# IEICE TRANSACTIONS

# on Fundamentals of Electronics, Communications and Computer Sciences

VOL. E101-A NO. 11
NOVEMBER 2018





A PUBLICATION OF THE ENGINEERING SCIENCES SOCIETY

The Institute of Electronics, Information and Communication Engineers
Kikai-Shinko-Kaikan Bldg., 5-8, Shibakoen 3 chome, Minato-ku, TOKYO, 105-0011 JAPAN

PAPER

# A New Discrete Gaussian Sampler over Orthogonal Lattices

**Dianyan XIAO**[†a)], **Yang YU**[††b)], ***Nonmembers***, ***and*** **Jingguo BI**[†c)], ***Member***

**SUMMARY** Discrete Gaussian is a cornerstone of many lattice-based cryptographic constructions. Aiming at the orthogonal lattice of a vector, we propose a discrete Gaussian rejection sampling algorithm, by modifying the dynamic programming process for subset sum problems. Within $O(nq^2)$ time, our algorithm generates a distribution statistically indistinguishable from discrete Gaussian at width $s > \omega(\log n)$. Moreover, we apply our sampling algorithm to general high-dimensional dense lattices, and orthogonal lattices of matrices $\mathbf{A} \in \mathbb{Z}_q^{O(1)\times n}$. Compared with previous polynomial-time discrete Gaussian samplers, our algorithm does not rely on the short basis.

***key words:*** *discrete Gaussian, rejection sampling, dynamic programming, orthogonal lattice*

## 1. Introduction

Lattice-based cryptography is the most promising candidate for conventional cryptography, especially in the upcoming era of quantum computing. Recently, a great deal of lattice-based cryptographic schemes were proposed, including trapdoor design [1], [2], public key encryption [3], digital signature [1], [4], identity-based encryption [1], [4], [5] and functional encryption [6]. Efficiently sampling a lattice point following discrete Gaussian is often a crucial primitive element in these cryptographic constructions.

Many discrete Gaussian sampling algorithms for cryptographic purpose have been developed since the trapdoor of [1] was proposed. A classical sampler is the randomized variant of Babai's nearest-plane algorithm [7]. This kind of sampler was proposed in [8] and [1], where the output follows discrete Gaussian distribution for width $s > \|\widetilde{\mathbf{B}}\| \cdot \omega(\sqrt{\log n})$ with $\widetilde{\mathbf{B}}$ the Gram-Schmidt orthogonalization of sampling basis $\mathbf{B}$. Subsequently, a parallel discrete Gaussian sampling algorithm was introduced by Peikert [9]. Afterwards, Ducas and Nguyen [10] optimized the asymptotic runtime by floating-point arithmetic and Brakerski et al. [11] refined the randomized nearest-plane algorithm for smaller parameter $s > \|\widetilde{\mathbf{B}}\| O(\sqrt{\log n})$. In 2015, discrete Gaussian samplers for arbitrary width $s > 0$ [12], [13] were proposed and applied to solve SVP and CVP.

Orthogonal lattice, as a special $q$-ary lattice, is of great interest in cryptography, especially in LWE/SIS based cryptographic schemes [1], [14], [15]. In this paper, we focus on these orthogonal lattices of a single vector, which are common and primitive. Notice that 85% full rank integer lattices can be represented as an orthogonal lattice of a vector [16], [17], that is, given a full rank lattice $\mathcal{L} \in \mathbb{Z}^n$, there usually exists a vector $\mathbf{a} \in \mathbb{R}^n$ such that $\mathcal{L} = \{\mathbf{v} \in \mathbb{Z}^n \mid \langle \mathbf{a}, \mathbf{v} \rangle = 0 \bmod \det(\mathcal{L})\}$. The orthogonal lattice of a single vector can be viewed as the set of integer solutions to a modular subset sum problem.

Dynamic programming is a classical method to solve dense subset sum problem [18], [19], which requires $O(nq)$ time and space. We refined the dynamic programming process to develop a new rejection sampling algorithm for discrete Gaussian over orthogonal lattices. It is noted that our sampling algorithm only requires $O(nq^2)$ time and $O(nq)$ space, which is polynomial when the modulus $q = poly(n)$. Different from previous polynomial-time sampling algorithms [1], [8]–[11], our algorithm works for width $s > \omega(\log n)$ and is independent of the specific basis.

We introduce some preliminaries in Sect. 2. Then we describe the detailed algorithm in Sect. 3 and generalize it in Sect. 4. Comparison with other samplers is presented in Sect. 5. Finally, we conclude and discuss open problems in Sect. 6.

## 2. Preliminary

We denote by $\|\cdot\|$ the Euclidean norm, by $\|\cdot\|_\infty$ the infinity norm and by $\langle\cdot,\cdot\rangle$ the inner product of $\mathbb{R}^n$. For a matrix $\mathbf{B} = (\mathbf{b}_1, \cdots, \mathbf{b}_n) \in \mathbb{R}^{n\times n}$, we denote by $\|\mathbf{B}\| = \max_i \|\mathbf{b}_i\|$. Let $\mathcal{B}_n^\infty(r) = \{\mathbf{x} \in \mathbb{R}^n \mid \|\mathbf{x}\|_\infty < r\}$ and $\overline{\mathcal{B}_n^\infty}(r) = \{\mathbf{x} \in \mathbb{R}^n \mid \|\mathbf{x}\|_\infty \le r\}$. For convenience of illustration, we denote by $\log n$ the natural logarithm of $n$ and by $\mathbb{Z}_q$ the ring $\mathbb{Z}/q\mathbb{Z}$ for any positive integers $n$ and $q$.

The statistical distance between distributions $D_1$ and $D_2$ over a countable domain $S$ is defined as: $\Delta(D_1, D_2) := \frac{1}{2}\sum_{x\in S} |D_1(x) - D_2(x)|$. We say that distributions $D_1$ and $D_2$ are statistically indistinguishable if $\Delta(D_1, D_2) \le n^{-\omega(1)}$.

### 2.1 Lattices

A full-rank *lattice* $\mathcal{L} \subset \mathbb{R}^n$ is the set of all integer combinations of $n$ linearly independent vectors $\mathbf{b}_1, \cdots, \mathbf{b}_n \in \mathbb{R}^n$, namely, $\mathcal{L}(\mathbf{b}_1, \cdots, \mathbf{b}_n) = \left\{\sum_{i=1}^n x_i \mathbf{b}_i \mid x_i \in \mathbb{Z}\right\}$. Since only

Manuscript received January 11, 2018.
Manuscript revised June 12, 2018.

[†]The authors are with the Institute for Advanced Study, Tsinghua University, Beijing, 100084, China.
[††]The author is with the Department of Computer Science and Technology, Tsinghua University, Beijing, 100084, China.
a) E-mail: xiaody12@mails.tsinghua.edu.cn
b) E-mail: y-y13@mails.tsinghua.edu.cn (Corresponding author)
c) E-mail: jingguobi@mail.tsinghua.edu.cn

DOI: 10.1587/transfun.E101.A.1880

full-rank lattices are used in this article, we refer to full-rank lattice by the term lattice. The matrix $\mathbf{B} = (\mathbf{b}_1, \cdots, \mathbf{b}_n)$ is called a *basis* of $\mathcal{L}$. It is noted that for any unimodular matrix $\mathbf{U} \in \mathbb{Z}^{n\times n}$, $\mathbf{BU}$ is a basis of $\mathcal{L}(\mathbf{B})$. We denote by $\widetilde{\mathbf{B}} = (\widetilde{\mathbf{b}}_1, \cdots, \widetilde{\mathbf{b}}_n)$ the *Gram-Schmidt Orthogonalization* of $\mathbf{B}$ where $\widetilde{\mathbf{b}}_i = \mathbf{b}_i - \sum_{j=1}^{i-1} \mu_{i,j}\widetilde{\mathbf{b}}_j$ for $\mu_{i,j} = \langle \mathbf{b}_i, \widetilde{\mathbf{b}}_j\rangle / \langle \widetilde{\mathbf{b}}_j, \widetilde{\mathbf{b}}_j\rangle$. The *determinant* (or *volume*) of lattice $\mathcal{L}$ equals $\det(\mathcal{L}) = |\det(\mathbf{B})|$, which is an invariant of $\mathcal{L}$. Notice that $\mathcal{L} \subset \mathbb{Z}^n$ is a discrete additive subgroup of $\mathbb{Z}^n$, thus the *quotient group* $\mathbb{Z}^n/\mathcal{L} := \{\mathbf{x} + \mathcal{L} \mid \mathbf{x} \in \mathcal{L}\}$ is a well-defined additive group, and $|\mathbb{Z}^n/\mathcal{L}| = \det(\mathcal{L})$. The *first minimum* $\lambda_1(\mathcal{L})$ (resp. $\lambda_1^\infty(\mathcal{L})$) is the minimum of Euclidean (resp. infinity) norm of all non-zero vectors of $\mathcal{L}$. The *dual lattice* of $\mathcal{L}$ is $\mathcal{L}^* = \{\mathbf{u} \in \text{span}(\mathcal{L}) \mid \langle \mathbf{u}, \mathbf{v}\rangle \in \mathbb{Z} \text{ for any } \mathbf{v} \in \mathcal{L}\}$. It is known that $\det(\mathcal{L}) \cdot \det(\mathcal{L}^*) = 1$[†].

Given $\mathbf{a} \in \mathbb{Z}_q^n$, the *orthogonal lattice*[††] of $\mathbf{a}$ is

$$\mathbf{a}^\perp := \{\mathbf{v} \in \mathbb{Z}^n | \langle \mathbf{a}, \mathbf{v}\rangle = 0 \bmod q\}.$$

Without loss of generality, we assume that the greatest common divisor of $a_1, \cdots, a_n$ and $q$ is 1 for $\mathbf{a} = (a_1, \cdots, a_n)$. Considering the quotient group

$$\mathbb{Z}^n/\mathbf{a}^\perp = \left\{\mathbf{c}_t + \mathbf{a}^\perp \mid t \in \mathbb{Z}_q, \mathbf{c}_t \in \mathbb{Z}^n, \langle \mathbf{c}_t, \mathbf{a}\rangle = t \bmod q\right\},$$

then we know $\det(\mathbf{a}^\perp) = |\mathbb{Z}^n/\mathbf{a}^\perp| = q$. Moreover, let

$$\mathcal{L}_q(\mathbf{a}) := \{\mathbf{v} \in \mathbb{Z}^n | \exists z \in \mathbb{Z} \text{ s.t.} \mathbf{v} = z \cdot \mathbf{a} \bmod q\},$$

then $\mathcal{L}_q(\mathbf{a}) = q(\mathbf{a}^\perp)^*$.

### 2.2 Discrete Gaussians

For $s > 0$, we define the *Gaussian function* of $\mathbb{R}$: $\rho_s(x) = e^{-\frac{\pi x^2}{s^2}}$, and the *Gaussian function* of $\mathbb{R}^n$: $\rho_s(\mathbf{x}) = \prod_{j=1}^n \rho_s(x_j) = e^{-\frac{\pi\|\mathbf{x}\|^2}{s^2}}$ for $\mathbf{x} = (x_1, \cdots, x_n)$. When $s = 1$, we omit the subscript. Given a discrete set $S \subset \mathbb{R}^n$, we define $\rho_s(S) = \sum_{\mathbf{x}\in S} \rho_s(\mathbf{x})$. For lattice $\mathcal{L} \subset \mathbb{R}^n$ and $\mathbf{c} \in \mathbb{R}^n$, we have the *Poisson summation formula* for $\rho_s(\mathbf{x})$: $\rho_s(\mathcal{L}) = s^n \det(\mathcal{L}^*)\rho_{1/s}(\mathcal{L}^*)$, $\rho_s(\mathbf{c} + \mathcal{L}) = s^n \det(\mathcal{L}^*) \sum_{\mathbf{y}\in\mathcal{L}^*} e^{-2\pi i\langle \mathbf{c},\mathbf{y}\rangle}\rho_{1/s}(\mathbf{y})$. It is easy to verify that $\rho_s(\mathbf{c} + \mathcal{L}) \le \rho_s(\mathcal{L})$ since $|e^{-2\pi i\langle \mathbf{c},\mathbf{y}\rangle}| \le 1$ for any $\mathbf{y} \in \mathcal{L}^*$. A *discrete Gaussian distribution* over $\mathcal{L}$ centered at $\mathbf{c}$ with *width* $s$ is $D_{\mathcal{L},\mathbf{c},s}(\mathbf{x}) = \frac{\rho_s(\mathbf{x}-\mathbf{c})}{\rho_s(\mathcal{L}-\mathbf{c})}$. We usually write $D_{\mathcal{L}-\mathbf{c},s}$ as the distribution $D_{\mathcal{L},\mathbf{c},s} - \mathbf{c}$.

For any lattice $\mathcal{L} \subset \mathbb{R}^n$ and positive real $\epsilon > 0$, the *smoothing parameter* $\eta_\epsilon(\mathcal{L})$ is defined as the unique real $s > 0$ such that $\rho_{1/s}(\mathcal{L}^* \setminus \{\mathbf{0}\}) = \epsilon$. For $s > \eta_\epsilon(\mathcal{L})$, any translation of the lattice will not change the total Gaussian measure essentially.

**Lemma 2.1** ([21], implicit in Lemma 4.4): For any full-rank lattice $\mathcal{L} \subset \mathbb{R}^n$ and $\epsilon \in (0, 1)$, $s > \eta_\epsilon(\mathcal{L})$, we have that for any $\mathbf{c} \in \mathbb{R}^n$,

$$\frac{1-\epsilon}{1+\epsilon} \le \frac{\rho_s(\mathbf{c} + \mathcal{L})}{\rho_s(\mathcal{L})} \le 1.$$

It is shown in the proof of Lemma 4.4 in [21] that $\rho_s(\mathbf{c}+\mathcal{L}) \ge s^n \det(\mathcal{L}^*)(1-\epsilon)$ and $\rho_s(\mathcal{L}) \le s^n \det(\mathcal{L}^*)(1+\epsilon)$ for $s > \eta_\epsilon(\mathcal{L})$, hence it follows that $\rho_s(\mathbf{c} + \mathcal{L})/\rho_s(\mathcal{L}) \ge (1-\epsilon)/(1+\epsilon)$.

The following lemmata estimate $\eta_\epsilon(\mathcal{L})$.

**Lemma 2.2** ([1], Lemma 3.1): For any $n$-dimensional lattice $\mathcal{L} \subset \mathbb{R}^n$ and real $\epsilon > 0$, we have

$$\eta_\epsilon(\mathcal{L}) \le \widetilde{bl}(\mathcal{L})\sqrt{\log(2n(1 + 1/\epsilon))/\pi},$$

where $\widetilde{bl}(\mathcal{L}) = \min_{\mathbf{B}} \|\widetilde{\mathbf{B}}\|$ is the minimum over all bases.

**Lemma 2.3** ([22], Lemma 3.5): For any lattice $\mathcal{L} \subset \mathbb{R}^n$ and $\epsilon > 0$,

$$\eta_\epsilon(\mathcal{L}) \le \frac{\sqrt{\log(2n(1 + 1/\epsilon))/\pi}}{\lambda_1^\infty(\mathcal{L}^*)}.$$

**Lemma 2.4** ([23], Lemma 2.10): For any $n$-dimensional lattice $\mathcal{L}$, $\mathbf{c} \in \mathbb{R}^n$ and any $r > 0$,

$$\frac{\rho((\mathcal{L} - \mathbf{c}) \setminus \overline{\mathcal{B}_n^\infty}(r))}{\rho(\mathcal{L})} < 2ne^{-\pi r^2}.$$

As a corollary, we have the follow tail inequality of discrete Gaussian with respect to infinity norm.

**Lemma 2.5:** For any lattice $\mathcal{L} \subset \mathbb{R}^n$, $\mathbf{c} \in \mathbb{R}^n$, $\mu > 0$ and $s > \eta_\epsilon(\mathcal{L})$, we have

$$\Pr_{\mathbf{y}\sim D_{\mathcal{L}-\mathbf{c},s}}[\|\mathbf{y}\|_\infty \ge \mu s] \le 2ne^{-\pi\mu^2} \cdot \frac{1+\epsilon}{1-\epsilon}.$$

*Proof* We notice that for arbitrary $\mu > 0$,

$$\begin{aligned}
&\Pr_{\mathbf{y}\sim D_{\mathcal{L}-\mathbf{c},s}}[\|\mathbf{y}\|_\infty > \mu s] \\
&= \frac{\rho_s\left((\mathcal{L} - \mathbf{c}) \setminus \overline{\mathcal{B}_n^\infty}(\mu s)\right)}{\rho_s(\mathcal{L} - \mathbf{c})} \\
&= \frac{\rho_s\left((\mathcal{L} - \mathbf{c}) \setminus \overline{\mathcal{B}_n^\infty}(\mu s)\right)}{\rho_s(\mathcal{L})} \cdot \frac{\rho_s(\mathcal{L})}{\rho_s(\mathcal{L} - \mathbf{c})} \\
&= \frac{\rho\left((\mathcal{L}/s - \mathbf{c}/s) \setminus \overline{\mathcal{B}_n^\infty}(\mu)\right)}{\rho(\mathcal{L}/s)} \cdot \frac{\rho_s(\mathcal{L})}{\rho_s(\mathcal{L} - \mathbf{c})}.
\end{aligned}$$

From Lemma 2.4, it can be derived that,

$$\frac{\rho\left((\mathcal{L}/s - \mathbf{c}/s) \setminus \overline{\mathcal{B}_n^\infty}(\mu)\right)}{\rho(\mathcal{L}/s)} < 2ne^{-\pi\mu^2}.$$

Hence with Lemma 2.1, we obtain that for $s > \eta_\epsilon(\mathcal{L})$,

$$\Pr_{\mathbf{y}\sim D_{\mathcal{L}-\mathbf{c},s}}[\|\mathbf{y}\|_\infty > \mu s] < 2ne^{-\pi\mu^2} \cdot \frac{1+\epsilon}{1-\epsilon}.$$

Notice that for arbitrary $\varepsilon > 0$, we have

$$\Pr_{\mathbf{y}\sim D_{\mathcal{L}-\mathbf{c},s}}[\|\mathbf{y}\|_\infty > (\mu - \varepsilon)s] < 2ne^{-\pi(\mu-\varepsilon)^2} \cdot \frac{1+\epsilon}{1-\epsilon}.$$

---

[†] We refer to [20] for a bibliography on lattices.

[††] It is essentially the $q$-ary lattice for matrices in $\mathbb{Z}_q^{1\times n}$. We refer to [1] for more details.

As $\varepsilon$ approaches 0, we have

$$\Pr_{\mathbf{y}\sim D_{\mathcal{L}-\mathbf{c},s}}[\|\mathbf{y}\|_\infty \geq \mu s] \leq 2ne^{-\pi\mu^2}\cdot\frac{1+\epsilon}{1-\epsilon}.$$

□

We recall the rejection sampling of discrete Gaussian over integers proposed in [1].

**Sample$\mathbb{Z}$** Let $t(n) \geq \omega(\sqrt{\log n})$ be some fixed function. On input $(s, c)$ and (implicitly) the security parameter $n$, choose an integer $x$ from $\mathbb{Z}\cap[c-t(n)\cdot s, c+t(n)\cdot s]$ uniformly at random. Then with probability $\rho_s(x-c) \in (0, 1]$ output $x$, otherwise repeat.

With reference to Lemma 4.2 of [1], for any $t(n) = \omega(\sqrt{\log n})$, $\epsilon \in (0, e^{-\pi})$ and $s > \eta_\epsilon(\mathbb{Z})$, the output is statistically close to $D_{\mathbb{Z},c,s}$ with overwhelming probability. The running time of Sample$\mathbb{Z}$ is $t(n)\cdot\omega(\log n)$.

## 3. Discrete Gaussian Sampler by Dynamic Programming

Dynamic programming (DP) is a classical method to find binary solutions to dense subset sum problems. By generalizing and refining the DP technique, we propose a rejection sampling algorithm for discrete Gaussian.

Firstly, we introduce a global rejection sampling algorithm DGS-GR (Algorithm 1) in Sect. 3.1, which is precise but costly. Then we explicate the refined algorithm DGS-LR (Algorithm 2) in Sect. 3.2, in which the DGS-GR is embedded in the head block.

### 3.1 Global Rejection Sampling of Discrete Gaussian

We recall the DP method to solve subset sum problems. Given $\mathbf{a} = (a_1, \cdots, a_n) \in \mathbb{Z}_q^n$ and $t \in \mathbb{Z}_q$ for $q > 0$, subset sum problem asks to find $\mathbf{x} \in \{0, 1\}^n$ such that $\langle\mathbf{a}, \mathbf{x}\rangle = t \bmod q$. We define the boolean-valued function $f(k, z)$ for $1 \leq k \leq n$ and $z \in \mathbb{Z}_q$ as:

$$f(k,z) = \begin{cases} 1, \text{ if } \exists x_i \in \{0,1\} \text{ s.t. } \sum_{i=1}^k a_i x_i = z \bmod q; \\ 0, \text{ otherwise.} \end{cases}$$

We let $f(0, 0) = 1$.

First of all, we establish a boolean table to store the values of $f(k, z)$: for $k$ ranging from 0 to $n$ and arbitrary $z \in \mathbb{Z}_q$, if $f(k, z) = 1$, then we set $f(k+1, z) = 1$, $f(k+1, (z+a_{k+1}) \bmod q) = 1$. We note that establishing and storing the table cost $O(nq)$ time and space.

The DP algorithm works as follows. For $j$ ranging from $n$ to 1, it chooses an $\alpha \in \{0, 1\}$ uniformly at random, and checks the value of $f$: if $f(j-1, (t-\alpha a_j) \bmod q) = 1$, then it assigns $x_j = \alpha$; otherwise, it assigns $x_j = 1-\alpha$, and then sets $t = (t - x_j a_j) \bmod q$. Finally it returns a binary solution $\mathbf{x}$. It is worth noting that randomly choosing an $\alpha \in \{0, 1\}$ gives a relatively fair backtrack for situations where both $f(j-1, t)$ and $f(j-1, (t-a_j) \bmod q)$ are 1.

Notice that the backtrack only includes addition operation, thus the complexity of DP mainly relies on computing the table for $f(k, z)$, which costs $O(nq)$ time and storage.

Now we generalize the DP algorithm to $(-r, r)^n$ for $0 < r \leq \frac{q}{2}$ and refine the process of choosing $x_i$. Define the generalized function $f_r(k, z)$ for $1 \leq k \leq n$ and $z \in \mathbb{Z}_q$ as

$$f_r(k,z) = \begin{cases} 1, \text{if } \exists x_i \in (-r,r) \text{ s.t. } \sum_{i=1}^k a_i x_i = z \bmod q; \\ 0, \text{otherwise.} \end{cases}$$

Similarly, we define $f_r(0, 0) = 1$ for any $r > 0$. In this case, it costs $O(rnq)$ time and space to establish the table for $f_r(k, z)$.

We are to describe our global rejection sampling algorithm for discrete Gaussians. We set the sampling interval for each $v_j$ as $(-\mu s, \mu s)$, where $\mu$ is a parameter related to the sample quality. For $j$ ranging from $n$ to 1, we pick a $v_j \in (-\mu s, \mu s)$ uniformly at random. Then we check that if $f_r(j-1, (t-v_j a_j) \bmod q) = 1$: if it is true, then we accept $v_j$ with probability $\rho_s(v_j)$ and set $t = (t - v_j a_j) \bmod q$; otherwise, we restart the algorithm. The detailed procedure is shown in Algorithm 1.

**Algorithm 1** DGS-GR[$\mathbf{a}, q, s, t, \mu$]

**Input:** $\mathbf{a} \in \mathbb{Z}_q^n$, modulus $q$, target value $t \in \mathbb{Z}_q$, width $s$ and parameter $\mu$.
**Output:** a vector $\mathbf{v}$ satisfying $\langle\mathbf{a}, \mathbf{v}\rangle = t \bmod q$.

1: **Preprocess**: establish a table for $f_{\mu s}(j, z)$ where $1 \leq j \leq n$ and $z \in \mathbb{Z}_q$.
2: assign $\bar{t} = t$.
3: **for** $j = n$ to 1 **do**
4: choose an integer $\alpha$ in interval $(-\mu s, \mu s)$ and a probability $p$ in [0,1) uniformly at random.
5: **if** $f_{\mu s}(j-1, \bar{t} - \alpha a_j) = 1$ and $p \leq \rho_s(\alpha)$ **then**
6: assign $v_j = \alpha$
7: $\bar{t} = (\bar{t} - \alpha a_j) \bmod q$
8: **else**
9: go to Step 2.
10: **end if**
11: **end for**
12: **return** $\mathbf{v}$

**Remark 3.1:** For each $v_j$, accepting $v_j$ with probability $\rho_s(v_j)$ means that sampling a $p$ uniformly distributed in $[0, 1)$, we accept $v_j$ if $p \leq \rho_s(v_j)$ and otherwise reset the algorithm. That's why we need to check $p \leq \rho_s(\alpha)$ in Step 5 of DGS-GR.

The following theorem gives an explicit analysis about the correctness and complexity of DGS-GR.

**Theorem 3.2:** Given $\mathbf{a} \in \mathbb{Z}_q^n$, $t \in \mathbb{Z}_q$, $s > \eta_\epsilon(\mathbf{a}^\perp)$ for $\epsilon \in (0, \frac{1}{2})$ and $\mu = \omega(\sqrt{\log n})$, the output of DGS-GR$(\mathbf{a}, q, s, t, \mu)$ follows a distribution statistically indistinguishable from $D_{\mathbf{c}_t+\mathbf{a}^\perp,s}$, where $\mathbf{c}_t$ satisfies $\langle\mathbf{a}, \mathbf{c}_t\rangle = t \bmod q$. The time for preprocessing is $O(\mu snq)$ and the expected time for sampling is $O(q(2\mu)^n)$.

*Proof* We write the distribution $D_{\mathbf{c}_t+\mathbf{a}^\perp,s}$ as $D$ for short. Let $Y$ be the output of the algorithm DGS-GR and $\widetilde{D}$ be the distribution of $Y$, then $Y \in \mathbf{c}_t + \mathbf{a}^\perp$ and $\|Y\|_\infty < \mu s$. Without loss of generality, we assume that $\mu s$ is an integer, then there are at most $2\mu s - 1$ integers in the interval $(-\mu s, \mu s)$. When $Y = \mathbf{v} = (v_1, \cdots, v_n)$, since $v_j$ is uniformly sampled from $(-\mu s, \mu s)$ and accepted with probability $\rho_s(v_j)$ in each round of the loop, we have that $\mathbf{v}$ is output with probability $\prod_{j=1}^{n} \frac{\rho_s(v_j)}{2\mu s-1} = \frac{\rho_s(\mathbf{v})}{(2\mu s-1)^n}$. Therefore we deduce that

$$\widetilde{D}(\mathbf{v}) = \Pr[Y = \mathbf{v}] = \frac{\rho_s(\mathbf{v})}{\sum_{\substack{\mathbf{y}\in\mathbf{c}_t+\mathbf{a}^\perp \\ \|\mathbf{y}\|_\infty<\mu s}} \rho_s(\mathbf{y})}. \tag{1}$$

A straightforward computation leads to that for $\mathbf{y} \in \mathbf{c}_t + \mathbf{a}^\perp$,

$$\sum_{\|\mathbf{y}\|_\infty<\mu s} \left| \frac{\rho_s(\mathbf{y})}{\sum_{\|\mathbf{y}\|_\infty<\mu s} \rho_s(\mathbf{y})} - \frac{\rho_s(\mathbf{y})}{\rho_s(\mathbf{c}+\mathbf{a}^\perp)} \right| = \frac{\sum_{\|\mathbf{y}\|_\infty\geq\mu s} \rho_s(\mathbf{y})}{\rho_s(\mathbf{c}+\mathbf{a}^\perp)}.$$

Notice that

$$\frac{\sum_{\|\mathbf{y}\|_\infty\geq\mu s} \rho_s(\mathbf{y})}{\rho_s(\mathbf{c}+\mathbf{a}^\perp)} = \Pr_{\mathbf{y}\sim D}[\|\mathbf{y}\|_\infty \geq \mu s] = \sum_{\|\mathbf{y}\|_\infty\geq\mu s} D(\mathbf{y}).$$

According to Lemma 2.5, the statistical distance $\widetilde{\Delta}$ between $\widetilde{D}$ and $D$ is

$$\begin{aligned}\widetilde{\Delta} &= \frac{1}{2}\sum_{\|\mathbf{y}\|_\infty\geq\mu s} |D(\mathbf{y})| + \frac{1}{2}\sum_{\|\mathbf{y}\|_\infty<\mu s} |D(\mathbf{y}) - \widetilde{D}(\mathbf{y})| \\ &= \sum_{\|\mathbf{y}\|_\infty\geq\mu s} |D(\mathbf{y})| \leq 2ne^{-\pi\mu^2} \cdot \frac{1+\epsilon}{1-\epsilon}.\end{aligned}$$

Therefore, when $\mu = \omega(\sqrt{\log n})$ and $\epsilon < \frac{1}{2}$, the distributions $\widetilde{D}$ and $D_{\mathbf{c}_t+\mathbf{a}^\perp,s}$ are statistically indistinguishable. For simplicity, we denote $\delta = 2ne^{-\pi\mu^2} \cdot \frac{1+\epsilon}{1-\epsilon}$.

Now we analyze the running time. For the preprocessing, we need $O(\mu snq)$ operations to establish the table of $f_{\mu s}(j, z)$ with $1 \leq j \leq n$ and $z \in \mathbb{Z}_q$. Since the table is binary, we need $O(nq)$ bits of storage.

The algorithm DGS-GR terminates once it successfully outputs a vector. The probability that DGS-GR successfully outputs $\mathbf{v}$ is $\frac{\rho_s(\mathbf{v})}{(2\mu s-1)^n}$. Thus the expected number of iterations before DGS-GR ends is

$$\frac{(2\mu s-1)^n}{\rho_s\left((\mathbf{c}_t+\mathbf{a}^\perp)\cap\mathcal{B}_n^\infty(\mu s)\right)} \leq \frac{(2\mu s-1)^n}{(1-\delta)\rho_s\left(\mathbf{c}_t+\mathbf{a}^\perp\right)}.$$

By Lemma 2.1, we obtain that for $s \geq \eta_\epsilon(\mathbf{a}^\perp)$,

$$\rho_s(\mathbf{c}_t+\mathbf{a}^\perp) \geq \frac{1-\epsilon}{1+\epsilon} \cdot \rho_s(\mathbf{a}^\perp).$$

The Poisson summation formula leads to

$$\rho_s(\mathbf{a}^\perp) = \frac{1}{\det(\mathbf{a}^\perp)} \cdot s^n \cdot \rho_{1/s}\left((\mathbf{a}^\perp)^*\right) \geq \frac{s^n}{q}.$$

Observing that $(2\mu s-1)^n < (2\mu s)^n$, we derive that

$$\frac{(2\mu s-1)^n}{(1-\delta)\rho_s(\mathbf{c}_t+\mathbf{a}^\perp)} \leq \frac{1+\epsilon}{(1-\delta)(1-\epsilon)} \cdot q(2\mu)^n.$$

Notice that for small $\epsilon, \delta > 0$, $\frac{1+\epsilon}{(1-\delta)(1-\epsilon)}$ can be bounded by a constant. Consequently, the expected running time is $O(q(2\mu)^n)$. □

As shown in Theorem 3.2, the time consumption of DGS-GR algorithm is quite expensive. Actually, it can be improved. We will modify the algorithm in Sect. 3.2 to achieve a better performance.

### 3.2 Local Rejection Sampling of Discrete Gaussian

Notice that Algorithm 1 is super-exponential because it would start a new iteration once the conditions in Step 5 can not be satisfied. A possible optimization would be for each $1 \leq j \leq n$, keeping sampling $(\alpha, p)$ until the conditions of Step 5 are reached. However, it may increase the gap between the output distribution and $D_{\mathbf{c}_t+\mathbf{a}^\perp,s}$. Indeed, we are able to apply a local optimization to improve the efficiency without affecting the distribution of outputs.

It is observed that, when $j$ is large, $f_{\mu s}(j, z) = 1$ for any $z \in \mathbb{Z}_q$, which means that the sampling process of $v_j$ seems independent of the value of $v_{j+1}, \cdots, v_n$. For these $j$'s, we can independently and repeatedly sample eligible $(\alpha, p)$.

We now elaborate our refined sampling algorithm DGS-LR (Algorithm 2). Let $n_0 = \lceil \frac{\log(nq)}{\log(2\mu)} \rceil$ and $\mathbf{a} = (\mathbf{a}_1, \mathbf{a}_2)$ where $\mathbf{a}_1 \in \mathbb{Z}_q^{n_0}$. It is worth noting that $n_0 \leq n$ is a necessary condition to ensure DGS-LR works, which is satisfied when

$$nq \leq (2\mu)^n. \tag{2}$$

We want to have $\lambda_1^\infty((\mathbf{a}_1^\perp)^*) \geq 1/(4\mu)$, that is $\lambda_1^\infty(\mathcal{L}_q(\mathbf{a}_1)) \geq q/(4\mu)$. If this condition could not be satisfied, then we cyclically left shift $\mathbf{a}$ by $n_0$ indices. Indeed, within $\log n$ such cyclic left shifts of $\mathbf{a}$, we can obtain that $\lambda_1^\infty(\mathcal{L}_q(\mathbf{a}_1)) \geq q/(4\mu)$ with high probability. Then we establish the table for $f_{\mu s}(j, z)$. For $j > n_0$, we run the algorithm Sample$\mathbb{Z}$ in [1] to obtain $v_j$ respectively and independently. For $j \leq n_0$, we invoke the small-scaled DGS-GR (Algorithm 1) with parameters $\left(\mathbf{a}_1, q, s, (t - \sum_{j>n_0} a_j v_j) \bmod q, \mu\right)$ to obtain $(v_1, \cdots, v_{n_0})$. Finally, we get the sample $\mathbf{v}$.

---

**Algorithm 2** DGS-LR$(\mathbf{a}, q, s, t, \mu)$

**Input:** $\mathbf{a} \in \mathbb{Z}_q^n$, modulus $q$, target value $t \in \mathbb{Z}_q$, width $s$ and parameter $\mu$.
**Output:** a vector $\mathbf{v}$ satisfying $\langle \mathbf{a}, \mathbf{v}\rangle = t \bmod q$.

1: **Preprocess I**: Let $n_0 = \lceil \frac{\log(nq)}{\log(2\mu)} \rceil$ and $\mathbf{a} = (\mathbf{a}_1, \mathbf{a}_2)$ where $\mathbf{a}_1 \in \mathbb{Z}_q^{n_0}$. Check if $\lambda_1^\infty(\mathcal{L}_q(\mathbf{a}_1)) \geq \frac{q}{4\mu}$. If so, set $L = 0$; otherwise, cyclic left shift $\mathbf{a}$ by $n_0$ indices and increase $L$ by 1 repeatly until $\lambda_1^\infty(\mathcal{L}_q(\mathbf{a}_1)) \geq \frac{q}{4\mu}$. If $L$ exceeds $\log n$, output $\perp$ and terminate.
2: **Preprocess II**: Establish a table of $f_{\mu s}(j, z)$ with $1 \leq j \leq n$ and $z \in \mathbb{Z}_q$
3: **for** $j = n$ to $n_0 + 1$ **do**
4: $\quad v_j$ = Sample$\mathbb{Z}(s, 0)$.
5: **end for**
6: Let $t' = (t - \sum_{i>n_0} a_i v_i) \bmod q$.
7: Run DGS-GR$(\mathbf{a}_1, q, s, t', \mu)$ to obtain $\mathbf{v}_1$.
8: Let $\mathbf{v}' = (\mathbf{v}_1, v_{n_0+1}, \cdots, v_n)$ and cyclic right shift $\mathbf{v}'$ by $Ln_0$ indices, then obtain $\mathbf{v}$.
9: **return** $\mathbf{v}$.

Before proving the correctness of Algorithm 2, we need some lemmata.

**Lemma 3.3:** For $\mathbf{a}$ uniformly distributed in $\mathbb{Z}_q^n$, we have that

$$\Pr[\lambda_1^\infty(\mathcal{L}_q(\mathbf{a})) \geq q/(2r)] \geq 1 - q\left(\frac{1}{r} + \frac{1}{q}\right)^n.$$

*Proof* For a vector $\mathbf{v} \in \mathbb{Z}^n$, we have that $\mathbf{v} \in \mathcal{L}_q(\mathbf{a})$ if and only if $\mathbf{v} = z\mathbf{a} \bmod q$ for some $z \in \mathbb{Z}_q$. Notice that for each $\mathbf{v}$, there are at most $q$ $\mathbf{a} \in \mathbb{Z}_q^n$ such that $\mathbf{v} \in \mathcal{L}_q(\mathbf{a})$ and the number of $\mathbf{v}$'s with $\|\mathbf{v}\|_\infty < \frac{q}{2r}$ is at most $\left(\frac{q}{r}+1\right)^n$. Hence we have that

$$\begin{aligned}\Pr_{\mathbf{a}\sim U(\mathbb{Z}_q^n)}[\lambda_1^\infty(\mathcal{L}_q(\mathbf{a})) < q/(2r)] &\leq \left(\frac{q}{r}+1\right)^n \cdot q \cdot \frac{1}{q^n} \\ &= q\left(\frac{1}{r} + \frac{1}{q}\right)^n.\end{aligned}$$

□

Due to Lemma 2.3 and the fact that $\mathcal{L}_q(\mathbf{a}) = q(\mathbf{a}^\perp)^*$, we have the following bound for $\eta_\epsilon(\mathbf{a}^\perp)$.

**Corollary 3.4:** For $\mathbf{a}$ uniformly distributed in $\mathbb{Z}_q^n$, it follows that

$$\eta_\epsilon(\mathbf{a}^\perp) \leq 2r\sqrt{\log(2n(1+1/\epsilon))/\pi}$$

with probability at least $1 - q\left(\frac{1}{r}+\frac{1}{q}\right)^n$.

We claim that the output of Algorithm 2 follows a distribution indistinguishable from $D_{\mathbf{c}_t+\mathbf{a}^\perp}$.

**Theorem 3.5:** For $\mathbf{a}$ uniformly distributed in $\mathbb{Z}_q^n$, $t \in \mathbb{Z}_q$, $\mu = \omega(\sqrt{\log n})$ and $s > 4\mu\sqrt{\log(2n(1+1/\epsilon))/\pi}$ with $\epsilon = n^{-\omega(1)}$, the output of DGS-LR$(\mathbf{a}, q, s, t, \mu)$ follows a distribution statistically indistinguishable from $D_{\mathbf{c}_t+\mathbf{a}^\perp,s}$, where $\mathbf{c}_t \in \mathbb{Z}^n$ satisfying $\langle \mathbf{a}, \mathbf{c}_t\rangle = t \bmod q$. The expected running time is $O(nq^2)$ and space complexity is $O(nq)$ if $q = poly(n)$.

*Proof* Given uniformly distributed $\mathbf{a} \in \mathbb{Z}_q^n$, we check that if $\lambda_1^\infty((\mathbf{a}_1^\perp)^*) \geq q/(4\mu)$. If not, we cyclic shift $\mathbf{a}$ by $n_0$ indices until $\lambda_1^\infty((\mathbf{a}_1^\perp)^*) \geq q/(4\mu)$. It is indeed easy to figure out $\lambda_1^\infty((\mathbf{a}_1^\perp)^*)$. Note that $(\mathbf{a}_1^\perp)^* = \frac{1}{q}\mathcal{L}_q(\mathbf{a}_1)$, by checking all $x \in \mathbb{Z}_q$ and reducing $x\mathbf{a}$ into $[-q/2, q/2]^n$, it suffices to obtain $\lambda_1^\infty$. Thus the time in Preprocess I is $O(nq\log n)$. Furthermore, notice that $\mathbf{a}$ is left shifted $n_0$ indices each time, thus $\mathbf{a}_1$ is uniformly random over $\mathbb{Z}_q^{n_0}$. According to Lemma 3.3, we have that for $n_0 = \lceil\frac{\log(nq)}{\log(2\mu)}\rceil \geq \frac{\log(nq)}{\log(2\mu)}$, there exists a constant $c > 0$, such that

$$\Pr\left[\lambda_1^\infty(\mathcal{L}_q(\mathbf{a}_1)) \geq q/(4\mu)\right] \geq 1 - q\left(\frac{1}{2\mu}+\frac{1}{q}\right)^{n_0} \geq 1 - \frac{c}{n}.$$

Thus the algorithm terminates in Preprocess I with probability at most $(1/n)^{\log n}$ which is negligible. Without loss of generality, we assume no cyclic shift occurs in later discussion, because the Gaussian measure keeps unchanged for cyclic shifted vectors.

Notice that for $\mathbf{a}_1^\perp$, $\lambda_1^\infty\left((\mathbf{a}_1^\perp)^*\right) = \frac{1}{q}\lambda_1^\infty(\mathcal{L}_q(\mathbf{a}_1)) \geq \frac{1}{4\mu}$, and for $\mathbb{Z}^{n-n_0}$, $\lambda_1^\infty((\mathbb{Z}^{n-n_0})^*) = \lambda_1^\infty(\mathbb{Z}^{n-n_0}) = 1$. Hence according to Lemma 2.3, we have that for $s > 4\mu\sqrt{\log(2n(1+1/\epsilon))/\pi}$,

$$\begin{aligned}\eta_\epsilon(\mathbf{a}_1^\perp) &\leq \frac{\sqrt{\log(2n_0(1+1/\epsilon))/\pi}}{\lambda_1^\infty\left((\mathbf{a}_1^\perp)^*\right)} < s,\\ \eta_\epsilon(\mathbb{Z}^{n-n_0}) &\leq \sqrt{\log(2(n-n_0)(1+1/\epsilon)/\pi)} < s.\end{aligned}$$

For $j > n_0$, we have $f_{\mu s}(j, z) = 1$ for any $z \in \mathbb{Z}_q$. Otherwise, there must exist $z_0 \in \mathbb{Z}_q$ such that $f(n_0, z_0) = 0$, which is $(\mathbf{c}_{z_0} + \mathbf{a}_1^\perp) \cap \mathcal{B}_{n_0}^\infty(\mu s) = \emptyset$. By Lemma 2.4, it leads to

$$\frac{\rho_s(\mathbf{c}_{z_0} + \mathbf{a}_1^\perp)}{\rho_s(\mathbf{a}_1^\perp)} = \frac{\rho_s((\mathbf{c}_{z_0} + \mathbf{a}_1^\perp) \setminus \mathcal{B}_{n_0}^\infty(\mu s))}{\rho_s(\mathbf{a}_1^\perp)} \leq 2n_0 e^{-\pi\mu^2}.$$

Observe that $2n_0e^{-\pi\mu^2} < \frac{1-\epsilon}{1+\epsilon}$ for $\mu = \omega(\sqrt{\log n})$ and $\epsilon \in (0,1)$, which conflicts with Lemma 2.1. Therefore, for arbitrary $\mathbf{y}_2 \in \mathbb{Z}^{n-n_0}$, we can always find $\mathbf{y}_1 \in \mathbb{Z}^{n_0}$ such that $(\mathbf{y}_1, \mathbf{y}_2) \in \mathbf{a}^\perp$.

We write the distribution $D_{\mathbf{c}_t+\mathbf{a}^\perp,s}$ as $D$ for short. We denote by $Y$ the output of algorithm DGS-LR$(\mathbf{a}, q, s, t, \mu)$ and $\hat{D}$ the distribution of $Y$. Let $Y = (Y_1, Y_2)$ where $Y_1$ is a random variable corresponding to the first $n_0$ entries of $Y$, then

$$\Pr[Y = \mathbf{v}] = \Pr[Y_1 = \mathbf{v}_1 | Y_2 = \mathbf{v}_2] \cdot \Pr[Y_2 = \mathbf{v}_2].$$

Since the last $n - n_0$ entries are sampled by Sample$\mathbb{Z}$ independently, the probability that $Y_2 = \mathbf{v}_2$ is that

$$\Pr[Y_2 = \mathbf{v}_2] = \frac{\rho_s(\mathbf{v}_2)}{\rho_s\left(\mathbb{Z}^{n-n_0} \cap \mathcal{B}_{n-n_0}^\infty(\mu s)\right)}.$$

Let $t(\mathbf{y}_2) = (t - \langle \mathbf{a}_2, \mathbf{y}_2\rangle) \bmod q$ for $\mathbf{y}_2 \in \mathbb{Z}^{n-n_0}$. It is noted that $\mathbf{v}_1$ is sampled by DGS-GR with target value $t(\mathbf{v}_2)$. Thus by Eq. (1) we have

$$\Pr[Y_1 = \mathbf{v}_1 \mid Y_2 = \mathbf{v}_2] = \frac{\rho_s(\mathbf{v}_1)}{\rho_s\left(\left(\mathbf{c}_{t(\mathbf{v}_2)} + \mathbf{a}_1^\perp\right) \cap \mathcal{B}_{n_0}^\infty(\mu s)\right)},$$

where $\mathbf{c}_{t(\mathbf{v}_2)} \in \mathbb{Z}^{n_0}$ is an arbitrary vector such that $\langle \mathbf{c}_{t(\mathbf{v}_2)}, \mathbf{a}_1\rangle = t(\mathbf{v}_2) \bmod q$. Thus, we get that

$$\hat{D}(\mathbf{v}) = \frac{\rho_s(\mathbf{v})}{\rho_s\left(\mathbb{Z}^{n-n_0} \cap \mathcal{B}_{n-n_0}^\infty(\mu s)\right)\rho_s\left(\left(\mathbf{c}_{t(\mathbf{v}_2)} + \mathbf{a}_1^\perp\right) \cap \mathcal{B}_{n_0}^\infty(\mu s)\right)}.$$

Let $\delta = 2ne^{-\pi\mu^2} \cdot \frac{1+\epsilon}{1-\epsilon}$, then $\delta \geq 2n'e^{-\pi\mu^2} \cdot \frac{1+\epsilon}{1-\epsilon}$ for any $n' \leq n$, including $n' = n_0$ and $n' = n - n_0$. On the basis of Lemma 2.5, we have that

$$\begin{aligned}\frac{\rho_s\left(\mathbb{Z}^{n-n_0} \cap \mathcal{B}_{n-n_0}^\infty(\mu s)\right)}{\rho_s(\mathbb{Z}^{n-n_0})} &\in (1-\delta, 1]\\ \frac{\rho_s\left(\left(\mathbf{c}_{t(\mathbf{v}_2)} + \mathbf{a}_1^\perp\right) \cap \mathcal{B}_{n_0}^\infty(\mu s)\right)}{\rho_s(\mathbf{c}_{t(\mathbf{v}_2)} + \mathbf{a}_1^\perp)} &\in (1-\delta, 1].\end{aligned} \quad (3)$$

Combining the fact that

$$\rho_s(\mathbf{c}_t + \mathbf{a}^\perp) = \sum_{\mathbf{y}_2 \in \mathbb{Z}^{n-n_0}} \rho_s(\mathbf{y}_2)\rho_s(\mathbf{c}_{t(\mathbf{y}_2)} + \mathbf{a}_1^\perp)$$

and

$$\frac{\rho_s(\mathbf{c}_{t(\mathbf{y}_2)} + \mathbf{a}_1^\perp)}{\rho_s(\mathbf{c}_{t(\mathbf{v}_2)} + \mathbf{a}_1^\perp)} \in \left[\frac{1-\epsilon}{1+\epsilon}, \frac{1+\epsilon}{1-\epsilon}\right],$$

for $\mathbf{y}_2 \in \mathbb{Z}^{n-n_0}$ from Lemma 2.1, we have that

$$\frac{\rho_s(\mathbf{c}_t + \mathbf{a}^\perp)}{\rho_s(\mathbb{Z}^{n-n_0})\rho_s(\mathbf{c}_{t(\mathbf{v}_2)} + \mathbf{a}_1^\perp)} \in \left[\frac{1-\epsilon}{1+\epsilon}, \frac{1+\epsilon}{1-\epsilon}\right].$$

Together with Eq. (3), for $\mathbf{y} \in (\mathbf{c}_t + \mathbf{a}^\perp) \cap \mathcal{B}_n^\infty(\mu s)$, it follows that

$$\frac{1-\epsilon}{1+\epsilon} \le \frac{\hat{D}(\mathbf{y})}{D(\mathbf{y})} \le \frac{1+\epsilon}{1-\epsilon} \cdot \frac{1}{(1-\delta)^2},$$

Besides, we know that $\sum_{\|\mathbf{y}\|_\infty > \mu s} D(\mathbf{y}) \le \delta$, which implies that the statistical distance $\hat{\Delta}$ between $\hat{D}$ and $D_{\mathbf{c}_t+\mathbf{a}^\perp}$ is

$$\begin{aligned}\hat{\Delta} &= \frac{1}{2}\sum_{\|\mathbf{y}\|_\infty > \mu s} |D(\mathbf{y})| + \frac{1}{2}\sum_{\|\mathbf{y}\|_\infty \le \mu s} |\hat{D}(\mathbf{y}) - D(\mathbf{y})| \\ &\le \frac{1}{2}\delta + \frac{1}{2}\left(\frac{1+\epsilon}{1-\epsilon} \cdot \frac{1}{(1-\delta)^2} - 1\right) \\ &\le 2\delta + 2\epsilon + 6\delta\epsilon\end{aligned}$$

since $\frac{1+\epsilon}{1-\epsilon} \le 1 + 4\epsilon$ and $\frac{1}{(1-\delta)^2} \le 1 + 3\delta$ when $\epsilon = n^{-\omega(1)}$ and $\mu = \omega(\sqrt{\log n})$. Thus the distribution $\widetilde{D}$ is statistically indistinguishable from $D$ when $\epsilon = n^{-\omega(1)}$ and $\mu = \omega(\sqrt{\log n})$.

Next we evaluate the running time of DGS-LR. As clarified in Theorem 3.2, the complexity for Preprocess II is $O(\mu snq)$ and thus that for the whole preprocessing is $O(\mu snq) + O(nq \log q) = O(\mu snq)$. The loop of Step 3 – 5 is $n - n_0$ rounds of Sample$\mathbb{Z}$. It is noted that the complexity for Sample$\mathbb{Z}$ is $\mu \cdot \omega(\log n)$, which can be bounded by $\mu \log^2 n$. Thus the cost of Step 3 – 5 is at most $\Theta(n\mu \log^2 n)$. Step 7 mainly calls DGS-GR without preprocessing, which costs $O(q(2\mu)^{n_0}) = O(nq^2)$. □

## 4. Applications to General Lattices

In this section, we will generalize the sampling algorithm DGS-LR (Algorithm 2) to some other lattices. We claim that DGS-LR is efficient for most high-dimensional dense lattices and $q$-ary lattices $\{\mathbf{v} \in \mathbb{Z}^n \mid \mathbf{A}\mathbf{v} = \mathbf{0} \bmod q\}$ for $\mathbf{A} \in \mathbb{Z}_q^{O(1)\times n}$ and $q = poly(n)$.

### 4.1 Application to High-Dimensional Dense Lattices

For full rank $\mathcal{L} \subset \mathbb{Z}^n$, according to Proposition 1 in [16], we know that there exists an $\mathbf{a} \in \mathbb{Z}^n_{\det(\mathcal{L})}$ such that

$$\mathcal{L} = \{\mathbf{v} \in \mathbb{Z}^n \mid \langle \mathbf{a}, \mathbf{v}\rangle = 0 \bmod \det(\mathcal{L})\}$$

if and only if the quotient group $\mathbb{Z}^n/\mathcal{L}$ is cyclic. The work in [17] proved that the natural density of such $\mathcal{L}$ over all full rank lattices of $\mathbb{Z}^n$ is approximately 0.85, which means that 85% full rank integer lattices are equivalent to an orthogonal lattice of a vector. Notice that such vector $\mathbf{a} \in \mathbb{Z}^n_{\det(\mathcal{L})}$ for $\mathcal{L}$ can be calculated in polynomial time (Proposition 2, [16]).

In line with Theorem 3.5, when the lattice $\mathcal{L}$ is dense, especially $\det(\mathcal{L}) = poly(n)$, our sampling algorithm DGS-LR can generate a discrete Gaussian distribution over $\mathcal{L}$ within polynomial time and space. However, when $\det(\mathcal{L})$ is large, such as the exponential of $n$, the sampler DGS-LR does not work as indicated by Eq. (2).

### 4.2 Discussion on General $q$-Ary Lattices

We also extend DGS-LR to general $q$-ary lattices. Given a matrix $\mathbf{A} \in \mathbb{Z}_q^{k\times n}$, we define its orthogonal lattice

$$\mathbf{A}^\perp = \{\mathbf{v} \in \mathbb{Z}^n \mid \mathbf{A}\mathbf{v} = \mathbf{0} \bmod q\}.$$

By similar analysis in Sect. 2.1, we have that $\det(\mathbf{A}^\perp) \le q^k$ with overwhelming probability and $(\mathbf{A}^\perp)^* = \frac{1}{q}\mathcal{L}_q(\mathbf{A})$ where

$$\mathcal{L}_q(\mathbf{A}) := \left\{\mathbf{v} \in \mathbb{Z}^n \mid \exists \mathbf{z} \in \mathbb{Z}^k \text{ s.t. } \mathbf{v} = \mathbf{z} \cdot \mathbf{A} \bmod q\right\}.$$

The first minimum $\lambda_1^\infty(\mathcal{L}_q(\mathbf{A}))$ also has a lower bound with a high probability when $\mathbf{A}$ is uniformly distributed in $\mathbb{Z}_q^{k\times n}$.

**Lemma 4.1:** Given $\mathbf{A}$ uniformly distributed in $\mathbb{Z}_q^{k\times n}$, it follows that

$$\Pr[\lambda_1^\infty(\mathcal{L}_q(\mathbf{A})) \ge q/(2r)] \ge 1 - q^k\left(\frac{1}{r} + \frac{1}{q}\right)^n.$$

*Proof* Given arbitrary $\mathbf{v} \in \mathbb{R}^n$, if $\mathbf{v} \in \mathcal{L}_q(\mathbf{A})$ for some $\mathbf{A}$, then $\mathbf{v} = \sum_{i=1}^k x_i\mathbf{a}_i \bmod q$ where $\mathbf{a}_1, \cdots, \mathbf{a}_k$ are the row vectors of $\mathbf{A}$ and $x_i \in \mathbb{Z}_q$. We observe that $x_k\mathbf{a}_k = \mathbf{v} - \sum_{i=1}^{k-1} x_i\mathbf{a}_i \bmod q$. Let $(x_i, \mathbf{a}_i)$ runs over $\mathbb{Z}_q \times \mathbb{Z}_q^n$ for $i = 1, \cdots, k-1$. Then we have the number of $\mathbf{A} \in \mathbb{Z}_q^{k\times n}$'s such that $\mathbf{v} \in \mathcal{L}_q(\mathbf{A})$ is at most $q^{nk-n+k}$.

Also, there are at most $\left(\frac{q}{r}+1\right)^n$ points in $(-\frac{q}{2r}, \frac{q}{2r})^n$, thus

$$\Pr\left[\lambda_1^\infty(\mathcal{L}_q(\mathbf{A})) < \frac{q}{2r}\right] \le \frac{(\frac{q}{r}+1)^n q^{nk-n+k}}{q^{nk}} \le q^k\left(\frac{1}{r}+\frac{1}{q}\right)^n.$$

□

We write $\mathbf{A} = (\hat{\mathbf{a}}_1, \cdots, \hat{\mathbf{a}}_n)$ where $\hat{\mathbf{a}}_i \in \mathbb{Z}_q^k$ for $i = 1, \cdots, n$. Comparably, we define the discriminant function $f_r(j, \hat{\mathbf{z}})$ for $0 < r < \frac{q}{2}$, $1 \le j \le n$ and $\hat{\mathbf{z}} \in \mathbb{Z}_q^k$:

$$f_r(j, \hat{\mathbf{z}}) = \begin{cases} 1, \text{if } \exists x_i \in (-r, r) \text{ s.t. } \sum_{i=1}^j \hat{\mathbf{a}}_i x_i = \hat{\mathbf{z}} \bmod q, \\ 0, \text{otherwise}\end{cases}$$

with $f_r(0, \hat{\mathbf{0}}) = 1$.

Given input $(\mathbf{A}, q, s, \hat{\mathbf{t}}, \mu)$, we set $n_0 = \lceil\frac{\log(nq^k)}{\log(2\mu)}\rceil$ and $\mathbf{A} = (\mathbf{A}_1, \mathbf{A}_2)$ where $\mathbf{A}_1 = (\hat{\mathbf{a}}_1, \cdots, \hat{\mathbf{a}}_{n_0}) \in \mathbb{Z}_q^{k\times n_0}$. We firstly check whether $\lambda_1^\infty(\mathcal{L}_q(\mathbf{A}_1^T)) \ge q/(4\mu)$. If not, we cyclically left shift the columns of $\mathbf{A}$ by $n_0$ indices. Assume that $\lambda_1^\infty(\mathcal{L}_q(\mathbf{A}_1^T)) \ge q/(4\mu)$ can be achieved within $\log n$ shifts, otherwise the algorithm would halt with failure. Then we establish the boolean table of size $n \times q^k$ for $f_{\mu s}(j, \hat{\mathbf{z}})$ with $1 \le j \le n$, $\hat{\mathbf{z}} \in \mathbb{Z}_q^k$. Similar with DGS-LR, Sample$\mathbb{Z}$ is called to sample $v_j$ for any $j > n_0$, and $\mathbf{v}_1 = (v_1, \cdots, v_{n_0})$ is generated by a vectorial DGS-GR(Algorithm 1) with input $(\mathbf{A}_1, q, s, \hat{\mathbf{t}} - \sum_{j>n_0} v_j \hat{\mathbf{a}}_j, \mu)$. Finally the algorithm return $\mathbf{v} = (\mathbf{v}_1, v_{n_0+1}, \cdots, v_n)$. We call this sampling algorithm GDGS-LR.

**Theorem 4.2:** For $\mathbf{A}$ uniformly distributed in $\mathbb{Z}_q^{k\times n}$, $\hat{\mathbf{t}} \in \mathbb{Z}_q^k$, $\mu = \omega(\sqrt{\log n})$ and $s > 4\mu\sqrt{\log(2n(1+1/\epsilon))/\pi}$ with $\epsilon = n^{-\omega(1)}$, the output of GDGS-LR$(\mathbf{A}, q, s, \hat{\mathbf{t}}, \mu)$ follows a distribution statistically indistinguishable from $D_{\mathbf{c}_{\hat{\mathbf{t}}}+\mathbf{a}^{\perp}, s}$, where $\mathbf{c}_{\hat{\mathbf{t}}} \in \mathbb{Z}^n$ satisfying $\mathbf{A}\mathbf{c}_{\hat{\mathbf{t}}} = \hat{\mathbf{t}} \bmod q$. The expected running time is $O(nq^{2k})$ and space complexity is $O(nq^k)$ if $q = poly(n)$.

**Remark 4.3:** Theorem 3.5 is essentially the case of $k = 1$ for Theorem 4.2. With a trivial generalization, the proof of Theorem 3.5 still applies to Theorem 4.2 and therefore we omit the proof. For those $q$-ary lattices where $k = O(1)$ and $q = poly(n)$, GDGS-LR still runs in polynomial time.

## 5. Comparison with Other Discrete Gaussian Samplers

We compare our algorithm with existing discrete Gaussian sampling algorithms.

From Theorem 3.5, sampling $D_{\mathbf{c}+\mathcal{L},s}$ for $s > \omega(\log n)$ can be achieved by DGS-LR within $O(nq^2)$ time. The table for $f_{\mu s}(j, z)$ is binary, thus the storage is $O(nq)$ bits. Hence when $q = poly(n)$, our sampling algorithm is polynomial-time. One highlight of DGS-LR is that it is applicable to any width $s > \omega(\log n)$ and independent of the basis.

Diversely, other two polynomial-time samplers proposed in [1] and [9] sample $D_{\mathbf{c}+\mathcal{L},s}$ with the help of a short basis $\mathbf{B}$. The sampler in [1] works for $s > \|\widetilde{\mathbf{B}}\|\omega(\sqrt{\log n})$. The usual cost is $\widetilde{O}(n^3)$ operations and $\Omega(n^3)$ bits of storage according to the analysis in [9], [10]. Utilizing the rounding technique and convolution theorem, Peikert presented an efficient and parallel sampler in [9] which applies for width $s > s_1(\mathbf{B})\omega(\sqrt{\log n})$ where $s_1(\mathbf{B})$ is the largest singular value of the basis $\mathbf{B}$. It requires $\widetilde{O}(n^3)$ for the offline computation and $\widetilde{O}(n^2)$ for the online [10], and $\widetilde{O}(n^2)$ bits for storage [9].

To get rid of the limitations of short basis and width, a sampling algorithm was proposed in [12], [13] that can sample vectors following $D_{\mathbf{c}+\mathcal{L},s}$ at any width $s > 0$ and does not require short basis in advance. However, the time and space complexity of this sampler are $2^{n+o(1)}$.

The detailed comparison of these discrete Gaussian samplers is listed in Table 1.

We remark that all these three existing algorithms [1], [9] and [12], [13] work for arbitrary $q$-array lattices $\mathcal{L} \subset \mathbb{R}^n$, while DGS-LR only works efficiently for specific high-dimensional dense lattices and $q$-ary lattices as clarified in Sect. 4.2.

**Table 1** Comparison with other samplers.

| Samplers | Time | Space | Needs for Short Basis | Width |
|---|---|---|---|---|
| DGS-LR | $O(nq^2)$ | $O(nq)$ | No | $\omega(\log n)$ |
| Alg. in [1] | $\widetilde{O}(n^3)$ | $\Omega(n^3)$ | Yes | $\|\widetilde{\mathbf{B}}\|\omega(\sqrt{\log n})$ |
| Alg. in [9] | $\widetilde{O}(n^3)$ | $\widetilde{O}(n^2)$ | Yes | $s_1(\mathbf{B})\omega(\sqrt{\log n})$ |
| Alg. in [12], [13] | $2^{n+o(1)}$ | $2^{n+o(1)}$ | No | $s > 0$ |

## 6. Conclusion

We propose a new discrete Gaussian sampler over orthogonal lattices by generalizing and refining dynamic programming. Our sampler is polynomial-time for high-dimensional dense lattices. It is worth noting that our sampler generates discrete Gaussian at any width $s > \omega(\log n)$, which is independent of the basis.

Notice that we exploit the basic dynamic programming for subset sum problems that needs space to store a large table. Exploiting optimized dynamic programming techniques may save space and time.

It would be interesting to improve the efficiency of our sampling algorithm for general $q$-ary lattice, which is crucial in the design and cryptanalysis of lattice-based cryptography. We leave it as future work.

## Acknowledgments

This research was supported by The National Key Research and Development Program of China (No. 2017YFA0303903) and National Natural Science Foundation of China (No. 61502269).

### References

[1] C. Gentry, C. Peikert, and V. Vaikuntanathan, "Trapdoors for hard lattices and new cryptographic constructions," STOC 2008, pp.197–206, 2008.
[2] D. Micciancio and C. Peikert, "Trapdoors for lattices: Simpler, tighter, faster, smaller," EUROCRYPT 2012, pp.700–718, 2012.
[3] V. Lyubashevsky, C. Peikert, and O. Regev, "A toolkit for Ring-LWE cryptography," EUROCRYPT 2013, pp.35–54, 2013.
[4] D. Cash, D. Hofheinz, E. Kiltz, and C. Peikert, "Bonsai trees, or how to delegate a lattice basis," EUROCRYPT 2010, pp.523–552, 2010.
[5] S. Agrawal, D. Boneh, and X. Boyen, "Efficient lattice (H)IBE in the standard model," EUROCRYPT 2010, pp.553–572, 2010.
[6] S. Agrawal, D.M. Freeman, and V. Vaikuntanathan, "Functional encryption for inner product predicates from learning with errors," ASIACRYPT 2011, pp.21–40, 2011.
[7] L. Babai, "On Lovász' lattice reduction and the nearest lattice point problem," Combinatorica, vol.6, no.1, pp.1–13, 1986.
[8] P.N. Klein, "Finding the closest lattice vector when it's unusually close," SODA 2000, pp.937–941, 2000.
[9] C. Peikert, "An efficient and parallel Gaussian sampler for lattices," CRYPTO 2010, pp.80–97, 2010.
[10] L. Ducas and P.Q. Nguyen, "Faster Gaussian lattice sampling using lazy floating-point arithmetic," ASIACRYPT 2012, pp.415–432, 2012.
[11] Z. Brakerski, A. Langlois, C. Peikert, O. Regev, and D. Stehlé, "Classical hardness of learning with errors," STOC 2013, pp.575–584, 2013.

[12] D. Aggarwal, D. Dadush, O. Regev, and N. Stephens-Davidowitz, "Solving the shortest vector problem in $2^n$ time using discrete Gaussian sampling: Extended abstract," STOC 2015, pp.733–742, 2015.
[13] D. Aggarwal, D. Dadush, and N. Stephens-Davidowitz, "Solving the closest vector problem in 2ˆn time – The discrete Gaussian strikes again!," FOCS 2015, pp.563–582, 2015.
[14] M. Ajtai and C. Dwork, "A public-key cryptosystem with worst-case/average-case equivalence," STOC 1997, pp.284–293, 1997.
[15] O. Regev, "On lattices, learning with errors, random linear codes, and cryptography," STOC 2005, pp.84–93, 2005.
[16] A. Paz and C. Schnorr, "Approximating integer lattices by lattices with cyclic factor groups," ICALP 1987, pp.386–393, 1987.
[17] P.Q. Nguyen and I.E. Shparlinski, "Counting co-cyclic lattices," SIAM J. Discrete Math., vol.30, no.3, pp.1358–1370, 2016.
[18] Z. Galil and O. Margalit, "An almost linear-time algorithm for the dense subset-sum problem," SIAM J. Comput., vol.20, no.6, pp.1157–1189, 1991.
[19] D. Lokshtanov and J. Nederlof, "Saving space by algebraization," STOC 2010, pp.321–330, 2010.
[20] O. Regev, "Lattices in computer science, fall 2004," ch. Introduction, 2004.
[21] 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.
[22] C. Peikert, "Limits on the hardness of lattice problems in $\ell_p$ norms," IEEE Conference on Computational Complexity, pp.333–346, 2008.
[23] W. Banaszczyk, "Inequalities for convex bodies and polar reciprocal lattices in $\mathbb{R}^n$," Discrete Comput. Geom., vol.13, no.1, pp.217–231, 1995.

**Dianyan Xiao** received her B.S. from School of Mathematics, Shandong University in 2008. She is currently a Ph.D. student of Institute for Advanced Study, Tsinghua University. Her research interests include (but not limited to) lattice-based cryptography, computational complexity and discrete logarithms.

**Yang Yu** received his B.S. in Computer Science from Tsinghua University in 2009. He is currently a Ph.D. student of Department of Computer Science and Technology, Tsinghua University. His current research interests include lattice reduction algorithm, lattice-based cryptography and public key schemes.

**Jingguo Bi** received the B.Sc. and Ph.D. degree in information security from Shandong University in 2007 and 2012, respectively. He is currently an associate researcher in Tsinghua University, Beijing, China. His research interests are designs and analysis of public key cryptosystems.