SCIENCE CHINA
Information Sciences

• LETTER •

March 2019, Vol. 62 039103:1–039103:3
https://doi.org/10.1007/s11432-017-9409-7

A rejection sampling algorithm for off-centered discrete Gaussian distributions over the integers

Yusong DU^{1,2*}, Baodian WEI^{1,2,3} & Huang ZHANG^{1}

^{1}*School of Data and Computer Science, Sun Yat-sen University, Guangzhou* 510006*, China;*
^{2}*Guangdong Key Laboratory of Information Security Technology, Guangzhou* 510006*, China;*
^{3}*Chongqing Key Lab of Computer Network and Communication Technology, Chongqing* 400065*, China*

Received 22 October 2017/Revised 25 January 2018/Accepted 28 February 2018/Published online 11 September 2018

Citation Du Y S, Wei B D, Zhang H. A rejection sampling algorithm for off-centered discrete Gaussian distributions over the integers. Sci China Inf Sci, 2019, 62(3): 039103, https://doi.org/10.1007/s11432-017-9409-7

Dear editor,
Discrete Gaussian sampling, that is, sampling from a discrete Gaussian distribution $D_{\Lambda,\sigma,\boldsymbol{c}}$ with parameter $\sigma > 0$ and center $\boldsymbol{c} \in \mathbb{R}^n$ over an n-dimensional lattice Λ, has been considered by the cryptography research community as one of the fundamental building blocks of lattice-based cryptography [1, 2]. The simplest lattice is the one-dimensional integer lattice $\mathbb{Z}$. Sampling from a discrete Gaussian distribution $D_{\mathbb{Z},\sigma,c}$ over the integers $\mathbb{Z}$, denoted by Sample$\mathbb{Z}$, is an important sub-problem of discrete Gaussian sampling, where parameter $\sigma > 0$ and center $c \in \mathbb{R}$. The first Sample$\mathbb{Z}$ algorithm, which uses rejection sampling and supports a varying center c, was given by Gentry et al. in [1]. This algorithm is not very efficient, because it requires at least about 10 trials on average before providing an integer as the output. Since many lattice-based cryptosystems only involve sampling from centered discrete Gaussian distributions (center $c = 0$), most of the improved Sample$\mathbb{Z}$ algorithms were designed only for centered discrete Gaussian distributions, such as [3,4]. However, they cannot be used in a sampling algorithm for Gaussian distributions over a general lattice, because sampling from $D_{\mathbb{Z},\sigma,c}$ with a varying center c is usually required as one of the kernel subroutines in a sampling algorithm, for Gaussian distributions over a general n-dimensional lattice Λ. Thus, it is interesting to design more efficient Sample$\mathbb{Z}$ algorithms for off-centered discrete Gaussian distributions over the integers, which support arbitrary and varying centers.

In this study, we propose an alternative rejection sampling algorithm for $D_{\mathbb{Z},\sigma,c}$, that is much more efficient than the widely used Sample$\mathbb{Z}$ algorithm proposed by Gentry et al. in [1]. With the support of double-precision floating-point arithmetic, this algorithm allows σ and c to be double-precision floating-point numbers, and it does not require any precomputation storage.

Discrete Gaussian distribution. We denote the set of real numbers by $\mathbb{R}$, the set of integers by $\mathbb{Z}$, and the set of non-negative integers by $\mathbb{Z}^+$, respectively. The Gaussian function on $\mathbb{R}$ with the parameter $\sigma > 0$, center $c \in \mathbb{R}$, and evaluated at $x \in \mathbb{R}$, is defined by $\rho_{\sigma,c}(x) = \exp\left(-(x-c)^2/2\sigma^2\right)$. For $\sigma > 0$ and $c \in \mathbb{R}$, the discrete Gaussian distribution over the integers $\mathbb{Z}$ is defined by $D_{\mathbb{Z},\sigma,c} = \rho_{\sigma,c}(x)/\rho_{\sigma,c}(\mathbb{Z})$, for $x \in \mathbb{Z}$.

Our algorithm. Sampling from $D_{\mathbb{Z},\sigma,c}$ is equivalent to sampling from $D_{\mathbb{Z},\sigma,\{c\}} + [c]$, where $[c]$ and $\{c\}$ are the integer and fractional parts of c, respectively, such that $0 \leqslant \{c\} < 1$. Furthermore, for any real $c \in [1/2, 1)$, sampling from $D_{\mathbb{Z},\sigma,c}$ is equivalent to sampling from $1 - D_{\mathbb{Z},\sigma,c'}$, where $c' \in (0, 1/2]$ such that $c = 1 - c'$. It suffices for us to observe sampling $D_{\mathbb{Z},\sigma,c}$ with $c \in (0, 1/2]$. Hence,

* Corresponding author (email: duyusong@mail.sysu.edu.cn)

our sampling algorithm is designed for $c \in (0, 1/2]$.

Algorithm 1 Off-centered Gaussian sampler over the integers

```
Sampling from $D_{\mathbb{Z},\sigma,c}$ with $\sigma > \sigma_2 = \sqrt{1/(2 \cdot \ln 2)}$, and $c \in (0, 1/2]$
Input: Double-precision numbers $\sigma$ and $c$
Output: An integer $z$ according to $D_{\mathbb{Z},\sigma,c}$
1: Set $q \leftarrow \sigma/\sigma_2$ with $\sigma_2 = \sqrt{1/(2 \cdot \ln 2)}$;
2: Sample $x \in \mathbb{Z}$ according to $D_{\mathbb{Z}^+,\sigma_2}$ and $y \in \mathbb{Z}$ uniformly in $\{0, 1, 2, \ldots, \lceil q \rceil - 1\}$;
3: Set $s \leftarrow \pm 1$ with equal probabilities;
4: Set $\delta \leftarrow \lceil xq + sc \rceil - xq - sc$ and goto Step 2 if $y + \delta \geqslant q$;
5: Set $z \leftarrow \lceil xq + sc \rceil + y$ and accept it with probability $\exp(-\frac{2xq(y+\delta)+(y+\delta)^2}{2q^2\sigma_2^2})$, otherwise goto Step 2;
6: return $s \cdot z$
```

Our algorithm can be viewed as an 'off-centralized' version of the sampling algorithm proposed by Ducas et al. [3]. It is also combined with the techniques that were used in Karney's sampling algorithm [5].

It can be proved that any possible output $z \in \mathbb{Z}$ can be uniquely written as $z = s(\lceil xq + sc \rceil + y)$ with $x \geqslant 0$ and $y \in [0, \lceil q \rceil)$, such that $\lceil xq + sc \rceil - xq - sc + y < q$. Meanwhile, we have

$$\rho_{\sigma_2}(x) \cdot \exp\left(-\frac{2xq(y+\delta)+(y+\delta)^2}{2q^2\sigma_2^2}\right) = \rho_{q\sigma_2,c}(s(\lceil xq + sc \rceil + y)) = \rho_{\sigma,c}(sz).$$

This implies that the returned value of $s \cdot z$ has the desired (relative) density $\rho_{\sigma,c}(sz)$.

The expected number of rejections that our algorithm needs, which is a vital complexity factor for a rejection sampling algorithm, can be estimated as follows. The expected number of rejections caused by Step 5 in Algorithm 1 has an upper-bound given by

$$\frac{2\lceil q \rceil \rho_{\sigma_2}(\mathbb{Z}^+)}{\rho_{q\sigma_2,c}(\mathbb{Z})} \leqslant \frac{\lceil q \rceil \rho_{\sigma_2}(\mathbb{Z}^+)}{q\sigma_2\sqrt{\pi/2} - 1}.$$

With the fact that the probability of $\lceil xq + sc \rceil - xq - sc + y \geqslant q$ is at most $1/\lceil q \rceil$, we have that the expected number of rejections Algorithm 1 needs in total has an upper-bound given by $(1/\tau) \cdot (\lceil q \rceil \rho_{\sigma_2}(\mathbb{Z}^+))/(q\sigma_2\sqrt{\pi/2} - 1)$, with $\tau = (1 - 1/\lceil q \rceil)$. Obviously, the bound is about $\rho_{\sigma_2}(\mathbb{Z}^+)/\sigma_2\sqrt{\pi/2} \approx 1.47$, for a large q. As a conclusion, we have the following theorem (see Appendix A for its complete proof).

Theorem 1. Algorithm 1 outputs an integer z according to a Gaussian distribution $D_{\mathbb{Z},\sigma,c}$, with $\sigma > \sigma_2$ and $c \in (0, 1/2]$. The expected number of rejections that the algorithm needs has an upper-bound given by $(1/\tau) \cdot (\lceil q \rceil \rho_{\sigma_2}(\mathbb{Z}^+))/(q\sigma_2\sqrt{\pi/2} - 1)$, with $\tau = (1 - 1/\lceil q \rceil)$.

Discussion. We remark that the algorithm also works for $c = 0$. There are two cases where $z = 0$ in the algorithm, when $c = 0$, i.e., $(x, y, s) = (0, 0, 1)$, and $(x, y, s) = (0, 0, -1)$. Therefore, we need to add one more step to the algorithm that is designed to avoid double counting 0 before it returns the result. This step can be described as '**goto** Step 2 if $c = 0$, $z = 0$ and $s = 1$'.

We take $\sigma_2 = \sqrt{1/(2 \cdot \ln 2)}$, because sampling $D_{\mathbb{Z}^+,\sigma_2}$ in this case can be very fast and it requires neither floating-point arithmetic nor precomputation storage (see Algorithm 10 in [3]). Our algorithm requires floating-point arithmetic mainly because of computing $\exp\left(-(2xq(y+\delta)+(y+\delta)^2)/2q^2\sigma_2^2\right)$. It allows σ and c to be double-precision floating-point numbers and does not require any precomputation storage.

Recent results on discrete Gaussian sampling precision suggested that the significand precision of 53 bits provided by double-precision floating arithmetic is sufficient for most of the security applications [2, 6]. Thus, our algorithm is only designed and implemented for double-precision floating-point parameters. One could also design an adapted algorithm for higher precision using the lazy floating-point arithmetic [7].

Experimental results. We compare our algorithm with other Sample$\mathbb{Z}$ algorithms that support varying centers. The experimental results show that using our algorithm, one can get about 10.5×10^6 samples per second from a discrete Gaussian distribution $D_{\mathbb{Z},\sigma,c}$, with σ ranging roughly from 4 to 2^{20}, and c picked uniformly from $[0, 1)$. Our algorithm has a significant performance advantage over the sampling algorithm proposed by Gentry et al. in [1]. Karney's algorithm [5] is also useful for varying centers, but it only accepts rational numbers in the form of p/q, with positive integers p and q. Our algorithm allows varying double-precision number c. Moreover, using Karney's algorithm one can get only about 4.6×10^6 samples per second. In addition, there will be a degradation in the performance, when $\sigma \geqslant 2^{18}$ for Karney's algorithm, while a large σ has no impact on the sampling efficiency of our algorithm. The sampling algorithms proposed by Micciancio et al. [2] and by Aguilar-Melchor et al. [8] can be used to sample from distributions with arbitrary and varying centers, but they rely on a large amount of precomputation storage or offline computation. In particular, Micciancio et al. [2] also suggested splitting their algorithm into online and offline phases, and they assumed that the offline phase is free (by using idle times, parallel devices, FPGAs or GPUs). In this case, its performance approaches 10.0×10^6, according to their experimental data. In fact, Algorithm 1 can also be split into online and offline phases, and we can also get a performance boost if the offline phase is free.

Strategies to realize time independence. Finally, we show that Algorithm 1 can be adapted for meeting the requirement of time independence, with only modest performance penalties. The side-channel leakage of discrete Gaussian sampling algorithms has been recognized as an important problem. For example, Bruinderink et al. [9] presented timing attacks on the sampling algorithms used in BLISS signature scheme. Karney's algorithm may also suffer from the same problem [2]. The implementations of discrete Gaussian sampling algorithms are also required by NIST (National Institute of Standards and Technology) to have resistance against side-channel attacks.

We assume that the double precision floating-point arithmetic does not leak the values of the operands. In Algorithm 1, the exponential function $\exp\left(-(2xq(y+\delta)+(y+\delta)^2)/2q^2\sigma_2^2\right)$, is computed by directly using (double precision) floating-point arithmetic, and thus it will not cause information leakage. However, it seems to be difficult for us to give a time independent implementation of discrete Gaussian distribution $D_{\mathbb{Z}^+,\sigma_2}$. Therefore, we need to mitigate the information leakage due to the implementation of $D_{\mathbb{Z}^+,\sigma_2}$.

Note that the samples from $D_{\mathbb{Z}^+,\sigma_2}$ (the values of x) can be generated in the offline phase within Algorithm 1, as the sampler for $D_{\mathbb{Z}^+,\sigma_2}$ is fixed and does not depend on the input of Algorithm 1. In this case, for a fixed sampler, Micciancio et al. [2] suggested generating the samples in large batches in the offline phase. In Algorithm 1, we could use this strategy simply by adding a minor performance penalty. Thereafter, one can look at the time required to generate a batch of samples from $D_{\mathbb{Z}^+,\sigma_2}$. Their aggregate generation time is sharply concentrated around the expectation, and can be made constant except with negligible probability.

The limitation of the mitigation strategy suggested by Micciancio et al. [2] is that it requires a cache of a large size to store a large batch of samples. We can show that randomly shuffling small batches of samples from $D_{\mathbb{Z}^+,\sigma_2}$ also works for mitigating the information leakage. In practice, we choose $d = 56$ and use Fisher-Yates algorithm to randomly shuffle every 56 samples from $D_{\mathbb{Z}^+,\sigma_2}$, so that the information leakage is invalid (see Appendix B for the detailed analysis). Applying this mitigation strategy, we evaluated that the performance decreases to about 9.2×10^6 samples per second in this case.

More information. The complete proof of Theorem 1 and the detailed analysis of the strategies to realize time independence can be found in Appendixes A and B, respectively. Appendix C provides a figure that shows the performance of Algorithm 1 compared with Gentry's and Karney's sampling algorithms.

Acknowledgements This work was supported by National Key Research and Development Program of China (Grant No. 2017YFB0802500), Science and Technology Planning Project of Guangdong Province (Grant No. 2014A010103017), Natural Science Foundation of Guangdong Province (Grant No. 2016A030313298), Fundamental Research Funds for the Central Universities (Grant No. 17lgjc45) and Opening Fund of Qiongqing Key Lab of Computer Network and Communication Technology (Grant No. CY-CNCL-2017-04).

Supporting information Appendixes A–C. The supporting information is available online at info.scichina.com and link.springer.com. The supporting materials are published as submitted, without typesetting or editing. The responsibility for scientific accuracy and content remains entirely with the authors.

References

1 Gentry C, Peikert C, Vaikuntanathan V. Trapdoors for hard lattices and new cryptographic constructions. In: Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, 2008. 197–206

2 Micciancio D, Walter M. Gaussian sampling over the integers: efficient, generic, constant-time. In: Proceedings of Annual International Cryptology Conference, Santa Barbara, 2017. 10402: 455–485

3 Ducas L, Durmus A, Lepoint T, et al. Lattice signatures and bimodal Gaussians. In: Proceedings of Annual Cryptology Conference, Santa Barbara, 2013. 8042: 40–56

4 Saarinen M J O. Arithmetic coding and blinding countermeasures for lattice signatures. J Cryptogr Eng, 2018, 8: 71–84

5 Karney C. Sampling exactly from the normal distribution. ACM Trans Math Softw, 2016, 42: 1–14

6 Prest T. Sharper bounds in lattice-based cryptography using the Rényi divergence. In: Proceedings of International Conference on the Theory and Application of Cryptology and Information Security, Hong Kong, 2017. 10624: 347–374

7 Ducas L, Nguyen P Q. Faster Gaussian lattice sampling using lazy floating-point arithmetic. In: Proceedings of International Conference on the Theory and Application of Cryptology and Information Security, Beijing, 2012. 7658: 415–432

8 Aguilar-Melchor C, Albrecht M R, Ricosset T. Sampling from arbitrary centered discrete Gaussians for lattice-based cryptography. In: Proceedings of International Conference on Applied Cryptography and Network Security, Kanazawa, 2017. 10355: 3–19

9 Bruinderink L G, Hülsing A, Lange T, et al. Flush, Gauss, and reload - a cache attack on the BLISS lattice-based signature scheme. In: Proceedings of International Conference on Cryptographic Hardware and Embedded Systems, Santa Barbara, 2016. 9813: 323–345