rceil$.

There are two such algorithms depending on a relation between the base b and the modulus q. If b is a power of q, there is nothing complex procedure in the algorithm since there is pretty simple basis of the lattice $\Lambda^{\perp}_{q}(\mathbf{G})$. However, if

q is not a power of b, it is not straightforward to do the second step (2) due to slightly different basis of $\Lambda_q^\perp(\mathbf{G})$. In order to do simpler sampling, the basis is factorized to two matrices, one of which is the sparse matrix $\hat{\mathbf{S}}$ and the other is triangular matrix $\hat{\mathbf{D}}$, then the $\mathbf{z}$ is transformed to another vector called target vector $\mathbf{t}$ in a lattice transformed by $\hat{\mathbf{S}}^{-1}$, that is $\mathbf{t} = \hat{\mathbf{S}}^{-1}\mathbf{z}$ outside $\mathcal{L}(\hat{\mathbf{D}})$. Then the step (2) is carried out over the lattice generated by the triangular matrix $\hat{\mathbf{D}}$, denoted by $\mathcal{L}(\hat{\mathbf{D}})$ centered at $-\mathbf{t}$. After step (3) the result is transformed to an element in the original lattice $\Lambda_q^\perp(\mathbf{G})$ by the sparse matrix. The transformation follows: $\hat{\mathbf{S}}(\mathbf{t} + \mathcal{L}(\hat{\mathbf{D}})) = \mathbf{z} + \hat{\mathbf{S}}\mathcal{L}(\hat{\mathbf{D}}) = \mathbf{z} + \Lambda_q^\perp(\mathbf{G}) = \Lambda_{\mathbf{u}}^\perp(\mathbf{G})$.

Overall, the algorithms are sequential in both cases, since the probability of sampling a point $\mathbf{e} \in \Lambda_q^\perp(\mathbf{G})$ depends on the distance from $\mathbf{z} \in \Lambda_{\mathbf{u}}^\perp(\mathbf{G})$ which is computed ahead. In other words, $\mathbf{e}$ must be chosen after $\mathbf{z}$ is determined.

1.2 Our Contribution

We improve an existing efficient subgaussian gadget decomposition algorithm [14] with a *noncentral* bounded uniform distribution which is a subgaussian distribution thanks to the generalisation of [25]. Our algorithm theoretically achieves Pythagorean error growth, linear time complexity as [14], and much faster computation cost. Hence, it can be an alternative of naive digit decomposition used in FHE schemes without any extra assumption in practice. As an experimental result, it performs much better (over 50%) than existing such algorithm [14]. Comparing to actual deterministic algorithm in HE schemes, it has only overhead with addition between two polynomials of degree n or vectors of dimension n.

Our algorithm consists of mainly two independent subalgorithms called `Decomposition` and `Sampling`. The `Decomposition` only depends on the input of the whole algorithm, hence the `sampling` which samples a random element from a bounded uniform distribution can be considered as offline phase which is a kind of precomputation.

Due to the bounded uniform distribution, we do not need a target vector in the step (2) for arbitrary q case. Regardless of a vector computed in the first step, a vector in $\Lambda_q^\perp(\mathbf{G})$ is sampled with the same (nonzero) probability in a bounded area and others are with zero probability. For more detail, we sample a vector from $\mathcal{L}(\hat{\mathbf{D}})$ with the bounded uniform distribution (in the step (2)) then transform it with $\hat{\mathbf{S}}$. In the step (3), we add the both independent two vectors and then the result is in the original coset: $\mathbf{z} + \hat{\mathbf{S}}\mathcal{L}(\hat{\mathbf{D}}) = \mathbf{z} + \Lambda_q^\perp(\mathbf{G}) = \Lambda_{\mathbf{u}}^\perp(\mathbf{G})$.

As a result, we can save the computation time in three ways: i) Computing the probability function (uniform) in Step (2) is simpler than [14]'s (triangular). ii) We do not need to compute the target vector. iii) Step (2) can be considered as precomputation (offline phase) due to its independent procedure to the first step. Indeed, we improve the efficiency by over 50% in terms of computation time. Due to the bounded uniform distribution, we can reduce the computation time of the algorithm, but it outputs the larger size of a vector with higher probability than using the triangular distribution in the same bounded area. However, the Euclidean norm of the output is also bounded by $b\sqrt{kn}$ as [14].

In addition, our algorithms can be applied to a system that uses the Chinese Remainder Transformation/Representation (CRT) gadgets for a large modulus discussed in Appendix A.

1.3 Road Map

We review subgaussian random variables and some basic notions of lattice in Section 2. We present our algorithms for two cases: for a special modulus and arbitrary modulus in Section 3. We also show that the output follows a subgaussian distribution in the section. We compare the magnitude of output and the computation time of our algorithm using PALISADE library [26] to an existing algorithm in Section 4.

2 Preliminaries

Notation: Numbers are denoted as small letters, such as $a \in \mathbb{Z}$, vectors as bold small letters, $\mathbf{a} \in \mathbb{Z}^n$, and matrices as capital bold letters, $\mathbf{A} \in \mathbb{R}^{n \times n}$. We denote by $\langle \mathbf{v}, \mathbf{w} \rangle$ the dot product of two vectors $\mathbf{v}, \mathbf{w}$. For a vector $\mathbf{x}$, $\mathbf{x}[i]$ or x_i denotes the i-th component scalar of $\mathbf{x}$. We use the Euclidean norm as a default norm for a vector $\mathbf{x}$. $[u]_b^k$ denotes the vector $\in \{0, \ldots, b-1\}^k$ which is b-ary decomposition of u and satisfies $\langle [1, b, b^2, \ldots, b^{k-1}], [u]_b^k \rangle = u$ for an integer base $b > 0$ and an integer $0 < u < b^k$. A notation $a \xleftarrow{\$} S$ means that a is chosen uniformly from a set S. We call a uniform distribution on a bounded set as a bounded uniform distribution.

2.1 Subgaussian Random Variables

We explain subgaussian random variables and the significant properties in this section.We describe the general definition for a univariate δ-subgaussian random variable for some $\delta \geq 0$ as in [22].

Definition 1. *A random variable V over $\mathbb{R}$ is δ-subgaussian ($\delta \geq 0$) with parameter $s > 0$ if its moment generating function satisfies*

$$\mathbb{E}[\exp(2\pi t V)] \leq \exp(\delta) \exp(\pi s^2 t^2)$$

for all $t \in \mathbb{R}$.

In many papers dealing with subgaussian random variable, 0-subgaussian random variable (when the mean is zero) is mainly considered. For better understanding, if a random variable V is 0-subgaussian with parameter s, then its tails are dominated by a Gaussian distribution parameterized by s. If a random variable V is centered (i.e. $\mathbb{E}[V] = 0$) and bounded with B, then V is 0-subgaussian with parameter $B\sqrt{2\pi}$ [27].

There is a simple and significant relation between 0-subgaussian and δ-subgaussian [25] for nonzero δ. We introduce some useful lemmas of them below.

Lemma 1. *Let V be a real-valued δ-subgaussian random variable with parameter s, for some $s \in \mathbb{R} > 0$. Then $\delta = 0$ if and only if $\mathbb{E}[V] = 0$.*

Lemma 2. *If V is a real-valued δ-subgaussian random variable with parameter s, then the centered random variable $V_0 = V - \mathbb{E}[V]$ is 0-subgaussian with parameter s.*

As defined in [25], a random variable V is a noncentral subgaussian with parameter s. In other words, Lemma 2 shows that a noncentral subgaussian random variable, equivalently δ-subgaussian random variable, can be described as 0-subgaussian by shifting its mean. Also, if there exists a 0-subgaussian random variable, we can make it to a δ-subgaussian random variable for nonzero δ by shifting the variable with some real number as stated in Lemma 3.

Lemma 3. *If $\overline{V}$ is a 0-subgaussian with parameter s, then the real-valued shifted random variable $V = \overline{V} + \alpha$ for $\alpha \in \mathbb{R}$ is a δ-subgaussian with parameter s for some nonnegative real-valued δ.*

There is one of significant properties of δ-subgaussian random variable, which is Pythagorean additivity generalised in [25].

Lemma 4. *Let V be noncentral subgaussian random variable, equivalently δ-subgaussian, with parameter α and W conditioned on V taking any value be δ-subgaussian with parameter β. Then, $V + W$ is δ-subgaussian with parameter $\sqrt{\alpha^2 + \beta^2}$.*

Using the proof of Lemma 4 (see Lemma 2.1 of [14]), we can show that if each coefficient of a random vector is subgaussian with parameter s, then the vector is also subgaussian with parameter s [25]. More general fact is below.

Lemma 5. *If $\mathbf{y}$ is a discrete random vector over $\mathbb{R}^n$ such that each component y_i of $\mathbf{y}$ is δ-subgaussian with parameter s_i, then the vector $\mathbf{y}$ is δ-subgaussian with parameter $\max_i\{s_i\}$, for $i \in \{1, \ldots, n\}$.*

We call such vector a δ-subgaussian (random) vector with parameter s, for some positive real number s. We call it a subgaussian vector if $\delta = 0$ throughout the paper for convenience. In our main algorithm, we use a linear transformation of a δ-subgaussian vector with a certain parameter, and its output is also a δ-subgaussian with a different parameter.

Lemma 6 ([21]). *Let $\delta, \alpha \geq 0$ and $\mathbf{x}$ be a random vector in $\mathbb{R}^n$, and let $\mathbf{A}$ be a $n \times n$ matrix. Suppose that the random vector $\mathbf{x}$ is δ-subgaussian with parameter α. Then $\mathbf{Ax}$ is δ-subgaussian with parameter $\alpha\lambda_{max}(\mathbf{AA}^t)^{\frac{1}{2}}$, where $\lambda_{max}(\cdot)$ is the largest eigenvalue.*

2.2 Lattices and Gadget Matrix

A lattice Λ with dimension n and basis $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_k]$ is a set of all linear combinations of the basis vectors with coefficients in $\mathbb{Z}$. A lattice of which the basis vectors consist of integer coordinates is called an integral lattice. i.e. a lattice which is a sublattice of $\mathbb{Z}^n$. We consider a special family of lattice, q-ary lattice. A q-ary lattice is a full-rank integral lattice with $q \cdot \mathbb{Z}^k$ as a sublattice. The lattice is defined by means of a matrix called primitive matrix [22]. Let $q > 0$ be an integer modulus and $m > l > n$. A matrix $\mathbf{P} \in \mathbb{Z}^{n \times m}$ is primitive if $\mathbf{P}\mathbb{Z}^m = \mathbb{Z}_q^n$. For a fixed matrix $\mathbf{P} \in \mathbb{Z}^{n \times m}$, we define the following lattice:

$$\Lambda_q^{\perp}(\mathbf{P}) = \{\mathbf{z} \in \mathbb{Z}^m \ : \ \mathbf{P}\mathbf{z} = \mathbf{0} \mod q\}.$$

For any $\mathbf{u} \in \mathbb{Z}_q^n$ admitting an integral solution to $\mathbf{P}\mathbf{z} = \mathbf{u} \mod q$, define the following shifted lattices as the cosets of $\Lambda_q^{\perp}(\mathbf{P})$:

$$\Lambda_{\mathbf{u}}^{\perp}(\mathbf{P}) = \{\mathbf{z} \in \mathbb{Z}^m \ : \ \mathbf{P}\mathbf{z} = \mathbf{u} \mod q\}.$$

Definition 2. *The Short Integer Solution problem (SIS) is defined as follows: given an integer q, a random integer matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and a real β, find a nonzero integer vector $\mathbf{z} \in \Lambda_q^{\perp}(\mathbf{A}) \backslash \{\mathbf{0}\}$ with $||\mathbf{z}|| \leq \beta$.*

If $\mathbf{u}$ is a zero vector, the problem is called Inhomogeneous Short Integer Solution problem (SIS). The above ISIS problem is known as hard as other lattice hard problems Gap-SVP, SIVP [1,15,23]. However, the problem becomes straightforward to solve if there exists an additional secret information called trapdoor. The problem is turned to a gadget decomposition problem which is much easier to solve with a special matrix $\mathbf{G}$ called a gadget matrix [22], given a valid trapdoor.

The gadget matrix denoted by $\mathbf{G}$ is also a primitive matrix of a lattice which satisfies $\mathbf{G} \cdot \mathbb{Z}^w = \mathbb{Z}_q^n$, where $\mathbf{G} \in \mathbb{Z}_q^{n \times w}$. The matrix $\mathbf{G}$ is constructed with a primitive vector (or gadget vector) $\mathbf{g} = [g_1, \ldots, g_k]$ which defines a k-dimensional lattice $\Lambda^{\perp}(\mathbf{g}^t) \in \mathbb{Z}^k$ such that the greatest common divisor of $(g_1, \ldots, g_k, q)$ is 1. We use the same gadget matrix $\mathbf{G}$ defined as $\mathbf{I} \otimes \mathbf{g}^t$ as [22], where $\mathbf{g} = [1, b, b^2, \ldots, b^{k-1}]$ for a positive integer b.

If the solution of a gadget decomposition algorithm follows a subgaussian distribution, we call it subgaussian (gadget) decomposition problem.

Definition 3. *Let $\mathbf{G} \in \mathbb{Z}_q^{n \times w}$ be a primitive matrix. The subgaussian decomposition problem with parameter s for $\mathbf{G}$ is to sample a δ-subgaussian vector $\mathbf{x} \in \mathbb{Z}^w$ with parameter s such that $\mathbf{G}\mathbf{x} = \mathbf{u} \mod q$ for arbitrary $\mathbf{u} \in \mathbb{Z}_q^n$.*

Then the problem is defined on the above integer cosets of $\Lambda_q^{\perp}(\mathbf{G})$, that is, the goal of an algorithm to solve the problem is to sample a vector from a subgaussian distribution in $\Lambda_{\mathbf{u}}^{\perp}(\mathbf{G}) = \mathbf{z} + \Lambda_q^{\perp}(\mathbf{G})$ for an arbitrary $\mathbf{z} \in \Lambda_{\mathbf{u}}^{\perp}(\mathbf{G})$. We call the algorithm subgaussian decomposition algorithm.

3 Subgaussian Decomposition Algorithms with Bounded uniform distribution

In this section, we present our efficient subgaussian decomposition algorithm using a bounded uniform distribution. We make the two main subalgorithms independent using a noncentral bounded uniform distribution, thus it can be divided into online and offline phases. We can set the offline phase as precomputation, hence it can save overall computation time. Then we show that a uniform distribution in a bounded set is δ-subgaussian with parameter α for some positive δ, α. We construct two different algorithms according to the relation between the modulus q and the base b.

In fact, the gadget matrix $\mathbf{G}$ is structured as $\mathbf{I}_n \otimes \mathbf{g}^t$, hence our algorithm solves $\mathbf{g}^{-1}(u_i)$ for $i \in [0, \ldots, n-1]$ component-wise, given $\mathbf{u} = [u_0, \ldots, u_{n-1}]$. In other words, the algorithm solves the above problem over the lattice $\Lambda_u^\perp(\mathbf{g}^t)$ for $u \in \mathbb{Z}_q$. Our algorithm does not directly choose an element at random from the lattice $\Lambda_u^\perp(\mathbf{g}^t)$. The algorithm consists of three subalgorithms, denoted by `Decomposition`, `Sampling`, and `Addition`. `Decomposition` computes a vector in $\Lambda_u^\perp(\mathbf{g}^t)$ using a deterministic b-ary decomposition, which is only affected by an input u. `Sampling` chooses a vector in $\Lambda_q^\perp(\mathbf{g}^t)$ with a bounded uniform distribution, not in $\Lambda_u^\perp(\mathbf{g}^t)$. We stress that the two subalgorithms are independent. The final result is a randomly chosen vector in $\Lambda_u^\perp(\mathbf{g}^t)$ by combining these two vectors in `Addition`.

Then, we can consider the two cases when the modulus q is a power of b (special case) and for arbitrary q (general case).

3.1 For Power-of-Base Modulus

The procedure becomes straightforward since the basis $\mathbf{S}$ of the lattice $\Lambda_q^\perp(\mathbf{g}^t)$ is a sparse and triangular matrix when q is a power of b. First, `Decomposition` computes $\mathbf{z} \in \Lambda_u^\perp(\mathbf{g}^t)$ such that $\mathbf{g}^t\mathbf{z} = u \bmod q$ where $0 \leq z_i \leq b-1$ for all $i = 0, \ldots, k-1$. It is possible by using a deterministic b-ary decomposition. Second, `Sampling` samples a subgaussian vector $\mathbf{e} = \mathbf{S}\mathbf{y} \in \Lambda_q^\perp(\mathbf{g}^t)$ using a bounded uniform distribution. For more detail, the algorithm samples $\mathbf{e}$ at random by choosing each entry of $\mathbf{y}$ from $\{-1, 0\}$. In fact, if $\mathbf{z}$ is a solution of the problem, $\mathbf{z} + \mathbf{S}\mathbf{y}$ is also a solution for any $\mathbf{y} \in \mathbb{Z}^k$. Since $\mathbf{S}\mathbf{y} = [by_0, -y_0 + by_1, \ldots, -y_{i-1} + by_i, \ldots, -y_{k-2} + by_{k-1}]$, the possible solutions are various according to y_i's for $i \in [0, k-1]$. However, the algorithm restricts the choice of y_i in the set $\{-1, 0\}$ to keep the size of a solution small due to maintaining the hardness of (I)SIS problem in many cryptographic applications. Finally, `Addition` computes an output $\mathbf{x} = \mathbf{z} + \mathbf{e}$.

We now prove that the output of Algorithm 1 is δ-subgaussian with a parameter s for some real value δ, s. We know that a centered uniform distribution on a bounded set is a 0-subgaussian with parameter s' for some $s' \in \mathbb{R} > 0$, but our distribution is not centered at 0 (see Figure 1). However, we can shift our distribution to make the random variables centered. Reversely, one can shift

Algorithm 1 Gadget Decomposition with Bounded Uniform for $q = b^k$

Input: $u \in \{0, \ldots, q-1\}$
Output: $\mathbf{x} \in \Lambda_u^{\perp}(\mathbf{g}^t)$ distributed uniformly in a bounded set.
Let $\mathbf{e}, \mathbf{y} \leftarrow \mathbf{0}$
$y_0 \xleftarrow{\$} \{-1, 0\}$
$e_0 \leftarrow b \cdot y_0$
for $i \leftarrow 1, \ldots, k-1$ **do**
  $y_i \xleftarrow{\$} \{-1, 0\}$
  $e_i \leftarrow -y_{i-1} + b \cdot y_i$
end for
Let $\mathbf{z} \coloneqq [u]_b^k$ ($[u]_b^k$ is u's b-ary decomposition)
$\mathbf{x} \leftarrow \mathbf{z} + \mathbf{e}$
return x

the mean of a centered bounded uniform distribution to have our distribution with the same parameter. Therefore, we show that our distribution is also a δ-subgaussian distribution with parameter s for some positive real value δ, s by Lemma 3.

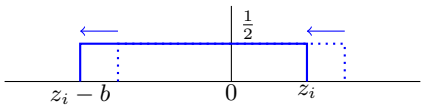


Fig. 1. A bounded uniform distribution on $\{z_i - b, z_i\}$

Theorem 1. *Let $b > 1$ be an integer base, $q > 1$ be an integer modulus, $k = \lceil \log_b q \rceil$ and $\mathbf{g}^t = [1, b, \ldots, b^{k-1}]$ be a gadget. Then the output vector of Algorithm 1 is $\frac{1}{6}$-subgaussian with parameter $b\sqrt{2\pi}$.*

Proof. Let $\hat{X}_i$ be chosen from $\{-\frac{b}{2}, \frac{b}{2}\}$ according to an uniform distribution. Then $\hat{X}_i$ is 0-subgaussian with parameter $\hat{s} = \frac{b}{2}\sqrt{2\pi}$ by Lemma 1, since $\hat{X}_i$ is $\frac{b}{2}$-bounded centered.

If a random variable X_i is an i-th component of a vector output by Algorithm 1, $X_i = z_i$ with probability $\frac{1}{2}$ or $X_i = z_i - b$ with probability $\frac{1}{2}$. Then X_i is expressed as $X_i = z_i - \frac{b}{2} + \hat{X}_i$. Let $\alpha = z_i - \frac{b}{2}$. If the parameter s of X_i is $s > \hat{s} \geq 0$ and $\delta \geq (\alpha^2\pi)/(s^2 - \hat{s}^2)$, the moment generating function of X_i satisfies

$$\begin{aligned}
\mathbb{E}[\exp(2\pi t X_i)] &= \mathbb{E}[\exp(2\pi t(\hat{X}_i + \alpha)] \\
&= \exp(2\pi t\alpha)\mathbb{E}[\exp(2\pi t\hat{X}_i)] \\
&\leq \exp(\delta)\exp\left(\pi s^2 t^2\right)
\end{aligned}$$

by the proof of Lemma 3 in [25]. Let the parameter of X_i be $s = b\sqrt{2\pi}$. Then the parameter s satisfies $b\sqrt{2\pi} > \frac{b}{2}\sqrt{2\pi}$. And since $\alpha^2 = (z_i - \frac{b}{2})^2 \leq (\frac{b}{2})^2$ and $s^2 - \hat{s}^2 = \frac{6b^2}{4}\pi$,

$$\frac{\alpha^2\pi}{s^2 - \hat{s}^2} \leq \frac{\pi b^2/4}{6\pi b^2/4} = \frac{1}{6}.$$

Since we show that each component of the output is $\frac{1}{6}$-subgaussian with parameter $b\sqrt{2\pi}$, the result holds by Lemma 5. □

3.2 For Arbitrary Modulus

For an arbitrary modulus q, the basis of the lattice $\Lambda_q^\perp(\mathbf{g}^t)$, denoted by $\mathbf{S}_q$, is somewhat complex to be dealt with since it is not a sparse matrix. Therefore, we use the fact that the basis $\mathbf{S}_q$ is factorized as sparse and triangular matrices $\mathbf{S}$ and $\mathbf{D}$. (i.e., $\mathbf{S}_q = \mathbf{SD}$.) The algorithm in [14] needs an additional subalgorithm called `Transformation` between `Decomposition` and `Sampling` to use $\mathbf{D}$. Moreover, the subalgorithms, except for `Decomposition`, are slightly different to the case when q is a power of b. The `Transformation` maps $\mathbf{z}$ to $\mathbf{t} = \mathbf{S}^{-1}\mathbf{z}$ in order to compute the probability in `Sampling`. They call such $\mathbf{t}$ a target vector. Then `Sampling` chooses a vector $\mathbf{Dy}$ from a lattice generated by $\mathbf{D}$, denoted by $\mathcal{L}(\mathbf{D})$, with a triangular distribution centered at $-\mathbf{t}$. Finally the `Addition` adds $\mathbf{t}$ and $\mathbf{Dy}$, then transforms the result to the lattice $\Lambda_u^\perp(\mathbf{g}^t)$ by multiplying $\mathbf{S}$.

However, our algorithm does not need `Transformation` even in this case since the probability to choose an element in $\Lambda_q^\perp(\mathbf{g}^t)$ is independent to $\mathbf{t}$ to which an output of `Decomposition` is transformed. For more detail, the algorithm `Sampling` samples a vector $\mathbf{Dy} \in \mathcal{L}(\mathbf{D})$ by choosing each entry of $\mathbf{y}$ from two possible values with the same probability $\frac{1}{2}$. That is, the probability is a constant, not a relation with a target vector. Then the result is transformed to $\Lambda_q^\perp(\mathbf{g}^t)$ by the matrix $\mathbf{S}$. Finally, `Addition` computes an output $\mathbf{x} = \mathbf{z}+\mathbf{e} = \mathbf{St}+\mathbf{SDy} \in \Lambda_u^\perp(\mathbf{g}^t)$.

In fact, when q is not a power of b, there is something more to be handled due to the different structure of the basis $\mathbf{S}_q$ of $\Lambda_q^\perp(\mathbf{g}^t)$. The last column vector of $\mathbf{S}_q$ is $[q]_b^k$, denoted by $[q_0, \ldots, q_{k-1}]$. We can represent the output $\mathbf{x}$ of our algorithm as $\mathbf{x} = \mathbf{z}+\mathbf{SDy} = \mathbf{z}+\mathbf{S}_q\mathbf{y}$, and the component of $\mathbf{x}$ is $x_i = z_i+(-y_{i-1}+b\cdot y_i)+q_i\cdot y_{k-1}$. The algorithm samples each y_i from $\{-1, 0\}$, hence, $-b \leq (-y_{i-1} + b \cdot y_i) \leq 1$. If the last component of $\mathbf{y}$ is -1 and $z_i < q_i$, then the value $z_i + q_i \cdot y_{k-1}$ is less than 0 for some $i \in \{0, \ldots, k-2\}$. Therefore, it may happen that the absolute value of x_i is larger than b, when $z_i \leq q_i$ for some $i \in \{0, \ldots, k-2\}$[1]. Then, the algorithm may output a vector of larger size with high probability than smaller one, which violates the aim of such algorithm. In order to fix the problem, if $y_i = -1$ and $z_i < q_i$, the algorithm in [14] adds basis vector of $\mathbf{D}$ to $\mathbf{Dy}$. That is, $y_i \leftarrow y_i + 1$. Then, $0 \leq (-y_{i-1} + b \cdot y_i) \leq b + 1$ unlike before. Since z_i and q_i are positive integers which is smaller than b, $-b < z_i + q_i \cdot y_{k-1} + b < 0$ in the case that $z_i < q_i$. As a result, the absolute value of x_i can be smaller than or equal to

[1] The last component of z, z_{k-1}, is always smaller than or equal to q_{k-1} Since $u < q$.

b. We also adapt their solution but in different subalgorithm which is `Addition` since we do not need a target vector.

Algorithm 2 Gadget Decomposition with Bounded Uniform for $q \neq b^k$

Input: $u \in \{0, \ldots, q-1\}$
Output: $\mathbf{x} \in \Lambda_u^{\perp}(\mathbf{g}^t)$ distributed uniformly in a bounded set.
Let $\mathbf{q} \coloneqq [q]_b^k$, $\mathbf{e}, \mathbf{y} \leftarrow \mathbf{0}$, $c \leftarrow 0, y_{-1} \leftarrow 0$
for $i \leftarrow 0, \ldots, k-1$ **do**
  $y_i \xleftarrow{\$} \{-1, 0\}$
end for
Let $\mathbf{z} \coloneqq [u]_b^k$ ($[u]_b^k$ is u's b-ary decomposition)
for $i \leftarrow 0, \ldots, k-2$ **do**
  if $z_i < q_i$ **then**
    $e_i \leftarrow b \cdot y_i - y_{i-1} + (q_i - b + c) \cdot y_{k-1}$
    $c \leftarrow 1$
  else if $z_i = q_i$ **then**
    $e_i \leftarrow b \cdot y_i - y_{i-1} + (q_i - c(b-1)) \cdot y_{k-1}$
  else
    $e_i \leftarrow b \cdot y_i - y_{i-1} + (q_i + c) \cdot y_{k-1}$
    $c \leftarrow 0$
  end if
end for
$x_{k-1} \leftarrow -y_{k-2} + (q_i + c) \cdot y_{k-1}$
$\mathbf{x} \leftarrow \mathbf{z} + \mathbf{e}$
return x

Likewise, Algorithm 2 also has to output the samples achieving a subgaussian distribution. We will show that Algorithm 2 is the sampling algorithm satisfying a subgaussian distribution similarly with Theorem 1. If $\mathbf{y}$ is a subgaussian, then the output is a subgaussian vector because a linear transformation of a subgaussian vector also follows a subgaussian distribution by Lemma 6.

Theorem 2. *Let $b > 1$ be an integer base, $q > 1$ be an integer modulus, $k = \lceil \log_b q \rceil$ and $\mathbf{g}^t = [1, b, \ldots, b^{k-1}]$ be a gadget. Then the output vector of Algorithm 2 is $\frac{1}{6}$-subgaussian with parameter $(b+1)\sqrt{\pi}$.*

Proof. The algorithm chooses y_i from $\{-1, 0\}$ and applies a transformation $\mathbf{S}$ at the end. Let $\hat{y_i}$ be chosen from $\{-1/2, 1/2\}$ and $\hat{Y_i}$ be ith component of a vector $\mathbf{D}\hat{\mathbf{y}}$. Then $\hat{Y_i}$ is 0-subgaussian with parameter $\hat{s} = \frac{1}{2}\sqrt{2\pi}$ by Lemma 1, since $\hat{Y_i}$ is $\frac{1}{2}$-bounded centered. Let $\hat{X_i}$ be a random variable after applying the transformation $\mathbf{S}$ to $\hat{Y_i}$. Since a transformation does not change δ, a random variable is also 0-subgaussian after a transformation. And the parameter becomes $\frac{1}{2}\sqrt{2\pi}\sqrt{\lambda_{max}(\mathbf{S}\mathbf{S}^t)}$ by Lemma 6. Since $\lambda_{max}(\mathbf{S}\mathbf{S}^t) \leq (b+1)^2$, the parmeter is $\hat{s} = \frac{b+1}{2}\sqrt{2\pi}$. Let a random variable X_i be an ith coordinate of a vector $\mathbf{z} + \mathbf{S}\mathbf{D}\mathbf{y}$. Then $X_i = z_i - \frac{b+1}{2} + \hat{X_i}$. Let $\alpha = z_i - \frac{b+1}{2}$, and the parameter $s = (b+1)\sqrt{2\pi}$.

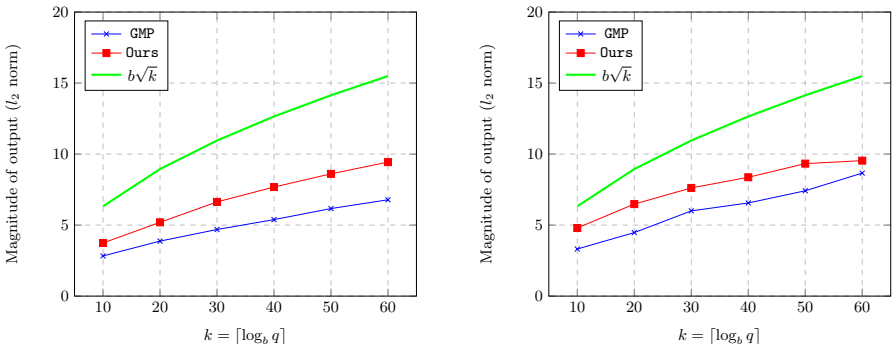
Then, by the proof of Lemma 3 in [25],

$$\frac{\alpha^2\pi}{s^2-\hat{s}^2} \leq \frac{\pi(b+1)^2/4}{6\pi(b+1)^2/4} = \frac{1}{6}.$$

Since we show that each component of the output is $\frac{1}{6}$-subgaussian with parameter $(b+1)\sqrt{2\pi}$, the result holds by Lemma 5. □

4 Experimental Results

Experimental Setup All experiments are performed on a laptop with Intel(R) Core(TM) i7-8565U CPU @ 1.80GHz. We use PALISADE Library [26] to compare our result to the previous work [14].



(a) for $q = b^k$ with Algorithm 1 (b) for $q \neq b^k$ with Algorithm 2

Fig. 2. The magnitude comparison, # trial= 10000

Magnitude of output and Computation time Figure 2 shows the magnitude of each output of our algorithm, denoted by `Ours`, and [14] denoted by `GMP`. Both cases show that our algorithms output a little bit larger size of output than `GMP`'s in our experiment (average value of 10000 trials with regard to Euclidean norm) due to the difference of two distributions, the bounded uniform distribution (`Ours`) and the triangular distribution (`GMP`). In fact, the sampling probability of larger output is higher than `GMP`'s. Nevertheless, the magnitude of both algorithms are bounded by $b\sqrt{k}$ with regard to Euclidean norm. Therefore, we show that the Euclidean norm of the output of two algorithms is indeed smaller than $b\sqrt{k}$ when q is a power of b and q is arbitrary, in Figure 2(a) and Figure 2(b), respectively. Therefore, it is adequate for `Ours` to be used in a lattice based cryptographic applications since it still outputs a small vector. In our experiment, we set $u = q/3$ in both cases because the size of output becomes too small with overwhelming

probability (close to 1) to compare if u is close to q in the case of GMP's. Therefore, we choose appropriate size of u to see various outputs.

q	Algorithm	Total	Decomposition	Transformation	Sampling	Addition
$q = b^k$	GMP[μs]	2.31487	1.04847	N/A	1.19871	0.06187
	Ours[μs]	1.89929	1.02334	N/A	0.81749	0.05826
$q \neq b^k$	GMP[μs]	2.43805	1.06996	0.187131	1.11734	0.06237
	Ours[μs]	1.88372	1.00613	N/A	0.81154	0.06588

Table 1. Detailed running time of each subalgorithm consisting of GMP [14] and ours without preprocessing for lattice dimension $n = 1$, the base $b = 2$, the modulus $q \approx 2^{60}$.

Table 1 shows that the detailed running time of all subalgorithms of Ours and GMP. Both algorithms are expected to perform slightly better when the modulus q is a power of the base b than the other case since GMP does not run additional subalgorithm Transformation and Ours has simpler Addition. However, Addition takes negligibly small time so that it barely affects the total computation time, whereas Transformation takes 4% of total in GMP. Ours is faster in any case even without precomputation: i) in the case of $q = b^k$, our Sampling is simpler than GMP's due to the constant probability. ii) For the case $q \neq b^k$, we do not run Transformation and have simpler sampling like the case of $q = b^k$, whereas GMP necessarily requires Transformation.

the base b	$k = \lceil \log_b q \rceil$	GMP[μs]	Ours[μs]
2	60	4947.93	2199.33
2^5	12	787.70	418.55
2^{10}	6	376.56	191.87
2^{15}	4	234.48	116.78
2^{20}	3	173.99	85.61

Table 2. Comparison of performance results between GMP and our algorithm with pre-computation for $n = 2048$, $\log_2 q \approx 60$ with the different base b

We apply our algorithm to the problem $\mathbf{Gx} = \mathbf{u}$ used in Ring-LWE based homomorphic encryption schemes (for the fixed recommended parameter [6]: n=2048, $q \approx 2^{60}$) then compare computation time with different b between two algorithms GMP and Ours with precomputation in Table 2. The larger the value of b, the smaller k for fixed q due to $k = \log_b q$. As b is increased, the running time becomes shorter with the larger size of output as a trade-off since the algorithm only runs kn times (see also Figure 2(b)).

In fact, the practical gadget decomposition algorithm used in many homomorphic encryption schemes is the same as Decomposition which is deterministic.

Then they heuristically assume that all the output coefficients independently follow a subgaussian distribution. However, we do not need the assumption since we give a subgaussian decomposition algorithm. As a result, there is only overhead with `Addition` which is just coefficient wise addition over two polynomial of degree n (or a vector of dimension n) in our case.

We note that we have time-memory trade-off due to preprocessing. In parameters $n = 2048, k = 60, b = 2$, our `Sampling` stores at most 15KB in advance as a preprocessing output, whereas `GMP`'s does not cost extra memory for that.

References

1. Ajtai, M.: Generating hard instances of the short basis problem. In: Proceedings of the 26th International Colloquium on Automata, Languages and Programming. p. 1–9. ICAL '99, Springer-Verlag, Berlin, Heidelberg (1999)
2. Alperin-Sheriff, J., Peikert, C.: Faster bootstrapping with polynomial error. In: Garay, J.A., Gennaro, R. (eds.) Advances in Cryptology – CRYPTO 2014, Part I. Lecture Notes in Computer Science, vol. 8616, pp. 297–314. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 17–21, 2014)
3. Boneh, D., Gentry, C., Gorbunov, S., Halevi, S., Nikolaenko, V., Segev, G., Vaikuntanathan, V., Vinayagamurthy, D.: Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In: Nguyen, P.Q., Oswald, E. (eds.) Advances in Cryptology – EUROCRYPT 2014. Lecture Notes in Computer Science, vol. 8441, pp. 533–556. Springer, Heidelberg, Germany, Copenhagen, Denmark (May 11–15, 2014)
4. Bonnoron, G., Ducas, L., Fillinger, M.: Large FHE gates from tensored homomorphic accumulator. In: Joux, A., Nitaj, A., Rachidi, T. (eds.) AFRICACRYPT 18: 10th International Conference on Cryptology in Africa. Lecture Notes in Computer Science, vol. 10831, pp. 217–251. Springer, Heidelberg, Germany, Marrakesh, Morocco (May 7–9, 2018)
5. Brakerski, Z., Halevi, S., Polychroniadou, A.: Four round secure computation without setup. In: Kalai, Y., Reyzin, L. (eds.) TCC 2017: 15th Theory of Cryptography Conference, Part I. Lecture Notes in Computer Science, vol. 10677, pp. 645–677. Springer, Heidelberg, Germany, Baltimore, MD, USA (Nov 12–15, 2017)
6. Chase, M., Chen, H., Ding, J., Goldwasser, S., Gorbunov, S., Hoffstein, J., Lauter, K., Lokam, S., Moody, D., Morrison, T., Sahai, A., Vaikuntanathan, V.: Security of homomorphic encryption. Tech. rep., HomomorphicEncryption.org, Redmond WA, USA (July 2017)
7. Chen, H., Chillotti, I., Song, Y.: Multi-key homomorphic encryption from TFHE. In: Galbraith, S.D., Moriai, S. (eds.) Advances in Cryptology – ASIACRYPT 2019, Part II. Lecture Notes in Computer Science, vol. 11922, pp. 446–472. Springer, Heidelberg, Germany, Kobe, Japan (Dec 8–12, 2019)
8. Chen, H., Dai, W., Kim, M., Song, Y.: Efficient multi-key homomorphic encryption with packed ciphertexts with application to oblivious neural network inference. Cryptology ePrint Archive, Report 2019/524 (2019), https://eprint.iacr.org/2019/524.pdf
9. 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, Part I. Lecture Notes in Computer

Science, vol. 10031, pp. 3–33. Springer, Heidelberg, Germany, Hanoi, Vietnam (Dec 4–8, 2016)
10. 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, Part I. Lecture Notes in Computer Science, vol. 10624, pp. 377–408. Springer, Heidelberg, Germany, Hong Kong, China (Dec 3–7, 2017)
11. Clear, M., McGoldrick, C.: Multi-identity and multi-key leveled FHE from learning with errors. In: Gennaro, R., Robshaw, M.J.B. (eds.) Advances in Cryptology – CRYPTO 2015, Part II. Lecture Notes in Computer Science, vol. 9216, pp. 630–656. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 16–20, 2015)
12. Dai, W., Doröz, Y., Polyakov, Y., Rohloff, K., Sajjadpour, H., Savaş, E., Sunar, B.: Implementation and evaluation of a lattice-based key-policy abe scheme. IEEE Transactions on Information Forensics and Security 13(5), 1169–1184 (2018)
13. Fan, J., Vercauteren, F.: Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive, Report 2012/144 (2012), https://eprint.iacr.org/2012/144
14. Genise, N., Micciancio, D., Polyakov, Y.: Building an efficient lattice gadget toolkit: Subgaussian sampling and more. In: Ishai, Y., Rijmen, V. (eds.) Advances in Cryptology – EUROCRYPT 2019, Part II. Lecture Notes in Computer Science, vol. 11477, pp. 655–684. Springer, Heidelberg, Germany, Darmstadt, Germany (May 19–23, 2019)
15. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: STOC. pp. 197–206 (2008), https://doi.org/10.1145/1374376.1374407
16. Gentry, C., Sahai, A., Waters, B.: Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In: Canetti, R., Garay, J.A. (eds.) Advances in Cryptology – CRYPTO 2013, Part I. Lecture Notes in Computer Science, vol. 8042, pp. 75–92. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 18–22, 2013)
17. Gorbunov, S., Vaikuntanathan, V., Wichs, D.: Leveled fully homomorphic signatures from standard lattices. In: Servedio, R.A., Rubinfeld, R. (eds.) 47th Annual ACM Symposium on Theory of Computing. pp. 469–477. ACM Press, Portland, OR, USA (Jun 14–17, 2015)
18. Halevi, S., Halevi, T., Shoup, V., Stephens-Davidowitz, N.: Implementing BP-obfuscation using graph-induced encoding. In: Thuraisingham, B.M., Evans, D., Malkin, T., Xu, D. (eds.) ACM CCS 2017: 24th Conference on Computer and Communications Security. pp. 783–798. ACM Press, Dallas, TX, USA (Oct 31 – Nov 2, 2017)
19. Kim, E., Lee, H.S., Park, J.: Towards round-optimal secure multiparty computations: Multikey FHE without a CRS. In: Susilo, W., Yang, G. (eds.) ACISP 18: 23rd Australasian Conference on Information Security and Privacy. Lecture Notes in Computer Science, vol. 10946, pp. 101–113. Springer, Heidelberg, Germany, Wollongong, NSW, Australia (Jul 11–13, 2018)
20. Lee, H.S., Park, J.: On the security of multikey homomorphic encryption. In: Albrecht, M. (ed.) Cryptography and Coding. pp. 236–251. Springer International Publishing, Cham (2019)
21. Lyubashevsky, V., Peikert, C., Regev, O.: A toolkit for ring-LWE cryptography. In: Johansson, T., Nguyen, P.Q. (eds.) Advances in Cryptology – EUROCRYPT 2013. Lecture Notes in Computer Science, vol. 7881, pp. 35–54. Springer, Heidelberg, Germany, Athens, Greece (May 26–30, 2013)

22. Micciancio, D., Peikert, C.: Trapdoors for lattices: Simpler, tighter, faster, smaller. In: Pointcheval, D., Johansson, T. (eds.) Advances in Cryptology – EUROCRYPT 2012. Lecture Notes in Computer Science, vol. 7237, pp. 700–718. Springer, Heidelberg, Germany, Cambridge, UK (Apr 15–19, 2012)
23. Micciancio, D., Regev, O.: Worst-case to average-case reductions based on gaussian measures. SIAM J. Comput. 37(1), 267–302 (Apr 2007), https://doi.org/10.1137/S0097539705447360
24. Mukherjee, P., Wichs, D.: Two round multiparty computation via multi-key FHE. In: Fischlin, M., Coron, J.S. (eds.) Advances in Cryptology – EUROCRYPT 2016, Part II. Lecture Notes in Computer Science, vol. 9666, pp. 735–763. Springer, Heidelberg, Germany, Vienna, Austria (May 8–12, 2016)
25. Murphy, S., Player, R.: δ-subgaussian random variables in cryptography. In: Information Security and Privacy. pp. 251–268. Springer (2019)
26. Polyakov, Y., Rohloff, K., Ryan., G.W.: Palisade lattice cryptography library. https://git. Accessed May (2020), https://gitlab.com/palisade/palisade-release
27. Stromberg, K.: Probability For Analysts. Chapman & Hall/CRC Probability Series, Taylor & Francis (1994), https://books.google.co.kr/books?id=gQaz79fv6QUC

A For CRT Representation

A modulus, which is larger than the native 64 bits in most modern hardware, is often used in many applications of lattice gadgets. Handling large modulus requires multi-precision numbers which is not the best option in practice. In order to avoid the use of multi-precision, one can pick a modulus of the form $q = \prod q_i$ with each co-prime q_i less than 64 bits. Then an element $u \in \mathbb{Z}_q$ can be stored as its Chinese Remainder representation (CRT form) and computed via the Chinese Remainder Theorem by using the fact below:

$$\mathbb{Z}_q \cong \mathbb{Z}_{q_1} \times \mathbb{Z}_{q_2} \times \cdots \times \mathbb{Z}_{q_l}.$$

The Chinese Remainder Transformation/Representation (CRT) gadgets, where $\mathbb{Z}_q$ when $q = q_1 q_2 \cdots q_l$ for co-prime q_i's, are used in many lattice-based cryptosystems where the modulus q is integer exceeding native 64-bits. The gadgets introduced by [4,18] are more compatible with CRT representation, but it is only applied to a limited setting where q has small prime factors.

After that, [14] introduced a general set of CRT gadgets for a wide class of moduli keeping arithmetic in the CRT representation. Since we use the same gadget, our algorithms are also suitable for large moduli. The algorithms in [14] can take input represented in CRT form. Therefore, the algorithms can be applied to the system that already store their numbers in CRT form, can operate on each CRT component independently (in-parallel), and only require arithmetic on small numbers q_i. On input $(u \mod q_1, \ldots, u \mod q_l)$, the sampling in CRT form, $\mathbf{g}_{CRT}^{-1}$, computes $\mathbf{g}_i^{-1}(u \mod q_i)$ which is $\mathbf{g}^{-1}$ for q_i and the corresponding base b_i, and each $\mathbf{g}_i^{-1}$ can be computed independently. Our algorithm also can support the same technique for computing $\mathbf{g}^{-1}$ since the difference is only the use of a uniform distribution. Therefore, our algorithms also achieve several theoretical and practical advantages which can be obtained by using algorithms in [14].