Towards Efficient Key Extraction of LBC over Ring: Fast Non-spherical G-lattice Sampler and Optimized Perturbation Generation

Hai Lu, Yan Zhu, *Member, IEEE,* Cecilia E Chen, *Member, IEEE,* and Di Ma, *Member, IEEE,*

***Abstract*—In the light of the advantages of ring, more and more Lattice-Based Cryptography (LBC) schemes are designed over it to provide small storage cost and high performance. Gaussian Sampler for Lattice Trapdoor (GSLT) plays an important role for these schemes, especially for key extraction. In this paper, we present an efficient GSLT scheme with On-line and Off-line stages. In the On-line stage, we extend the fast non-spherical Gadget-lattice sampling into the ring setting for high performance, and analyze the covariance matrix of output vectors. Subsequently, two optimized perturbation sampling constructions are designed for non-spherical Gadget-lattice sampler to avoid inefficient Cholesky decomposition during Off-line stage. The first construction aims to the spherical Gaussian distribution of preimage vectors, which is beneficial for theoretical analysis. In contrast, the second one is designed on the non-spherical distribution to improve the efficiency of perturbation sampling without leakage of trapdoor in statistic, and we further provide the method how to choose the Gaussian parameters. The complexity analysis and experimental results show that the On-line stage of our scheme has a better performance in comparison with the other works. In the Off-line stage, both of two perturbation sampling constructions can avoid low efficiency of Cholesky decomposition, and are more suitable for the non-spherical G-lattice sampling. In short, our work provides two candidates on either Gaussian parameter or sampling efficiency, thereby offering more options for key generation in LBC schemes.**

***Index Terms*—Lattice-based Cryptography, Fast Non-spherical G-lattice Sampling, Ring Setting, Optimized Perturbation Generation.**

I. Introduction

Lattice-Based Cryptography (LBC), as typified by anti-quantum cryptography, has gained more attentions with fast development of quantum computing. Michele Mosca, a founder of the University of Waterloo's Institute of Quantum Computing, believes that "quantum computing has a one-in-seven chance of breaking RSA-2048 encryption by 2026 and a 50-50 chance of doing so by 2031". To end it, many LBC schemes have been designed to provide anti-quantum security for various kinds of scenarios. These LBC schemes can also be applied for anti-forensics to protect the privacy of core secret.

As an important component, Gaussian Sampling for Lattice Trapdoor (GSLT) is widely employed to construct key generator in various kinds of LBC encryption schemes. For example, GSLT is usually used as key extraction algorithm in some encryption schemes, such as Identity-Based Encryption [1], [2], Attribute-Based Encryption [3], [4], and Fully Homomorphic Encryption [5].

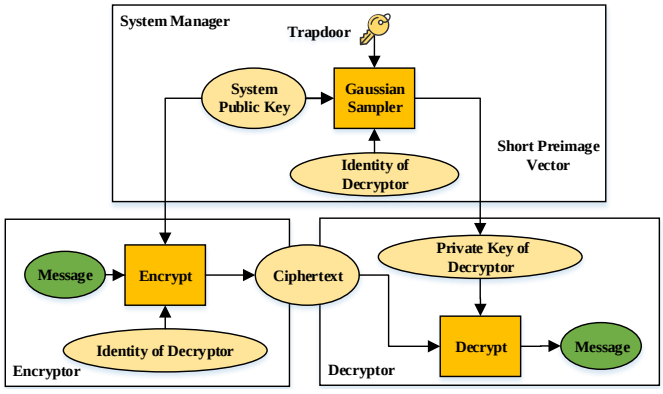

Fig. 1. Diagram of the LBC identity-based encryption.

We take the Identity-Based Encryption scheme as an example, and provide the simplified diagram of it shown as Fig. 1. The encryptor can encrypt the message to the ciphertext by system public key and the decryptor's identity, and the decryptor can use his unique private key to decrypt it. In the process, the system manager needs to invoke the Gaussian sampler through his trapdoor to extract the short preimage vector as the decryptor's private key that cannot be forged by the attackers. This key extraction essentially solves the Short Integer Solution (SIS) problem with the trapdoor of system manager. Given the modulus q, the trapdoor $\mathbf{T}$, the public matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, a syndrome $\mathbf{u} \in \mathbb{Z}_q^n$ and bound parameter $\beta > 0$, it can search the vector $\mathbf{x} \in \mathbb{Z}_q^m$ satisfying $\mathbf{Ax} = \mathbf{u}$ and $\|\mathbf{x}\| \leq \beta$. It is hard for attackers to search the short preimage vector without the trapdoor $\mathbf{T}$, so as to ensure the unforgeability of private key corresponding to any identity.

Micciancio and Peikert [6] presented an efficient GSLT scheme (in short MP12) with a specific framework. MP12 no longer focuses on constructing the short basis as the trapdoor, but rather on exploring a special lattice, i.e., Gadget-lattice (G-lattice), for more efficient sampling. Hence, the G-lattice sampling algorithm is easy-to-sample and parallelizable. In the aspect of preimage sampling, MP12 regards the linear factor as the trapdoor $\mathbf{T}$, and maps the vector sampled from G-lattice to that of the specific lattice by linear transformation. MP12's framework consists of two stages:

- **Off-line stage**: which is used to sample perturbation

H. Lu, Y. Zhu and C. E Chen are with the School of Computer and Communication Engineering, University of Science and Technology Beijing, Beijing, 100871 China (e-mail: luhai@xs.ustb.edu.cn; zhuyan,chene@ustb.edu.cn).

D. Ma is with the Computer and Information Science Department, College of Engineering and Computer Science, University of Michigan-Dearborn, Michigan 48128, USA (e-mail: dmadma@umich.edu).

TABLE I
COMPARISON OF G-SAMPLING ALGORITHMS IN [9] AND [7]

	Integer arithmetic	Floating point arithmetic	Calls to SampleZ	Integer storage	Floating point storage
GM18	$13k$	$7k$	$2k$	$2k$	$4k$
HJ19	$2k$	1	k	$3k$	1

vector $\mathbf{p}$ according to the specific covariance matrix. This stage is designed for security, and used to cover the statistical trace of trapdoor caused by linear transformation. Moreover, it can be performed in advance.

- **On-line stage**: which involves two main steps as follows:
  1) **G-lattice sampling**: which is used to sample a vector $\mathbf{z}$ from the specific coset of G-lattice with a discrete Gaussian-like distribution.
  2) **Linear transformation**: which is used to linearly expand $\mathbf{z}$ to $\mathbf{y} = \mathbf{p} + \begin{bmatrix}\mathbf{T}\\ \mathbf{I}\end{bmatrix} \cdot \mathbf{z}$ according to the precomputed perturbation $\mathbf{p}$ and trapdoor $\mathbf{T}$.

As shown above, because the off-line stage can be carried out in advance, the computation and storage cost of on-line stage directly affects the efficiency of GSLT, further influences the performance of LBC. As an important role of on-line stage, a G-lattice sampling algorithm with low time and space complexity does not only indicate a high-performance GSLT scheme, but also facilitates the equipment with limited computation ability.

Under the above MP12's framework, Hu and Jia (HJ19) [7] presented a fast non-spherical G-lattice sampling algorithm. Given a n-dimensional syndrome $\mathbf{u} \in \mathbb{Z}^n$ and an arbitrary modulus q, this algorithm controls the final component of output vector $\mathbf{z}$ to make $\mathbf{z}$ belong to the specific coset of G-lattice. Unlike traditional G-lattice sampling algorithms [6], [8], [9], the output vectors follow a non-spherical Gaussian distribution. It essentially sacrifice the efficiency of perturbation sampling to improve the performance of G-lattice sampling. Under an arbitrary modulus, HJ19 can achieve linear time and space complexity, and its performance is better than that of MP12 (it operates on real in $O(n\log q)^2$ time and $O(\log q)^2$ space with preprocessing, such as Gram-Schmidt orthogonalization). Comparing with GM18 [9] that can also achieve linear time and space complexity, the performance of HJ19 is also better, as shown in Table I.

Recently, on the basis of more efficient Ring-LWE (R-LWE) [10], more LBC schemes [11]–[13] are designed under the ring setting to provide small storage cost and high performance. However, HJ19 [7] only focuses on the construction of fast non-spherical G-lattice sampler, but give no deep discussion of the method how to apply it into GSLT scheme under the ring setting. Specifically, it does not give the specific method to convert the non-spherical G-lattice sampler to the one under the ring setting, as well as the analysis of covariance matrix. Moreover, HJ19 [7] indicates that the efficiency of the presented fast non-spherical G-lattice sampler in the on-line stage is built on the increased computational cost in the off-line stage. However, this work does not provide the reasons and analyze how to reduce the computational cost in the off-line stage. Therefore, this paper mainly considers these problems, and discusses how to construct a GSLT scheme under ring setting based on the fast non-spherical G-lattice sampler.

Contributions: This paper mainly explore the specific implementation of more efficient GSLT scheme over ring to provide anti-quantum and anti-forensics. Aiming to the fast non-spherical G-lattice sampler, our scheme carries out exploratory researches on both on-line stage and off-line stage in GSLT over ring. The contributions are shown as follows:

- We extend HJ19's integer G-lattice sampling algorithm to the ring setting. Furthermore, the covariance matrix of the output vectors from this G-lattice sampling algorithm are provided. Despite the covariance matrix belongs to $\mathcal{P}_n$, each entry of this matrix is a constant rather than a polynomial. Therefore it may be stored as a real matrix to reduce the storage cost. Meanwhile, it can be operated as a real matrix in calculation for some special cases.
- In order to reduce the computation and storage cost of perturbation sampling algorithm, we introduce the main idea of GM18's perturbation sampling into our presented GSLT scheme over ring by avoiding the inefficient Cholesky decomposition. Furthermore, two new constructions on perturbation sampling, *SampleP1overRing* and *SampleP2overRing*, are presented under different Gaussian parameters. The former aims to the spherical Gaussian distribution of preimage vector, and the latter is designed on non-spherical Gaussian distribution to improve the efficiency of perturbation sampling without leakage of trapdoor in statistic. Moreover, we analyze how to choose the Gaussian parameters for the *SampleP2overRing*.

These two constructions offer two candidates on either Gaussian parameter or sampling efficiency for preimage sampling. *SampleP1overRing* facilitates theoretical analysis across various LBC cryptosystems, whereas *SampleP2overRing* is more suited for practical applications such as the key generation or the signature implementation in LBC cryptosystems. As a result, an appropriate construction can be selected according to different requirements of LBC design. More details are shown in the last paragraph of Sec. V-B.

Current state of the art. Traditional GSLT scheme [8] over integer lattices usually uses Ajtai's trapdoor generation algorithm [14] to produce a public hard basis and a "good" short basis of a random integer lattice, where the short basis is regarded as the trapdoor. In the aspect of Gaussian preimage sampling, this scheme presented a Gaussian sampling algorithm, called Randomized Nearest Plane (RNP). It can sample vectors from the discrete Gaussian distribution of the specific integer lattice according to the trapdoor (i.e., the generated

short basis). Furthermore, Alwen and Peikert [15] improved the work of [14], and presented two kinds of trapdoor generation algorithms, where the quality of generated trapdoor is asymptotically optimal in the second construction. However, these two constructions involve many costly operations, such as seeking Hermite normal form and inverse matrix for a specific matrix.

Micciancio and Peikert [6] (i.e., MP12) improved the work of [8], and presented a GSLT scheme over integer lattice based on a special lattice, i.e., G-lattice. In the aspect of trapdoor generation, this scheme regards a randomly generated matrix, which is used for linear transformation, as the trapdoor $\mathbf{T}$ rather than a short basis, then extend another random matrix into a public matrix $\mathbf{A}$ according to $\mathbf{T}$. In the aspect of Gaussian preimage sampling, MP12 consists of both on-line and off-line stages, where on-line stage involves G-lattice sampling and linear expansion, and off-line stage involves perturbation sampling.

When the modulus $q = 2^k$, MP12 provides an optimized G-lattice sampling algorithm. The time complexity is $O(n\log q)$, and the space complexity is $O(\log q)$. However, this case cannot meet the requirement of actual cryptographic scheme. When q is an arbitrary prime integer, MP12 can only use the inefficient RNP algorithm to sample vectors from G-lattice. The time complexity is $O(n\log^2 q)$, and the space complexity is $O(\log^2 q)$. Genise et al. [9] (GM18) presented a fast G-lattice sampling algorithm under an arbitrary modulus. This algorithm decomposes the basis of G-lattice into a approximately orthogonal basis used for sampling and a matrix used for linear factor. Then, this algorithm linearly maps the vector sampled by the basis into the G-lattice. The time complexity of this algorithm is $\mathrm{O}(n\log q)$ which is faster than MP12. Hu et al. [7] also presented a fast G-lattice sampling algorithm under an arbitrary modulus. Despite the output vector follows a non-spherical Gaussian distribution, this algorithm avoids perturbation sampling in on-line stage, and achieves lower time and space complexity than GM18.

In order to apply the GSLT scheme over integer lattice to the ring setting, several works [9], [11], [12], [16] are presented. In the aspect of G-lattice sampling over ring, Bert et al. [11] converts the basis $\mathbf{B}$ over R_q into its anti-circular form $\phi_n(\mathbf{B})$, and directly sample vectors according to the basis $\phi_n(\mathbf{B})$ by RNP algorithm. In the aspect of perturbation sampling, GM18 provides a fast perturbation sampling algorithm under the ring setting, which can avoid the inefficient Cholesky decomposition, and achieve the quasi-linear time complexity. Moreover, Bert et al. [16] applied GM18's perturbation sampling to another hard problem, i.e., Module SIS (M-SIS) problem.

In order to reduce the dimensions of lattice and improve the performance of LBC scheme, Chen et al. [17] presented the Approximate Inhomogeneous Short Integer Solution (Approximate-ISIS) problem, and reduce its hardness to the LWE problem. Based on this hard problem, Chen et al. [17] design a novel GSLT scheme with an approximate trapdoor. Jia et al. [18] further introduce the non-spherical Gaussian distribution into the above work, and discuss the choice of parameter in terms of security and storage cost under the ring setting.

II. Preliminary

We use $\mathbb{R}$ and $\mathbb{Z}$ to denote real numbers and integers, respectively. Vectors (in column form) are represented by bold lower-case letters, e.g., $\mathbf{v}$, and the i-th component of $\mathbf{v}$ is denoted by v_i. Matrices are written as bold capital letters, e.g., $\mathbf{B}$. Note that $\mathbf{I}_k$ is used to denote an k-dimensional identity matrix. In this paper, l_2 norm is used to measure the length of a vector $\mathbf{v}$, i.e., $\|\mathbf{v}\|$, and the length of a matrix $\mathbf{B}$ is measured by $\|\mathbf{B}\| = \max_i \|\mathbf{b}_i\|$. For any two matrices $\mathbf{A}$ and $\mathbf{B}$, $\mathbf{A} \otimes \mathbf{B}$ is used to denote their tensor product.

Moreover, we use $\Sigma \succ 0$ (resp., $\Sigma \succeq 0$) to denote Σ is positive definite (resp., semidefinite) if $\mathbf{x}^t\Sigma\mathbf{x} > 0$ (resp., $\mathbf{x}^t\Sigma\mathbf{x} \geq 0$) for all nonzero $\mathbf{x} \in \mathbb{R}^m$. For clarity, $\Sigma \succeq \eta\mathbf{I}_m$ (resp., $\Sigma \succ \eta\mathbf{I}_m$) is abbreviated as $\Sigma \succeq \eta$ (resp., $\Sigma \succ \eta$). Suppose that $\Sigma = \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{B}^t & \mathbf{D} \end{bmatrix} \in \mathbb{R}^{m\times m}$, where $\mathbf{D}$ is a principal submatrix of Σ and invertible. $\mathbf{D}$'s Schur component in Σ is defined as $\Sigma/\mathbf{D} = \mathbf{A} - \mathbf{B}\mathbf{D}^{-1}\mathbf{B}^t$. For any real matrix $\mathbf{B} \in \mathbb{R}^{m\times n}$, $\mathrm{S}_i(\mathbf{B})$ is denoted as the i-th singular value of $\mathbf{B}$, and $\mathrm{S}_1(\mathbf{B}) = \max_{\mathbf{u}}\|\mathbf{B}\mathbf{u}\| = \max_{\mathbf{u}}\|\mathbf{B}^t\mathbf{u}\|$, where the maxima are taken over all units vectors $\mathbf{u} \in \mathbb{R}^k$.

A. Lattices, Gaussians and Ideal Lattices

Lattices: A lattice Λ is a discrete subgroup of $\mathbb{R}^m$. It can be defined as $\Lambda = \mathcal{L}(\mathbf{B}) = \{\mathbf{B}\mathbf{z} : \mathbf{z} \in \mathbb{Z}^m\}$ by m linearly independent vectors $\mathbf{B} = \{\mathbf{b}_1, \mathbf{b}_2, \cdots, \mathbf{b}_m\}$. $\Lambda^* = \{\mathbf{x} \in \mathbb{R}^n : \forall \mathbf{v} \in \Lambda, \mathbf{x}\cdot\mathbf{v} \in \mathbb{Z}\}$ represents the dual lattice of Λ.

Let $n, m, q \in \mathbb{Z}^+$, $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ and $\mathbf{u} \in \mathbb{Z}_q^n$. Most current LBC schemes usually use the lattices defined by:

$$\begin{cases} \Lambda_q(\mathbf{A}) &= \{\mathbf{x} \in \mathbb{Z}^m s.t. \exists \mathbf{s} \in \mathbb{Z}^n, \mathbf{A}^\mathbf{t}\mathbf{s} = \mathbf{x} \bmod q\}, \\ \Lambda_q^\perp(\mathbf{A}) &= \{\mathbf{x} \in \mathbb{Z}^m s.t. \mathbf{A}\mathbf{x} = 0 \bmod q\}, \\ \Lambda_q^\mathbf{u}(\mathbf{A}) &= \{\mathbf{x} \in \mathbb{Z}^m s.t. \mathbf{A}\mathbf{x} = \mathbf{u} \bmod q\}, \end{cases} \tag{1}$$

where, q is a prime modulus, and $\mathbf{u}$ is a syndrome. Note that the two lattices $\Lambda_q(\mathbf{A})$ and $\Lambda_q^\perp(\mathbf{A})$ defined above are dual. Moreover, $\Lambda_q^\mathbf{u}(\mathbf{A})$ is regarded as the "shifted lattice" of $\Lambda_q^\perp(\mathbf{A})$ for an arbitrary syndrome $\mathbf{u} \in \mathbb{Z}_q^n$. If $\mathbf{t} \in \Lambda_q^\mathbf{u}(\mathbf{A})$, $\Lambda_q^\mathbf{u}(\mathbf{A}) = \Lambda_q^\perp(\mathbf{A}) + \mathbf{t}$.

Gaussian distribution: Suppose that $\sigma > 0$ is a positive Gaussian parameter and $\mathbf{c} \in \mathbb{Z}^m$ is a center. The Gaussian function of a lattice $\Lambda \subset \mathbb{Z}^m$ is defined as $\forall \mathbf{x} \in \Lambda, \rho_{\sigma,\mathbf{c}}(\mathbf{x}) = \exp(-\pi\|\mathbf{x}-\mathbf{c}\|^2/\sigma^2)$. Then, the discrete gaussian distribution of $\Lambda \subset \mathbb{Z}^m$ can be defined as $D_{\Lambda,\sigma,\mathbf{c}}$ as $\forall \mathbf{x} \in \Lambda, D_{\Lambda,\sigma,\mathbf{c}}(\mathbf{x}) = \rho_{\sigma,\mathbf{c}}(\mathbf{x})/\rho_{\sigma,\mathbf{c}}(\Lambda)$, where $\rho_{\sigma,\mathbf{c}}(\Lambda) = \sum_{\mathbf{y}\in\Lambda}\rho_{\sigma,\mathbf{c}}(\mathbf{y})$. For clarity, we rewrite $\rho_{\sigma,0}$ as ρ_σ, if the center is 0. Furthermore, we define the skewed gaussian function $\rho_{\sqrt{\Sigma},\mathbf{c}}$ as $\forall \mathbf{x} \in \Lambda, \rho_{\sqrt{\Sigma}}(\mathbf{x}) = \exp(-\pi(\mathbf{x}-\mathbf{c})^t\Sigma^{-1}(\mathbf{x}-\mathbf{c}))$, and the corresponding discrete gaussian distribution is denoted by $D_{\Lambda,\sqrt{\Sigma},\mathbf{c}}$.

Here, we provide some definitions related to smoothing parameter $\eta_\epsilon(\Lambda)$ of a lattice $\Lambda \subset \mathbb{R}^m$, shown as follows:

Definition 1 ([19], [20]): Suppose $\Lambda \subset \mathbb{R}^m$ be a lattice with a basis $\mathbf{B}$ and $\widetilde{\mathbf{B}}$ be the Gram-Schimidt orthogonalization of $\mathbf{B}$. Then, for any $\epsilon > 0$, we have $\eta_\epsilon(\Lambda) \leq \|\widetilde{\mathbf{B}}\| \cdot \sqrt{\ln(2m(1+1/\epsilon))/\pi}$.

Definition 2 ([8], Lemma 3.1): For any m-dimensional lattice Λ, center $\mathbf{c} \in \mathbb{R}^m$, reals $0 < \epsilon < 1$ and $\sigma \geq \eta_\epsilon(\Lambda)$, if $\mathbf{x}$ is from the distribution $D_{\Lambda,\sigma,\mathbf{c}}$, then we have $\Pr[\|\mathbf{x}-\mathbf{c}\| > \sigma\sqrt{m}] \leq \frac{1+\epsilon}{1-\epsilon}\cdot 2^{-m}$.

Ideal Lattice: In this paper, we mainly consider the special ideal lattice with a polynomial structure. Let n is a power of 2. We represent R as the ring $\mathbb{Z}[x]/\langle x^n+1\rangle$ and set $R_q = R/qR = \mathbb{Z}_q[x]/\langle x^n+1\rangle$. Moreover, $\mathcal{P}_i$ is defined as $\mathbb{R}[x]/\langle x^i+1\rangle$ for an arbitrary integer i, note that both of R and R_q can be regarded as subsets of $\mathcal{P}_n$. Suppose that a polynomial $a = \sum_{j=0}^{n-1} a_j x^j \in \mathcal{P}_n$, where a_j is the j-th coefficient of a. In this paper, the coefficient vector $\mathbf{a} = (a_0, a_1, \cdots, a_{n-1})$ is equivalent to the polynomial a, and the coefficient vector $\mathbf{a}$ can be denoted as $\mathbf{a} = \gamma(a)$. ϕ_n is represented as the ring homomorphic mapping from $\mathcal{P}_n$ to $\mathbb{R}^{n\times n}$. For an element $a = \sum_{j=0}^{n-1} a_j x^j \in \mathcal{P}_n$, we have

$$\phi_n(a) = \begin{bmatrix} a_0 & -a_{n-1} & \cdots & -a_1 \\ a_1 & a_0 & & \vdots \\ \vdots & & \ddots & -a_{n-1} \\ a_{n-1} & \cdots & a_1 & a_0 \end{bmatrix}. \tag{2}$$

Moreover, $\phi_i()$ and $\gamma()$ can be also extended to matrices and vectors over $\mathcal{P}_i$ respectively by component-wise application. We take $\mathbf{v} \in \mathcal{P}_n^m$ as an example, and have $\gamma(\mathbf{v}) = (\gamma(v_1), \gamma(v_2), \cdots, \gamma(v_m))$.

B. Trapdoor Preimage Sampling Algorithm over Ring from MP12

Several works [11], [12], [16], [21] have applied MP12 to the ring setting. Algorithm 1 [12] provides the details of trapdoor construction:

Algorithm 1 TrapGen(q, σ, n)

Input: A ring modulus q, a Gaussian parameter δ, an integer n that is power of 2, $k = \lceil \log_2 q \rceil$.

Output: A public vector $\mathbf{a} \in R_q^{k+2}$ and trapdoor $\mathbf{T} = (\mathbf{t}_1, \mathbf{t}_2) \in R_q^{2\times k}$.

1: $a \leftarrow_R R_q$, which represents randomly choosing a from R_q.
2: $\mathbf{t}_1 \leftarrow (t_1^{(1)}, t_2^{(1)}, \cdots, t_k^{(1)})^t$, where $t_i^{(1)} \leftarrow D_{R,\sigma}$ for $i = 1, \cdots, k$.
3: $\mathbf{t}_2 \leftarrow (t_1^{(2)}, t_2^{(2)}, \cdots, t_k^{(2)})^t$, where $t_i^{(2)} \leftarrow D_{R,\sigma}$ for $i = 1, \cdots, k$.
4: $\mathbf{a} \leftarrow (1, a, g_1 - (t_1^{(1)} + a t_1^{(2)}), \cdots, g_k - (t_k^{(1)} + a t_k^{(2)}))$.
5: **return** $\mathbf{a}, \mathbf{T} = (\mathbf{t}_1, \mathbf{t}_2)^t$.

Note that, $g_i \in R$ for $i \in [1, k]$ is the i-th element of $\mathbf{g} = (1, 2, 2^2, \cdots, 2^{k-1}) \in R_q^k$. Combined with the above trapdoor, MP12's GSLT over ring is shown as Algorithm 2.

Algorithm 2 includes two important functions:

1) **Perturbation sampling function** $\mathrm{SamplePoverRing}(n, k, q, \Sigma_s, \mathbf{T}, \sigma)$. It takes $n, k, q, \Sigma_s, \mathbf{T}$ and σ as inputs, and outputs a perturbation $\mathbf{p} \in D_{R,\Sigma_p}$, where Σ_p is the covariance matrix of $\mathbf{p}$, and it can be computed according to $\Sigma_s, \mathbf{T}$ and σ.
2) **G-lattice sampling function over ring** $\mathrm{SampleGoverRing}(n, k, q, \sigma, v)$. It takes n, k, q, σ and v as inputs, and outputs a vector $\mathbf{z} \in D_{R,\Sigma_G}$, where Σ_G is the covariance matrix of $\mathbf{z}$, and is related to σ.

Algorithm 2 SamplePreoverRing$(\mathbf{a}, \mathbf{T}, u, \sigma, \Sigma_s)$

Input: Public vector $\mathbf{a}$, trapdoor $\mathbf{T}$, syndrome $u \in R_q$, Gaussian parameter σ used for G-lattice sampling, Gaussian parameter Σ_s used for preimage sampling.

Output: $\mathbf{z} \leftarrow D_{R_q,\Sigma_s}$.

Off-line stage:

1: Sample a perturbation vector $\mathbf{p} \leftarrow \mathrm{SamplePoverRing}(n, k, q, \Sigma_s, \sigma) \in R^{k+2}$.

On-line stage:

1: Compute $v = u - \mathbf{ap} \in R_q$.
2: Sample $\mathbf{z} \leftarrow \mathrm{SampleGoverRing}(n, k, q, \sigma, v) \in R^k$.
3: Compute $\mathbf{y} = \mathbf{p} + \begin{bmatrix} \mathbf{T} \\ \mathbf{I} \end{bmatrix} \mathbf{z} \in R_q^{k+2}$.
4: **return** $\mathbf{y}$.

III. Technique Overview

Fig. 2 provides the technique overview of our GSLT scheme, including On-line and Off-line stages. In the Off-line stage, we present two new perturbation sampling constructions, **SampleP1overRing** and **SampleP2overRing**, tailored specifically for the non-spherical G-lattice sampling. Construction 1 facilitates theoretical analysis across various LBC cryptosystems, whereas Construction 2 is more suited for practical applications such as the key generation or the signature implementation in LBC cryptosystems. To reduce the computational cost, each of two constructions is equipped with a precomputation algorithm (**SampleP1-Precomp** or **SampleP2-Precomp**) to generate intermediate parameters. During the On-line stage, we provide a rearranging method to extend the fast non-spherical G-lattice sampling algorithm into the ring setting to efficiently sample G-lattice vectors. Finally, the G-lattice vectors are linearly expanded to the preimage vectors according to the pre-sampled perturbation vectors. The technical details are elaborated in terms of On-line and Off-line stages, shown as follows:

1) **On-line stage**: The work [21] introduced the rearranging method, yet the authors only analyzes the covariance matrix of the output vector when employing spherical integer G-lattice sampling. In this paper, we extend the fast non-spherical G-lattice sampling into the ring setting, and analyze the covariance matrix of output vectors under this rearrangement. According to the special structure of the covariance matrix, it can be stored and operated as a real matrix despite it is actually a matrix with elements in polynomial ring. Such a special structure not only enhances the efficiency of perturbation sampling, but also reduces the storage requirements of this matrix.

2) **Off-line stage**: To avoid the inefficient Cholesky decomposition, two perturbation sampling constructions are designed for the non-spherical G-lattice sampling. The main idea of the presented constructions are summarized as follows:

(1) For the given covariance matrix $\Sigma_p \in \mathcal{P}_n^{k+2}$ of perturbations, a vector $\mathbf{p}_s \in R^k$ is sampled according to the easy-to-sample portion of Σ_p. As a result, Σ_p is reduced to a (2×2)-dimensional matrix Σ over $\mathcal{P}_n$.

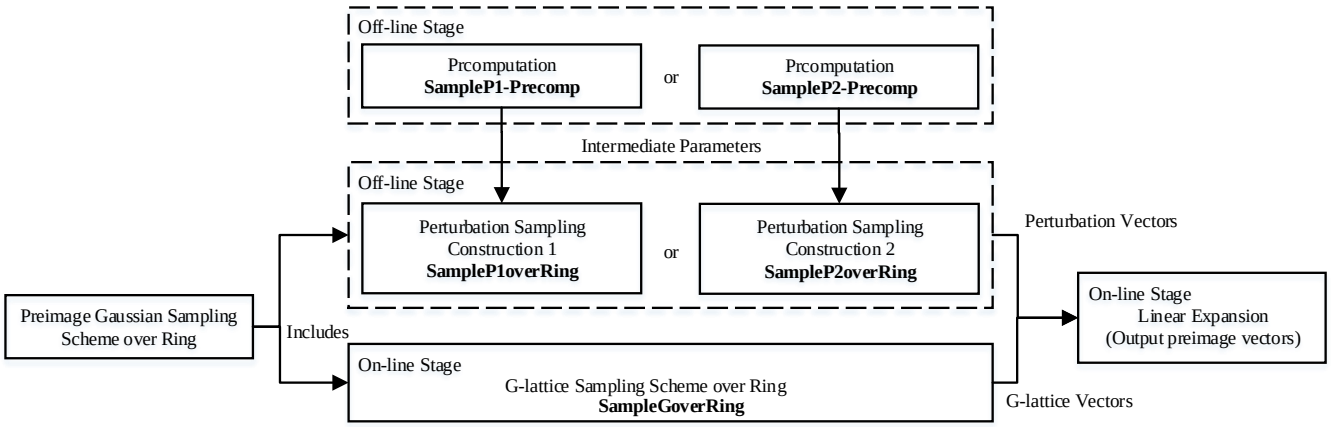

Fig. 2. Technique overview of our GSLT scheme.

(2) The vector $(p_1, p_2) \in R_q^2$ can be then extracted based on the Σ through iteratively sampling. Finally, the perturbation $\mathbf{p} = (p_1, p_2, \mathbf{p}_s) \in R^{k+2}$ can be obtained by combining (p_1, p_2) with $\mathbf{p}_s$.

Next, the difference between GM18 [9] and our presented constructions can be described as follows. GM18 mainly focuses on spherical G-lattice sampler, but this method cannot be directly used for the non-spherical G-lattice sampling. The reason is that the distribution of final preimage vectors will leak the trapdoor information in statistics. In order to reduce the cost of perturbation sampling, we design two constructions on perturbation sampling which are suitable for the fast non-spherical G-lattice sampler. The Construction 1 is tailored for the spherical distribution of preimage vectors, but its time & space complexity is high, and it needs high computational accuracy. The reason is that the easy-to-sample part of Σ_p no longer corresponds to a spherical Gaussian distribution. To reduce the computational cost during the Off-line stage, the Construction 2 is presented by modifying the easy-to-sample portion. Despite the final preimage vector follows a non-spherical Gaussian distribution by using this construction, the Gaussian parameter will not leak the information of trapdoor in statistics.

IV. Fast Non-spherical G-lattice Sampling Algorithm over Ring

In this section, we introduce how to extend the fast non-spherical G-lattice sampling algorithm in HJ19 to the ring setting. Before the description, we recall the G-lattice and G-lattice sampling algorithm in HJ19.

G-lattice is defined as $\Lambda_q^{\perp}(\mathbf{G})$ based on a special matrix $\mathbf{G} = \mathbf{I}_n \otimes \mathbf{g}^t \in \mathbb{Z}^{n \times kn}$, where $\mathbf{g}^t = (1, 2, \cdots, 2^{k-1})$. Given a syndrome $\mathbf{u} = (u_1, u_2, \cdots, u_n) \in \mathbb{Z}^n$, the G-lattice sampling algorithm can be carried out by independently sampling n vectors from $D_{\Lambda_q^{u_i}(\mathbf{g}^t),\sigma}$ for $i \in [1, n]$, where σ is the Gaussian parameter.

HJ19 provides a fast non-spherical G-lattice sampling algorithm to capture vectors from $\Lambda_q^u(\mathbf{g}^t)$ with a parameter σ, the details are shown in Algorithm 3.

Algorithm 3 HJ19-SampleG(n, k, q, σ, u)

Input: parameter n, k and q, Gaussiann parameter σ used for G-lattice sampling and an arbitrary syndrome $u \in \mathbb{Z}_q$;
Output: a vector $\mathbf{z}$ following the distribution $D_{\Lambda_q^u(\mathbf{g}^t),\sqrt{\sigma^2\Sigma_0}}$.

1: Let $\mathbf{u} = [u]_2^k$ and $\mathbf{q} = [q]_2^k$ be the bit sequences of u and q, respectively.
2: Compute $c = -\frac{u}{q}$, and choose $y \leftarrow D_{\mathbb{Z},\frac{\sigma}{2},c}$.
3: $\mathbf{v} = \mathbf{u} + y \cdot \mathbf{q}$, where $\mathbf{v} = (v_1, v_2, \cdots, v_k)$.
4: **for** $i \in [1, k-1]$ **do**
5: Choose $z_i \leftarrow D_{2\mathbb{Z}+v_i,\sigma}$.
6: Set $v_{i+1} = v_{i+1} + \frac{v_i - x_i}{2} \in \mathbb{Z}$.
7: **end for**
8: Set $z_k = v_k$.
9: **return** $\mathbf{z} = (z_1, z_2, \cdots, z_k)$.

In this algorithm, the covariance matrix of $\mathbf{z}$ equals to $\sigma^2\Sigma_0$ rather than $\sigma^2\mathbf{I}$ in MP12 and GM18, where

$$\Sigma_0 = \begin{bmatrix} & & -\left(\frac{1}{2}\right)^{k-1} \\ & \mathbf{I}_{k-1} & \vdots \\ & & -\frac{1}{2} \\ -\left(\frac{1}{2}\right)^{k-1} & \cdots \quad -\frac{1}{2} & \alpha \end{bmatrix} \in \mathbb{R}^{k \times k}, \quad (3)$$

and $\alpha = \sum_{i=1}^{k-1} \left(\frac{1}{4}\right)^i + \frac{q^2}{4^k}$. It means that $\mathbf{z}$ follows a non-spherical Gaussian distribution.

In the light of the fact that G-lattice sampling algorithm in HJ19 is more efficient than the other algorithms (as shown in Table I), we intend to use the idea from the work of [21] to convert the HJ19's G-lattice sampling to the ring setting. Unlike the work [11], the main idea is shown as Fig. 3.

Notice that $\mathbf{g}^t = (1, 2, \cdots, 2^{k-1}) \in R_q^k$ is regarded as the vector over R_q in the top half of Fig. 3, while $\mathbf{g}^t \in \mathbb{Z}_q^k$ is in the bottom part of Fig. 3. Here, $z_i^{(j)} \in \mathbb{Z}$ and $u_i \in \mathbb{Z}$ denotes the i-th coefficient of $z_j \in R_q$ and $u \in R_q$, respectively. As shown in Fig. 3, the operation for sampling a vector $\mathbf{z} = (z_1, z_2, \cdots, z_k) \in R_q^k$ from $\Lambda_q^u(\mathbf{g}^t) = \{\mathbf{z} \in R_q^k, \mathbf{g}^t \in R_q^k | \mathbf{g}^t \cdot \mathbf{z} = u\}$ can be carried out by sampling the integer vector $\mathbf{z}' = (z_1^{(1)}, z_1^{(2)}, \cdots, z_1^{(k)}, \cdots, z_n^{(1)}, \cdots, z_n^{(k)}) \in \mathbb{Z}^{kn}$ from the integer G-lattice $\Lambda_q^{\mathbf{u}}(\mathbf{G}) = \{\mathbf{G} \in \mathbb{Z}^{n \times nk}, \mathbf{z}' \in \mathbb{Z}_q^{nk} | \mathbf{G}\mathbf{z}' =$

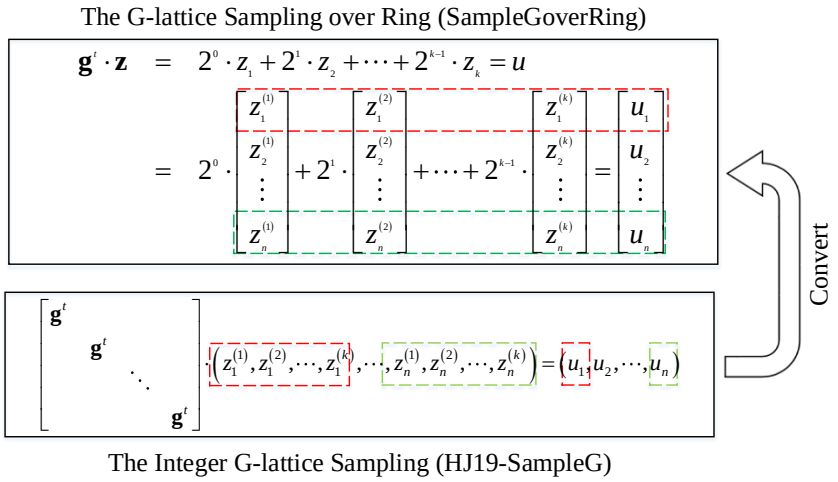

Fig. 3. The rearranging method to convert integer G-lattice sampling to that over ring.

$\mathbf{u} \bmod q\}$ for $\mathbf{u} = \gamma(u)$, i.e., the coefficient vector of u. The steps of Fig. 3 are shown as follows:

1) Invoke G-lattice sampling algorithm to sample a vector $\mathbf{z}' \in \mathbb{Z}^{kn}$ from $\Lambda_q^{\mathbf{u}}(\mathbf{G})$;
2) Rearrange $\mathbf{z}'$ into $\mathbf{z} = (z_1^{(1)}, z_2^{(1)}, \cdots, z_n^{(1)}, \cdots, z_1^{(k)}, \cdots, z_n^{(k)})$ according to the Fig. 3. Then, this coefficient vector $\mathbf{z}$ can be regarded as the vector $\mathbf{z} = (z_1, \cdots, z_k)$ over R_q. Here, we use $\mathrm{Rearrange}()$ to denote this rearranging process for clarity.

Algorithm 4 provides the details of this G-lattice sampling algorithm over ring.

Algorithm 4 SampleGoverRing(n, k, q, σ, u)

Input: parameter n, k and q, Gaussian parameter σ used for G-lattice sampling, syndrome $u = \sum_{i=1}^{n} u_i \cdot x^i \in R_q$;
Output: a vector $\mathbf{y} \leftarrow D_{\Lambda_q^{\mathbf{u}}(\mathbf{g}^t), \sqrt{\Sigma_G}}$;

1: Initialize $\mathbf{z}' = \phi$.
2: **for** $i \in [1, n]$ **do**
3: Invoke $\mathrm{HJ19-SampleG}(n, k, q, \sigma, u_i)$ to sample a vector $(z_i^{(1)}, z_i^{(2)}, \cdots, z_i^{(k)})$.
4: $\mathbf{z}' = (\mathbf{z}', z_i^{(1)}, z_i^{(2)}, \cdots, z_i^{(k)})$.
5: **end for**
6: $\mathbf{z} = \mathrm{ReArrange}(\mathbf{z}')$.
7: **return** $\mathbf{z}$.

Here, we discuss the covariance matrix Σ_G of vectors sampled from Algorithm 4 shown as follows.

Proposition 1: The output vectors from Algorithm 4 follows the Gaussian-like distribution $D_{\Lambda_q^{\mathbf{u}}(\phi_n(\mathbf{g}^t)), \sqrt{\Sigma_G}}$, where, $\mathbf{u}$ is the coefficient vector of u, the covariance matrix $\Sigma_G = \sigma^2 \Sigma_1$, and $\Sigma_1 \in \mathbb{R}^{kn \times kn}$ is

$$\Sigma_1 = \begin{bmatrix} & & -\left(\frac{1}{2}\right)^{k-1}\mathbf{I}_n \\ & \mathbf{I}_{(k-1)n} & \vdots \\ & & -\frac{1}{2}\mathbf{I}_n \\ -\left(\frac{1}{2}\right)^{k-1}\mathbf{I}_n & \cdots & -\frac{1}{2}\mathbf{I}_n \quad \alpha\mathbf{I}_n \end{bmatrix}. \quad (4)$$

Proof 1: According to the proof of Theorem 1 from HJ19 [7], the vector sampled from $\mathrm{HJ19-SampleG}(n, k, q, \sigma, u_i)$ follows the non-spherical Gaussian-like distribution $D_{\Lambda_q^{u_i}(\mathbf{g}^t), \sigma \cdot \sqrt{\Sigma_0}}$ for $\mathbf{g}^t \in \mathbb{Z}^k$. Therefore, the covariance matrix of the $\mathbf{z}'$ in the sixth step is $\Sigma'_G = \mathbf{I}_n \otimes (\sigma^2 \cdot \Sigma_0)$, and the center is 0.

Then, the order of components in $\mathbf{z}'$ is rearranged by $\mathrm{ReArrange}()$ to get $\mathbf{z}$ in the sixth step of Algortihm 4. According to the definition of covariance matrix, the rows and columns of Σ'_G should be also rearranged by $\mathrm{ReArrange}()$ to get the covariance matrix Σ_G of $\mathbf{z}$, i.e., $\Sigma_G = \sigma^2 \Sigma_1$. Meanwhile, we also have $\phi_n(\mathbf{g}^t) \cdot \mathbf{z} = \mathbf{u}$ for the vector $\mathbf{g}^t = (1, 2, \cdots, 2^{k-1}) \in R^k$, i.e., $\mathbf{z}$ belongs to the integer lattice $\Lambda_q^{\mathbf{u}}(\phi_n(\mathbf{g}^t))$. Therefore, the above $\mathbf{z}$ follows the Gaussian distribution $D_{\Lambda_q^{\mathbf{u}}(\phi_n(\mathbf{g}^t)), \sigma\sqrt{\Sigma_1}}$.

In this paper, the above coefficient vector $\mathbf{z} = (z_1^{(1)}, \cdots, z_n^{(1)}, \cdots, z_1^{(k)}, \cdots, z_n^{(k)}) \in \mathbb{Z}^{kn}$ can be regarded as the vector $\mathbf{z} = (z_1, z_2, \cdots, z_k) \in R_q^k$. We further observe that Σ_1 can be also regarded as the anti-circular matrix form of $\Sigma_2 \in \mathcal{P}_n^{k \times k}$, where Σ_2 is shown as:

$$\Sigma_2 = \begin{bmatrix} & & -\frac{1}{2^{k-1}} \\ & \mathbf{I}_{k-1} & \vdots \\ & & -\frac{1}{2} \\ -\frac{1}{2^{k-1}} & \cdots \quad -\frac{1}{2} & \alpha \end{bmatrix} \in \mathcal{P}_n^{k \times k}. \quad (5)$$

For some appropriate matrices, such as $\mathbf{T} \in \mathcal{P}_n^{2 \times k}$, we have $\phi_n(\mathbf{T}) \cdot \Sigma_1 = \phi_n(\mathbf{T} \cdot \Sigma_2)$. Therefore, this vector $\mathbf{z} \in R_q^k$ actually follows the Gaussian-like distribution $D_{\Lambda_q^{u}(\mathbf{g}^t), \sigma\sqrt{\Sigma_2}}$ for $\mathbf{g}^t \in R_q^k$. Note that, each entry of Σ_2 is a real number rather than a polynomial. Therefore, this matrix can be stored and operated as a real matrix in certain calculations, e.g., Step 1 of Algorithm 5 in Section V-A. It can not only improve the computational efficiency, but also reduce the dimension from $(kn \times kn)$ to $k \times k$.

V. Perturbation Sampling

Perturbation sampling plays an important role in GSLT schemes. It can protect the trapdoor from the leakage caused by the linear transformation. In this section, we suppose

that $\Sigma_s \in \mathbb{R}^{(k+2)n\times(k+2)n}$ is the Gaussian parameter that represents the covariance matrix of preimage vectors sampled from the GSLT scheme over ring. Then the covariance matrix of perturbations is defined as $\Sigma_p = \Sigma_s - \begin{bmatrix}\phi_n(\mathbf{T})\\ \mathbf{I}\end{bmatrix}\Sigma_G\begin{bmatrix}\phi_n(\mathbf{T})^t & \mathbf{I}\end{bmatrix} \in \mathbb{R}^{(k+2)n\times(k+2)n}$. For perturbation sampling, the most straightforward method is to decompose Σ_p into $\sqrt{\Sigma_p}$ by Cholesky decomposition, and sample perturbation $\mathbf{p} \in R^{k+2}$ according to $\sqrt{\Sigma_p}$. However, this method is inefficient, because it needs $O(nk)^3$ precomputation (i.e., Cholesky decomposition), as well as $O(nk)^2$ space complexity and $O(nk)^2$ time complexity.

To avoid Cholesky decomposition, GM18 provides a fast perturbation sampling algorithm under the ring setting, and achieve quasi-linear time complexity. The most important part of GM18 is the function $\mathrm{SampleF}_z(f, c)$ that can sample an element $p \in R$ according to a covariance polynomial $f \in \mathcal{P}_n$ and a center $c \in \mathcal{P}_n$. Moreover, GM18 mainly relies on the following important lemma.

Lemma 1 ([16], Lemma 5): For any real $0 < \epsilon \leq \frac{1}{2}$, positive integers r and s, vector $\mathbf{c} = (\mathbf{c}_1, \mathbf{c}_2) \in \mathbb{R}^{r+s}$, positive definite $\Sigma = \begin{bmatrix}\mathbf{A} & \mathbf{B}\\ \mathbf{B}^t & \mathbf{D}\end{bmatrix} \in \mathbb{R}^{(r+s)\times(r+s)}$ composed of blocks $\mathbf{A} \in \mathbb{R}^{r\times r}$, $\mathbf{B} \in \mathbb{R}^{r\times s}$, $\mathbf{D} \in \mathbb{R}^{s\times s}$, we define the following random process:

- $\mathbf{x}_2 \leftarrow D_{\mathbb{Z}^s,\sqrt{\mathbf{D}},\mathbf{c}_2}$;
- $\mathbf{x}_1 \leftarrow D_{\mathbb{Z}^r,\sqrt{\Sigma/\mathbf{D}},\mathbf{c}_1+\mathbf{B}\mathbf{D}^{-1}(\mathbf{x}_2-\mathbf{c}_2)}$.

If $\Sigma_p \geq \eta_\epsilon^2(\mathbb{Z}^{r+s})$, then this process outputs a vector $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2) \in \mathbb{Z}^{r+s}$ whose distribution is statistically indistinguishable from $D_{\mathbb{Z}^{r+s},\sqrt{\Sigma},\mathbf{c}}$.

However, GM18 only considers the case of spherical G-lattice sampler, i.e., the sampled G-lattice vector's covariance matrix follows $\Sigma_G = \sigma^2 \cdot \mathbf{I}$ for some appropriate parameters σ. For the non-spherical G-lattice sampler, GM18 should set perturbation's covariance matrix as

$$\Sigma_p = \Sigma_s - \sigma_g^2\begin{bmatrix}\phi_n(\mathbf{T})\\ \mathbf{I}\end{bmatrix}\cdot\begin{bmatrix}\phi_n(\mathbf{T})^t & \mathbf{I}\end{bmatrix} \tag{6}$$

where σ_g^2 is regarded as the eigenvalue of Σ_G. If $\Sigma_G = \sigma^2\Sigma_1$, then we have $\sigma_g = \sigma \cdot \sqrt{\mathrm{S}_1(\Sigma_1)}$. Therefore, the covariance matrix Σ_y of final output vectors $\mathbf{y} = \mathbf{p} + \begin{bmatrix}\mathbf{T}\\ \mathbf{I}\end{bmatrix}\mathbf{z}$ in the linear expansion process is shown as

$$\begin{aligned}\Sigma_y &= \Sigma_p + \begin{bmatrix}\phi_n(\mathbf{T})\\ \mathbf{I}\end{bmatrix}\Sigma_G\begin{bmatrix}\phi_n(\mathbf{T})^t & \mathbf{I}\end{bmatrix}\\ &= \Sigma_s - \sigma_g^2\begin{bmatrix}\phi_n(\mathbf{T})\\ \mathbf{I}\end{bmatrix}\begin{bmatrix}\phi_n(\mathbf{T})^t & \mathbf{I}\end{bmatrix} +\\ &\quad \sigma^2\begin{bmatrix}\phi_n(\mathbf{T})\\ \mathbf{I}\end{bmatrix}\Sigma_1\begin{bmatrix}\phi_n(\mathbf{T})^t & \mathbf{I}\end{bmatrix}\\ &= \Sigma_s + \begin{bmatrix}\phi_n(\mathbf{T})\\ \mathbf{I}\end{bmatrix}(\sigma^2\Sigma_1 - \sigma_g^2\mathbf{I})\begin{bmatrix}\phi_n(\mathbf{T})^t & \mathbf{I}\end{bmatrix},\end{aligned} \tag{7}$$

and this matrix involves the trapdoor information. Notice that the structure of $(\sigma^2\Sigma_1 - \sigma_g^2\mathbf{I})$ in the above equation is shown as

$$= \begin{bmatrix}(\sigma^2\Sigma_1 - \sigma_g^2\mathbf{I}) & & & \\ & & & -(\frac{1}{2})^{k-1}\sigma^2\\ & \sigma^2-\sigma_g^2 & & -(\frac{1}{2})^{k-2}\sigma^2\\ & & \ddots & \vdots\\ -(\frac{1}{2})^{k-1}\sigma^2 & -(\frac{1}{2})^{k-2}\sigma^2 & \cdots & \alpha\sigma^2-\sigma_g^2\end{bmatrix}. \tag{8}$$

This matrix involves several non-zero entries, and may cause the leakage of trapdoor information in statistic.

In short, we introduce the main idea of GM18 into the perturbation sampling, and design two new constructions that are suitable for the non-spherical G-lattice sampler. The difference of them is the choice of Σ_s that represents covariance matrix of preimage vectors. According to the different LBC design requirements, theoretical analysis or practical application, an appropriate construction can be selected for perturbation sampling. The details will be shown in two next subsections.

A. Construction 1

Being similar to the traditional GSLT schemes over ring, Construction 1 aims to a spherical Gaussian distribution of the output preimage vector, i.e., $\Sigma_s = s^2\mathbf{I}$ for some appropriate s. Therefore, the structure of Σ_p can be defined as:

$$\begin{aligned}\Sigma_p &= s^2\mathbf{I} - \begin{bmatrix}\overline{\mathbf{T}}\\ \mathbf{I}\end{bmatrix}\Sigma_G\begin{bmatrix}\overline{\mathbf{T}}^t & \mathbf{I}\end{bmatrix}\\ &= \begin{bmatrix}s^2\mathbf{I}_{2n} - \sigma^2\overline{\mathbf{T}}\Sigma_1\overline{\mathbf{T}}^t & -\sigma^2\overline{\mathbf{T}}\Sigma_1\\ -\sigma^2\Sigma_1\overline{\mathbf{T}}^t & s^2\mathbf{I}_{kn} - \sigma^2\Sigma_1\end{bmatrix},\end{aligned} \tag{9}$$

where we set $\overline{\mathbf{T}} = \phi_n(\mathbf{T})$ for clarity.

Then, we use Lemma 1 to sample $\mathbf{p}$ according to the above Σ_p. First, we initialize a covariance matrix $\Sigma = (s^2\mathbf{I}_{kn} - \sigma^2\Sigma_1) \in \mathbb{R}^{kn\times kn}$ and a center $\mathbf{c} = 0$. Then, a vector $\mathbf{p}_s$ is sampled from $D_{\mathbb{Z}^{kn},\sqrt{\Sigma},\mathbf{c}}$. Note that this vector can be also regarded as $\mathbf{p}_s \in R^k$, i.e., $\mathbf{p}_s \leftarrow D_{R^k,\sqrt{s^2\mathbf{I}_k-\sigma^2\Sigma_2}}$. Furthermore, the covariance matrix Σ and center $\mathbf{c}$ are updated as follows:

$$\begin{cases}\Sigma &= \Sigma_p/(s^2\mathbf{I}_{nk} - \sigma^2\Sigma_1)\\ &= s^2\mathbf{I}_{2n} - \sigma^2\overline{\mathbf{T}}\Sigma_1\overline{\mathbf{T}}^t - \sigma^4\overline{\mathbf{T}}\Sigma_1(s^2\mathbf{I}_{nk} - \sigma^2\Sigma_1)^{-1}\\ &\quad\cdot\Sigma_1\overline{\mathbf{T}}^t \in \mathbb{R}^{2n\times 2n},\\ \mathbf{c} &= -\sigma^2\overline{\mathbf{T}}\Sigma_1(s^2\mathbf{I}_{nk} - \sigma^2\Sigma_1)^{-1}\cdot\mathbf{p}_s \in \mathbb{R}^{2n}.\end{cases} \tag{10}$$

Here, we assume that $\mathbf{c} = (\gamma(c_1), \gamma(c_2))$ for $(c_1, c_2) \in R^2$, and

$$\Sigma = \begin{bmatrix}\phi_n(a) & \phi_n(b)\\ \phi_n(b^*) & \phi_n(d)\end{bmatrix}, \tag{11}$$

where, $b^* \in \mathcal{P}_n$ satisfies that $\phi_n(b^*) = \phi_n(b)^t$. Finally, $\mathrm{SampleF}_z()$ is iteratively invoked to sample a vector $(p_1, p_2) \in R^2$ according to Σ and $\mathbf{c}$, i.e., sample $p_2 \leftarrow \mathrm{SampleF}_z(d, c_2)$ and $p_1 \leftarrow \mathrm{SampleF}_z(a - bd^{-1}b^*, c_1 + bd^{-1}(p_2 - c_2))$ sequentially. According to Lemma 1, the vector $\mathbf{p} = (p_1, p_2, \mathbf{p}_s) \in R^{k+2}$ is the required perturbation.

In order to reduce the time cost for sampling perturbation, two matrices in Equation (10), Σ and $\overline{\mathbf{T}}\Sigma_1$, can be precomputed, and stored as matrices over the ring $\mathcal{P}_n$ to reduce the storage cost. Algorithm 5 and 6 give the details of precomputation and perturbation sampling algorithms.

We provide the optimized operations in Algorithm 5 and 6.

Algorithm 5 SampleP1-Precomp$(\mathbf{T}, \overline{s}, \widetilde{s}, \sigma)$

Input: Trapdoor $\mathbf{T}$, preimage gaussian parameters $\overline{s}, \widetilde{s}$ and G-lattice sampling gaussian parameter σ;
Output: Two matrices $\Sigma_{t_1} \in \mathcal{P}_n^{2\times k}$ and $\Sigma_{t_2} \in \mathcal{P}_n^{2\times 2}$;

1: Compute $(s^2\mathbf{I}_k - \sigma^2\Sigma_2)^{-1}$.
2: Compute $\Sigma_{t_1} = \mathbf{T}\Sigma_2$
3: Compute $\Sigma_{t_2} = s^2\mathbf{I}_2 - \sigma^2\Sigma_{t_1}\mathbf{T}^t - \sigma^4\Sigma_{t_1} \cdot (s^2\mathbf{I}_k - \sigma^2\Sigma_2)^{-1} \cdot \Sigma_{t_1}^t$.
4: Compute $\Sigma_{t_1} = -\sigma^2 \cdot \Sigma_{t_1}(s^2\mathbf{I}_k - \sigma^2\Sigma_2)^{-1}$.
5: **return** $(\Sigma_{t_1}, \Sigma_{t_2})$.

Algorithm 6 SampleP1overRing$(n, k, q, \overline{s}, \widetilde{s}, \sigma, \Sigma_{t_1}, \Sigma_{t_2})$

Input: Parameters n, k and q, Gaussian parameters $\overline{s}, \widetilde{s}$ used for perturbation sampling, σ used for G-Sampling and two matrices, Σ_{t_1} and Σ_{t_2};
Output: The perturbation vector $\mathbf{p} \in R^{k+2}$;

1: Sample a vector $\mathbf{p}_s$ from the distribution $D_{R^k, \sqrt{s^2\mathbf{I}_k - \sigma^2\Sigma_2}}$.
2: Compute $\mathbf{c} = (c_1, c_2) = \Sigma_{t_1}\mathbf{p}_s$.
3: Regard Σ_{t_2} as $\begin{bmatrix} a & b \\ b^* & d \end{bmatrix}$.
4: Invoke $\mathrm{SampleF}_z(d, c_2)$ to sample $p_2 \in R$.
5: Invoke $\mathrm{SampleF}_z(a - bd^{-1}b^*, c_1 + bd^{-1}(p_2 - c_2))$ to sample $p_1 \in R$.
6: **return** $\mathbf{p} = (p_1, p_2, \mathbf{p}_s)$.

1) In the first step of Algorithm 6, $\mathbf{p}_s \leftarrow D_{R^k, \sqrt{s^2\mathbf{I}_k - \sigma^2\Sigma_2}}$ can be carried out by Equation (12) according to the work [22].

$$\left\lceil \sqrt{(s^2 - \theta^2)\mathbf{I}_{kn} - \sigma^2\Sigma_1} \cdot D_{\mathbb{R},1}^{kn} \right\rfloor_\theta, \tag{12}$$

where, $D_{\mathbb{R},1}^{kn}$ denotes the (kn)-dimensional continuous Gaussian distribution over $\mathbb{R}$ with the Gaussian parameter 1, $\lceil\cdot\rfloor_\theta$ denotes the randomized rounding operation with a parameter θ.
If we directly use Cholesky decomposition to compute $(\sqrt{(s^2 - \theta^2)\mathbf{I}_{kn} - \sigma^2\Sigma_1})$, the computation cost is heavy because the time complexity is $O(nk)^3$. However, we observe that the structure of $((s^2 - \theta^2)\mathbf{I}_{(k-1)n} - \sigma^2\Sigma_1)$ is shown as the equation:

$$\begin{bmatrix} & & & \frac{\sigma^2}{2^{k-1}}\mathbf{I}_n \\ & (s^2 - \sigma^2 - \theta^2)\mathbf{I}_{(k-1)n} & & \vdots \\ & & & \frac{\sigma^2}{2}\mathbf{I}_n \\ \frac{\sigma^2}{2^{k-1}}\mathbf{I}_n & \cdots & \frac{\sigma^2}{2}\mathbf{I}_n & (s^2 - \theta^2 - \alpha\sigma^2)\mathbf{I}_n \end{bmatrix}. \tag{13}$$

Because of the above special structure, the Cholesky decomposition form of this matrix has a fixed structure shown as the equation,

$$\begin{bmatrix} \sqrt{s^2 - \theta^2 - \sigma^2}\mathbf{I}_{(k-1)n} & 0 \\ \alpha'_{k-1}\mathbf{I}_n & \cdots & \alpha'_1\mathbf{I}_n & \alpha'\mathbf{I}_n \end{bmatrix}, \tag{14}$$

where, $\alpha'_i = \frac{\sigma^2}{2^i\sqrt{s^2 - \theta^2 - \sigma^2}}$, and $\alpha' = \sqrt{s^2 - \theta^2 - \alpha\sigma^2 - \sum_{i=1}^{k-1}\alpha_i'^2}$. Therefore, the parameter $\sqrt{(s^2 - \theta^2)\mathbf{I}_{kn} - \sigma^2\Sigma_1}$ can be generated directly without the inefficient Cholesky decomposition operation. Moreover, this matrix can be also regarded as a $(k \times k)$-dimensional matrix over $\mathcal{P}_n$, even $\mathbb{R}$.

2) For the first step of Algorithm 5, we need to compute the inverse of $(s^2\mathbf{I}_k - \sigma^2\Sigma_2) \in \mathcal{P}_n^{k\times k}$. We observe that each entry of this matrix is a constant rather than a polynomial. Therefore we can regard it as a $(k \times k)$-dimensional matrix over $\mathbb{R}$, and compute its inverse, instead of inversing the $(kn \times kn)$-dimensional matrix $\phi_n(s^2\mathbf{I}_k - \sigma^2\Sigma_2)$.

B. Construction 2

According to Construction 1, for some parameter $\Sigma_s = s^2\mathbf{I}$, the lower right part of Σ_p is $(s^2\mathbf{I}_{kn} - \sigma^2\Sigma_1)$ that no longer corresponds to a spherical Gaussian distribution. Both of sampling and inversing operations for this matrix are more inefficient than those for $(s^2 - \sigma^2)\mathbf{I}_{kn}$.

In order to improve the efficiency of perturbation sampling, we set the Gaussian parameter Σ_s of the output preimage vector as the equation

$$\Sigma_s = \begin{bmatrix} \overline{s}^2\mathbf{I}_{2n} & & & -\sigma^2 \cdot \frac{1}{2^{k-1}}\mathbf{I}_n \\ & \widetilde{s}^2\mathbf{I}_{(k-1)n} & & \vdots \\ & & & -\sigma^2 \cdot \frac{1}{2}\mathbf{I}_n \\ & -\sigma^2 \cdot \frac{1}{2^{k-1}}\mathbf{I}_n \cdots -\sigma^2 \cdot \frac{1}{2}\mathbf{I}_n & & \widetilde{s}^2 + \sigma^2 \cdot (\alpha - 1)\mathbf{I}_n \end{bmatrix}. \tag{15}$$

Clearly, the structure of Σ_s does not have any information of $\mathbf{T}$. Therefore, the output vectors will not leak $\mathbf{T}$ statistically despite the output vectors follow a non-spherical Gaussian distribution.

Moreover, Σ_p have the following structure:

$$\begin{aligned} \Sigma_p &= \Sigma_s - \begin{bmatrix} \overline{\mathbf{T}} \\ \mathbf{I} \end{bmatrix} \Sigma_G \begin{bmatrix} \overline{\mathbf{T}}^t & \mathbf{I} \end{bmatrix} \\ &= \begin{bmatrix} \overline{s}^2\mathbf{I}_{2n} - \sigma^2\overline{\mathbf{T}}\Sigma_1\overline{\mathbf{T}}^t & -\sigma^2\overline{\mathbf{T}}\Sigma_1 \\ -\sigma^2\Sigma_1\overline{\mathbf{T}}^t & (\widetilde{s}^2 - \sigma^2)\mathbf{I}_{kn} \end{bmatrix}. \end{aligned} \tag{16}$$

We observe that the lower right part of Σ_p is $(\widetilde{s}^2 - \sigma^2)\mathbf{I}_{kn}$ corresponding to a spherical Gaussian distribution. Therefore, the operation $\mathbf{p}_s \leftarrow D_{R_q^k, \sqrt{\widetilde{s}^2 - \sigma^2}}$ can be carried out by independently sampling (kn) integers from the discrete Gaussian distribution $D_{\mathbb{Z}, \sqrt{\widetilde{s}^2 - \sigma^2}}$. Since the inverse of $(\widetilde{s}^2 - \sigma^2)\mathbf{I}_{kn}$ can be easily computed (i.e., $\frac{1}{\widetilde{s}^2 - \sigma^2}\mathbf{I}_{kn}$), its Schur complement in Σ_p can be easily computed as

$$\begin{aligned} &\Sigma_p / \left((\widetilde{s}^2 - \sigma^2)\mathbf{I}_{kn}\right) \\ =\ &\overline{s}^2\mathbf{I}_{2n} - \sigma^2\overline{\mathbf{T}}\Sigma_1\overline{\mathbf{T}}^t - \frac{\sigma^4}{\widetilde{s}^2 - \sigma^2}\overline{\mathbf{T}}\Sigma_1^2\overline{\mathbf{T}}^t \in \mathbb{R}^{2n\times 2n}. \end{aligned} \tag{17}$$

Based on the above description, the complete steps of Construction 2 are provided as Algorithm 7 and Algorithm 8.

Our presented constructions are designed for different Gaussian parameters, in order to meet the varying requirements in the design of LBC schemes. In Construction 1, the Gaussian parameter $\Sigma_s = s^2\mathbf{I}$ has a simple structure (in fact, it can be operated as a real number). It is beneficial for

Algorithm 7 SampleP2-Precomp$(\mathbf{T}, \overline{s}, \widetilde{s}, \sigma)$

Input: Trapdoor $\mathbf{T}$, preimage gaussian parameters $\overline{s}, \widetilde{s}$ and G-lattice sampling gaussian parameter σ;
Output: Two matrices $\Sigma_{t_1} \in \mathcal{P}_n^{2\times k}$ and $\Sigma_{t_2} \in \mathcal{P}_n^{2\times 2}$;

1: Compute $\Sigma_{t_1} = \mathbf{T}\Sigma_2$.
2: Compute $\Sigma_{t_2} = \overline{s}^2\mathbf{I} - \sigma^2\Sigma_{t_1}\mathbf{T}^t - \frac{\sigma^4}{\widetilde{s}^2-\sigma^2}\Sigma_{t_1}\Sigma_{t_1}^t$.
3: Compute $\Sigma_{t_1} = -\frac{\sigma^2}{\widetilde{s}^2-\sigma^2}\Sigma_{t_1}$.
4: **return** $(\Sigma_{t_1}, \Sigma_{t_2})$.

Algorithm 8 SampleP2overRing$(n, k, q, \overline{s}, \widetilde{s}, \sigma, \Sigma_{t_1}, \Sigma_{t_2})$

Input: Parameters n, k and q, Gaussian parameters $\overline{s}, \widetilde{s}$ used for perturbation sampling, σ used for G-lattice sampling and two matrices, Σ_{t_1} and Σ_{t_2};
Output: The perturbation vector $\mathbf{p} \in R^{k+2}$

1: Sample a vector $\mathbf{p}_s$ from the distribution $D_{R^k,\sqrt{\widetilde{s}^2-\sigma^2}}$.
2: Compute $\mathbf{c} = (c_1, c_2) = \Sigma_{t_1}\mathbf{p}_s$.
3: regard Σ_{t_2} as $\begin{bmatrix} a & b \\ b^* & d \end{bmatrix}$.
4: Invoke $\mathrm{SampleF}_z(d, c_2)$ to sample $p_2 \in R$.
5: Invoke $\mathrm{SampleF}_z(a - bd^{-1}b^*, c_1 + bd^{-1}(p_2 - c_2))$ to sample $p_1 \in R$.
6: **return** $\mathbf{p} = (p_1, p_2, \mathbf{p}_s)$.

analyzing cumulative error constraints and storage space of preimage vectors in LBC scheme design, e.g., Attribute-Based Encryption [12] and Identity-Based Encryption [11]. While, Construction 2 selects the optimized Gaussian parameter with a greater consideration for sampling efficiency. It might be a better choice when there are no strict requirements for the Gaussian parameter [18].

C. Correctness of Construction 1 and Construction 2

The correctness of two constructions is summarized by the following theorem.

Theorem 2: Suppose that $\mathbf{T} \in R_q^{2\times k}$ is the trapdoor, $s, \overline{s}, \widetilde{s}, \sigma > 0$ are four positive real numbers, $\Sigma_p = \Sigma_s - \begin{bmatrix} \mathbf{T} \\ \mathbf{I} \end{bmatrix} \Sigma_G \begin{bmatrix} \overline{\mathbf{T}}^t & \mathbf{I} \end{bmatrix}$ is the covariance matrix of perturbation.

If $\Sigma_p \succeq \eta_\epsilon^2(\mathbb{Z}^{(k+2)n})$, then the algorithms, SampleP1overRing and SampleP$-$ 2overRing, can return vectors $\mathbf{p} \in R^{(k+2)}$ whose distribution is statistically indistinguishable from $D_{R^{k+2},\sqrt{\Sigma_p}}$.

Before the proof of Theorem 2, we provide the following lemma:

Lemma 2 ([16], Lemma 6): Let $\epsilon > 0$, r and s be positive integers, and $\Sigma = \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{B}^t & \mathbf{D} \end{bmatrix} \in \mathbb{R}^{(r+s)\times(r+s)}$ be a positive definite matrix made out of blocks $\mathbf{A} \in \mathbb{R}^{r\times r}$, $\mathbf{B} \in \mathbb{R}^{r\times s}$, $\mathbf{D} \in \mathbb{R}^{s\times s}$. If $\Sigma_p \succeq \eta_\epsilon^2(\mathbb{Z}^{r+s})$, then $\mathbf{D} \succeq \eta_\epsilon^2(\mathbb{Z}^s)$ and $\Sigma/\mathbf{D} \succeq \eta_\epsilon^2(\mathbb{Z}^r)$.

Proof 2: In order to simplify description, we let $\eta = \eta_\epsilon(\mathbb{Z}^{(k+2)n})$. The proof of correctness of Construction 1 is shown as follows:

According to Lemma 1 and $\Sigma_p \succeq \eta^2$, the operation for sampling $\mathbf{p} \in R^{k+2}$ by Σ_p can be carried out by sampling two vectors, $\mathbf{p}_s$ and (p_1, p_2), where $\mathbf{p}_s \in R^k$ is sampled with the covariance matrix $(s^2\mathbf{I}_{nk} - \sigma^2\Sigma_1)$, and $(p_1, p_2) \in R^2$ is sampled with the covariance matrix $\Sigma = \Sigma_p/(s^2\mathbf{I}_{nk} - \sigma^2\Sigma_1)$ and the center $\mathbf{c} = -\sigma^2\overline{\mathbf{T}}\Sigma_1(s^2\mathbf{I}_{nk} - \sigma^2\Sigma_1)^{-1} \cdot \mathbf{p}_s$. Note that we have $(s^2\mathbf{I}_{nk} - \sigma^2\Sigma_1) \succ \eta^2$ and $\Sigma \succ \eta^2$ because of Lemma 2 and $\Sigma_p \succeq \eta^2$.

Let $\Sigma = \begin{bmatrix} \phi_n(a) & \phi_n(b) \\ \phi_n(b^*) & \phi_n(d) \end{bmatrix}$, and $\mathbf{c} = (\gamma(c_1), \gamma(c_2))$. According to $\Sigma \succ \eta^2$ and Lemma 1, the operation for sampling (p_1, p_2) can be carried out by invoking $\mathrm{SampleF}_z$, i.e., sequentially sampling $p_2 \leftarrow \mathrm{SampleF}_z(d, c_2)$ and $p_1 \leftarrow \mathrm{SampleF}_z(a - bd^{-1}b^*, c_1 + bd^{-1}(p_2 - c_2))$. Moreover, Lemma 2 and $\Sigma \succ \eta^2$ induce that each covariance given to $\mathrm{SampleF}_z$ can meet the condition of Lemma 1, i.e., $\phi_n(d) \succ \eta^2$ and $\phi_n(a - bd^{-1}b^*) \succ \eta^2$. Therefore, the correctness of Construction 1 is proven.

Similarly, correctness of Construction 2 can be also proven by iteratively using Lemma 1 and Lemma 2.

VI. Analysis of Parameters in Construction 2

We further analyze how to choose $\overline{s}$ and $\widetilde{s}$ to design the Gaussian parameter Σ_s in Construction 2. To do it, the parameters of Construction 2 should meet two following conditions:

1) According to Theorem 2, the condition $\Sigma_p \succeq \eta_\epsilon^2(\mathbb{Z}^{(k+2)n})$ should be met.
2) In order to ensure the correctness of GSLT scheme, the condition presented in Theorem 2 of HJ19 should also be met, i.e., $\Sigma_s \succeq \begin{bmatrix} \mathbf{T} \\ \mathbf{I} \end{bmatrix} (a + \Sigma_G) \begin{bmatrix} \overline{\mathbf{T}}^t & \mathbf{I} \end{bmatrix}$. It means that the condition $\Sigma_p \succeq a \begin{bmatrix} \mathbf{T} \\ \mathbf{I} \end{bmatrix} \begin{bmatrix} \overline{\mathbf{T}}^t & \mathbf{I} \end{bmatrix}$ should be met.

Note that there are two cases for generating parameter a in HJ19: 1

1) If $q \geq 2^{k-1+\frac{1}{2}}$, $a = \frac{q^2}{q^2-4^{k-1}} \in (\frac{4}{3}, 2)$. This case is the optimal case.
2) If $q < 2^{k-1+\frac{1}{2}}$, the Gaussian parameter should be set as $\Sigma_G^* = 2 \cdot \Sigma_G$, i.e., set the Gaussian parameter of Algorithm 4 as 2σ. In this case, $a = \frac{q^2}{q^2-4^{k-1-\frac{1}{2}}}$ is also in $(\frac{4}{3}, 2)$.

In this paper, we mainly consider the first case, because it means a high quality of output vectors. The discussion of $\overline{s}$ and $\widetilde{s}$ is shown as follows:

For the first condition, we set $\eta = \eta_\epsilon(\mathbb{Z}^{(k+2)n})$, and expect $(\Sigma_p - \eta^2\mathbf{I})$ is a positive definite matrix. Firstly, $(\Sigma_p - \eta^2\mathbf{I})$ have the following construction:

$$\Sigma_p - \eta^2\mathbf{I} = \begin{bmatrix} (\overline{s}^2 - \eta^2)\mathbf{I}_{2n} - \sigma^2\overline{\mathbf{T}}\Sigma_1\overline{\mathbf{T}}^t & -\sigma^2\overline{\mathbf{T}}\Sigma_1 \\ -\sigma^2\Sigma_1\overline{\mathbf{T}}^t & (\widetilde{s}^2 - \sigma^2 - \eta^2)\mathbf{I}_{kn} \end{bmatrix}. \quad (18)$$

Therefore, this condition requires that both of $(\widetilde{s}^2 - \sigma^2 - \eta^2)\mathbf{I}_{kn}$ and its Schur complement

$$(\Sigma_p - \eta^2\mathbf{I})/((\widetilde{s}^2 - \sigma^2 - \eta^2)\mathbf{I}_{kn}) = (\overline{s}^2 - \eta^2)\mathbf{I}_{2n} - \left(\sigma^2\overline{\mathbf{T}}\Sigma_1\overline{\mathbf{T}}^t + \frac{\sigma^4}{\widetilde{s}^2-\sigma^2-\eta^2}\overline{\mathbf{T}}\Sigma_1^2\overline{\mathbf{T}}^t\right) \quad (19)$$

are positive definite matrices. The former requires that $\widetilde{s} > \sqrt{\sigma^2+\eta^2}$. For the latter, suppose that $s_g > \mathrm{S}_1(\Sigma_1)$ is a positive real, so $s_g\mathbf{I} \succ \Sigma_1$. Therefore, we have

$$\begin{array}{rl} & \sigma^2\overline{\mathbf{T}}\Sigma_1\overline{\mathbf{T}}^t + \frac{\sigma^4}{\overline{s}^2-\sigma^2-\eta^2}\overline{\mathbf{T}}\Sigma_1^2\overline{\mathbf{T}}^t \\ \prec & \left(\sigma^2\cdot s_g + \frac{\sigma^4\cdot s_g^2}{\overline{s}^2-\sigma^2-\eta^2}\right)\overline{\mathbf{T}}\overline{\mathbf{T}}^t \\ = & \frac{C}{\overline{s}^2-\sigma^2-\eta^2}\overline{\mathbf{T}}\overline{\mathbf{T}}^t, \end{array} \tag{20}$$

where $C = s_g\sigma^2\widetilde{s}^2 - s_g\sigma^2\eta^2 + s_g\sigma^4(s_g-1)$. According to Equation (19) and Equation (20), the Schur complement is positive definite, if

$$(\overline{s}^2-\eta^2)\mathbf{I}_{2n} - \frac{C}{\widetilde{s}^2-\sigma^2-\eta^2}\overline{\mathbf{T}}\overline{\mathbf{T}}^t \succ 0. \tag{21}$$

For Equation (21), we have

$$\begin{array}{rl} & (\overline{s}^2-\eta^2)\mathbf{I}_{2n} - \frac{C}{\overline{s}^2-\sigma^2-\eta^2}\overline{\mathbf{T}}\overline{\mathbf{T}}^t \\ = & (\overline{s}^2-\eta^2)\left(\mathbf{I}_{2n} - \frac{C}{(\overline{s}^2-\eta^2)(\overline{s}^2-\sigma^2-\eta^2)}\overline{\mathbf{T}}\overline{\mathbf{T}}^t\right). \end{array} \tag{22}$$

It requires that $\mathbf{I}_{2n} - \frac{C}{(\overline{s}^2-\eta^2)(\overline{s}^2-\sigma^2-\eta^2)}\overline{\mathbf{T}}\overline{\mathbf{T}}^t$ is positive definite, i.e., $1 - \frac{C}{(\overline{s}^2-\eta^2)(\overline{s}^2-\sigma^2-\eta^2)}\mathrm{S}_1^2(\overline{\mathbf{T}}) > 0$. Based on the above analysis, we have $\overline{s} > \sqrt{\frac{C\cdot\mathrm{S}_1^2(\overline{\mathbf{T}})}{\widetilde{s}^2-\sigma^2-\eta^2}+\eta^2}$.

For the second condition, we expect $\Sigma_p - a\begin{bmatrix}\mathbf{T}\\ \mathbf{I}\end{bmatrix}\begin{bmatrix}\overline{\mathbf{T}}^t & \mathbf{I}\end{bmatrix}$ is a positive definite matrix. Firstly, we observe it has the following construction:

$$\begin{array}{rl} & \Sigma_p - a\begin{bmatrix}\overline{\mathbf{T}}\\ \mathbf{I}\end{bmatrix}\begin{bmatrix}\overline{\mathbf{T}}^t & \mathbf{I}\end{bmatrix} \\ = & \begin{bmatrix}\overline{s}^2\mathbf{I}_{2n} - \sigma^2\overline{\mathbf{T}}\Sigma_1\overline{\mathbf{T}}^t - a\overline{\mathbf{T}}\overline{\mathbf{T}}^t & -\sigma^2\overline{\mathbf{T}}\Sigma_1 - a\overline{\mathbf{T}} \\ -\sigma^2\Sigma_1\overline{\mathbf{T}}^t - a\overline{\mathbf{T}}^t & (\widetilde{s}^2-\sigma^2-a)\mathbf{I}_{kn}\end{bmatrix}. \end{array} \tag{23}$$

It requires that both of $(\widetilde{s}^2-\sigma^2-a)\mathbf{I}_{kn}$ and its Schur complement

$$\begin{array}{rl} & (\Sigma_p - a\begin{bmatrix}\mathbf{T}\\ \mathbf{I}\end{bmatrix}\begin{bmatrix}\overline{\mathbf{T}}^t & \mathbf{I}\end{bmatrix})/(\widetilde{s}^2-\sigma^2-a)\mathbf{I}_{kn} \\ = & \overline{s}^2\mathbf{I}_{2n} - \sigma^2\overline{\mathbf{T}}\Sigma_1\overline{\mathbf{T}}^t - a\overline{\mathbf{T}}\overline{\mathbf{T}}^t \\ & -\frac{1}{\overline{s}^2-\sigma^2-a}(\sigma^2\overline{\mathbf{T}}\Sigma_1 + a\overline{\mathbf{T}})(\sigma^2\Sigma_1\overline{\mathbf{T}}^t + a\overline{\mathbf{T}}^t) \end{array} \tag{24}$$

are positive definite. The former requires that $\widetilde{s} > \sqrt{\sigma^2+a}$. Then, we discuss the latter. By using $s_g > \mathrm{S}_1(\Sigma_1)$, we have

$$\begin{array}{rl} & (\Sigma_p - a\begin{bmatrix}\mathbf{T}\\ \mathbf{I}\end{bmatrix}\begin{bmatrix}\overline{\mathbf{T}}^t & \mathbf{I}\end{bmatrix})/(\widetilde{s}^2-\sigma^2-a)\mathbf{I}_{kn} \\ \succ & \overline{s}^2\mathbf{I}_{2n} - \left((s_g\sigma^2+a)\overline{\mathbf{T}}\overline{\mathbf{T}}^t + \frac{(s_g\sigma^2+a)^2}{\widetilde{s}^2-\sigma^2-a}\overline{\mathbf{T}}\overline{\mathbf{T}}^t\right), \\ = & \overline{s}^2\mathbf{I}_{2n} - \frac{C'}{\overline{s}^2-\sigma^2-a}\overline{\mathbf{T}}\overline{\mathbf{T}}^t \end{array} \tag{25}$$

where, $C' = (s_g\sigma^2+a)(\widetilde{s}^2-\sigma^2-a)+(s_g\sigma^2+a)^2$. Therefore, Equation (25) requires that $\overline{s} > \sqrt{\frac{C'}{\widetilde{s}^2-\sigma^2-a}}\cdot\mathrm{S}_1(\overline{\mathbf{T}})$ is met.

Based on the above analysis, the two parameters, $\overline{s}$ and $\widetilde{s}$, should meet Equation (26).

$$\begin{cases} \widetilde{s} > \sqrt{\sigma^2+\eta^2} \quad \text{and} \quad \overline{s} > \sqrt{\frac{C\cdot\mathrm{S}_1(\overline{\mathbf{T}})}{\widetilde{s}^2-\sigma^2-\eta^2}+\eta^2}, \\ \widetilde{s} > \sqrt{\sigma^2+a} \quad \text{and} \quad \overline{s} > \sqrt{\frac{C'}{\widetilde{s}^2-\sigma^2-a}}\cdot\mathrm{S}_1(\overline{\mathbf{T}}). \end{cases} \tag{26}$$

VII. Complexity Analysis and Experiments

Recently, more LBC encryption schemes under ring setting are applied to protect the security of private data in different scenarios, such as cloud computing [23] and Internet of things [24]. In the above schemes, GSLT is usually used as the core technology of the process of key extractor that extracts short preimage vectors to constitute users' private keys. Specifically, the presented GSLT can be employed (but not limited to) the following LBC encryption:

- **Identity-based encryption** [11], [12]: which employs user's identity as the minimum decryption authorization unit. In the scheme, the private data is encrypted by the public key corresponding to a user's identity, while the trusted third-party invokes GSLT to extract private key for this user according to his identity.
- **Attribute-based encryption** [13], [23]: which uses user's attributes as the minimum decryption authorization unit, and it is more fine-grained than the identity-based encryption. In this scheme, the private data is encrypted by attributes and policy rather than identity, while the trusted third-party invokes GSLT to extract private keys for this user according to his attributes.
- **Fully homomorphic encryption** [25]: which can perform arithmetic operations on encrypted data without decrypting it. In this scheme, GSLT is used to extract private keys for users. The security of the encryption scheme relies on the hardness of finding the trapdoor for a given lattice function, which is related to problems such as SIS and LWE over ring.

Therefore, the research of GSLT is meaningful since it improves the performance of LBC encryption schemes, especially the key generation process. To estimate the performance of the presented GSLT, we here provide the time and space complexity of the presented GSLT scheme over ring, including off-line and on-line stages, and compare them with some existing schemes. Furthermore, the experiments are designed for evaluating the execution time of the presented GSLT scheme and these existing schemes. These experiments are implemented by Mathematica with the default computational accuracy, and executed on 64-bit Windows 10 under Intel(R) Core(TM) i7-10750H CPU @2.60 GHz, 16.0 G ROM.

A. Off-line stage

In this subsection, we analyze the performance of two presented constructions on perturbation sampling, including Construction 1 and Construction 2, in terms of time and space complexity. Here, we assume that M_R and $M_{\mathcal{P}_{2^i}}$ are represented as the execution time of multiplications over R and $\mathcal{P}_{2^i}$. $Inv_{\mathcal{P}_{2^i}}$ is denoted as the inversion operation for element over $\mathcal{P}_{2^i}$. $\mathrm{SampleZ}$ and $\mathrm{SampleF}$ represent the Gaussian sampler over $\mathbb{Z}$ and $\mathbb{R}$, respectively. Moreover, the addition over $\mathcal{P}_n$ and the multiplication between a real scalar and a vector or matrix over $\mathcal{P}_n$ are neglected, because these operations are more efficient than the multiplication over $\mathcal{P}_n$.

As shown in Table II and III, Construction 1 and 2 involve $6k^2+8k$ and $2k^2+8k$ multiplication operations over $\mathcal{P}_n$ at the

precomputation stage, respectively. For the perturbation sampling stage, Construction 1 involves k^2 extra multiplications over $\mathcal{P}_n$ in comparison with Construction 2, and requires kn sampling operations on real number. The above analysis also explains the reason why Construction 1 is less efficient than Construction 2.

Currently, there lacks the optimized perturbation sampling for non-spherical G-lattice vectors, thus the MP12's original perturbation sampling (i.e., Cholesky decomposition) is considered as the only choice. However, our presented constructions have better performances in comparison with MP12's original method. Suppose that the multiplication $M_{\mathcal{P}_n}$ is regarded as n^2 multiplications over $\mathbb{R}$ (actually, $M_{\mathcal{P}_n}$ can be further optimized to lower time complexity). The comparison between MP12 and two constructions is shown as follows:

- **Original perturbation sampling:** which needs $O(nk)^3$ precomputation (Cholesky decomposition), and operates on reals in $O(nk)^2$ time complexity.
- **Construction 1:** which requires $O(nk)^2$ precomputation, and $O(k^2 \cdot n \cdot \log n)$ time complexity.
- **Construction 2:** which requires $O(nk)^2$ precomputation, and $O(k \cdot n \cdot \log n)$ time complexity.

The above results indicate that the efficiency of two constructions is higher than the original one, and are more suitable for the non-spherical G-lattice sampling.

Furthermore, HJ19 has indicated that high efficiency of non-spherical G-lattice sampling is implemented on extra computational overheads of perturbation sampling, but lacks enough evidences on this point. Therefore, we provide more analyses by comparing GM18's perturbation sampling with our constructions. Table II and III show the time complexity of precomputation and perturbation sampling algorithms in these schemes, respectively. It is worthy to note that our constructions are aimed at the non-spherical G-lattice sampling, whereas GM18 is at the spherical one.

TABLE II
THE TIME COMPLEXITY OF PRECOMPUTATION ALGORITHMS

	Multiplication over R	Multiplication over $\mathcal{P}_n$	Matrix inverse over $\mathbb{R}^{k\times k}$
GM18	$2k$	-	$2k$
Construction 1	-	$6k^2+8k$	1
Construction 2	-	$2k^2+8k$	-

As shown in Table II, GM18 only performs $2k$ multiplication operations over R, because it only need to precompute the matrix $\mathbf{TT}^t \in R^{2\times 2}$, but the constructions need $O(k^2)$ multiplication operations over $\mathcal{P}_n$. The reason is that the covariance matrix of output vectors from Algorithm 4 is $\Sigma_G = \sigma^2 \cdot \Sigma_2$ rather than $\sigma^2 \cdot \mathbf{I}$, and $\phi_n(\Sigma_2)$ involves several real numbers. In order to clearly show the execution efficiency of the above algorithms, two experiments are designed to estimate their execution time under different parameters:

1) k (i.e., the bits of q) is fixed to 20, and n increases form 8 to 512. The execution time of the three algorithms are shown as Fig. 4(a).
2) n is fixed to 64, and k increases from 5 to 30. The execution time of the three algorithms are shown as Fig. 4(b).

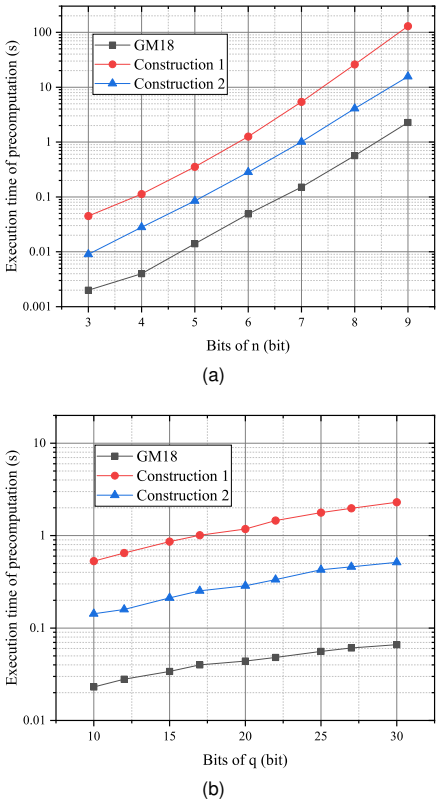

Fig. 4. Execution time of precomputation

As shown in Fig. 4, as n increases from 8 to 512, the execution time of GM18's precomputaion algorithm is 0.002s - 2.274s, and the execution time of precomputation in Construction 1 and Construction 2 are 0.045s - 129.363s and 0.009s - 15.679s, respectively. When k increases from 10 to 30, the execution time of GM18's precomputaion algorithm is 0.023s - 0.066s, and the execution time of precomputation in Construction 1 and Construction 2 are 0.053s - 2.301s and 0.143s - 0.514s, respectively.

Next, we turn attention to time and space complexity of perturbation sampling. As shown in Table III, GM18 requires $O(k)$ multiplication operations over R. However, Construction 1 and 2 need $O(k^2)$ and $O(k)$ multiplications on $\mathcal{P}_n$ respectively, which are less efficient than those over R. The reason is that the center is computed as $\mathbf{c} = \Sigma_{t_1}\mathbf{p}_s$ in the second step of Algorithm 6 and Algorithm 8, and $\phi_n(\Sigma_{t_1}) \in \mathbb{R}^{2n\times kn}$ involves real entries. The fundamental reason is $\Sigma_G = \sigma^2\Sigma_1$, which corresponds to a non-spherical Gaussian distribution, rather than $\sigma^2\mathbf{I}$. Moreover, Construction 1 samples $\mathbf{p}_s$ according to Equation (12), and $\mathbf{p}_s$ follows a non-spherical Gaussian distribution. Consequently, it will cause:

1) Construction 1 needs to call $\mathrm{SampleF}$ for (kn) times rather than $\mathrm{SampleZ}$.
2) Unlike Construction 2, Construction 1 needs to perform k^2 extra multiplication operations to sample $\mathbf{p}_s$.

In order to clearly show the execution efficiency of three perturbation sampling algorithms, two experiments are designed to evaluate their execution time under different parameters:

TABLE III
THE TIME AND SPACE COMPLEXITY OF PERTURBATION SAMPLING ALGORITHMS.

	Multiplication over R	Multiplication over $\mathcal{P}_{2^i}$	Inverse over $\mathcal{P}_{2^i}$	Calls to SampleZ	Calls to SampleF	Storage over $\mathcal{P}_n$
GM18	$2k$	$3\sum_{i=1}^{\log n} M_{\mathcal{P}_{2^i}}$	$\sum_{i=1}^{\log n} Inv_{\mathcal{P}_{2^i}}$	$(k+2)n$	-	2
Construction 1	-	$3\sum_{i=1}^{\log n} M_{\mathcal{P}_{2^i}}$ $+(k^2+2k)\cdot M_{\mathcal{P}_n}$	$\sum_{i=1}^{\log n} Inv_{\mathcal{P}_{2^i}}$	$2n$	kn	k^2+2
Construction 2	-	$3\sum_{i=1}^{\log n} M_{\mathcal{P}_{2^i}}$ $+2k\cdot M_{\mathcal{P}_n}$	$\sum_{i=1}^{\log n} Inv_{\mathcal{P}_{2^i}}$	$(k+2n)$	-	2

1) k is fixed to 20, and n increases form 8 to 512. The execution time of three algorithms is shown as Fig. 5(a).
2) n is fixed to 64, and k increases from 5 to 30. The execution time of three algorithms is shown as Fig. 5(b).

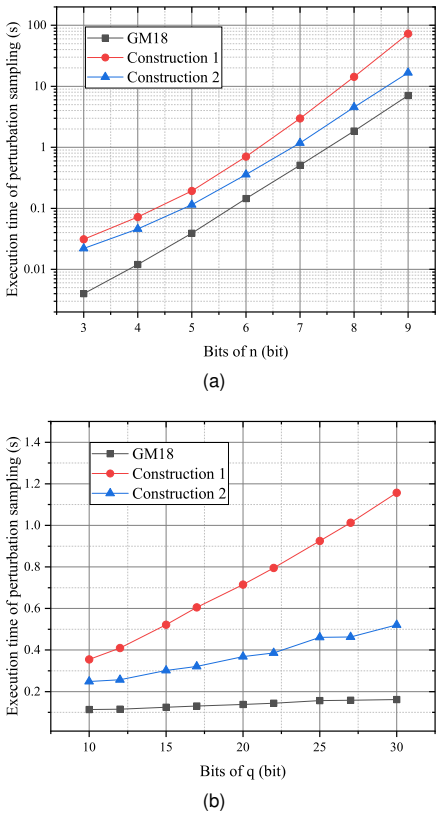

Fig. 5. Execution time of perturbation sampling

As shown in Fig. 5, when n increases from 8 to 512, the execution time of GM18's perturbation sampling algorithm is 0.004s - 7.049s. The execution time of Construction 1 and Construction 2 are 0.031s - 73.049s and 0.022s - 16.675s, respectively. When k increases from 10 to 30, the execution time of GM18's perturbation sampling algorithm is 0.023s - 0.066s. The execution time of Construction 1 and Construction 2 are 0.530s - 2.301s and 0.143s - 0.514s, respectively.

Since the above experiments are carried out by Mathematica platform, the accuracy of real numbers will affect the computational efficiency of the multiplications over $\mathcal{P}_{2^i}$ for some integers i. We take the second steps of Algorithm 6 and Algorithm 8 as examples. When $n = 128$ and $k = 20$, the accuracy of the entries of Σ_{t_1} should reach 10^{-6} in Algorithm 8, while that of Algorithm 6 should reach 10^{-18}. It makes the execution time for computing center $\mathbf{c}$ in Algorithm 6 higher than that of Algorithm 8. The reason of higher accuracy in Algorithm 6 is that Construction 1 needs to compute $(s^2\mathbf{I}_k - \sigma^2\Sigma_2)^{-1}$, and the accuracy of each entry in this matrix is high. It further results in the high accuracy of $\Sigma_{t_1} = -\sigma^2\mathbf{T}\Sigma_2(s^2\mathbf{I}_k - \sigma^2\Sigma_2)^{-1}$. To clearly show the executive efficiency of the second steps of Algorithm 6 and Algorithm 8, we set $k = 20$, and evaluate the execution time with n increasing from 8 to 512. The experimental results are shown in Fig. 6.

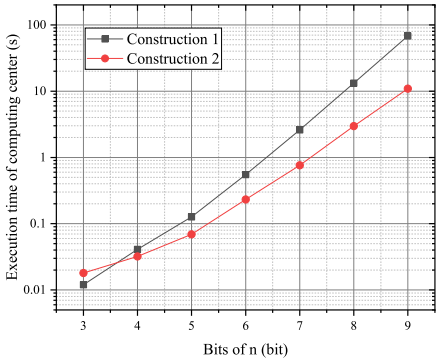

Fig. 6. Execution time of computing center

In summary, we provide two distinct perturbation sampling constructions tailored specifically for non-spherical G-lattice sampling. These constructions can avoid the Cholesky decomposition, and offer two candidates according to different requirements in LBC design. Our research also provides the instructive foundation for converting the non-spherical G-lattice sampling to the spherical one.

B. On-line stage

In this subsection, we design experiments to evaluate the execution performance of on-line stage in our proposed GSLT scheme over ring, and compare it with the ones of MP12 and GM18. Note that their on-line stage includes three following steps:

1) For a syndrome $u \in R_q$, compute $v = u - \mathbf{ap} \in R_q$.
2) Sample a vector $\mathbf{z} \in R_q^k$ by a specified G-lattice sampling algorithm over ring.
3) Compute $\mathbf{y} = \mathbf{p} + \begin{bmatrix}\mathbf{T}\\ \mathbf{I}\end{bmatrix}\mathbf{z} \in R_q^{k+2}$.

The details of experiments are provided as follows:

1) k is fixed to 20, and n increases from 8 to 512. The execution time of three algorithms is shown as Fig. 7(a).

2) n is fixed to 64, and k increases from 5 to 30. The execution time of three algorithms is shown as Fig. 7(b).

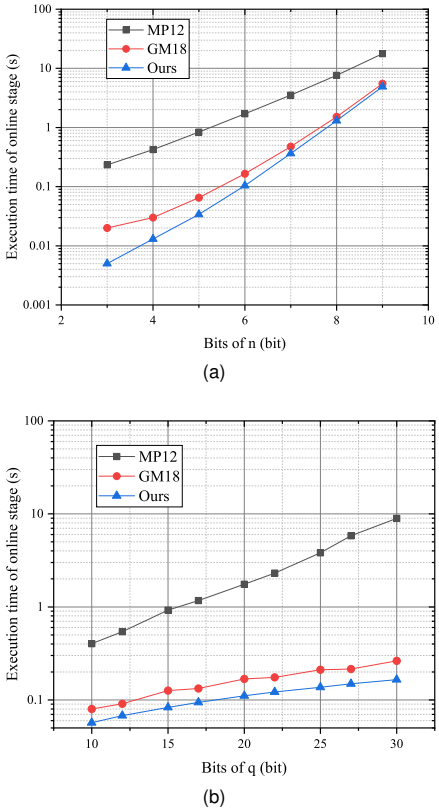

Fig. 7. Execution time of on-line stage

As shown in Fig. 7, as n increases from 8 to 512, the execution time of MP12's and GM18's on-line stage are from 0.234s to 17.711s and 0.02s to 5.498s, respectively. In comparison with these schemes, the execution time of on-line stage in our proposed GSLT scheme is only 0.005s to 4.893s. As k increases from 10 to 30, the execution time of MP12's and GM18's on-line stage are from 0.402s to 8.961s and 0.08s to 0.263s, respectively. The execution time of on-line stage in our proposed GSLT scheme is only 0.057s to 0.165s. Fig. 7(a) indicates the execution time of GM18 and our scheme become increasingly close when $n > 512$. The reason is that this experiment is used to evaluate the execution time of the entire on-line stage, including G-lattice sampling and linear expansion, and the latter further corresponds to multiple multiplication operations over R. As n gradually increases, the execution time of G-lattice sampling exhibits a slower growth rate compared to that of multiplication M_R. Actually, the time complexity of G-lattice sampling is linear (i.e., $O(n)$) in both GM18 and our scheme, while M_R exceeds linear time complexity (at least $O(n \cdot \log n)$). Consequently, the execution time of both GM18 and our scheme are relatively close during the online stage, but the execution time of GM18 cannot be less than that of ours.

We recall the comparison result in Table I. The non-spherical one requires fewer arithmetic operations and storage spaces for integer, and needs fewer sampling operations from discrete Gaussian distribution (i.e., SampleZ), in contrast with GM18's spherical G-lattice sampling. Meanwhile, both arithmetic operations and storage spaces of floating point numbers are reduced from $O(k)$ to $O(1)$. Therefore, this G-lattice sampling has low computational and storage complexity to make it more suitable for devices with limited computation and storage capabilities. As a result, our scheme has a better performance over ring at the on-line stage in comparison with the existing GSLT schemes.

VIII. Conclusion

In this paper, we present a Gaussian sampling scheme for trapdoor lattice under ring setting based on fast non-spherical G-lattice sampler and optimized perturbation generation. This scheme is designed under the MP12's framework, and consists of on-line and off-line stages. In the on-line stage, we apply the fast non-spherical G-lattice sampler to the ring setting to improve the efficiency. Then, we discover the entries of covariance matrix of the output vectors are all constants rather than polynomials. Therefore, it can be operated as a real matrix in calculation for some special cases. For the off-line stage, two constructions are designed on generating perturbation. The first construction aims to spherical Gaussian distribution, but it needs high computational accuracy and time & space complexity. In order to improve the performance, the second one is designed on non-spherical Gaussian distribution which will not leak any information of trapdoor in statistics. Meanwhile, the discussion of parameter choice for the second construction is also provided. The complexity analysis and experimental results show that our scheme has a better performance than the existing schemes in the on-line stage. For the off-line stage, two constructions can avoid low efficiency of Cholesky decomposition.

Acknowledge

This work was supported by the National Natural Science Foundation of China (61972032) and the Beijing Natural Science Foundation (M23017).

References

[1] P. Dutta, W. Susilo, D. H. Duong, and P. S. Roy, "Collusion-resistant identity-based proxy re-encryption: Lattice-based constructions in standard model," *Theor. Comput. Sci.*, vol. 871, pp. 16–29, 2021.

[2] P. Dutta, M. Jiang, D. H. Duong, W. Susilo, K. Fukushima, and S. Kiyomoto, "Hierarchical identity-based puncturable encryption from lattices with application to forward security," in *Asia Conference on Computer and Communications Security*. ACM, 2022, pp. 408–422.

[3] E. Chen, Y. Zhu, G. Zhu, K. Liang, and R. Feng, "How to implement secure cloud file sharing using optimized attribute-based access control with small policy matrix and minimized cumulative errors," *Computers & Security*, vol. 107, p. 102318, 2021.

[4] W. Susilo, P. Dutta, D. H. Duong, and P. S. Roy, "Lattice-based hra-secure attribute-based proxy re-encryption in standard model," in *26th European Symposium on Research in Computer Security*, vol. 12973. Springer, 2021, pp. 169–191.

[5] C. Gentry, A. Sahai, and B. Waters, "Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based," in *33rd Annual Cryptology Conference*, vol. 8042. Springer, 2013, pp. 75–92.

[6] D. Micciancio and C. Peikert, "Trapdoors for lattices: Simpler, tighter, faster, smaller," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2012, pp. 700–718.

[7] Y. Hu and H. Jia, "A new gaussian sampling for trapdoor lattices with arbitrary modulus," *Des. Codes Cryptogr.*, vol. 87, no. 11, pp. 2553–2570, 2019.
[8] C. Gentry, C. Peikert, and V. Vaikuntanathan, "Trapdoors for hard lattices and new cryptographic constructions," in *Proceedings of the 40th Annual ACM Symposium on Theory of Computing*. ACM, 2008, pp. 197–206.
[9] N. Genise and D. Micciancio, "Faster gaussian sampling for trapdoor lattices with arbitrary modulus," in *37th Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2018, pp. 174–203.
[10] L. Ducas and A. Durmus, "Ring-lwe in polynomial rings," in *15th International Conference on Practice and Theory in Public Key Cryptography*, vol. 7293. Springer, 2012, pp. 34–51.
[11] P. Bert, P. Fouque, A. Roux-Langlois, and M. Sabt, "Practical implementation of ring-sis/lwe based signature and IBE," in *9th International Conference on Post-Quantum Cryptography*, vol. 10786. Springer, 2018, pp. 271–291.
[12] K. D. Gür, Y. Polyakov, K. Rohloff, G. W. Ryan, H. Sajjadpour, and E. Savas, "Practical applications of improved gaussian sampling for trapdoor lattices," *IEEE Trans. Computers*, vol. 68, no. 4, pp. 570–584, 2019.
[13] S. Zhao, R. Jiang, and B. K. Bhargava, "RL-ABE: A revocable lattice attribute based encryption scheme based on R-LWE problem in cloud storage," *IEEE Trans. Serv. Comput.*, vol. 15, no. 2, pp. 1026–1035, 2022.
[14] M. Ajtai, "Generating hard instances of the short basis problem," in *26th International Colloquium on Automata, Languages and Programming*, vol. 1644. Springer, 1999, pp. 1–9.
[15] J. Alwen and C. Peikert, "Generating shorter bases for hard random lattices," *Theory Comput. Syst.*, vol. 48, no. 3, pp. 535–553, 2011.
[16] P. Bert, G. Eberhart, L. Prabel, A. Roux-Langlois, and M. Sabt, "Implementation of lattice trapdoors on modules and applications," in *12th International Workshop on Post-Quantum Cryptography*, vol. 12841. Springer, 2021, pp. 195–214.
[17] Y. Chen, N. Genise, and P. Mukherjee, "Approximate trapdoors for lattices and smaller hash-and-sign signatures," in *25th International Conference on the Theory and Application of Cryptology and Information Security*, vol. 11923. Springer, 2019, pp. 3–32.
[18] H. Jia, Y. Hu, and C. Tang, "Lattice-based hash-and-sign signatures using approximate trapdoor, revisited," *IET Inf. Secur.*, vol. 16, no. 1, pp. 41–50, 2022.
[19] D. Micciancio and O. Regev, "Worst-case to average-case reductions based on gaussian measures," *SIAM J. Comput.*, vol. 37, no. 1, pp. 267–302, 2007.
[20] C. Peikert and A. Rosen, "Efficient collision-resistant hashing from worst-case assumptions on cyclic lattices," in *Third Theory of Cryptography Conference*, vol. 3876. Springer, 2006, pp. 145–166.
[21] R. E. Bansarkhani and J. Buchmann, "Improvement and efficient implementation of a lattice-based signature scheme," in *20th International Conference on Selected Areas in Cryptography*, vol. 8282. Springer, 2013, pp. 48–67.
[22] C. Peikert, "An efficient and parallel gaussian sampler for lattices," in *Annual Cryptology Conference*, vol. 6223. Springer, 2010, pp. 80–97.
[23] Y. Yang, J. Sun, Z. Liu, and Y. Qiao, "Practical revocable and multi-authority CP-ABE scheme from RLWE for cloud computing," *J. Inf. Secur. Appl.*, vol. 65, p. 103108, 2022.
[24] A. Khalid, S. McCarthy, M. O'Neill, and W. Liu, "Lattice-based cryptography for iot in A quantum world: Are we ready?" in *IEEE 8th International Workshop on Advances in Sensors and Interfaces*. IEEE, 2019, pp. 194–199.
[25] A. Acar, H. Aksu, A. S. Uluagac, and M. Conti, "A survey on homomorphic encryption schemes: Theory and implementation," *ACM Computing Surveys*, vol. 51, no. 4, pp. 1–35, 2018.

Hai Lu received the master's degree from School of Computer Science, Shaanxi Normal University, Xi'an, Shaanxi, China in 2019. He is currently a PhD candidate with the department of School of Computer and Communication Engineering, University of Science and Technology Beijing, China. His research interests include access control and lattice-based cryptography. (Email: luhai@xs.ustb.edu.cn)

Yan Zhu received the M.S. and Ph.D. degrees in applied computer technology from Harbin Engineering University, Harbin, China, in 2002 and 2005, respectively. He is currently a Professor with the School of Computer and Communication Engineering, University of Science and Technology Beijing, China. From 2007 to 2013, he was an Associate Professor of computer science with the Institute of Computer Science and Technology, Peking University, Beijing. From 2008 to 2009, he was a Visiting Scholar with Arizona State University and with the University of Michigan-Dearborn, in 2012. His research interests include cryptography, secure computation, and network security. (Email:zhuyan@ustb.edu.cn)

Cecilia E Chen is currently a lecturer with the Department of the School of Computer and Communication Engineering, University of Science and Technology Beijing, China. She received B.S. and Ph.D. degrees from the Department of the School of Mathematics and Physics and the School of Computer and Communication Engineering, University of Science and Technology Beijing, China, in 2013 and 2021, respectively. Her research interests include blockchain, smart contracts, and lattice cryptography. (Email: chene@ustb.edu.cn)

Di Ma is currently a Professor in the Computer and Information Science (CIS) Department, College of Engineering and Computer Science (CECS), at the University of Michigan-Dearborn. She is also serving as the Associate Dean for Graduate Education and Research and the director of the Cybersecurity Center for Education, Research, and Outreach, CECS. She is broadly interested in the general area of security, privacy, and applied cryptography. Her research spans a wide range of topics, including connected and autonomous vehicle security, smartphone and mobile device security, RFID and sensor security, data privacy, and so on. Her research is supported by NSF, NHTSA, AFOSR, Intel, Ford, and Research in Motion. She received the PhD degree from the University of California, Irvine, in 2009. She was with IBM Almaden Research Center in 2008 and the Institute for Infocomm Research, Singapore in 2000-2005. She was the recipient of the Trevor O. Jones Outstanding Paper Award from Society of Automobile Engineers (SAE) in 2019, the Distinguished Research Award from the College of Engineering and Computer Science of UM-Dearborn in 2017, and the Tan Kah Kee Young Inventor Award in 2004. (Email: dmadma@umich.edu)