DE GRUYTER
OPEN

TATRA
MOUNTAINS
Mathematical Publications

DOI: 10.2478/tmmp-2014-0022
Tatra Mt. Math. Publ. **60** (2014), 1–23

GAUSSIAN SAMPLING IN LATTICE BASED CRYPTOGRAPHY

János Folláth

ABSTRACT. Modern lattice-based cryptosystems require sampling from discrete Gaussian distributions. We review lattice based schemes and collect their requirements for sampling from discrete Gaussians. Then we survey the algorithms implementing such sampling and assess their practical performance. Finally we draw some conclusions regarding the best candidates for implementation on different platforms in the typical parameter range.

1. Introduction

Lattice based cryptography began with the seminal work of Ajtai, who built a one-way function based on the worst case hardness based on certain lattice problems [1]. These lattice problems are believed to be hard even in the presence of large quantum computers and such a promising post-quantum replacement for standard cryptography. The most general public key primitives like encryption schemes [28] and digital signatures [27] already have practical lattice based instantiations.

Many recent lattice based schemes require sampling from discrete Gaussians (for example, see [5], [12], [15], [25], [27], [30], [36], [38]). The parameters of discrete Gaussians are governed by the security proofs of the particular schemes. A finite machine cannot sample from a discrete Gaussian distribution, hence one has to sample from a distribution close to it. It is a common practice to require that the statistical distance of the sampled distribution from the desired discrete Gaussian be less than 2^{100}.

2010 Mathematics Subject Classification: 94A60, 65C10.
Keywords: lattice-based cryptography, Gaussian sampling, practicality, Knuth-Yao, Ziggurat, BLISS, learning with errors, Lyubashevsky signature.
This research was supported by the TÁMOP-4.2.2.C-11/1/KONV-2012-0001 "Future Internet Research, Services and Technology" project supported by the European Union, co-financed by the European Social Fund.

Computing the probabilities requires floating point operations of at least 100 bit precision if one wants to achieve a statistical distance less than 2^{100}. Whereas any precomputation means storing a variable amount of values of the same precision. This can highly affect the sampling performance on personal computers and even make the implementation completely impractical on constrained devices. W e i d e n et al. [43] report that the Gaussian sampling takes up 50 % of the running time of L y u b a s h e v s k y' s signature scheme [27]. Thus efficient sampling from discrete Gaussians plays a crucial role in the performance of these primitives.

As in [14] by a constrained device we will think of an embedded or portable device with a small amount of memory (measured in kilobytes instead of gigabytes) and a modest processor that has to be economical with respect to power usage. Also these kind of devices not necessarily come with floating point arithmetic capability. Even if a platform provides floating point arithmetic, the required precision is usually not supported natively. This means that software libraries have to be used for this functionality, and these have significantly worse performance and take up additional space in the already tight memory.

The particular discrete Gaussian samplers apply different techniques to increase the performance and reduce or avoid the floating point operations, which usually utilize precomputed tables (with the notable exception of Algorithm 6 (see also [21, Algorithm D]), requiring neither floating point arithmetic nor precomputed tables). Many factors can affect the performance and memory consumption (i.e., the size and number of the potential precomputed tables). Such factors are the size of the Gaussian parameter, whether the center is zero or not, and whether the parameters are fixed or changing (or more precisely, the number of the needed parameter combinations). To evaluate the practicality of the discrete Gaussian samplers in lattice based cryptography one needs to assess the parameters of the distributions required by the different cryptographic schemes.

The techniques utilized by different samplers require various amount of memory and floating point operations, which result in different overall performance on the particular platforms. Thus for the evaluation of their practical performance one needs to collect the characteristics of the discrete Gaussian samplers too.

In Section 2 we will give the basic definitions and main issues regarding discrete Gaussians. Section 3 contains the overview of the lattice based cryptographic schemes using discrete Gaussians at some point. Section 4 is about the known methods for sampling from discrete Gaussians and in Section 5 there is a brief summary of the information gathered and the resulting conclusions.

2. Preliminaries

In this section we overview the basic definitions and fundamental problems related to discrete Gaussian sampling.

2.1. Discrete Gaussian distribution

Definition 1 (discrete Gaussian distribution)**.** For any center $c \in \mathbb{R}$, and Gaussian parameter $s \in \mathbb{R}^+$, define the discrete Gaussian distribution as

$$D_{s,c}(x) = \frac{\rho_{s,c}(x)}{\sum_{y=-\infty}^{\infty} \rho_{s,c}(y)}, \tag{1}$$

$\forall x \in \mathbb{Z}$, where ρ denotes the Gaussian function $\rho_{s,c}(x) = e^{-\pi|x-c|^2/s^2}$.

It is worth to mention that sometimes in the literature this definition is formulated with the parameter $\sigma = s/\sqrt{2\pi}$. For the sake of uniformity we will use the Gaussian parameter s of Definition 1, to describe a discrete Gaussian distribution throughout this paper.

2.2. Statistical distance

To keep the security proofs of the cryptographic schemes in Section 3 valid, we need the actual sampled distribution to be statistically close to the theoretical discrete Gaussian.

Definition 2 (statistical distance)**.** Let X and Y be two random variables corresponding to given distributions over the integers. Then the statistical distance of their distribution is defined by

$$\Delta(X,Y) = \frac{1}{2} \sum_{x=-\infty}^{\infty} \left|\Pr[X = x] - \Pr[Y = x]\right|.$$

Since the proofs assume a perfect discrete Gaussian distribution, they do not give a well defined bound on the required statistical distance, thus we will rely on the common practice to require it to be less than $2^{-\lambda}$ with λ between 90 and 128.

Clearly no finite machine can sample exactly from the discrete Gaussian distribution, algorithms usually just sample from a finite range large enough to comply to the statistical distance requirement. To determine a safe tailcut one may use the following lemma.

Lemma 1 ([15, Lemma 3.1])**.** *For any $\epsilon > 0$, any $s \geq \eta_\epsilon(\mathbb{Z})$ and any $t > 0$, we have*

$$\Pr_{x \leftarrow D_{s,c}}[|x - c| \geq t \cdot s] \leq 2e^{-\pi t^2} \cdot \frac{1+\epsilon}{1-\epsilon}.$$

Here η denotes the smoothing parameter introduced in [31]. Applying [31, Lemma 3.2] we get $\eta_{\frac{1}{2}}(\mathbb{Z}) \leq 1$ and from Lemma 1 with simple computation we conclude that if $t > 4.72$ and $s > 1$, then the probability of the tails of $D_{s,c}$ is less than 2^{-100}.

2.3. Computing the discrete Gaussian distribution

If the sampler needs to compute the probabilities, then it has to do it at least to precision λ to have an output distribution with statistical difference less than $2^{-\lambda}$. This means that floating point precision of around 100 is required, thus the IEEE standard double-precision is not enough, consequently higher precision arithmetic (simulated by software libraries) has to be used, which are typically $10 - 20$ times slower for quad precision and even more slower for arbitrary precision [11].

It is clear from Definition 1, that computing the probabilities corresponding to $D_{s,c}$ requires the computation of the exponential function e^x. There are multiple methods to perform this task, for a brief survey we refer to [14, Subsection 4.1.] and to [8], [32], [35], [42] for the particular methods. As it is summarized in [14] all of these methods require either a large number of floating point operations or large precomputed tables. This makes computing the exponential function relatively expensive operation on most platforms and even completely impractical on devices with constrained memory and without high precision floating point arithmetic capability.

To sufficiently approximate the denominator in (1) it is enough to compute

$$A_{s,c} = \sum_{y=-ts}^{ts} \rho_{s,c}(y). \tag{2}$$

The sum is extremely close to s for large s, regardless of the value of c [14]. If $c \neq 0$ then the sum has to be computed every time the parameters change.

Some of the samplers in Section 4 are using precomputed tables that depend on the parameters of the distribution. Usually the size of these tables depends on the Gaussian parameters and the tailcut. Also if the scheme requires to sample from distributions with different center or Gaussian parameter, then a new table is required.

3. Lattice based cryptography

As it is seen in Section 2, many factors affect the actual performance and memory consumption of computing the probabilities and the size and number of the potential precomputed tables. Such factors are the size of the Gaussian parameter, whether the center is zero or not, and whether the parameters are fixed or changing (or more precisely, the number of the needed parameter combinations).

In this section we review some lattice based cryptographic schemes with the goal of assessing the requirements against the Gaussian samplers and the typical parameters of the discrete Gaussian distribution to sample.

3.1. LWE based encryption

In [39] Regev described an average-case problem called Learning With Errors (LWE) and reduced the worst-case lattice problems such as GapSVP and SIVP to it. In his work he also constructed a public-key cryptosystem based on the LWE problem. Regev's reduction is almost entirely classical, but it uses a quantum step too. In [37] Peikert removed the quantum step, making the reduction from GapSVP completely classical. Unfortunately the modulus in the LWE problem has to be exponentially large for the classical reduction to work and thus usually the results of the quantum reduction are considered when determining the security parameters of an LWE based scheme.

Also a nice property of Regev's cryptosystem was proven by Akavia, Goldwasser, and Vaikuntanathan [2], namely that it stays secure even if almost the entire secret key is leaked.

A drawback of Regev's encryption scheme is that the encrypted message is an $\mathcal{O}(n \log n)$ times longer than the plaintext. Kawachi, Tanaka, and Xagawa [22] came up with a modified version of the scheme that has a reduced penalty factor of $\mathcal{O}(n)$. Peikert, Vaikuntanathan, and Waters [38] made an even greater improvement by reducing the penalty factor to $\mathcal{O}(1)$. In the following we will use this scheme to determine some concrete requirements against the Gaussian samplers used in the implementations.

With the parameters suggested by the authors the Gaussian sampler has to produce an output distribution with $51336 \leq s \leq 102672$. The center of the distribution is a half-integer and the Gaussians are needed only from a single distribution, thus the parameters are fixed.

The scheme needs discrete Gaussians only in the key generation phase. In the case of constrained devices the key can be supplied with the device, and even with on-board key generation implemented, the efficiency of the Gaussian sampler has a limited impact on the overall performance.

Lindner and Peikert [25] also proposed an encryption scheme based on the LWE problem with much better key sizes. In their variant discrete Gaussians are required both at key generation and encryption and the Gaussian parameters are ranging from 8.35 to 13.01 (see [25, Figure 4.] in the case of instances with reasonable security. Also the Gaussian parameter is fixed and the center is always zero.

3.2. GPV signatures

One of the early proposals for Lattice based signature was the GGH scheme [18] by Goldreich, Goldwasser, and Halevi. Although it was built

on certain lattice problems directly, it lacked of a security proof and was later broken by Nguyen and Regev [33]. The scheme used a "good" basis of the lattice (one with short Gram-Schmidt vectors) as a secret key and a "bad" one (one in Hermite normal form) as a public key. The main problem was, that the signatures leaked the geometry of the secret basis and it could be determined by the attacker.

Later in [15] Gentry, Peikert and Vaikuntanathan showed a theoretically sound and secure way to use a short basis of a lattice as a trapdoor. Their construction relies on their newly defined cryptographic primitive, the so called *one-way preimage samplable trapdoor function*, which can be used in certain situations in the place of trapdoor permutations.

Definition 3 (one-way preimage samplabe trapdoor function)**.** A one-way preimage samplable trapdoor function is a tuple of probabilistic polynomial time algorithms (TrapGen, SampleDom, SamplePre) which satisfies the following:

1. Generating a function with trapdoor:
   $TrapGen(1^n)$ outputs (a, t), where a is a description of an efficiently--computable function $f_a : D_n \to R_n$ (for some efficiently-recognizable domain D_n and range R_n depending on n), and t is some trapdoor information for f_a.
2. Domain sampling with uniform output:
   $SampleDom(1^n)$ outputs x such that $f_a(x)$ is uniform over R_n.
3. Preimage sampling with trapdoor:
   For every $y \in R_n$ the distribution of $SamplePre(t, y)$ output is the conditional distribution $x \leftarrow SampleDom(1^n)$, given $f_a(x) = y$.
4. One-wayness:
   for any probabilistic poly-time algorithm $\mathcal{A}$, the probability that
   $$\mathcal{A}(1^n, a, y) \in f_a^{-1}(y) \subseteq D_n$$
   is negligible, where the probability is taken over the choice of a, the target value $y \leftarrow R_n$ chosen uniformly at random, and $\mathcal{A}$'s random coins.

The classical hash-and-sign paradigm was suggested in [10]. Later it was formalized in [6], and also it was shown, that this scheme (called Full-Domain Hash) is existentially unforgeable under chosen-message attacks when instantiated with a trapdoor permutation and the hash function is modeled as a random oracle.

In [15] the authors gave a version of the Full-Domain Hash scheme using a one-way preimage samplable trapdoor function instead of a trapdoor permutation. The security of the scheme lies on the hardness of the SIS problem. In order for the security reduction to work, the signer must give out at most one preimage of a given point.

The key element of both the preimage sampling and the trapdoor inversion algorithm is a subroutine that samples from a discrete Gaussian distribution over a lattice. The algorithm proposed in [15] was to use a randomized variant of Babai's nearest plane algorithm [4], which is equivalent to the one proposed by Klein [23] in another context. This algorithm chooses the next plane according to a discrete Gaussian distribution instead of selecting the nearest one. This is inherently sequential and it requires to sample from discrete Gaussian distributions over the integers with varying center and Gaussian parameter. Peikert [36] proposed an improved method for sampling discrete Gaussians over lattices that was not only highly parallelizeable but also required to sample from discrete Gaussian distributions over the integers with the Gaussian parameter fixed (and only q different centers when sampling from q-ary lattices).

Micciancio and Peikert [30] introduced a special kind of trapdoor, for which sampling discrete Gaussians over the lattice can be reduced to sample from (possibly non-spherical) discrete Gaussians over $\mathbb{Z}^m$ and over so called *primitive lattices*.

Sampling discrete Gaussians over $\mathbb{Z}^m$ can be done by sampling a corresponding continuous Gaussian and independently randomized rounding the coordinates to nearby integers [36, Theorem 3.1] (rounding essentially means sampling from discrete Gaussians with fixed Gaussian parameter). Since this part of the scheme can be done offline, these values can be precomputed and stored on the devices not capable of floating point arithmetic. This solution limits the number of signatures the device can sign over its lifetime though.

In the case of the trapdoor generation and sampling of [30] it proposes $n = 284$ implying a Gaussian parameter of 17 (in [30, page 25] the authors mention that $s > \sqrt{n}$ typically holds). The trapdoor generation of [3] does not seem to use discrete Gaussian distributions and according to [30, Figure 2] a security parameter of reasonable security is $n = 436$. We will accept the argument of [14] that the security parameters of [15] should be at least as big as of [3], and use the aforementioned $s > \sqrt{n}$ bound to estimate the Gaussian parameter required by the original trapdoor in [15] to be around 21. Also it is worth mentioning that in both cases these values were suggested assuming that the statistical error of the randomized rounding (i.e. discrete Gaussian sampling over the integers) is at most 2^{-90}.

3.3. Lyubashevsky signatures

Early lattice based signature schemes [18]–[20], did not have security reductions and leaked information about the secret key and thus were broken [13], [16], [34]. The signatures overviewed in Subsection 3.2 built a theoretically sound trapdoor to avoid this weakness. The signature schemes proposed by Lyubashevsky [12], [18], [26], [27] take a direct approach similar to the

early schemes, but hide the geometry of the secret basis with rejection sampling. These schemes are based on the Fiat-Shamir paradigm, namely the signature serves as a proof of knowledge of the private key. He also provided variants of this scheme and also gave security reductions of them to lattice problems like LWE and SIS. Clearly the Gaussian sampling part has a fixed Gaussian parameter and the center is always zero. The suggested Gaussian parameters were ranging from 6 738 to 754 310 with corresponding signature sizes between 15 and 165 kilobits.

D u c a s, D u r m u s, L e p o i n t and L y u b a s h e v s k y [12] improved on this scheme and gave a security reduction to non-standard (generalized) versions of the usual lattice problems. They used an NTRU-like, ring based variant and further implementation tricks and optimizations when instantiating the scheme resulting in a signature algorithm called "BLISS" (Bimodal Lattice Signature Scheme). BLISS still requires sampling from discrete Gaussians with the center of zero and the suggested Gaussian parameters are ranging from 269 to 680 with corresponding signature sizes between 5 and 6.5 kilobits.

B a i and G a l b r a i t h [5] gave a scheme with proof of security and with signature sizes between 9 and 15 kilobits. This scheme requires Gaussian sampling only in the key generation phase (in the case of constrained devices this task can be delegated to another device). The center is always zero, the Gaussian parameter is fixed and ranges from 146 to 562.

3.4. Summary

As mentioned earlier, the size of the Gaussian parameter and the number of parameter combinations can affect the performance of Gaussian samplers heavily. In this section we overviewed the particular schemes and now we summarize the collected information in Table 1.

TABLE 1. Characteristics of Gaussian sampling in particular schemes.

Scheme	Source	Gaussian parameter	Center	Usage
LWE - plain	[38]	50,000-100,000	$c + \frac{i}{2}$	key generation
LWE - dual	[25]	8-13	0	encryption
GPV	[15], [36]	21	$c + \frac{i}{q}$	signature
GPV+	[30]	17	$c + \frac{i}{2}$	signature
Lyubashevsky	[27]	7,000-700,000	0	signature
BLISS	[12]	270-680	0	signature
Bai-Galbraith	[5]	15-560	0	signature

The only scheme that requires Gaussian sampling with variable Gaussian parameters is the original [15] GPV signature algorithm. In the table we considered this algorithm with the application of Peikert's [36] Gaussian sampling and thus all the schemes in Table 1 are requiring Gaussian sampling from distributions with fixed Gaussian parameter.

In the case of the two older schemes huge Gaussian parameters are needed. These huge parameters are not necessarily to be taken into consideration when aiming for practical implementations though: the original LWE requires Gaussian sampling only at the key generation phase and Lyubashevsky's scheme has highly unfavorable signature sizes compared to its successors.

In the center column there is the form of the center of the distributions prescribed by the particular schemes. Most schemes only sample discrete Gaussians with the center of zero, and in some cases both zero and one half (in the case of GPV+ [30] primitive lattices with power of two modulus were assumed). This means that in these cases only one or two distributions are to sample and consequently only one or two tables are required (if the Gaussian sampler uses any).

The value q is determined by the q-ary lattice used, and can be relatively large. In this case sampling from distributions with q different centers is required, which can mean a huge additional storage requirement if the Gaussian sampler uses any tables.

4. Gaussian samplers

In the schemes discussed in Section 3 sampling from discrete Gaussians over lattices plays a crucial role. These schemes are using algorithms for the task to include sampling from discrete Gaussians over the integers as a subroutine. In this section we survey the methods proposed for performing this task.

4.1. Rejection sampling

The most natural method to sample from discrete Gaussians is the rejection sampling and also it was the first method proposed to apply in lattice based cryptography [15].

In the pseudocode description of the algorithms we will use floating point numbers with explicitly determined precision. To describe the floating point numbers we will adopt the notation of [11].

Definition 4. Let $\mathbb{FP}_m$ denote the floating point numbers with a mantissa of m and precision $\epsilon = 2^{-m+1}$. A floating point number $f \in \mathbb{FP}_m$ is a triplet $f = (s, e, v)$, where $s \in \{0, 1\}$, $e \in \mathbb{Z}$ and $v \in \mathbb{N}_{2^m-1}$, which represents the real number $f = (-1)^s \cdot 2^{e-m} \cdot v \in \mathbb{R}$.

Algorithm 1 Basic Rejection Sampling

procedure SAMPLE$\mathbb{Z}_m(t \in \mathbb{FP}_m, s \in \mathbb{FP}_m, c \in \mathbb{FP}_m)$
  $h \leftarrow -\pi/s^2 \in \mathbb{FP}_m$
  do
    $x \leftarrow [c - ts, c + ts] \cap \mathbb{Z}$ uniformly at random
    $r \leftarrow [0, 1) \subset \mathbb{FP}_m$ uniformly at random
    $p \leftarrow e^{h \cdot (x-c)^2} \in \mathbb{FP}_m$
  while $r < p$
  return x

Algorithm 1 is the basic rejection sampling algorithm, formulated as in [11].

D u c a s and N g u y e n noticed [11] that in most cases the most significant bits are enough to decide about the rejection and constructed a "lazy" variant of the algorithm (see Algorithm 2). Their algorithm uses floating point numbers of two different precision. They report significant speedup with practical parameters and using the IEEE standard double precision as the lower precision.

Algorithm 2 Lazy Rejection Sampling

procedure LAZYSAMPLE$\mathbb{Z}_{m,m'}(s', c', t, \delta_p \in \mathbb{FP}_{m'}, c, s \in \mathbb{FP}_m)$
  $h \leftarrow -\pi/s^2 \in \mathbb{FP}_m$
  $h' \leftarrow -\pi/s'^2 \in \mathbb{FP}_{m'}$
  highprec $\leftarrow$ false
  do
    $x \leftarrow [c' - ts', c' + ts'] \cap \mathbb{Z}$ uniformly at random
    $r \leftarrow [0, 1) \subset \mathbb{FP}_{m'}$ uniformly at random
    if not highprec **then**
      $p' \leftarrow e^{h' \cdot (x-c')^2} \in \mathbb{FP}_{m'}$
      if $|r' - p'| \leq \delta_p$ **then**
        highprec $\leftarrow$ true
      else
        if $r' < p'$ **then**
          return x
    else
      $r \leftarrow$ an extension of r' from $\mathbb{FP}_m$ uniformly at random
      $p \leftarrow e^{h \cdot (x-c)^2} \in \mathbb{FP}_m$
      highprec $\leftarrow$ false
      if $r < p$ **then**
        return x
  while true
  return x

This variant works the same way and uses low precision floating point arithmetic (that is arithmetic in $\mathbb{FP}_{m'}$) until the difference is over certain threshold (the parameter δ_p), in which case it switches to higher precision until the decision.

The rejection sampling does not use precomputed tables and thus its memory consumption is not effected by the number of the possible parameters. But this comes at a price: it has to evaluate the Gaussian function every time it decides about rejection and since rejected rounds do not produce output, the Gaussian function potentially has to be evaluated multiple times before outputting a sample. The algorithm needs an average of $poly(\log n)$ trials until acceptance [36], which makes the procedure even more expensive computationally.

4.2. Inversion method

In the inversion method, we generate a uniform random variate U in the interval $[0, 1]$ and determine the output according to the following inequality:

$$F(X-1) = \sum_{i<X} p_i < U \leq \sum_{i \leq X} = F(X), \tag{3}$$

where p_i denotes $\Pr[X = i]$. P e i k e r t [36] proposed to use this method to sample discrete Gaussians. His exact method was to precompute and store the values of $F(X)$ in a table and solve the inequality (3) through performing a binary search in the table.

This method is fast and does not require high precision floating point arithmetic, but it also requires the presence of precomputed tables. These tables can be relatively large, depending on the Gaussian parameter and the tailcut, and we also need multiple instances of them: one for each different parameter combination.

4.3. Discrete Ziggurat

B u c h m a n et al. [7] adapted the Ziggurat method (a method used for sampling from continuous Gaussians [29]) to the discrete case. The Ziggurat method uses the symmetry of the distribution and generates a sample from the positive half and generates the sign separately. As a precomputation, the probability density function (PDF) is encapsulated in multiple rectangles with the same area. First the algorithm selects a rectangle uniformly at random, and then an x coordinate inside the rectangle also uniformly at random. If the coordinate is in the part of the rectangle completely within the area of the PDF (green area on Figure 1), then it is accepted as a sample. Otherwise a y coordinate is generated uniformly at random and x is accepted as a sample only if (x, y) is in the area of the PDF.

Let (x_i, y_i) be the lower right corner of each rectangle R_i, and let $\bar{x}$ denote the n bit fixed-point representation of x. The algorithm first selects an integer i and then an x integer coordinate inside R_i, both uniformly at random.

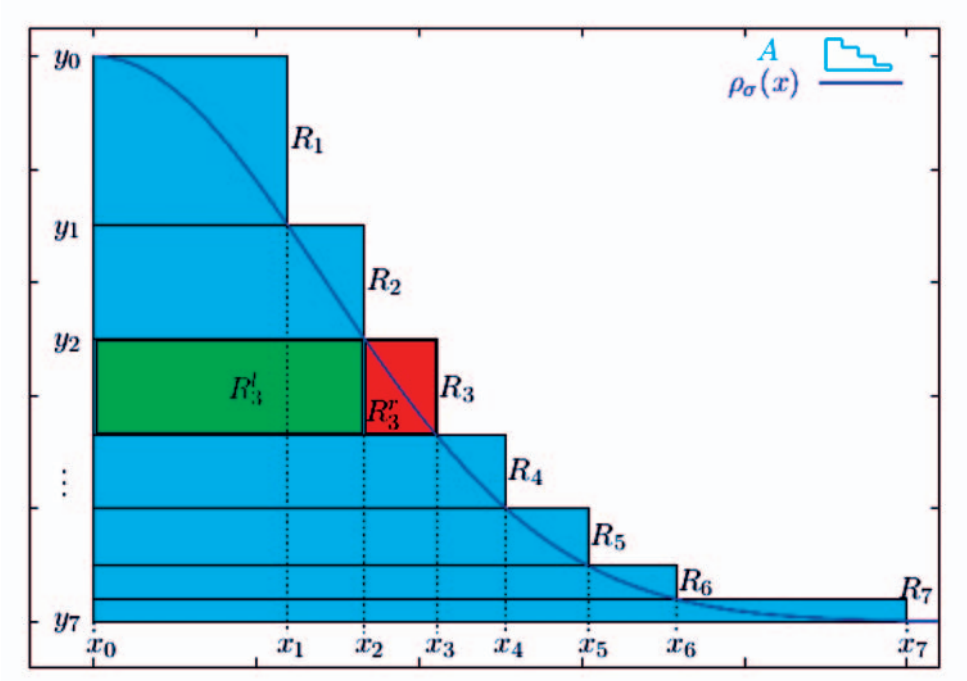

FIGURE 1. A partition of Ziggurat [7, Figure 1].

Then it has to sample from $\bar{y} \in [\bar{y}_i, \bar{y}_{i-1}]$ uniformly and accept if $\bar{y} \leq \rho_s(x)$. To perform this task, the algorithm samples uniformly from $\bar{y} \in \left[0, 2^{\omega}(\bar{y}_{i-1} - \bar{y}_i)\right]$ and accepts if $\bar{y} \leq 2^{\omega}\big(\rho_s(x) - \bar{y}_i\big)$.

The authors also performed a careful analysis of their algorithm and gave an upper bound on the statistical distance of the output from the theoretical discrete Gaussian.

THEOREM 1 (Statistical distance of Ziggurat [7]). *The statistical distance of the discrete Gaussian distribution D_s and the distribution $\bar{D}_s$ output of Algorithm 3 is bounded by*

$$\Delta(D_s, \bar{D}_s) \leq \tau e^{(1-\tau^2)/2} + \frac{|B_0^+|}{\rho_s(B^+) + \frac{1}{2}}(2^{-\omega+1} + 2^{-n}),$$

where B_0^+ denotes the support of $\bar{D}_s^+$ (the non-negative half of the distribution), $B^+ = B_0^+ \setminus \{0\}$ and $\tau = t/\sqrt{2\pi}$.

Using Theorem 1 one can derive a tailcut similar to the general case (see the discussion after Lemma 1), and that if ω and n are roughly the same, then

$n \geq \lambda + \log_2 st$ (notice that $|B_0^+| = \lfloor st/2 \rfloor$) is required to achieve statistical distance less than $2^{-\lambda}$.

This method precomputes and stores the rectangles resulting from the partitioning. The number of the partitions can be arbitrary and a finer partitioning (more rectangles, larger memory requirement) can promise increase in performance. On the other hand, it still uses a large number of high precision floating point operations because of the rejection sampling performed in some regions. Furthermore in its current form it can only sample from a discrete Gaussian with a center of zero, but it seems to be extensible to the general case.

4.4. Knuth-Yao algorithm

The previous methods are based on the assumption that a perfect uniform $[0, 1]$ random variate generator is available. Another model was developed by K n u t h and Y a o [24] which is based on the presence of a perfect random bit generator and measures the cost of an algorithm in terms of the number of bits required to generate a random variate (this is called *random bit model*). Their method is capable to generate finite discrete distributions and the notion of *DDG-tree* plays a central role in it.

Algorithm 3 Discrete Ziggurat algorithm [43]

procedure SAMPLEDZ(m,σ,$\lfloor x_1 \rfloor, \lfloor x_2 \rfloor, \ldots, \lfloor x_m \rfloor, \bar{y}_0, \bar{y}_1, \ldots, \bar{y}_m, \omega$)
  while true **do**
    $i \leftarrow \{1, \ldots, m\}$ uniformly at random
    $s \leftarrow \{-1, 1\}$ uniformly at random
    $x \leftarrow \{0, \ldots, \lfloor x_i \rfloor\}$ uniformly at random
    if $0 < x \leq \lfloor x_{i-1} \rfloor$ **then**
      return sx
    else
      if $x = 0$ **then**
        $b \leftarrow \{0, 1\}$ uniformly at random
        if $b = 0$ **then**
          return sx
        else
          continue
      else
        $y' \leftarrow \{0, \ldots, 2^\omega - 1\}$ uniformly at random
        $\bar{y} \leftarrow y' \cdot (\bar{y}_{i-1} - \bar{y}_i)$
        if $\bar{y} \leq 2^\omega \cdot (\rho_s(x) - \bar{y}_i)$ **then**
          return sx
        else
          continue

Definition 5 (DDG-tree [9])**.** Let X be a random variate to generate with probability vector $p_1, p_2, \ldots$, then the (potentially infinite) binary tree that contains two types of nodes

1. Internal nodes, having two children and the left and right outgoing edges labeled with 0 and 1 respectively,
2. Terminal nodes without children labeled with an integer,

is called DDG-tree (Discrete Distribution Generating tree) of X if

$$\sum_{k\geq 0} \frac{t_i(k)}{2^k} = p_i \qquad \text{for all} \quad i, \tag{4}$$

where $t_i(k)$ denotes the number of terminal nodes labeled with i on the kth level.

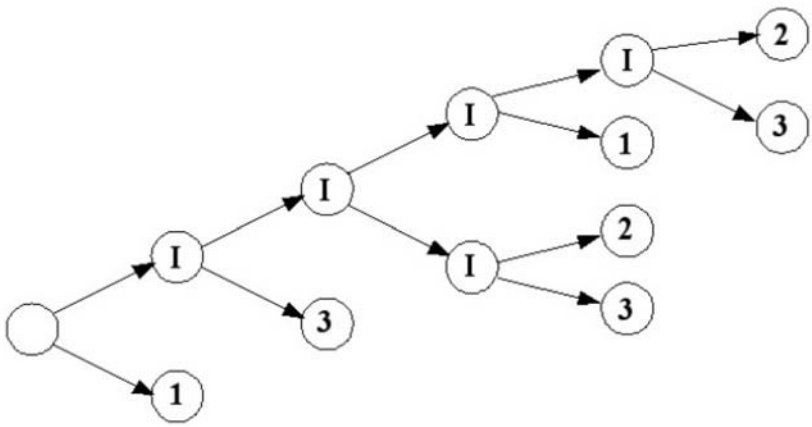

Figure 2. A Knuth-Yao DDG tree [14, Figure 2].

The algorithm traverses this tree randomly, starting at the root and choosing an edge at each level uniformly at random, according to the random bit generator, until it hits a terminal node, in which case the algorithm outputs the label of the node.

Another way to formulate (4) is, that there is a leaf labeled i on the level k if and only if the kth binary digit of p_i is one.

Although D e v r o y e [9] refers to this method as DDG-tree algorithm, we will adapt to [14] and use the name Knuth-Yao algorithm.

The main advantage of this algorithm is, that it is almost perfect in an information theoretic sense: the expected number of the uniformly generated input

Algorithm 4 Knuth-Yao algorithm [40]

```
procedure SampleKY(P ∈ Z_2^{λ×2st})
    d ← 0
    Hit ← 0
    col ← 0
    while Hit = 0 do
        r ← Z_2 uniformly at random
        d ← 2d + 1 − r
        for row ← st down to 0 do
            d ← d − P[row][col]
            if d = −1 then
                S ← row
                Hit ← 1
                break
        col ← col + 1
    return S
```

bits is at most two more than the entropy of the distribution (see [24, Theorem 2.1 and Corollary to Theorem 3.1]), that is $\approx 2.72 + \log_2 s$ (where the entropy is approximated by the entropy of the corresponding continuous Gaussian). The expected number of the random bits used is the primary performance characteristic of samplers analyzed in the random bit model. There are two major drawbacks of this measurement though: firstly any comparison requires that both samplers are analyzed in the random bit model and secondly its impact on the practical performance is highly dependent on the source of randomness used.

The binary expansions of the probabilities are potentially infinite, resulting in an infinite DDG-tree. To make the method finite, one has to truncate the binary expansions of the probabilities. Clearly at least λ bits of the binary expansions are needed in order to achieve a statistical distance less than $2^{-\lambda}$.

The tree can be represented as a table [14] and the columns of this table can easily be constructed with the knowledge of the binary expansions of the probabilities [40]. Algorithm 4 takes this latter approach and as such, it's input is a bitmatrix constructed from the binary expansions of the probabilities.

Clearly this method also requires a precomputed table, namely the binary expansions of the probabilities up to at least λ bits. D w a r a k a n a t h and G a l b r a i t h [14] suggest to perform the algorithm in multiple stages, dividing the distribution into blocks of roughly the same probability, select a block with the Knuth-Yao algorithm, and then perform another Knuth-Yao for the distribution inside the block. They suggest to further improve the performance

by storing only the probabilities corresponding to the middle of the distribution and compute the tails on the fly. This can reduce the size of the table significantly but requires a relatively large number of floating point operations.

In [40] the authors introduce multiple implementation tricks to improve on the efficiency. They use the specific structure of the tree to optimize the performance with the help of the relative distance of the internal nodes (see Algorithm 4). The authors also noticed that in the binary representations there are many leading zeroes and they reduce the size of the precomputed tables by omitting the leading zeroes.

4.5. Binary method

D u c a s et al. [12] beyond their new signature scheme also proposed a new method to sample from discrete Gaussians. The approach is a combination of the inversion method and the rejection sampling. They introduced two major improvements to the original methods:

1. They use the combination of Bernoulli variables to avoid the computation of transcendental functions during rejection sampling.
2. The inversion method is used in combination with rejection sampling to sample from D_{s_2} (the so-called binary discrete Gaussian distribution). This distribution has cumulative probabilities with a very special binary representations so that they can computed on the fly (without evaluating transcendental functions) and no precomputed table is needed.

To sample from $\mathcal{B}_{exp(-x/f)}$ (i.e., the Bernoulli distribution with bias $e^{-x/f}$) for arbitrary integer x and real f one only needs to combine variables with distribution $\mathcal{B}_{exp(2^i/f)}$ according to the binary representation of x. This fact is utilized at the final rejection sampling stage in the algorithm (Algorithm 5) and it means a table of $\log_2(ts)$ values each of which is λ bit long.

The binary discrete Gaussian distribution is D_{s_2} with $s_2 = \sqrt{\pi/\ln 2}$ and thus with probabilities

$$\rho_{s_2} = e^{-\pi x^2/s_2^2} = 2^{-x^2} \qquad \text{for} \quad x \in \mathbb{Z}.$$

Consequently the binary expansions of the scaled cumulative probabilities are of the following form:

$$\rho_{s_2}(\{0,\dots,j\}) = \sum_{i=0}^{j} 2^{-i^2} = 1.1001\underbrace{0\dots0}_{4}1\underbrace{0\dots0}_{6}1\ \dots\ \underbrace{0\dots0}_{2(j-2)}1\underbrace{0\dots0}_{2(j-1)}1\,.$$

Thus one can sample from D_{s_2} by generating a $u \in [0,2)$ bit by bit uniformly at random, rejecting if it is above $\rho_{s_2}(\mathbb{Z}^+)$ and outputting i if

$$\rho_{s_2}(\{0,\dots,i-1\}) \le x < \rho_{s_2}(\{0,\dots,i\})$$

(see [12, Algorithm 10]).

Algorithm 5 Binary Method [12, Algorithm 11]

procedure SAMPLEBI($k \in \mathbb{Z}^+$)
  do
    $x \leftarrow \mathbb{Z}^+$ random according to $D^+_{s_2}$
    $y \leftarrow \{0, \ldots, k-1\}$ uniformly at random
    $z \leftarrow kx + y$
    $b \leftarrow \{0, 1\}$ random according to $\mathcal{B}_{exp(-\pi y(y+2kx)/(ks_2)^2)}$
  while $b = 0$
  return z

The binary method first generates a random variate already close to $D^+_{ks_2}$ for some $k \in \mathbb{Z}^+$ and corrects it with rejection sampling. Algorithm 5 samples only from the positive half of the distribution, to sample from the full one has reject the zero values with a probability of 1/2 and sample a sign uniformly (see [12, Algorithm 12]).

The algorithm was also analyzed in the random bit model: the expected number of random bits consumed is $\approx 1.35 + 2\log_2 s$ ([12, Remark 6.7]). Although that is about two times the cost of the Knuth-Yao algorithm, the impact of this measure on the practical performance is not clear and varies depending on the random source used.

The single disadvantage of the binary method is, that it can only sample from a D_s with s being an integer multiple of s_2.

4.6. D Algorithm

This Algorithm 6 is essentially a rejection scheme which uses von Neumann's algorithm to sample from the exponential distribution [21, N Algorithm]. It has the remarkable property that it uses neither floating point arithmetic, nor precomputed tables.

The algorithm was also analyzed in the random bit model: the expected number of random bits consumed is $\approx -1.85 + 1.39\log_2 s$ ([21]). Although that is between the cost of the Knuth-Yao algorithm and the binary method, the impact of this measure on the practical performance is not clear and varies depending on the random source used.

4.7. Performance

Buchmann et al. [7] also performed practical experiments regarding the performance of the four basic Gaussian sampling algorithms for different choices of the parameter s (see Figure 3). Their implementation uses Shoup's NTL library [41] with precision of 106 bit (the Ziggurat algorithm needs 106 bit precision to achieve statistical distance less than 2^{100}, see [7, Section 3.2]). They used a tailcut of $t = 5$ and parameters $s \in \{25, 80, 2506, 4 \cdot 10^5\}$. The parameter $s = 25$

Algorithm 6 Algorithm D [21]

procedure SAMPLED(μ, σ)
  Select integer $k \geq 0$ with probability $\exp(-\frac{1}{2}k)(1 - 1/\sqrt{e})$
  Accept k with probability $\exp(-\frac{1}{2}k)(k-1)$; otherwise go to step 1
  With equal probabilities set $s \leftarrow \pm 1$.
  Set $i_0 \leftarrow \lceil \sigma k + s\mu \rceil$.
  Select an integer j from $[0, \lceil \sigma \rceil)$ uniformly.
  Set $x \leftarrow ((i_0 - (\sigma k + s\mu)) + j)/\sigma$.
  if $x \geq 1$ **then**
    go to step 1
  if $k = 0, x = 0$ and $s < 0$ **then**
    go to step 1
  Accept x with probability $\exp(-\frac{1}{2}x(2k + x))$; otherwise go to step 1
  Set $i \leftarrow s(i_0 + j)$.
  return i

was based on the requirement of the worst-to-average-case reduction of [39], the $s = 4 \cdot 10^5$ was chosen according to [14] and the other two arbitrarily in between. Although these parameters do not match the ones summarized in Table 1 exactly, these tests still give valuable information about the performance of the particular sampling methods.

As it is seen on the graphs, the rejection sampling algorithm was implemented with precomputed tables. The Ziggurat algorithm's observable drop in performance for bigger table sizes is due to the fact that the table sizes in this cases outgrow the processors fastest cache. Although the graph's show that the Knuth--Yao method requires more memory than the inversion method, using the table representation of [14] this can be reduced to match the inversion method's requirement.

The running time of algorithm D depends weakly on s on the parameter range in question and requires between $0.4\mu s$ and $0.5\mu s$ to generate a sample (that is, it generates $2 \cdot 10^6$ samples per second). Although it is slower than the other methods, keep in mind that it requires neither floating point arithmetic, nor precomputed tables.

5. Conclusion

The parameters of the distribution have a great impact on the performance of discrete Gaussian samplers, that is why we reviewed the recent lattice based schemes utilizing discrete Gaussian sampling, and assessed the parameters of the distributions required. The conclusion is that the most recent and efficient schemes are using Gaussian parameters approximately in the range of $s \in [10, 700]$

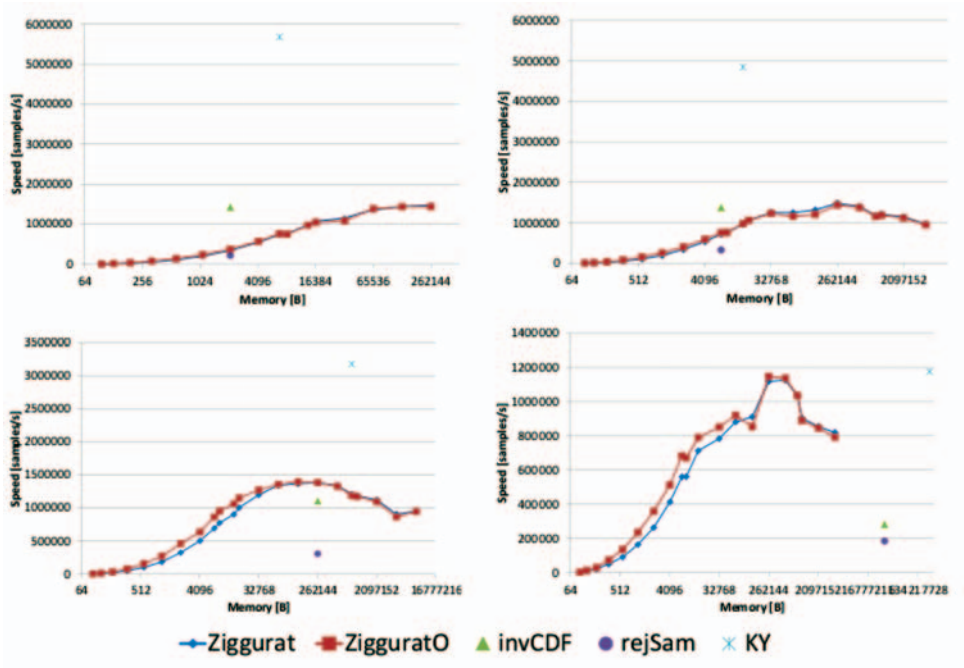

FIGURE 3. Results for inverse CDF, rejection sampling, Knuth-Yao, and discrete Ziggurat with and without optimization for parameters $s = 25, 80, 2506, 4 \cdot 10^5$, respectively. [7, Figure 3].

and with the exception of the original GPV [15] scheme they only need sampling with one or two different centers and all of them require distribution(s) with fixed Gaussian parameter.

To evaluate their practical applicability, we surveyed the known discrete Gaussian sampling algorithms. All of the methods in question have a fixed memory requirement, except the discrete Ziggurat algorithm, which offers great flexibility and a way to reduce the memory consumption at the expense of performance.

Although the Knuth-Yao algorithm is the fastest, it also has the biggest memory consumption among the tested methods (see Figure 3). Still, on a general platform a tree of one megabyte is affordable, and also this can be significantly reduced with the table representation of [14].

In the case of small parameters and constrained devices with sufficient memory (an extra 1–2 kbyte), the Knuth-Yao algorithm is still a good choice. If the Gaussian parameter is on the upper end of the interval and/or the memory is extremely constrained, then the best option is the Ziggurat or algorithm D, depending on the floating point capabilities of the device.

REFERENCES

[1] MIKLÓS, A.: *Generating hard instances of lattice problems,* Electronic Colloquium on Computational Complexity (ECCC) **3** (1996), 29 p.

[2] AKAVIA, A.—GOLDWASSER, S.—VAIKUNTANATHAN, V: *Simultaneous hardcore bits and cryptography against memory attacks,* in: Theory of Cryptography, 6th Theory of Cryptography Conf.—TCC '09, San Francisco, CA, USA, 2009 (O. Reingold, ed.), Lecture Notes in Comput. Sci., Vol. 5444, Springer, Berlin, 2009, pp. 474–495.

[3] ALWEN, J.—PEIKERT, CH.: *Generating shorter bases for hard random lattices,* in: 26th Internat. Symp. on Theoretical Aspects of Comput. Sci.—STACS '09, Freiburg, Germany, 2009 (S. Albers et al., eds.), Leibniz Internat. Proc. in Informatics (LIPICS), Vol. 3, Schloss Dagstuhl – Leibniz Zentrum für Informatik, Wadern, 2009, pp. 75–86 (electronic only).

[4] BABAI, L.: *On Lovász' lattice reduction and the nearest lattice point problem,* Combinatorica **6** (1986), 1–13.

[5] BAI, SH.—GALBRAITH, S. D.: *An improved compression technique for signatures based on learning with errors,* in: Topics in Cryptology—CT-RSA '14, The Cryptographer's Track at the RSA Conf. 2014, San Francisco, CA, USA, 2014 (J. Benaloh, ed.), Lecture Notes in Comput. Sci., Vol. 8366, Springer, Berlin, 2014, pp. 28–47.

[6] BELLARE, M.—ROGAWAY, P.: *Random oracles are practical: A paradigm for designing efficient protocols,* in: Proc. of the 1st ACM Conf. on Computer and Communications Security—CCS '93, ACM, New York, NY, USA, 1993, pp. 62–73.

[7] BUCHMANN, J.—CABARCAS, D.—GÖPFERT, F.—HÜLSING, A.—WEIDEN, P.: *Discrete Ziggurat: A time-memory trade-off for sampling from a Gaussian distribution over the integers,* in: Selected Areas in Cryptography—SAC '13, Burnaby, BC, Canada, 2013 (P. Lisonek et al., eds.), Lecture Notes in Comput. Sci., Vol. 8282, Springer, Berlin, 2013, pp. 402–417.

[8] DETREY, J.—DE DINECHIN, F.: *Table-based polynomials for fast hardware function evaluation,* in: Application-Specific Systems, Architecture Processors—ASAP '05, 16th IEEE Internat. Conf., EEE Computer Society Washington, DC, USA, 2005, pp. 328–333.

[9] DEVROYE, L.: *Non-Uniform Random Variate Generation.* Springer, New York, 1986.

[10] DIFFIE, W.—HELLMAN, M.: *New directions in cryptography,* IEEE Trans. Inform. Theory **22** (2006), 644–654.

[11] DUCAS, L.—NGUYEN, P. Q.: *Faster Gaussian lattice sampling using lazy floating-point arithmetic,* in: Advances in Cryptology—ASIACRYPT '12, 18th Internat. Conf. on the Theory and Appl. of Crypt. and Inform. Security, Beijing, China, 2012 (X. Wang et al., eds.), Lecture Notes in Comput. Sci., Vol. 7658, Springer, Berlin, 2012, pp. 415–432.

[12] DUCAS, L.—DURMUS, A.—LEPOINT, T.—LYUBASHEVSKY, V.: *Lattice signatures and bimodal gaussians,* in: Advances in Cryptology—CRYPTO '13, 33rd Annual Cryptology Conf., Santa Barbara, CA, USA, 2013 (R. Canetti et al., eds.), Lecture Notes in Comput. Sci., Vol. 8042, Springer, Berlin, 2013, pp. 40–56.

[13] DUCAS, L.—NGUYEN, P. Q.: *Learning a zonotope and more: Cryptanalysis of NTRUSign countermeasures,* in: Advances in Cryptology—ASIACRYPT '12, 18th Internat. Conf. on the Theory and Appl. of Cryptology and Information Security, Beijing, China, 2012 (X. Wang and K. Sako, eds.), Lecture Notes in Comput. Sci., Vol. 7658, Springer, Berlin, 2012, pp. 433–450.

[14] DWARAKANATH, N. C.—GALBRAITH, S. D.: *Sampling from discrete gaussians for lattice-based cryptography on a constrained device,* Appl. Algebra Engrg. Comm. Comput. **25** (2014), 159–180.

[15] GENTRY, C.—PEIKERT, CH.—VAIKUNTANATHAN, V.: *Trapdoors for hard lattices and new cryptographic constructions,* in: Proc. of the 40th Annual ACM Symp. on Theory of Computing—STOC '08, Victoria, Canada, ACM, New York, 2008, pp. 197–206.

[16] GENTRY, C.—SZYDLO, M.: *Cryptanalysis of the revised NTRU signature scheme,* in: Advances in Cryptology—EUROCRYPT '02, 21st Internat. Conf. on the Theory and Appl. of Cryptographic Techniques, Amsterdam, the Netherlands, 2002 (L. Knudsen, ed.), Lecture Notes in Comput. Sci., Vol. 2332, Springer, Berlin, 2002, pp. 299–320.

[17] GÜNEYSU, T.—LYUBASHEVSKY, V.—PÖPPELMANN, TH.: *Practical lattice-based cryptography: A signature scheme for embedded systems,* in: Cryptographic Hardware and Embedded Systems—CHES '12, 14th Internat. Workshop, Leuven, Belgium, 2012, (E. Prouff and P. Schaumont, eds.), Lecture Notes in Comput. Sci., Vol. 7428, Springer, Berlin, 2012, pp. 530–547.

[18] GOLDREICH, O.—GOLDWASSER, SH.—HALEVI, SH.: *Public-key cryptosystems from lattice reduction problems,* in: Advances in Cryptology—CRYPTO '97, 17th Annual Internat. Cryptology Conf., Santa Barbara, CA, USA, 1997, (B. S. Kaliski, jr., ed.), Lecture Notes in Comput. Sci., Vol. 1294, Springer, Berlin, 1997, pp. 112–131.

[19] HOFFSTEIN, J.—HOWGRAVE-GRAHAM, N.—PIPHER, J.—SILVERMAN, J. H.––WHYTE, W.: *NTRUSign: Digital signatures using the NTRU lattice,* in: Topics in Cryptology—CT-RSA '03, The Cryptographers' Track at the RSA Conf., San Francisco, CA, USA, 2003 (M. Joye, ed.), Lecture Notes in Comput. Sci., Vol. 2612, Springer, Berlin, 2003, pp. 122–140.

[20] HOFFSTEIN, J.—PIPHER, J.—SILVERMAN, J. H.: *Nss: An ntru lattice-based signature scheme,* in: Advances in Cryptology—EUROCRYPT '01, Internat. Conf. on the Theory and Appl. of Cryptographic Tech., Innsbruck, Austria, 2001 (B. Pfitzmann, ed.), Lecture Notes in Comput. Sci., Vol. 2045, Springer, Berlin, 2001, pp. 211–228.

[21] KARNEY, C. F. F.: *Sampling exactly from the normal distribution,* 2014, `http://arxiv.org/abs/1303.6257`.

[22] KAWACHI, A.—TANAKA, K.—XAGAWA, K.: *Multi-bit cryptosystems based on lattice problems,* in: Public Key Cryptography—PKC '07, 10th Internat. Conf. on Practice and Theory in Public-Key Cryptography, Beijing, China, 2007 (T. Okamoto et al., eds.), Lecture Notes in Comput. Sci., Vol. 4450, Springer, Berlin, 2007, pp. 315–329.

[23] KLEIN, P.: *Finding the closest lattice vector when it's unusually close,* in: Proc. of the 11th Annual ACM-SIAM Symposium on Discrete Algorithms, San Francisco, CA, USA, 2000, SIAM, Philadelphia, PA, 2000, pp. 937–941.

[24] KNUTH, D. E.—YAO, A. C.: *The complexity of nonuniform random number generation,* in: Algorithms and Complexity: New Directions and Recent Results, Pittsburgh, 1976 (J. F. Traub, ed.), Academic Press, New York, 1976.

[25] LINDNER, R.—PEIKERT, CH.: *Better key sizes (and attacks) for lwe-based encryption,* in: Topics in cryptology—CT-RSA '11, The Cryptographers' Track at the RSA Conf., 2011, San Francisco, CA, USA, 2011, (A. Kiayias, ed.), Lecture Notes in Comput. Sci., Vol. 6558, Springer, Berlin, 2011, pp. 319–339.

[26] LYUBASHEVSKY, V.: *Fiat-shamir with aborts: Applications to lattice and factoring-based signatures,* in: Advances in Cryptology—ASIACRYPT '09, 15th Internat. Conf. on the Theory and Appl. of Cryptology and Inform. Security, Tokyo, Japan, 2009 (M. Matsui, ed.), Lecture Notes in Comput. Sci., Vol. 5912, Springer, Berlin, 2009, pp. 598–616.

[27] LYUBASHEVSKY, V.: *Lattice signatures without trapdoors,* in: Advances in Cryptology––EUROCRYPT '12, 31st Annual Internat. Conf. on the Theory and Appl. of Cryptographic Techniques, Cambridge, UK, 2012 (D. Pointcheval and Th. Johansson, eds.), Lecture Notes in Comput. Sci., Vol. 7237, Springer, Berlin, 2012, pp. 738–755.

[28] LYUBASHEVSKY, V.—PEIKERT, CH.—REGEV, O.: *On ideal lattices and learning with errors over rings,* in: Advances in Cryptology—EUROCRYPT '10, 29th Annual Internat. Conf. on the Theory and Appl. of Cryptographic Techniques, French Riviera, 2010 (H. Gilbert, ed.), Lecture Notes in Comput. Sci., Vol. 6110, Springer, Berlin, 2010, pp. 1–23.

[29] MARSAGLIA, G.—TSANG, W. W.: *The ziggurat method for generating random variables,* J. Statist. Software **5** (2000), 1–7.

[30] MICCIANCIO, D.—PEIKERT, CH.: *Trapdoors for lattices: Simpler, tighter, faster, smaller,* in: in: Advances in Cryptology–EUROCRYPT '12, 31st Annual Internat. Conf. on the Theory and Appl. of Cryptographic Techniques, Cambridge, UK, 2012 (D. Pointcheval and Th. Johansson, eds.), Lecture Notes in Comput. Sci., Vol. 7237, Springer, Berlin, 2012, pp. 700–718.

[31] MICCIANCIO, D.—REGEV, O.: *Worst-case to average-case reductions based on gaussian measures,* SIAM J. Comput. **37** (2007), 267–302.

[32] MULLER, J.-M.: *Elementary Functions. Algorithms and Implementation.* Birkhäuser, Boston, 1997.

[33] NGUYEN, P. Q.—REGEV, O.: *Learning a parallelepiped: Cryptanalysis of ggh and ntru signatures,* in: Advances in Cryptology—EUROCRYPT '06, 25th Annual Internat. Conf. on the Theory and Appl. of Cryptographic Techniques, St. Petersburg, Russia, 2006 (S. Vaudenay, ed.), Lecture Notes in Comput. Sci., Vol. 4004, Springer, Berlin, 2006, pp. 271–288.

[34] NGUYEN, P. Q.—REGEV, O.: *Learning a parallelepiped: Cryptanalysis of GGH and NTRU signatures,* J. Cryptology **22** (2009), 139–160.

[35] OLVER, F. W.—LOZIER, D. W.—BOISVERT, R. F.—CLARK, CH. W.: *NIST Handbook of Mathematical Functions.* Cambridge University Press, Cambridge, 2010.

[36] PEIKERT, CH.: *An efficient and parallel gaussian sampler for lattices,* in: Advances in Cryptology—CRYPTO '10, 30th Annual Cryptology Conf., Santa Barbara, CA, USA, 2010 (T. Rabin, ed.), Lecture Notes in Comput. Sci., Vol. 6223, Springer, Berlin, 2010, pp. 80–97.

[37] PEIKERT, CH.: *Public-key cryptosystems from the worst-case shortest vector problem: Extended abstract,* in: Proc. of the 41 Annual ACM Symp. on Theory of Comput.––STOC '09, ACM, New York, NY, USA, 2009, pp. 333–342.

[38] PEIKERT, CH.—VAIKUNTANATHAN, V.—WATERS, B.: *A framework for efficient and composable oblivious transfer,* in: Advances in Cryptology—CRYPTO '08, 28th Annual Internat. Cryptology Conf., Santa Barbara, CA, USA, 2008 (D. Wagner, ed.), Lecture Notes in Comput. Sci., Vol. 5157, 554-571 (2008) Springer, Berlin, 2008, pp. 554–571.

[39] REGEV, O.: *On lattices, learning with errors, random linear codes, and cryptography,* in: Proc. of the 37th Annual ACM Symp. on Theory of Computing—STOC '05, Baltimore, MD, USA, 2005, ACM, New York, NY, pp. 84–93.

[40] ROY, S. S.—VERCAUTEREN, F.—VERBAUWHEDE, I.: *High precision discrete gaussian sampling on FPGAs,* in: Selected Areas in Cryptography—SAC '13, 20th Internat. Conf., Burnaby, BC, Canada, 2013 (T. Lange et al., eds.), Lecture Notes in Comput. Sci., Vol. 8282, Springer, Berlin, 2013, pp. 383–401.

[41] SHOUP, V.: *NTL: A library for doing number theory,* 2013, `www.shoup.net/ntl`.

[42] SPECKER, W. H.: *A class of algorithms for* $\ln x$, $\exp x$, $\sin x$, $\cos x$, $\tan^{-1} x$, *and* $\cot^{-1} x$, in: j-IEEE-trans-elec-comput, EC-14(1):85–86, 1965.

[43] WEIDEN, P.—HÜLSING, A.—CABARCAS, D.—BUCHMANN, J.: *Instantiating treeless signature schemes,* Cryptology ePrint Archive 65:1–18, 2013.

Received August 25, 2014

Department of Computer Science
Faculty of Informatics
University of Debrecen
Kassai Street 26
H-4028 Debrecen
HUNGARY

E-mail: follath.janos@inf.unideb.hu