A Survey on Discrete Gaussian Samplers in Lattice Based Cryptography

Jiaxin Deng[1], Simin Chen[1], Jiageng Chen[1(✉)], and Weizhi Meng[2]

[1] School of Computer Science, Central China Normal University, Wuhan, China
chinkako@gmail.com
[2] Technical University of Denmark, Kongens Lyngby, Denmark

Abstract. Lattice-based cryptography is one of the most competitive algorithms in post-quantum algorithms. The discrete Gaussian sampler is a fundamental building block in lattice-based cryptography, but it is still challenging to construct a generic, efficient and secure discrete Gaussian sampler. In this work, we survey the existing discrete Gaussian samplers and summarize the characteristics and improvements of each sampler in detail. In addition, we discuss the evaluation criteria for samplers which we believe that a good scheme should use less precision to achieve the same level of security. The survey can help the reader to focus on the development of discrete Gaussian samplers and apply the discrete Gaussian sampler to lattice-based cryptography in a black-box manner.

Keywords: Lattice-based cryptography · Discrete Gaussian sampler

1 Introduction

Lattice-based cryptography has attracted more and more attention, not only within the cryptographic community, but also in the area of computer security in both research and industry, for at least two reasons: first, lattice-based cryptosystems are often algorithmically simple and highly parallelizable. Second, lattices allow to construct versatile and advanced cryptographic schemes that go beyond classical public key encryption, like identity-based encryption (IBE), fully homomorphic encryption (FHE), and attribute based encryption (ABE). Cryptographic primitives built on lattices, including digital signature schemes, IBE, and FHE, have been inspired by the Gaussian distribution reduction method proposed by [37]. Discrete Gaussian sampling was used for the first time by Gentry, Peikert, and Vaikuntanathan [23] as a trapdoor function, i.e. *preimage sampleable functions* (PSFs). They used PSFs to construct a hash-and-sign scheme, in which the message is hashed to a point in the lattice, and its signature is a nearby lattice point obtained by the sampling function. However, discrete Gaussian sampling occupies about 50% of the time in digital signature schemes [48]. Sampling with discrete Gaussian distribution is a fundamental problem that arises in almost every application of lattice cryptography, and it can be both time consuming and challenging to implement. Therefore, it is crucial to design an efficient discrete Gaussian sampler.

J. Chen et al. (Eds.): EISA 2022, CCIS 1641, pp. 87–107, 2022.
https://doi.org/10.1007/978-3-031-23098-1_6

Existing samplers generally sample from a uniform distribution or other easy distributions (called approximate distribution) to obtain a target distribution (called ideal distribution). How good or bad the closeness of two distributions is determined by the relative error, and traditionally, most cryptographic schemes use statistical distance as a measure. Because the cryptographic scheme requires high quality, it is rigorous for statistical distance to be used in discrete Gaussian sampling with high quality.

There are two types of sampling algorithms: rejection sampling and inversion sampling. The rejection sampling algorithm first generates a sample x, and if x is greater than the probability of the target distribution, it will be rejected (otherwise accepted). The rejection sampling algorithm has many advantages, such as flexibility, and less memory, because the parameters of the discrete Gaussian distribution can change in real-time, and the memory consumed is independent of the parameters. The inversion sampling algorithm first computes the cumulative distribution table (CDT), then generates the random bits to look up the corresponding probability in the table, and finally inverses the cumulative distribution function to obtain the sampled value. We can divide the inversion sampling algorithm into two phases, namely the offline phase and the online phase. The CDT can be pre-computed in the offline phase, and the table can be directly looked up in the online phase without computing the complicated probability density function. Therefore inversion sampling is usually faster than rejecting sampling, but the former requires a huge pre-computed storage cost. Moreover, the pre-computed table is also related to the parameters of the Gaussian distribution, which makes inversion sampling less generic than rejection sampling (different schemes require different parameters). Recently, studies have shown that the execution time of the sampler can cause side-channel attacks [44], so the cryptographic scheme using discrete Gaussian sampling needs to consider the realization of constant time, but this is in conflict with efficiency. In other words, better use of discrete Gaussian sampling in cryptographic schemes requires a trade-off between memory consumption, sampling speed, parameter flexibility, and security. Over the two fundamental samplers, many works have improved these characteristics, generating a large number of new and more effective samplers.

Designing a generic, efficient, and secure discrete Gaussian sampler is a challenging task, and Gentry et al. [23] first adapted Klein's algorithm to a subroutine of their sampling algorithm, which is a randomized nearest-plane algorithm. Next, Peikert [41] proposed a randomized Babai's simple rounding-off algorithm, where the rounding operation used the inversion sampling. Buchmann et al. [8] proposed discrete ziggurat sampling, a rejection sampling algorithm with a time-memory trade-off. The discrete ziggurat sampling algorithm improves efficiency compared to the original [23] sampling algorithm, but it is still not a practical sampling algorithm because it requires large memory when the parameters are large, and the running time of the algorithm is not constant time. Karney [32] proposed a rejection sampling algorithm based on the von Neumann algorithm, which samples from the exponential distribution, does not require

floating-point arithmetic, and the output distribution is exact. Recently, the (continuous) rounded Gaussian has been proved by Hülsing et al. [28] that it is secure for the cryptographic scheme. Compared with the discrete Gaussian, continuous Gaussian sampling has many useful algorithm libraries, such as the Box-muller algorithm. Dwarakanath et al. [18] adapted the Knuth-Yao algorithm to discrete Gaussian sampling for the first time. The algorithm is very efficient, but the disadvantage is that it relies on a pre-computed table and requires huge memory (when the standard deviation is large). Ducas et al. [14] proposed a Bernoulli-type sampling algorithm that first samples from a simple discrete Gaussian distribution and second performs rejection sampling on the sample values. The first step can be run in the offline phase and requires a relatively small amount of memory; The second step samples from a Bernoulli distribution without computing transcendental function. Because the sampler proposed by [14] is efficient and versatile, it has received a lot of attention, such as in terms of efficiency, generality, and security [3,11,13,27,43,46,49,51,53]. Micciancio and Walter [39] used the convolution-like technique in [41] to propose a significantly efficient and generic sampler. Inspired by the work of [4,39,50] improved the shortcoming of the gadget trapdoor in [38], that is, it can only be constructed on a power-of-q moduli lattice. Finally, [17] used a fast Fourier orthogonalization technique that greatly improves the sampling algorithm in [23] and applied it to the Falcon [21] signature.

Our Contribution: Our main contribution is to sort out the progress of discrete Gaussian samplers in the past ten years. We list the measures that Gaussian samplers need to use, as well as improvements in efficiency, generality, and security, as shown in Table 1. We divide Gaussian samplers into three types: rejection samplers, inversion samplers, and hybrid samplers. We believe that the survey of these samplers can help readers quickly study discrete Gaussian sampling, and provide some ideas for those interested in lattice-based cryptography.

Related Work: [20] surveyed some samplers, but it did not cover the variety of samplers, did not discuss the effect of measures, nor did it talk about constant-time improvements. [26] implemented a suite for testing discrete Gaussian sampling outputs, and [27] proposed a new test suite named SAGA. [6] investigated the hardware implementation for generating discrete Gaussian distributions, but we mainly discuss the algorithm and its improvements. [24] implemented a new sampling algorithm for lattice trapdoor, and also evaluated the recently improved Gaussian sampling algorithm, but it was not enough and the samplers were not sorted.

Organization of the Paper: We introduce some notions and discrete Gaussian distributions used in this paper in Sect. 2. We discuss measures for evaluating output distributions in Sect. 3. In Sect. 4, we extensively describe proposed samplers and their variants. Finally, we summarize the work of this paper in Sect. 5.

Table 1. Comparison between different sampler improvements: measures, efficiency, parameters, and security

Sampler	Memory	Speed	Centered	Fixed Deviation	Constant-Time	Measures
Rejection [16,23,35]	$-/-/-$	$(2\tau/\sqrt{2\pi})/6\tau/(2\tau/\sqrt{2\pi})$	✗/✗/✗	✗/✗/✗	-/-/-	S/S/S
CDT [1,10,16,29,36,41]	$O(\lambda\sigma)/O(\lambda\sigma)/O(\lambda\sigma)/$ $O(\lambda\sigma)/O(\lambda\sigma^*)/O(\lambda\sigma^*)$	$O(\lambda\log\sigma)/O(\lambda\log\sigma)/O(\lambda\log\sigma)/$ $O(\lambda\log\lambda)/O(\lambda\sigma^*)/O(\lambda\log\lambda)$	✓/✓/✗/✗/✓/✓	✓/✓/✓/✓/✓/✗	-/-/-/-/I	S/S/S/R/S/K
Discrete Ziggurat [7,8,25,40]	var	var	✓/✓/✓/✓	✓/✓/✓/✓	-/I/-/I	S/K/S/S
Knuth-Yao [18,25,30,31,44,45]	$O(\lambda\sigma)/O(\lambda\sigma)/O(\lambda\sigma)/$ $O(\lambda\sigma)/O(\lambda\sigma)/O(\lambda\sigma^*)$	$O(\lambda\log\lambda)/O(\lambda\log\lambda)/O(\lambda\log\lambda)/$ $O(\lambda\log\lambda)/O(\lambda\log\lambda)/O(\lambda\log\lambda)$	✓/✓/✓/✓/✓	✓/✓/✓/✓/✓	-/-/I/I/I/I	S/S/K/K/S
Bernoulli-type [13,14,25,43,53] [3,11,27,46,49,51]	$O(\lambda\log\sigma)/O(\lambda\log\sigma)/-$ $/O(\lambda\log\sigma)/O(\lambda)/O(\lambda)/$ $O(\lambda)/O(\lambda)/O(1)/O(1)$	$\leq 1.47/13/\approx 1.47/\leq 2/\leq 2/$ $\leq 2/\leq 2/\approx 1.47/\leq 3/\leq 2/\leq 2$	✓/✓/✗/✗/✗/ ✓/✗/✗/✗/✓/✓	✗/✗/✗/✗/✗ /✓/✗/✗/✗/✓/✗	-/I/I/I/I /I/II/- /II/I/I	S/K/M/R/R/ R/R/S/R/R/R
Karney [12,13,32]	-/-/-	$\approx 2.03/\approx 1.47/\approx 3.68$	✗/✗/✗	✗/✗/✗	-/I/-	-/M/-
Convolution-like [4,15,19,22,29,50]	var	var	✗/✗/✗/✗/✗/✓	✗/✗/✗/✗/✗/✓	I/I/-/I/-	M/S/K/M/R/M
Rounded-Gaussian [5,7,28,51,52]	-/-/-/-/-	-/2/-/-/2	✗/✗/✗/✗/✗	✗/✗/✗/✗/✗	I/I/I/I/I	R/R/R/R/R
Fast Fourier [17,21]	$-/O(\sigma_{\max})$	-/-	✗/✗	✗/✗	-/-	S/R

Table 1 summarizes the types of existing samplers, the memory used for sampling, the sampling speed (the time complexity of search algorithms or average iterations of rejection sampling), whether the center of discrete Gaussian is fixed, whether the standard deviation of discrete Gaussian is fixed, resisting side channel attack and the evaluation criteria between an ideal distribution and actual distribution.

- ▶ Bit precision is represented by the parameter λ.
- ▶ The Gaussian distribution's standard deviation is represented by the parameter σ.
- ▶ τ is the tail-cut of the distribution.
- ▶ "var" means memory or speed is variable.
- ▶ Smaller parameters are marked with *.
- ▶ Side channel attacks can be resisted by: Constant-Time (Type I) and Isochronous (Type II).
- ▶ There are four measures: Statistical distance (S), KL-divergence (K), Rényi-divergence (R) and Max-Log distance (M).

2 Preliminaries

2.1 Notion

We denote the integers by $\mathbb{Z}$, the rationals by $\mathbb{Q}$, and the reals by $\mathbb{R}$. We extend any real function $f(\cdot)$ to a countable set A by defining $f(A) = \sum_{x \in A} f(x)$.

We use bold lower-case letters (e.g., $\mathbf{x}$) to denote vectors in $\mathbb{R}^n$, for an undetermined positive integer dimension n that remains the same throughout the paper. A lattice Λ is a discrete additive subgroup of $\mathbb{R}^n$. In this work, we are only concerned with full-rank lattices, which are generated by some non-singular basis $B \in \mathbb{R}^{n \times n}$.

We use standard big-O notation to classify the growth of functions, and say that $f(n) = \widetilde{O}(g(n))$ if $f(n) = O(g(n) \cdot \log^c n)$ for some fixed constant c.

2.2 Discrete Gaussian Distribution

Definition 1. *For any center $c \in \mathbb{R}$ and Gaussian parameter $s \in \mathbb{R}^+$, let*

$$\rho_{s,c}(x) = e^{-\pi(x-c)^2/s^2}$$

be a Gaussian function centered in c scaled by a factor of s. This definition is sometimes formulated with the parameter $\sigma = s/\sqrt{2\pi}$. It is worth to mention that the integration of $\rho_{s,c}$ is $\int_{-\infty}^{+\infty} \rho_{s,c}(x) = s$. Therefore, we can define the continuous Gaussian distribution around c with parameter s by its probability density function

$$\forall x \in \mathbb{R}, D_{s,c}(x) = \frac{\rho_{s,c}(x)}{\int_{-\infty}^{+\infty} \rho_{s,c}(x)} = \frac{\rho_{s,c}(x)}{s}.$$

We extend Gaussian function to any countable set A by $\rho_{s,c}(A) = \sum_{x \in A} \rho_{s,c}(x)$. The discrete Gaussian distribution over the integers, denoted by

$$\forall x \in \mathbb{Z}, \mathcal{D}_{\mathbb{Z},c,s}(x) = \frac{D_{s,c}(x)}{D_{s,c}(\mathbb{Z})} = \frac{\rho_{s,c}(x)}{\rho_{s,c}(\mathbb{Z})}.$$

Naturally, we can extend the discrete Gaussian over the integer to the discrete Gaussian over the n-dimension lattice, where $\rho_{s,\mathbf{c}}(\mathbf{x}) = e^{-\pi\|\mathbf{x}-\mathbf{c}\|^2/s^2}$ and $\mathcal{D}_{\Lambda,\mathbf{c},s}(\mathbf{x}) = \frac{\rho_{s,\mathbf{c}}(\mathbf{x})}{\rho_{s,\mathbf{c}}(\Lambda)}$.

2.3 Smoothing Parameter

Definition 2 [37, *Definition 3.1*]**.** *For any $\epsilon > 0$, the smoothing parameter of the integers is the smallest $s > 0$ such that $\rho(s\mathbb{Z}) \leq 1 + \epsilon$.*

Lemma 1 [37, *Lemma 3.3*]. *For any integer $\mathbb{Z}$ and positive real $\epsilon > 0$, the smoothing parameter satisfies*

$$\eta_\epsilon(\mathbb{Z}) \leq \sqrt{\ln(2 + 2/\epsilon)/\pi}.$$

For example, $\eta_\epsilon(\mathbb{Z}) < 6$ is a relatively small constant for very small values of $\epsilon < 2^{-160}$.

We will use the following lemma to show how to limit the tail-cut τ,

Lemma 2 [23, *Lemma 4.2*]. *For any $\epsilon > 0$, any $s \geq \eta_\epsilon(\mathbb{Z})$, and any $\tau > 0$,*

$$\Pr_{x \leftarrow \mathcal{D}_{\mathbb{Z},s,c}}[|x - c| \geq \tau \cdot s] \leq 2e^{-\pi\tau^2} \cdot \frac{1+\epsilon}{1-\epsilon}.$$

It is too difficult or costly to sample from the ideal distribution, hence we expect the actual sampled distribution to be statistically close to the theoretical discrete Gaussian. There are several measures to keep the security proofs of the cryptographic schemes correct.

3 Measures

In this section, we overview the criteria for evaluating the output distribution of a sampler. Many security reductions for lattice-based cryptographic primitives assume that the primitive has access to samplers for ideal distribution. However, sampling from the ideal distribution is too difficult or expensive, so we just only sample from the approximate distribution. Traditionally, the quality between the approximation distribution and ideal distribution has been measured in terms of the statistical distance. Generally, the security required by a cryptographic scheme is $2^{-\lambda}$. In recent years, several statistical measures have been used to analyze the closeness of the approximate distribution to the ideal distribution. With the new measure, the analysis of cryptographic security becomes simpler and the same security can be achieved using fewer bits.

3.1 Statistical Distance

Definition 3. *The statistical distance between two distributions $\mathcal{P}$ and $\mathcal{Q}$ over the same support S is defined as*

$$\Delta(\mathcal{P}, \mathcal{Q}) = \frac{1}{2} \sum_{x \in S} |\mathcal{P}(x) - \mathcal{Q}(x)|.$$

The statistical distance is a naive and most commonly used measure. The following lemma uses statistical distance to show the relationship between the Gaussian distribution and the uniform parallelepiped.

Lemma 3 [37, *Lemma 4.1*]. *For any $s > 0, \boldsymbol{c} \in \mathbb{R}^n$, and lattice $\Lambda(B)$, the statistical distance between $\mathcal{D}_{s,\boldsymbol{c}}$ mod $\mathcal{P}(B)$ and the uniform distribution over $\mathcal{P}(B)$ is at most $1/2\rho_{1/s}(\Lambda(B)^* \backslash \{0\})$. In particular, for any $\epsilon > 0$ and any $s \geq \eta_\epsilon(B)$, the statistical distance is at most*

$$\Delta(\mathcal{D}_{s,\boldsymbol{c}} \bmod \mathcal{P}(B), U(\mathcal{P}(B))) \leq \epsilon/2.$$

Thus, to achieve λ-bit security, the accuracy of the statistical distance requires $\lambda + 1$ bits.

3.2 Kullback-Leibler Divergence

Definition 4. *Let $\mathcal{P}$ and $\mathcal{Q}$ be two distributions over a common countable set Ω, and let $S \subset \Omega$ be the strict support of $\mathcal{P}(\mathcal{P}(i) > 0 \text{ if } i \in S)$. The Kullback-Leibler divergence, noted D_{KL} of $\mathcal{Q}$ from $\mathcal{P}$ is defined as:*

$$D_{KL}(\mathcal{P}\|\mathcal{Q}) \leq 2\sum_{i\in S} \ln\left(\frac{\mathcal{P}(i)}{\mathcal{Q}(i)}\right)\mathcal{P}(i).$$

with the convention that $\ln(x/0) = +\infty$ for any $x > 0$.

Compared with statistical distance, KL-divergence has many good properties, such as additivity and non-decreasing. An important difference though is that it is not symmetric. The following lemma shows that KL divergence provides higher security,

Lemma 4 [42, *Lemma 1*]. *Let $\varepsilon^{\mathcal{P}}$ be an algorithm making at most q queries to an oracle sampling from a distribution $\mathcal{P}$ and returning a bit. Let $\epsilon \geq 0$, and $\mathcal{Q}$ be a distribution such that $D_{KL}(\mathcal{P}\|\mathcal{Q}) \leq \epsilon$. Let x (resp. y) denote the probability that $\varepsilon^{\mathcal{P}}$ (resp. $\varepsilon^{\mathcal{Q}}$) outputs 1. Then, $|x - y| \leq \sqrt{q\epsilon/2}$.*

We give an example below to illustrate the advantages of KL divergence in terms of security. Let $\mathcal{B}_c$ be the Bernoulli variable that returns 1 with probability c, then we have $D_{KL}(\mathcal{B}_{\frac{1-\epsilon}{2}}\|\mathcal{B}_{\frac{1}{2}}) \approx \epsilon^2/2$ (by Definition 4). Note that one requires $q = O(1/\epsilon^2)$ samples to distinguish two distribution with constant advantage rather than $q = O(1/\epsilon)$ queries for the statistical distance.

Many works use KL-divergence instead of statistical distance, see Table 1 for details. Pöppelmann et al. [42] first use KL-divergence to improve the convolution Lemma of Peikert [41] and construct a sampler using convolutions, resulting in the smoothing condition reduced by a factor of about $\sqrt{2}$. They also present a CDT sampling algorithm with reduced table size; [25,30] proposed a more efficient and more secure Gaussian sampling algorithm by using KL-divergence.

3.3 Rényi Divergence

Definition 5. *For any two discrete probability distribution P and Q such that $Supp(P) \subseteq Supp(Q)$ and $a \in (1, +\infty)$, we define the Rényi divergence of order a by*

$$R_a(P\|Q) = \left(\sum_{x\in Supp(P)} \frac{P(x)^a}{Q(x)^{a-1}}\right)^{\frac{1}{a-1}}.$$

We omit the a subscript when $a = 2$. We define the Rényi divergences of order 1 and $+\infty$ by

$$R_1(P\|Q) = \exp\left(\sum_{x\in SuppP} P(x)\log\frac{P(x)}{Q(x)}\right) \quad \textit{and} \quad R_\infty(P\|Q) = \max_{x\in Supp(P)} \frac{P(x)}{Q(x)}.$$

As we can see, the divergence R_1 is the (exponential of) the KL-divergence.

Similarly, replacing the statistical distance can reduce the required parameter s for the smoothing parameter by a factor of $\Theta(\sqrt{\lambda})$. [2] analyzed the BLISS sampler using Rényi divergences, and specifically the divergence of order infinity is closely related to the relative error discussed in [47]. Again, see Table 1 for using divergence in discrete Gaussian sampling.

3.4 Max-Log Distance

The above measures either do not have strong security or depend on some specific conditions. Micciancio and Walter [39] discussed the properties of a useful or efficient measure in a cryptographic scheme, and proposed a new notation, namely, max-log distance.

Definition 6. *The max-log distance between two distributions P and Q over the same support S is*

$$\Delta(\mathcal{P}, \mathcal{Q}) = \max_{x \in S} |\ln \mathcal{P}(x) - \ln \mathcal{Q}(x)|.$$

[39] performed a detailed analysis of ML-distance that satisfies three useful properties, including probability preservation, sub-additivity, and data processing inequality. They call any measure that satisfies these three properties a useful measure. Furthermore, a measure will be called λ-efficient measure if it satisfies λ-pythagorean property. The following lemma shows that max-log distance is more competitive than other measures.

Lemma 5 [39, *Lemma 3.3*]. *Let $S^{\mathcal{P}}$ be a standard cryptographic scheme with oracle to a probability distribution ensemble $\mathcal{P}_\theta$. If $S^{\mathcal{P}}$ is k-bit secure and $\delta(\mathcal{P}_\theta, \mathcal{Q}_\theta) \leq 2^{-k/2}$ for some $2^{-k/2}$-efficient measure δ, then $S^{\mathcal{Q}}$ is $(k-3)$-bit secure.*

According to Lemma 5, we can build a 128-bit secure encryption scheme but only need to store data in variables with 63-bit precision. For more details we refer the reader to [39] and list papers that use different measures in recent years, see Table 1.

4 Samplers

Now we know what a discrete Gaussian distribution over the integers or a lattice is, we will review the current sampling algorithms. Such algorithms are known as Gaussian samplers, and there were two main types of Gaussian samplers, including rejection samplers and inversion samplers. Finally, we discuss the hybrid sampler, which is closely related to lattice-based cryptographic schemes, such as improving the efficiency of lattice signatures.

4.1 Rejection Sampler

The basic rejection sampling is as follows: sampling $x \in S$ from some easy distribution (typically the uniform or exponential distribution), where S is a set, and then accepting the sample within the probability proportional to $\rho_{s,c}(x)$ (otherwise rejecting). We survey several variants of rejection sampling and give more details below.

4.1.1 Klein's Sampler

Klein's sampler was first introduced in [33] and used in a cryptographic context for the first time by Gentry, Peikert and Vaikuntanathan [23]. Algorithm 1 describes the details of the sampler, which is a subroutine of the lattice Gaussian sampler in [23], namely a randomized rounding according to a discrete Gaussian over $\mathbb{Z}$.

Algorithm 1. Klein's Sampler

Output: a sample value $x \in D_{\mathbb{Z},\sigma,c}$

1: **do**
2: $x \leftarrow \mathbb{Z} \cap [c - \tau\sigma, c + \tau\sigma]$
3: $y \in [0, 1)$ uniformly at random
4: **while** $y < \rho_{\sigma,c}(x)$
5: **return** x

The above algorithm is simple and intuitive and does not require a precomputed table. Thus, its memory consumption is independent of the parameters. However, since the algorithm needs to consume a lot of resources to compute the transcendental function e, and involves arbitrary precision operations; the rejection rate of Klein's sampler is so high that it is rarely used directly in the practical scheme of lattice-based cryptography.

Ducas and Nguyen [16] found that in most cases the most significant bits are enough to reject the samples. Their algorithm uses floating point numbers to compute transcendental functions, and the laziness technique they proposed can significantly speed up sampling algorithms. Specially, in NTRUSign lattice, laziness can decrease the complexity of preimage sampling function from $\widetilde{\mathcal{O}}(n^3)$ to $\widetilde{\mathcal{O}}(n^2)$ or even $\widetilde{\mathcal{O}}(n)$.

4.1.2 Discrete Ziggurat Sampler

Buchmann et al. [8] adapted the continuous Ziggurat algorithm to the discrete case. The discrete Ziggurat algorithm first generates multiple adjacent rectangles with the same area from top to bottom, the y-axis is the left side of the rectangle, the lower right corner of the rectangle is on the Gaussian distribution function, and the coordinates (x_i, y_i) of the lower right corner of the rectangle are stored in precomputation table. When sampling, the algorithm picks a rectangle R_i uniformly at random and then samples an x coordinate inside the rectangle also uniformly

at random. Rectangle R_i can be split into a left rectangle R_i^l that lies completely within the area of the probability density function (PDF) and a right rectangle R_i^r that is only partially covered by the PDF. If the coordinate x is in the left rectangle R_i^l, then it is accepted as a sample. Otherwise, we perform rejection sampling in the right rectangle R_i^r. Algorithm 2 shows the full sampling process.

Algorithm 2. Discrete Ziggurat Sampler

Input: $m, \sigma, \lfloor x_1 \rfloor, \cdots, \lfloor x_m \rfloor, \bar{y}_0, \bar{y}_1, \cdots, \bar{y}_m, \omega$
Output: a sample value $x \in D_{\mathbb{Z},\sigma,c}$

while true **do**
  $i \xleftarrow{\$} \{1, \cdots, m\}, s \xleftarrow{\$} \{-1, 1\}, x \xleftarrow{\$} \{0, \cdots, \lfloor x_i \rfloor\}$
  // choose rectangle, sign and value
  if $0 < x \leq \lfloor x_{i-1} \rfloor$ **then return** sx;
  else
    if $x = 0$ **then**
      $b \xleftarrow{\$} \{0, 1\}$
      if $b = 0$ **then return** sx;
      else
        continue;
      end if
    else
      //in rejection area R_i^r now
      $y' \xleftarrow{\$} \{0, \cdots, 2^\omega - 1\}, \bar{y} = y' \cdot (\bar{y}_{i-1} + \bar{y}_i)$;
      if $\bar{y} \leq 2^\omega \cdot (\bar{\rho}_\sigma(x) - \bar{y}_i)$ **then**
        return x
      end if
    end if

  end if
end while
return x

The expensive part of Algorithm 2 is computing $\rho_\sigma(x)$ if x does not in R_i^l. If we use more rectangles, the ratio between the left and the right rectangle becomes comparatively bigger. Hence, we accept an x without computing $\rho_\sigma(x)$ with higher probability. The discrete Ziggurat sampler provides a time-memory trade-off by controlling the number of rectangles used (more rectangles, larger memory requirement). Compared with other sampling algorithms, Algorithm 2 can only sample from a discrete Gaussian with a fixed center and is not resistant to side-channel attacks. Howe et al. [25] presented the first hardware implementation of a discrete Gaussian ziggurat sampler. They achieved constant time by controlling signals through a state machine to ensure that all comparisons were performed, with only a few extra clock cycles. Since the original ziggurat algorithm changes its time-memory requirement takes about 1000 s seconds, More and Katti [40] proposed an improved ziggurat algorithm, which can eliminate this large computational overhead and switch different "time-memory" options on the fly by storing a minimal number of extra points $(x_i, \rho(x_i))$.

4.1.3 Karney's Sampler

Karney's sampler [32] is essentially a rejection method which uses von Neumann's algorithm to sample from the exponential distribution. Intuitively, this algorithm

Algorithm 3. Karney's Sampler

Output: a sample $x \in$ normal distribution $\phi(x)$

1: **Step1.**[Sample integer part of deviate k] Select integer $k \geq 0$ with probability $\exp(-\frac{1}{2}k)(1-1/\sqrt{e})$
2: **Step2.**[Adjust relative probability of k by rejection.]Accept k with probability $\exp(-\frac{1}{2}k(k-1))$; otherwise go to **Step1**.
3: **Step3.**[Sample fractional part of deviate x.] Set $x \leftarrow U$, where U is a uniform deviate $U \in (0,1)$
4: **Step4.**[Adjust relative probability of x by rejection.]Accept x with probability $\exp(-\frac{1}{2}x(2k+x))$; otherwise go to **Step1**.
5: **Step5.**[Combine integer and fraction.]Set $y \leftarrow k+x$
6: **Step6.**[Assign a sign.]With probability $\frac{1}{2}$,set $y \leftarrow -y$
7: **Step7.**[Return result.] Set $N \leftarrow y$

requires neither pre-computed tables nor floating point arithmetic. All the steps can be carried out exactly. It is a very nice property since most sampling algorithms require floating-point arithmetic, which is difficult to run on constrained devices. Overall, the probability of success in the **Step2** is $(1-1/\sqrt{e})G \approx 0.690$, where $G = \sum_{k=0}^{\infty} \exp(-1/2k^2) \approx 1.753$ and **Step4** succeeds with probability $\sqrt{\pi/2}/G \approx 0.715$. Therefore, **Step1** is executed $\sqrt{2/\pi}/(1-1/\sqrt{e}) \approx 2.03$ time on average. Du et al. [12] study the computational complexity of Karney's exact sampling algorithm under the random deviate model. They give an estimate of the expected number of uniform deviates used by Karney's algorithm, and present an improved algorithm with 2.018 uniform deviates rather than 2.194, which has better actual performance than Karney's algorithm. Although Algorithm 3 can only sample from the normal distribution, Du et al. [13] can achieve an off-centered discrete Gaussian distribution in combination with the exact property, see Sect. 4.3.1.

4.1.4 Rounded-Gaussian Sampler

Hülsing et al. [28] proposed to replace discrete Gaussian with a different distribution, namely the rounded Gaussian distribution. This distribution has the same benefits as the discrete Gaussian, but the security analysis of using rounded Gaussian in Lyubashevsky's scheme [34] is more involved than discrete Gaussian.

Formally, the rounded Gaussian distribution is obtained by rounding samples from a continuous Gaussian distribution to the nearest integer x_i. To compute the probability at an integer x_i, we compute the integral over the interval $(x_i - \frac{1}{2}, x_i + \frac{1}{2}]$.

Definition 7. *The rounded Gaussian distribution over* $\mathbb{Z}^m$ *centered at some* $\boldsymbol{v} \in \mathbb{Z}^m$ *with parameter* σ *is defined for* $\boldsymbol{x} \in \mathbb{Z}^m$ *as*

$$R^m_{\boldsymbol{v},\sigma}(\boldsymbol{x}) = \int_{A_{\boldsymbol{x}}} \rho^m_{\boldsymbol{v},\sigma}(\boldsymbol{s}) d\boldsymbol{s} = \int_{A_{\boldsymbol{x}}} \left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^m \exp\left(\frac{-\|\boldsymbol{s}-\boldsymbol{v}\|^2}{2\sigma^2}\right) d\boldsymbol{s}$$

where $A_{\mathbf{x}}$ denotes the area defined by $[x_1 - 1/2, x_1 + 1/2) \times \cdots \times [x_m - 1/2, x_m + 1/2)$. A very efficient and easy way to generate samples from the continuous Gaussian distribution is based on the Box-Muller transform.

Algorithm 4. Rounded-Gaussian Sampler

Input: The required number of samples $m = 2n$ and parameter σ
Output: m independent rounded Gaussian distribution integers $z_0, \cdots, z_{2n-1}$ with parameter σ

1: **for** i=0 **do**n-1
2: Generate two uniform random numbers u_1, u_2
3: $x_1, x_2 \leftarrow BoxMuller(u_1, u_2)$
4: $z_{2i} \leftarrow \lfloor x_1 \cdot \sigma \rceil$
5: $z_{2i+1} \leftarrow \lfloor x_2 \cdot \sigma \rceil$
6: **end for**
7: **return** $(z_0, \cdots, z_{2n-1})$

Hülsing et al. [28] show that it is secure enough to use rounded Gaussian, while the resulting rejection rate is identical to before. The implementation of the sampling algorithm is within less than 40 lines of C++ source code. Brannigan et al. [5] presented the first secure algorithm for implementing the Box-Muller Gaussian sampling algorithm. Recently, Zhao et al. [52] implemented an arbitrary-centered discrete Gaussian sampling algorithm over integers based on [28]. Compared to previous arbitrary-centered discrete Gaussian sampling techniques, their algorithm does not require any pre-computations and only requires a low number of trials close to 2 per sample on average.

4.2 Inversion Sampler

The inversion method is based upon the following theorem:

Theorem 1 [9, *Chapter 2*]. *Let* F *be a continuous distribution function on* R *with inverse* F^{-1} *defined by* $F^{-1}(u) = \{x : F(x) = u, 0 < u < 1\}$. *If* U *is a uniform [0,1] random variable, then* $F^{-1}(U)$ *has distribution function* F.

4.2.1 CDT Sampler

The cumulative distribution table (CDT) sampler was first proposed by Peikert [41]. The algorithm needs to compute the discrete Gaussian distribution $p_z \in Pr(x \leq z; x \leftarrow D_\sigma)$, and store the p_z in the cumulative distribution table. Since

the distribution is symmetric about the y-axis, taking $z \in [0, \tau\delta]$ can save half of the storage space, generally the required storage space is about $\lambda\tau\delta$ bits. We can sample directly from discrete Gaussian distribution by choosing a uniformly random $x \in [0, 1)$ and performing a binary search through the table for the $\widetilde{z} \in \mathbb{Z}$ such that $x \in [p_{\widetilde{z}-1}, p_{\widetilde{z}})$.

The CDT algorithm avoids the exponential function and floating-point arithmetic, but due to the whole distribution table, it needs a lot of storage. Because precomputing the CDT needs to know the center c, the CDT algorithm is usually used as an offline phase. The variant [16] of Peikert's offline algorithm, by using laziness techniques, runs in average time $\widetilde{O}(n)$ in some ring setting (where n is the lattice dimension). Du et al. [10] further optimized the CDT sampler. They found that when comparing two large random bits (such as 112 bits), the size of the two random numbers can often be determined by comparing the first 8 bits, thus saving $112 - 8 = 104$ random bits are used, and the remaining bits need to be compared only if the current 8 bits are the same. Melchor et al. [36] proposed a Twin-CDT algorithm, which can sample from a non-centered discrete Gaussian by precomputing the CDT for a relatively small number of centers. They decreased the number of pre-computed CDTs by using a Taylor expansion rather than the laziness technique. Aguilar-Melchor and Ricosset [1] use Rényi divergence to improve the Twin-CDT algorithm, that is, only double precision can be used for the usual lattice signature parameters and fixing the issue where prior construction is not constant-time. Karabulut et al. [29] propose a more efficient, flexible, constant-time Gaussian sampler combined with fusion trees. The results show that the size of the cumulative distribution table can be reduced by up to 86%, and the parameters are suitable for post-quantum digital signature schemes such as qTesla, Falcon, etc.

4.2.2 Knuth-Yao Sampler

Dwarakanath et al. [18] first considered using the Knuth-Yao algorithm for discrete Gaussian sampling. This algorithm is based on a discrete distribution generating (DDG) tree which is constructed from a probability matrix, where the probability matrix is denoted by $P_1, P_2, \cdots P_n$, while each sample can be represented as a binary expansion of λ bits, the probability matrix P can be written as a $N \times \lambda$ binary matrix. The DDG tree contains two types of nodes, namely internal nodes, and terminal nodes. The number of terminal nodes in the i-th level of the tree is equal to the number of non-zeros in the i-th column of the probability matrix. Each terminal node is marked as the row number of matrix P. The sampling process is a random walk from the tree root to the terminal node. When it hits the terminal node, it stops walking and outputs the sampling result. Algorithm 5 formally describes this process. From the perspective of information theory, the algorithm is perfect, it requires random input bits that are only 2 larger than the entropy of the distribution.

Roy et al. [45] used hardware to construct the Knuth-Yao sampler for the first time. They optimized the structure of the DDG tree. It is not necessary to store the entire DDG tree. Instead, one just stores the node of the current level and the

Algorithm 5. Knuth-Yao Sampler

Require: Three integers d, hit and ctr;
Discrete samplers of Gaussian distribution as matrix P with $N \times \lambda$ dimension and $N = \tau \times \sigma$;
Sample bits uniformly in {0,1}, store in array r;
Column-wise Hamming distance of P, i.e., $h\_dist[j] = \sum_{i=0}^{N} P[i][j]$;
Ensure: $d \leftarrow 0; hit \leftarrow 0; ctr \leftarrow 0;$
for int $col \leftarrow 0; col < \lambda; col \leftarrow col + 1$ **do**
  $d \leftarrow 2d + (!r[ctr++]) - h\_dist[col];$
  if d<0 **then**
    for int $row \leftarrow 0; row < N; row \leftarrow row + 1$ **do**
      $d \leftarrow d + P[row][col];$
      if d == 0 **then**
        $hit \leftarrow 1;$
        break;
      end if
    end for
    if hit **then**
      break;
    end if
  end if
end for
return $(-1)^{r[ctr++]}$**.row**

information of the next level; from this information, it is simple to construct the full tree. [25,30,31,44] all improve the problem that the Knuth-Yao operation is non-constant time. To achieve time-independent, [25] use a simple shuffler to shuffle discrete Gaussian samples after the generation of a complete block. [30] proposed a bit-slicing Knuth-Yao algorithm, which is 2.4 times faster than the shuffle-based constant-time implementation and consumes significantly less memory. [44] observed an interesting property of the mapping from input random bit strings to samples and proposed an efficient method to minimize the Boolean expression of the mapping. The overall performance of the signature algorithm drops by up to 33% due to the additional overhead of "constant-time" sampling.

4.3 Hybrid Sampler

The samplers we introduced above are either sampled over the integers or a single algorithm. In this section, we summarize the samplers actually used in lattice-based cryptography, which are efficient, generic, and secure at the same time. We also introduce some accelerated methods at the end, which have performed well in recent NIST submissions.

4.3.1 Bernoulli-Type Sampler

Ducas et al. [14] first proposed the Bernoulli-type sampler, which combines rejection sampling and inversion sampling, and applied it to BLISS digital signature

scheme. This algorithm uses rejection sampling from the binary discrete Gaussian distribution, denoted by $D_{\mathbb{Z}+,\sigma_2}$. Firstly, Sampling $x \in \mathbb{Z}^+$ from $D_{\mathbb{Z}+,\sigma_2}$ can be very efficient because the pre-computed table for $\sigma_2 = \sqrt{1/(2 \cdot \ln 2)}$ is small. Secondly, it samples $y \in \mathbb{Z}$ uniformly in $\{0, 1, 2, \cdots, k+1\}$ and accepts $z = kx+y$ with probability $\exp -(y^2 + 2kxy)/2k^2\sigma_2^2$. Finally, it outputs a signed integer, z or $-z$, with equal probabilities. Algorithm 6 gives a detailed description.

Algorithm 6. Bernoulli-Type Sampler

Require: An integer $k \in \mathbb{Z}(\sigma = k\sigma_2)$
Ensure: An integer $z \in \mathbb{Z}^+$ according to D_σ^+
1: sample $x \in \mathbb{Z}$ according to $D_{\sigma_2}^+$
2: sample $y \in \mathbb{Z}$ uniformly in $\{0, \cdots, k-1\}$
3: $z \leftarrow kx + y$
4: sample $b \leftarrow \mathcal{B}_{\exp(-y(y+2kx)/(2\sigma^2))}$
5: **if** $\neg b$ **then restart**
6: **return** z
7: **end if**

A complete analysis of Algorithm 6 is given by Ducas et al. [14] Compared with the original rejection sampling algorithm, the average number of rejections is smaller than 1.47, so this is a very practical algorithm. The disadvantage of the Bernoulli-type sampling algorithm is that it is centered and not constant time. A lot of work has proposed improvements based on this algorithm, such as [3,11,25,27,43,53] to compensate for the security of the sampler; [13,46,49] to improve the efficiency and generality. [3,27] found that a polynomial approximation of the transcendental function could be used to achieve constant time. For sampling from the binary distribution, both need to look up the full CDT. To achieve constant-time, the FACCT method [53] used floating point multiplications instead to compute the exponential. Otherwise, the GALACTICS method [3] used integer polynomials to avoid floating operations. Howe et al. [27] show that their sampler is isochronous, i.e. running time is independent of the inputs σ, c and of the output z. Isochrony is weaker than being constant-time, but it suffices to argue security against timing attacks. More precisely, they used a base sampler and Bernoulli rejection sampler, in which the base sampler needs to look up the entire CDT to achieve isochronous, and the Bernoulli rejection sampler approximates the transcendental function by using the GALACTICS method [3]. Du et al. [11] proposed a constant-time algorithm, which requires no precomputation storage and its entropy consumption is smaller than that of the full-table access algorithm. Replacing the binary sampling with Karney's algorithm [32], Xie et al. [49] proposed a new algorithm without floating-point arithmetic, which supports a variable center of precision up to 20 bits. [46] constructed a Gaussian sampler that is generic, efficient, and resistant to timing attacks. Compared with the polynomial approximation method, their algorithm reduces the floating-point multiplications significantly.

4.3.2 Convolution-Like Sampler

There are two strategies to design generic Gaussian samplers. One is based on rejection sampling, and another is based on the convolution-like technique of discrete Gaussian distributions. The convolution-like technique was first applied to lattice Gaussian sampling by Peikert [41], and [38] used this technique to build a very efficient trapdoor for the lattice scheme. All convolution-like samplers invoke the following theorem.

Theorem 2. *Let $\Sigma_1, \Sigma_2 > 0$ be positive difinite matrices, with $\Sigma = \Sigma_1 + \Sigma_2 > 0$ and $\Sigma_3^{-1} = \Sigma_1^{-1} + \Sigma_2^{-1} > 0$. Let Λ_1, Λ_2 be lattices such that $\sqrt{\Sigma_1} \geq \eta_\epsilon(\Lambda_1)$ and $\sqrt{\Sigma_3} \geq \eta_\epsilon(\Lambda_2)$ for some positive $\epsilon \leq 1/2$, and let $\boldsymbol{c}_1, \boldsymbol{c}_2 \in \mathbb{R}^n$ be arbitrary. Consider the following probabilistic experiment:*

$$\text{Choose } \boldsymbol{x}_2 \leftarrow D_{\Lambda_2 + \boldsymbol{c}_2, \sqrt{\Sigma_2}}, \text{then choose } \boldsymbol{x}_1 \leftarrow \boldsymbol{x}_2 + D_{\Lambda_1 + \boldsymbol{c}_1 - \boldsymbol{x}_2, \sqrt{\Lambda_1}}.$$

The marginal distribution of $\boldsymbol{x}_1$ is within statistical distance 8ϵ of $D_{\Lambda_1 + \boldsymbol{c}_1, \sqrt{\Lambda}}$. In addition, for any $\bar{\boldsymbol{x}}_1 \in \Lambda_1 + \boldsymbol{c}_1$, the conditional distribution of $\bar{\boldsymbol{x}}_2 \in \Lambda_2 + \boldsymbol{c}_2$ given $\boldsymbol{x}_1 = \bar{\boldsymbol{x}}_1$ is within statistical distance 2ϵ of $\boldsymbol{c}_3 + D_{\Lambda_2 + \boldsymbol{c}_2 - \boldsymbol{c}_3, \sqrt{\Sigma_3}}$, where $\Sigma_3^{-1} = \Sigma_1^{-1} \bar{\boldsymbol{x}}_1$

Micciancio and Walter [39] adapt the above theorem to get a generic integer sampler, which can generate a relatively small number of samples coming from a discrete Gaussian distribution for a fixed small deviation, and then recombine them into arbitrary standard deviation s and center c. [38] constructed a gadget trapdoor, which is divided into two phases: an offline phase and an online phase. The perturbation vector is obtained by sampling in the offline phase, and the target vector is obtained by sampling in the online phase. The convolution theorem ensures that the target vector does not reveal the key or trapdoor information. Recently, the gadget trapdoor has attracted great interest due to its simplicity and parallelization. For example, [22] developed a new algorithm for the off-line perturbation sampling phase of [38] in the ring setting. Their algorithm is based on a variant of the Fast Fourier Orthogonalization (FFO) algorithm (we will talk about in the next section), but avoids the need to pre-compute and store the FFO matrix; Bert et al. [4] presented the first efficient implementation of a lattice-based signature scheme in the standard model by generalizing the work of [38] and [22]; to avoid using floating-number arithmetic in the perturbation sampling, [50] exploit an integral matrix decomposition which is inspired by [44]. Recently, Karabulut et al. [29] significantly optimize CDT sampling, combining convolution techniques and fusion tree searching, resulting in smaller table size and comparator size. And then, Verilog was used to achieve constant time.

4.3.3 Fast-Fourier Sampler

The fast Fourier sampler was first proposed by Ducas and Prest [17]. The sampler, which combines the quality of Klein's algorithm and the efficiency of Peikert's algorithm, improves the nearest plane algorithm through fast Fourier transformation and can be implemented on the NTRU lattice. Falcon [21] is the result

of combining GPV framework, NTRU lattices, and Fast Fourier sampling. Falcon was accepted in NIST third-round submission, and it has the smallest total size among all the post-quantum digital signature schemes at the same security level. The Fast Fourier sampling was employed by Falcon for the phase of key generation. Algorithm 7 shows the detailed process of Falcon's fast Fourier sampling.

Algorithm 7. ffSampling$_n$(**t**,**T**)

Require: $\mathbf{t} = (t_0, t_1) \in \text{FFT}(\mathbb{Q}[x]/(x^n+1)^2)$,a FALCON tree T. All polynomials are in FFT representation
Ensure: $\mathbf{z} = (z_0, z_1) \in \text{FFT}(\mathbb{Z}[x]/(x^n+1)^2)$
if $n == 1$ **then**
  $\sigma' \leftarrow \mathbf{T}$.value ▷ It is always the case that $\sigma' \in [\sigma_{\min}, \sigma_{\max}]$
  $z_0 \leftarrow \text{SamplerZ}(t_0, \sigma')$ ▷ Since $n = 1, t_i = \text{invFFT}(t_i) \in \mathbb{Q}$ and $z_i = \text{invFFT}(z_i) \in \mathbb{Z}$
  $z_1 \leftarrow \text{SamplerZ}(t_1, \sigma')$
  return $\mathbf{z} = (z_0, z_1)$
end if
$(\ell, \mathbf{T}_0, \mathbf{T}_1) \leftarrow$ (**T**.value,**T**.leftchild,**T**.rightchild)
$\mathbf{t}_1 \leftarrow \text{splitfft}(t_1)$ ▷ $t_0, t_1 \in \text{FFT}\left(\mathbb{Q}[x]/(x^{n/2}+1)\right)^2$
$\mathbf{z}_1 \leftarrow \text{ffSampling}_{n/2}(t_1, T_1)$ ▷ First recursive call to ffSampling$_{n/2}$
$z_1 \leftarrow \text{mergefft}(\mathbf{z}_1)$ ▷ $\mathbf{z}_0, \mathbf{z}_1 \in \text{FFT}\left(\mathbb{Z}[x]/(x^{n/2}+1)\right)^2$
$t_0' \leftarrow t_0 + (t_1 - z_1) \bigodot \ell$
$\mathbf{t}_0 \leftarrow \text{splitfft}(t_0')$
$\mathbf{z}_0 \leftarrow \text{ffSampling}_{n/2}(\mathbf{t}_0, \mathbf{T}_0)$ ▷ Second recursive call to ffSampling$_{n/2}$
$z_0 \leftarrow \text{mergefft}(\mathbf{z}_0)$
return $\mathbf{z} = (z_0, z_1)$

When sampling a short vector, ffSampling is invoked twice for each leaf of the LDL* tree. Each invocation should produce an integer value from a Gaussian distribution that is centered on a value μ and with a standard deviation σ. In Algorithm 7, splitfft is a subroutine of the inverse fast Fourier transform, more precisely the part which from FFT(f) computes two FFT's twice smaller. mergefft is a step of the fast Fourier transform: it is the reconstruction step that from two small FFT's computes a larger FFT.

Note that the sampler execution time in the original Falcon is not constant, so it is hard to protect against the timing and side-channel attacks. We can see that in Algorithm 7 because samplerZ determines the security of sampling, Howe et al. [27] propose a constant-time samplerZ algorithm that uses a CDT sampler and Bernoulli rejection sampling.

5 Conclusion

Discrete Gaussian sampling plays an essential role in lattice-based cryptography. Many works in recent years have focused more on constant-time implementations

and generic constructions. On the one hand, discrete Gaussian sampling needs to comply with NIST's standards for resisting side-channel attacks if it is used in lattice cryptographic schemes. On the other hand, the variable-parameter sampler can be applied to more schemes since it is easy to adjust and the efficiency will not decrease too much. This paper is devoted to summarizing the progress of discrete Gaussian samplers in measures, efficiency, generality, and security. And how to make discrete Gaussian samples more efficient and practical is still an open problem.

References

1. Aguilar-Melchor, C., Ricosset, T.: Cdt-based gaussian sampling: From multi to double precision. IEEE Trans. Comput. **67**, 1610–1621 (2018)
2. Bai, S., Lepoint, T., Roux-Langlois, A., Sakzad, A., Stehlé, D., Steinfeld, R.: Improved security proofs in lattice-based cryptography: using the rényi divergence rather than the statistical distance. J. Cryptol. **31**, 610–640 (2017)
3. Barthe, G., Belaïd, S., Espitau, T., Fouque, P.A., Rossi, M., Tibouchi, M.: Galactics: Gaussian sampling for lattice-based constant- time implementation of cryptographic signatures, revisited. In: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security (2019)
4. Bert, P., Eberhart, G., Prabel, L., Roux-Langlois, A., Sabt, M.: Implementation of lattice trapdoors on modules and applications. In: Cheon, J.H., Tillich, J.-P. (eds.) PQCrypto 2021 2021. LNCS, vol. 12841, pp. 195–214. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-81293-5_11
5. Brannigan, S., O'Neill, M., Khalid, A., Rafferty, C.: A secure algorithm for rounded gaussian sampling. In: Krenn, S., Shulman, H., Vaudenay, S. (eds.) CANS 2020. LNCS, vol. 12579, pp. 593–612. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-65411-5_29
6. Brannigan, S., et al.: An investigation of sources of randomness within discrete gaussian sampling. IACR Cryptol. ePrint Arch. **2017**, 298 (2017)
7. Brannigan, S.: Secure Gaussian sampling for lattice-based signatures: new directions for reaching high standard deviation. Ph.D. thesis, Queen's University Belfast (2021)
8. Buchmann, J.A., Cabarcas, D., Göpfert, F., Hülsing, A., Weiden, P.: Discrete ziggurat: a time-memory trade-off for sampling from a gaussian distribution over the integers. IACR Cryptol. ePrint Arch. **2013**, 510 (2013)
9. Devroye, L.: Non-uniform random variate generation (1986)
10. Du, C., Bai, G.: Towards efficient discrete gaussian sampling for lattice-based cryptography. In: 2015 25th International Conference on Field Programmable Logic and Applications (FPL), pp. 1–6 (2015)
11. Du, Y., Fan, B., Wei, B.: A constant-time sampling algorithm for binary gaussian distribution over the integers. Inf. Process. Lett. **176**, 106246 (2022)
12. Du, Y., Fan, B., Wei, B.: An improved exact sampling algorithm for the standard normal distribution. Comput. Stat. **37**, 721–737 (2022)
13. Du, Y., Wei, B., Zhang, H.: A rejection sampling algorithm for off-centered discrete gaussian distributions over the integers. Sci. China Inf. Sci. **62**, 1–3 (2017)
14. Ducas, L., Durmus, A., Lepoint, T., Lyubashevsky, V.: Lattice signatures and bimodal gaussians. IACR Cryptol. ePrint Arch. **2013**, 383 (2013)

15. Ducas, L., Galbraith, S., Prest, T., Yu, Y.: Integral matrix gram root and lattice gaussian sampling without floats. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020. LNCS, vol. 12106, pp. 608–637. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45724-2_21
16. Ducas, L., Nguyen, P.Q.: Faster gaussian lattice sampling using lazy floating-point arithmetic. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 415–432. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-34961-4_26
17. Ducas, L., Prest, T.: Fast fourier orthogonalization. In: Proceedings of the ACM on International Symposium on Symbolic and Algebraic Computation (2015)
18. Dwarakanath, N.C., Galbraith, S.D.: Sampling from discrete gaussians for lattice-based cryptography on a constrained device. Appl. Algebra Eng. Commun. Comput. **25**, 159–180 (2014)
19. Espitau, T., et al.: MITAKA: a simpler, parallelizable, maskable variant of FALCON. In: Dunkelman, O., Dziembowski, S. (eds.) EUROCRYPT 2022. LNCS, vol. 13277, pp. 222–253. Springer, Cham (2021). https://doi.org/10.1007/978-3-031-07082-2_9
20. Folláth, J.: Gaussian sampling in lattice based cryptography. Tatra Mount. Math. Publ. **60**, 1–23 (2014)
21. Fouque, P.A., et al.: Falcon: Fast-Fourier lattice-based compact signatures over NTRU (2019)
22. Genise, N., Micciancio, D.: Faster gaussian sampling for trapdoor lattices with arbitrary modulus. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018. LNCS, vol. 10820, pp. 174–203. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-78381-9_7
23. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: Proceedings of the fortieth Annual ACM Symposium on Theory of Computing, pp. 197–206 (2008)
24. Gür, K.D., Polyakov, Y., Rohloff, K.R., Ryan, G.W., Savaş, E.: Implementation and evaluation of improved gaussian sampling for lattice trapdoors. In: Proceedings of the 6th Workshop on Encrypted Computing & Applied Homomorphic Cryptography (2017)
25. Howe, J., Khalid, A., Rafferty, C., Regazzoni, F., O'Neill, M.: On practical discrete gaussian samplers for lattice-based cryptography. IEEE Trans. Comput. **67**, 322–334 (2018)
26. Howe, J., O'Neill, M.: GLITCH: a discrete gaussian testing suite for lattice-based cryptography. In: SECRYPT (2017)
27. Howe, J., Prest, T., Ricosset, T., Rossi, M.: Isochronous gaussian sampling: from inception to implementation. In: Ding, J., Tillich, J.-P. (eds.) PQCrypto 2020. LNCS, vol. 12100, pp. 53–71. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-44223-1_4
28. Hülsing, A., Lange, T., Smeets, K.: Rounded gaussians - fast and secure constant-time sampling for lattice-based crypto. IACR Cryptol. ePrint Arch. **2017**, 1025 (2017)
29. Karabulut, E., Alkim, E., Aysu, A.: Efficient, flexible, and constant-time gaussian sampling hardware for lattice cryptography. IEEE Trans. Comput. **71**, 1810–1823 (2022)
30. Karmakar, A., Roy, S.S., Reparaz, O., Vercauteren, F., Verbauwhede, I.M.R.: Constant-time discrete gaussian sampling. IEEE Trans. Comput. **67**, 1561–1571 (2018)

31. Karmakar, A., Roy, S.S., Vercauteren, F., Verbauwhede, I.M.R.: Pushing the speed limit of constant-time discrete gaussian sampling. a case study on the falcon signature scheme. In: 2019 56th ACM/IEEE Design Automation Conference (DAC), pp. 1–6 (2019)
32. Karney, C.F.F.: Sampling exactly from the normal distribution. ACM Trans. Math. Softw. (TOMS) **42**, 1–14 (2016)
33. Klein, P.N.: Finding the closest lattice vector when it's unusually close. In: SODA 2000 (2000)
34. Lyubashevsky, V.: Lattice signatures without trapdoors. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 738–755. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-29011-4_43
35. Lyubashevsky, V., Prest, T.: Quadratic time, linear space algorithms for gram-schmidt orthogonalization and gaussian sampling in structured lattices. IACR Cryptol. ePrint Arch. **2015**, 257 (2015)
36. Aguilar-Melchor, C., Albrecht, M.R., Ricosset, T.: Sampling from arbitrary centered discrete gaussians for lattice-based cryptography. In: Gollmann, D., Miyaji, A., Kikuchi, H. (eds.) ACNS 2017. LNCS, vol. 10355, pp. 3–19. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-61204-1_1
37. Micciancio, D., Regev, O.: Worst-case to average-case reductions based on gaussian measures. In: 45th Annual IEEE Symposium on Foundations of Computer Science, pp. 372–381 (2004)
38. Micciancio, D., Peikert, C.: Trapdoors for lattices: simpler, tighter, faster, smaller. IACR Cryptol. ePrint Arch. **2011**, 501 (2011)
39. Micciancio, D., Walter, M.: Gaussian sampling over the integers: efficient, generic, constant-time. IACR Cryptol. ePrint Arch. **2017**, 259 (2017)
40. More, S., Katti, R.S.: Discrete gaussian sampling for low-power devices. In: 2015 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM), pp. 181–186 (2015)
41. Peikert, C.: An efficient and parallel gaussian sampler for lattices. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 80–97. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14623-7_5
42. Pöppelmann, T., Ducas, L., Güneysu, T.: Enhanced lattice-based signatures on reconfigurable hardware. IACR Cryptol. ePrint Arch. **2014**, 254 (2014)
43. Prest, T., Ricosset, T., Rossi, M.: Simple, fast and constant-time gaussian sampling over the integers for falcon (2019)
44. Roy, S.S., Reparaz, O., Vercauteren, F., Verbauwhede, I.M.R.: Compact and side channel secure discrete gaussian sampling. IACR Cryptol. ePrint Arch. **2014**, 591 (2014)
45. Sinha Roy, S., Vercauteren, F., Verbauwhede, I.: High precision discrete gaussian sampling on FPGAs. In: Lange, T., Lauter, K., Lisoněk, P. (eds.) SAC 2013. LNCS, vol. 8282, pp. 383–401. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-43414-7_19
46. Sun, S., Zhou, Y., Ji, Y.S., Zhang, R., Tao, Y.: Generic, efficient and isochronous gaussian sampling over the integers. Cybersecurity **5**, 1–22 (2021)
47. Walter, M.: Sampling the integers with low relative error. In: Buchmann, J., Nitaj, A., Rachidi, T. (eds.) AFRICACRYPT 2019. LNCS, vol. 11627, pp. 157–180. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-23696-0_9
48. Weiden, P., Hülsing, A., Cabarcas, D., Buchmann, J.A.: Instantiating treeless signature schemes. IACR Cryptol. ePrint Arch. **2013**, 65 (2013)
49. Xie, S., Zhuang, S., Du, Y.: Improved bernoulli sampling for discrete gaussian distributions over the integers. Mathematics **9**, 378 (2021)

50. Zhang, S., Yu, Y.: Towards a simpler lattice gadget toolkit. In: Hanaoka, G., Shikata, J., Watanabe, Y. (eds.) PKC 2022. LNCS, vol. 13177, pp. 498–520. Springer, Cham (2021)
51. Zhao, K.: Efficient Implementation Techniques for Lattice-based Cryptosystems. Ph.D. thesis, Monash University (2022)
52. Zhao, R.K., Steinfeld, R., Sakzad, A.: COSAC: COmpact and scalable arbitrary-centered discrete gaussian sampling over integers. In: Ding, J., Tillich, J.-P. (eds.) PQCrypto 2020. LNCS, vol. 12100, pp. 284–303. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-44223-1_16
53. Zhao, R.K., Steinfeld, R., Sakzad, A.: FACCT: fast, compact, and constant-time discrete Gaussian sampler over integers. IEEE Trans. Comput. **69**, 126–137 (2020)