An Efficient and Parallel Gaussian Sampler for Lattices

Chris Peikert*

Georgia Institute of Technology

Abstract. At the heart of many recent lattice-based cryptographic schemes is a polynomial-time algorithm that, given a 'high-quality' basis, generates a lattice point according to a Gaussian-like distribution. Unlike most other operations in lattice-based cryptography, however, the known algorithm for this task (due to Gentry, Peikert, and Vaikuntanathan; STOC 2008) is rather inefficient, and is inherently sequential.
We present a new Gaussian sampling algorithm for lattices that is *efficient* and *highly parallelizable*. We also show that in most cryptographic applications, the algorithm's efficiency comes at almost no cost in asymptotic security. At a high level, our algorithm resembles the "perturbation" heuristic proposed as part of NTRUSign (Hoffstein *et al.*, CT-RSA 2003), though the details are quite different. To our knowledge, this is the first algorithm and rigorous analysis demonstrating the security of a perturbation-like technique.

1 Introduction

In recent years, there has been rapid development in the use of *lattices* for constructing rich cryptographic schemes.[1] These include digital signatures (both 'tree-based' [13] and 'hash-and-sign' [8, 6]), identity-based encryption [8] and hierarchical IBE [6, 1], noninteractive zero knowledge [19], and even a fully homomorphic cryptosystem [7].

The cornerstone of many of these schemes (particularly, but not exclusive to, those that 'answer queries') is the polynomial-time algorithm of [8] that samples from a so-called *discrete Gaussian* probability distribution over a lattice Λ. More precisely, for a vector $\mathbf{c} \in \mathbb{R}^n$ and a "width" parameter $s > 0$, the distribution $D_{\Lambda+\mathbf{c},s}$ assigns a probability proportional to $\exp(-\pi\|\mathbf{v}\|^2/s^2)$ to each $\mathbf{v} \in \Lambda + \mathbf{c}$ (and probability zero elsewhere). Given $\mathbf{c}$, a basis $\mathbf{B}$ of Λ, and a sufficiently large s (related to the 'quality' of $\mathbf{B}$), the GPV algorithm outputs a sample from a

* This material is based upon work supported by the National Science Foundation under Grant CNS-0716786. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

[1] A lattice $\Lambda \subset \mathbb{R}^n$ is a periodic 'grid' of points, or more formally, a discrete subgroup of $\mathbb{R}^n$ under addition. It is generated by a (not necessarily unique) *basis* $\mathbf{B} \subset \mathbb{R}^{n\times k}$ of k linearly independent vectors, as $\Lambda = \{\mathbf{Bz} : \mathbf{z} \in \mathbb{Z}^k\}$. In this paper we are concerned only with *full-rank* lattices, i.e., where $k = n$.

distribution statistically close to $D_{\Lambda+\mathbf{c},s}$. (Equivalently, by subtracting $\mathbf{c}$ from the output, it samples a lattice point from a Gaussian distribution centered at $-\mathbf{c}$.) Informally speaking, the sampling algorithm is 'zero-knowledge' in the sense that it leaks no information about its input basis $\mathbf{B}$ (aside from a bound on its quality), because $D_{\Lambda+\mathbf{c},s}$ is defined without reference to any particular basis. This zero-knowledge property accounts for its broad utility in lattice-based cryptography.

While the sampling algorithm of [8] has numerous applications in cryptography and beyond, for both practical and theoretical purposes it also has some drawbacks:

- First, it is rather *inefficient*: on an n-dimensional lattice, a straightforward implementation requires exact arithmetic on an $n \times n$ matrix having $\Omega(n)$-bit entries (even ignoring some additional $\log n$ factors). While approximate arithmetic and other optimizations may be possible in certain cases, great care would be needed to maintain the proper output distribution, and the algorithm's essential structure appears difficult to make truly practical.
- Second, it is *inherently sequential*: to generate a sample, the algorithm performs n adaptive iterations, where the choices made in each iteration affect the values used in the next. This stands in stark contrast to other 'embarrassingly parallelizable' operations that are typical of lattice-based cryptography.

1.1 Contributions

We present a new algorithm that samples from a discrete Gaussian distribution $D_{\Lambda+\mathbf{c},s}$ over a lattice, given a 'high-quality' basis for Λ. The algorithm is especially well-suited to 'q-ary' integer lattices, i.e., sublattices of $\mathbb{Z}^n$ that themselves contain $q\mathbb{Z}^n$ as a sublattice, for some known and typically small $q \geq 2$. These include NTRU lattices [10] and the family of random lattices that enjoy 'worst-case hardness,' as first demonstrated by Ajtai [3]. Most modern lattice-based cryptographic schemes (including those that rely on Gaussian sampling) are designed around q-ary lattices, so they are a natural target for optimization.

The key features of our algorithm, as specialized to n-dimensional q-ary lattices, are as follows. It is:

- *Offline / online*: when the lattice basis is known in advance of the point $\mathbf{c}$ (which is the norm in cryptographic applications), most of the work can be performed as offline precomputation. In fact, the offline phase may be viewed simply as an extension of the application's key-generation algorithm.
- *Simple and efficient*: the online phase involves only $O(n^2)$ integer additions and multiplications modulo q or q^2, where the O-notation hides a small constant ≈ 4.
- *Fully parallelizable*: for any P up to n^2, the online phase can allocate $O(n^2/P)$ of its operations to each of P processors.

– *High-quality*: for random bases that are commonly used in cryptographic schemes, our algorithm can sample from a Gaussian of essentially the same 'quality' as the prior GPV algorithm; this is important for the concrete security of applications. See Section 1.2 below for a full discussion.

We emphasize that for a practical implementation, parallelized operations on small integers represent a significant performance advantage. Most modern computer processors have built-in support for "vector" instructions (also known as "single instruction, multiple data"), which perform simple operations on entire vectors of small data elements simultaneously. Our algorithm can exploit these operations very naturally. For a detailed efficiency comparison between our algorithm and that of [8], see Section 1.2 below.

At a very high level, our algorithm resembles the "perturbation" heuristic proposed for the NTRUSign signature scheme [9], but the details differ significantly; see Section 1.3 for a comparison. To our knowledge, this is the first algorithm and analysis to demonstrate the theoretical soundness of a perturbation-like technique. Finally, the analysis of our algorithm relies on some new general facts about 'convolutions' of discrete Gaussians, which we expect will be applicable elsewhere. For example, these facts allow for the use of a clean *discrete* Gaussian error distribution (rather than a 'rounded' Gaussian) in the "learning with errors" problem [20], which may be useful in certain applications.

1.2 Comparison with the GPV Algorithm

Here we give a detailed comparison of our new sampling algorithm to the previous one of [8]. The two main points of comparison are the width ('quality') of the sampled Gaussian, and the algorithmic efficiency.

Gaussian Width One of the important properties of a discrete Gaussian sampling algorithm is the *width* s of the distribution it generates, as a function of the input basis. In cryptographic applications, the width is the main quantity governing the concrete security and, if applicable, the approximation factor of the underlying worst-case lattice problems. This is because in order for the scheme to be secure, it must hard for an adversary to find a lattice point within the likely radius $s\sqrt{n}$ of the Gaussian (i.e., after truncating its negligibly likely tail). The wider the distribution, the more leeway the adversary has in an attack, and the larger the scheme's parameters must be to compensate. On the other hand, a more efficient sampling algorithm can potentially allow for the use of larger parameters without sacrificing performance.

The prior sampling algorithm of [8], given a lattice basis $\mathbf{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_n\}$, can sample from a discrete Gaussian having width as small as $\|\widetilde{\mathbf{B}}\| = \max_i \|\widetilde{\mathbf{b}_i}\|$, where $\widetilde{\mathbf{B}}$ denotes the Gram-Schmidt orthogonalization of $\mathbf{B}$.[2] (Actually, the width also includes a small $\omega(\sqrt{\log n})$ factor, which is also present in our new algorithm,

[2] In the Gram-Schmidt orthogonalization $\widetilde{\mathbf{B}}$ of $\mathbf{B}$, the vector $\widetilde{\mathbf{b}_i}$ is the projection of $\mathbf{b}_i$ orthogonally to $\mathrm{span}(\mathbf{b}_1, \ldots, \mathbf{b}_{i-1})$.

so for simplicity we ignore it in this summary.) As a point of comparison, $\|\widetilde{\mathbf{B}}\|$ is always at most $\max_i \|\mathbf{b}_i\|$, and in some cases it can be substantially smaller.

In contrast, our new algorithm works for a width s as small as the *largest singular value* $s_1(\mathbf{B})$ of the basis $\mathbf{B}$, or equivalently, the square root of the largest eigenvalue of the Gram matrix $\mathbf{B}\mathbf{B}^t$. It is easy to show that $s_1(\mathbf{B})$ is always at least $\max_i \|\mathbf{b}_i\|$, so our new algorithm cannot sample from a narrower Gaussian than the GPV algorithm can. At the same time, any basis $\mathbf{B}$ can always be efficiently processed (without increasing $\|\widetilde{\mathbf{B}}\|$) to guarantee that $s_1(\mathbf{B}) \leq n \cdot \|\widetilde{\mathbf{B}}\|$, so our algorithm is at worst an n factor looser than that of [8].

While a factor of n gap between the two algorithms may seem rather large, in cryptographic applications this *worst-case* ratio is actually immaterial; what matters is the relative performance on the *random* bases that are used as secret keys. Here the situation is much more favorable. First, we consider the basis-generation algorithms of [4] (following [2]) for 'worst-case-hard' q-ary lattices, which are used in most theoretically sound cryptographic applications. We show that with a minor modification, one of the algorithms from [4] outputs (with overwhelming probability) a basis $\mathbf{B}$ for which $s_1(\mathbf{B})$ is only an $O(\sqrt{\log q})$ factor larger than $\|\widetilde{\mathbf{B}}\|$ (which itself is asymptotically optimal, as shown in [4]). Because q is typically a small polynomial in n, this amounts to a cost of only an $O(\sqrt{\log n})$ factor in the width of the Gaussian. Similarly, when the vectors of $\mathbf{B}$ are themselves drawn from a discrete Gaussian, as in the basis-delegation technique of [6], we can show that $s_1(\mathbf{B})$ is only a $\omega(\sqrt{\log n})$ factor larger than $\|\widetilde{\mathbf{B}}\|$ (with overwhelming probability). Therefore, in cryptographic applications the performance improvements of our algorithm can come at almost no asymptotic cost in security. Of course, a concrete evaluation of the performance/security trade-off for real-world parameters would require careful analysis and experiments, which we leave for later work.

Efficiency We now compare the efficiency of the two known sampling algorithms. We focus on the most common case of q-ary n-dimensional integer lattices, where a 'good' lattice basis (whose vectors having length much less than q) is initially given in an offline phase, followed by an online phase in which a desired center $\mathbf{c} \in \mathbb{Z}^n$ is given. This scenario allows for certain optimizations in both algorithms, which we include for a fair comparison.

The sampling algorithm from [8] can use the offline phase to compute the Gram-Schmidt orthogonalization of its given basis; this requires $\Omega(n^4 \log^2 q)$ bit operations and $\Omega(n^3)$ bits of intermediate storage. The online phase performs n sequential iterations, each of which computes an inner product between a Gram-Schmidt vector having $\Omega(n)$-bit entries, and an integer vector whose entries have magnitude at most q. In total, these operations require $\Omega(n^3 \log q)$ bit operations. In addition, each iteration performs a certain randomized-rounding operation, which, while asymptotically poly$(\log n)$-time, is not especially practical (nor precomputable) because it uses rejection sampling on a value that is not known until the online phase. Lastly, while the work within each iteration may be parallelized, the iterations themselves must be performed sequentially.

Our algorithm is more efficient and practical in the running time of both phases, and in the amount of intermediate storage between phases. The offline phase first computes a matrix inverse modulo q^2, and a 'square root' of a matrix whose entries have magnitude at most q; these can be computed in $O(n^3 \log^2 q)$ bit operations. Next, it generates and stores one or more short integer 'perturbation' vectors (one per future call to the online phase), and optionally discards the matrix square root. The intermediate storage is therefore as small as $O(n^2 \log q)$ bits for the matrix inverse, plus $O(n \log q)$ bits per perturbation vector. Optionally, the offline phase can also precompute the randomized-rounding operations, due to the small number of possibilities that can occur online. The online phase simply computes about $4n^2$ integer additions and multiplications ($2n^2$ of each) modulo q or q^2, which can be fully parallelized among up to n^2 processors.

Lastly, we mention that our sampling algorithm translates very naturally to the setting of *compact* q-ary lattices and bases over certain rings R that are larger than $\mathbb{Z}$, where security is based on the worst-case hardness of *ideal lattices* in R (see, e.g., [16, 21, 14]). In contrast to GPV, our algorithm can directly take advantage of the ring structure for further efficiency, yielding a savings of an $\tilde{\Omega}(n)$ factor in the computation times and intermediate storage.

1.3 Overview of the Algorithm

The GPV sampling algorithm [8] is based closely on Babai's "nearest-plane" decoding algorithm for lattices [5]. Babai's algorithm takes a point $\mathbf{c} \in \mathbb{R}^n$ and a lattice basis $\mathbf{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_n\}$, and for $i = n, \ldots, 1$ computes a coefficient $z_i \in \mathbb{Z}$ for $\mathbf{b}_i$ by iteratively projecting ('rounding') $\mathbf{c}$ orthogonally to the nearest hyperplane of the form $z_i \mathbf{b}_i + \mathrm{span}(\mathbf{b}_1, \ldots, \mathbf{b}_{i-1})$. The output is the lattice vector $\sum_i z_i \mathbf{b}_i$, whose distance from the original $\mathbf{c}$ can be bounded by the quality of $\mathbf{B}$. The GPV algorithm, whose goal is instead to sample from a discrete Gaussian centered at $\mathbf{c}$, uses *randomized rounding* in each iteration to select a 'nearby' plane, under a carefully defined probability distribution. (This technique is also related to another randomized-rounding algorithm of Klein [11] for a different decoding problem.)

In addition to his nearest-plane algorithm, Babai also proposed a simpler (but somewhat looser) lattice decoding algorithm, which we call "simple rounding." In this algorithm, a given point $\mathbf{c} \in \mathbb{R}^n$ is rounded to the lattice point $\mathbf{B}\lfloor \mathbf{B}^{-1}\mathbf{c} \rceil$, where each coordinate of $\mathbf{B}^{-1}\mathbf{c} \in \mathbb{R}^n$ is independently rounded to its nearest integer. With precomputation of $\mathbf{B}^{-1}$, this algorithm can be quite practical — especially on q-ary lattices, where several more optimizations are possible. Moreover, it is trivially parallelized among up to n^2 processors. Unfortunately, its deterministic form it turns out to be completely insecure for 'answering queries' (e.g., digital signatures), as demonstrated by Nguyen and Regev [18].

A natural question, given the approach of [8], is whether a *randomized* variant of Babai's simple-rounding algorithm is secure. Specifically, the natural way of randomizing the algorithm is to round each coordinate of $\mathbf{B}^{-1}\mathbf{c}$ to a nearby integer (under a discrete Gaussian distribution over $\mathbb{Z}$, which can be sampled efficiently), then left-multiply by $\mathbf{B}$ as before. Unlike with the randomized nearest-plane

algorithm, though, the resulting probability distribution here is unfortunately not spherical, nor does it leak zero knowledge. Instead, it is a 'skewed' (elliptical) Gaussian, where the skew mirrors the 'geometry' of the basis. More precisely, the *covariance* matrix $\mathrm{E}_{\mathbf{x}}[(\mathbf{x}-\mathbf{c})(\mathbf{x}-\mathbf{c})^t]$ of the distribution (about its center $\mathbf{c}$) is approximately $\mathbf{B}\mathbf{B}^t$, which captures the entire geometry of the basis $\mathbf{B}$, up to rigid rotation. Because covariance can be measured efficiently from only a small number of samples, the randomized simple-rounding algorithm leaks this geometry.[3]

Our solution prevents such leakage, in a manner inspired by the following facts. Recall that if X and Y are two independent random variables, the probability distribution of their sum $X+Y$ is the *convolution* of their individual distributions. In addition, for continuous (not necessarily spherical) Gaussians, covariance matrices are additive under convolution. In particular, if Σ_1 and Σ_2 are covariance matrices such that $\Sigma_1 + \Sigma_2 = s^2\mathbf{I}$, then the convolution of two Gaussians with covariance matrices Σ_1, Σ_2 (respectively) is a spherical Gaussian with standard deviation s.

The above facts give the basic idea for our algorithm, which is to convolve the output of the randomized simple-rounding algorithm with a suitable non-spherical (continuous) Gaussian, yielding a spherically distributed output. However, note that we want the algorithm to generate a *discrete* distribution — i.e., it must output a lattice point — so we should not alter the output of the randomized-rounding step. Instead, we *first* perturb the desired center $\mathbf{c}$ by a suitable non-spherical Gaussian, then apply randomized rounding to the resulting perturbed point. Strictly speaking this is not a true convolution, because the rounding step depends on the output of the perturbation step, but we can reduce the analysis to a true convolution using bounds related to the "smoothing parameter" of the lattice [17].

The main remaining question is: for a given covariance matrix $\Sigma_1 = \mathbf{B}\mathbf{B}^t$ (corresponding to the rounding step), for what values of s is there an efficiently sampleable Gaussian having covariance matrix $\Sigma_2 = s^2\mathbf{I} - \Sigma_1$? The covariance matrix of any (non-degenerate) Gaussian is symmetric positive definite, i.e., all its eigenvalues are positive. Conversely, every positive definite matrix is the covariance of some Gaussian, which can sampled efficiently by computing a 'square root' of the covariance matrix. Since any eigenvector of Σ_1 (with eigenvalue $\sigma^2 > 0$) is also an eigenvector of $s^2\mathbf{I}$ (with eigenvalue s^2), it must be an eigenvector of Σ_2 (with eigenvalue $s^2 - \sigma^2$) as well. Therefore, a necessary and sufficient condition is that all the eigenvalues of Σ_1 be less than s^2. Equivalently, the algorithm works for any s that exceeds the largest singular value of the

[3] Given the above, one might still wonder whether the covariance $\mathbf{B}\mathbf{B}^t$ could be simulated efficiently (without any privileged knowledge about the lattice) when $\mathbf{B}$ is itself drawn from a 'nice' distribution, such as a discrete Gaussian. Indeed, if the vectors of $\mathbf{B}$ were drawn independently from a *continuous* Gaussian, the matrix $\mathbf{B}\mathbf{B}^t$ would have the so-called *Wishart distribution*, which can be generated 'obliviously' (without knowledge of $\mathbf{B}$ itself) using the *Bartlett decomposition*. (See, e.g., [12] and references therein). Unfortunately, these facts do not quite seem to carry over to discrete Gaussians, though they may be useful in another cryptographic context.

given basis $\mathbf{B}$. More generally, it can sample any (possibly non-spherical) discrete Gaussian with covariance matrix $\Sigma > \Sigma_1$ (i.e., $\Sigma - \Sigma_1$ is positive definite).

In retrospect, the high-level structure of our algorithm resembles the "perturbation" heuristic proposed for NTRUSign [9], though the details are quite different. First, the perturbation and rounding steps in NTRUSign are both *deterministic* with respect to two or more bases, and there is evidence that this is insecure [15], at least for a large polynomial number of signatures. Interestingly, randomization also allows for improved efficiency, since our perturbations can be chosen with offline precomputation (as opposed to the deterministic method of [9], which is inherently online). Second, the signing and perturbation bases used in NTRUSign are chosen independently, whereas our perturbations are carefully chosen to conceal the statistics that would otherwise be leaked by randomized rounding.

2 Preliminaries

2.1 Notation

For a countable set X and a real-valued function f, we write $f(X)$ to denote $\sum_{x \in X} f(x)$. A nonnegative function $f\colon \mathbb{N} \to \mathbb{R}$ is called *negligible*, written $f(n) = \mathrm{negl}(n)$, if it vanishes faster than any inverse polynomial, i.e., $f(n) = o(n^{-c})$ for every constant $c \geq 0$. A function $g\colon \mathbb{N} \to [0,1]$ is called *overwhelming* if it is $1 - \mathrm{negl}(n)$. The *statistical distance* between two distributions X and Y (or two random variables have those distributions, respectively) is defined as $\Delta(X,Y) = \sup_{A \subseteq D} |X(A) - Y(A)|$. When D is a countable set, we have $\Delta(X,Y) = \frac{1}{2} \sum_{d \in D} |X(d) - Y(d)|$.

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. We use bold upper-case letters (e.g., $\mathbf{B}$) for ordered sets of vectors, and identify the set with the matrix having the vectors as its columns. We frequently use upper-case Greek letters such as Σ to denote (symmetric) *positive (semi)definite* matrices, defined below. In contexts where a matrix is expected, we sometimes use a scalar $s \in \mathbb{R}$ to denote $s \cdot \mathbf{I}$, where $\mathbf{I}$ is the identity matrix of appropriate dimension. We let $\|\mathbf{B}\| = \max_i \|\mathbf{b}_i\|$, where $\|\cdot\|$ denotes the Euclidean norm.

2.2 Linear Algebra

A symmetric matrix $\Sigma \in \mathbb{R}^{n \times n}$ is *positive definite*, written $\Sigma > \mathbf{0}$, if $\mathbf{x}^t \Sigma \mathbf{x} > 0$ for all nonzero $\mathbf{x} \in \mathbb{R}^n$. Equivalently, its spectral decomposition is

$$\Sigma = \mathbf{Q}\mathbf{D}^2\mathbf{Q}^{-1} = \mathbf{Q}\mathbf{D}^2\mathbf{Q}^t,$$

where $\mathbf{Q} \in \mathbb{R}^{n \times n}$ is an orthogonal matrix (i.e., one for which $\mathbf{Q}^t\mathbf{Q} = \mathbf{Q}\mathbf{Q}^t = \mathbf{I}$) whose columns are eigenvectors of Σ, and $\mathbf{D}$ is the real diagonal matrix of the square roots of the corresponding eigenvalues, all of which are positive. We

have $\Sigma > \mathbf{0}$ if and only if $\Sigma^{-1} > \mathbf{0}$. We say that Σ is positive *semi*definite, written $\Sigma \geq \mathbf{0}$, if $\mathbf{x}^t \Sigma \mathbf{x} \geq 0$ for all $\mathbf{x} \in \mathbb{R}^n$; such a matrix may not be invertible. Positive (semi)definiteness defines a partial ordering on symmetric matrices: we say that $\Sigma_1 > \Sigma_2$ if $(\Sigma_1 - \Sigma_2) > \mathbf{0}$, and likewise for $\Sigma_1 \geq \Sigma_2$. It is the case that $\Sigma_1 \geq \Sigma_2 > \mathbf{0}$ if and only if $\Sigma_2^{-1} \geq \Sigma_1^{-1} > \mathbf{0}$, and likewise for the analogous strict inequalities.

For any nonsingular matrix $\mathbf{B} \in \mathbb{R}^{n \times n}$, the symmetric matrix $\Sigma = \mathbf{B}\mathbf{B}^t$ is positive definite, because

$$\mathbf{x}^t \Sigma \mathbf{x} = \langle \mathbf{B}^t \mathbf{x}, \mathbf{B}^t \mathbf{x} \rangle = \|\mathbf{B}^t \mathbf{x}\|^2 > 0$$

for nonzero $\mathbf{x} \in \mathbb{R}^n$. We say that $\mathbf{B}$ is a *square root* of $\Sigma > \mathbf{0}$, written $\mathbf{B} = \sqrt{\Sigma}$, if $\mathbf{B}\mathbf{B}^t = \Sigma$. Every $\Sigma > \mathbf{0}$ has a square root $\mathbf{B} = \mathbf{Q}\mathbf{D}$, where $\Sigma = \mathbf{Q}\mathbf{D}^2\mathbf{Q}^t$ is the spectral decomposition of Σ as above. Moreover, the square root is unique up to right-multiplication by an orthogonal matrix, i.e., $\mathbf{B}' = \sqrt{\Sigma}$ if and only if $\mathbf{B}' = \mathbf{B}\mathbf{P}$ for some orthogonal matrix $\mathbf{P}$. A square root of particular interest is given by the *Cholesky decomposition* $\Sigma = \mathbf{L}\mathbf{L}^t$, where $\mathbf{L}$ is a (unique) lower-triangular matrix. Given Σ, its Cholesky decomposition can be computed efficiently in fewer than n^3 multiplication and addition operations (on real numbers of sufficient precision).

For a nonsingular matrix $\mathbf{B}$, a *singular value decomposition* is $\mathbf{B} = \mathbf{Q}\mathbf{D}\mathbf{P}^t$, where $\mathbf{Q}, \mathbf{P} \in \mathbb{R}^{n \times n}$ are orthogonal matrices, and $\mathbf{D}$ is a diagonal matrix with positive entries $s_i > 0$ (called the singular values) on the diagonal, in non-increasing order. Under this convention, $\mathbf{D}$ is uniquely determined, and $s_1(\mathbf{B}) = \max_{\mathbf{u}} \|\mathbf{B}\mathbf{u}\| = \max_{\mathbf{u}} \|\mathbf{B}^t\mathbf{u}\|$, where the maximum is taken over all unit vectors $\mathbf{u} \in \mathbb{R}^n$. Note that

$$\Sigma = \mathbf{B}\mathbf{B}^t = \mathbf{Q}\mathbf{D}\mathbf{P}^t\mathbf{P}\mathbf{D}^t\mathbf{Q}^t = \mathbf{Q}\mathbf{D}^2\mathbf{Q}^t,$$

so the eigenvalues of Σ are the squares of the singular values of $\mathbf{B}$.

2.3 Gaussians

The n-dimensional Gaussian function $\rho : \mathbb{R}^n \to (0, 1]$ is defined as

$$\rho(\mathbf{x}) \stackrel{\Delta}{=} \exp(-\pi \cdot \|\mathbf{x}\|^2) = \exp(-\pi \cdot \langle \mathbf{x}, \mathbf{x} \rangle).$$

Applying a linear transformation given by a nonsingular matrix $\mathbf{B}$ yields the Gaussian function

$$\rho_{\mathbf{B}}(\mathbf{x}) \stackrel{\Delta}{=} \rho(\mathbf{B}^{-1}\mathbf{x}) = \exp\left(-\pi \cdot \left\langle \mathbf{B}^{-1}\mathbf{x}, \mathbf{B}^{-1}\mathbf{x} \right\rangle\right) = \exp\left(-\pi \cdot \mathbf{x}^t \Sigma^{-1} \mathbf{x}\right),$$

where $\Sigma = \mathbf{B}\mathbf{B}^t > \mathbf{0}$. Because $\rho_{\mathbf{B}}$ is distinguished only up to Σ, we usually refer to it as $\rho_{\sqrt{\Sigma}}$.

Normalizing $\rho_{\sqrt{\Sigma}}$ by its total measure $\int_{\mathbb{R}^n} \rho_{\sqrt{\Sigma}}(\mathbf{x})\, d\mathbf{x} = \sqrt{\det \Sigma}$ over $\mathbb{R}^n$, we obtain the probability distribution function of the (continuous) *Gaussian distribution* $D_{\sqrt{\Sigma}}$. It is easy to check that a random variable $\mathbf{x}$ having distribution

$D_{\sqrt{\Sigma}}$ can be written as $\sqrt{\Sigma} \cdot \mathbf{z}$, where $\mathbf{z}$ has spherical Gaussian distribution D_1. Therefore, the random variable $\mathbf{x}$ has *covariance*

$$\mathop{\mathrm{E}}_{\mathbf{x} \sim D_{\sqrt{\Sigma}}} \left[\mathbf{x} \cdot \mathbf{x}^t\right] = \sqrt{\Sigma} \cdot \mathop{\mathrm{E}}_{\mathbf{z} \sim D_1} \left[\mathbf{z} \cdot \mathbf{z}^t\right] \cdot \sqrt{\Sigma}^t = \sqrt{\Sigma} \cdot \frac{\mathbf{I}}{2\pi} \cdot \sqrt{\Sigma}^t = \frac{\Sigma}{2\pi},$$

by linearity of expectation. (The $\frac{\mathbf{I}}{2\pi}$ covariance of $\mathbf{z} \sim D_1$ arises from the independence of its entries, which are each distributed as D_1 in one dimension, and have variance $\frac{1}{2\pi}$.) *For convenience, in this paper we implicitly scale all covariance matrices by a 2π factor, and refer to Σ as the covariance matrix of $D_{\sqrt{\Sigma}}$.*

The following standard fact, which will be central to the analysis of our sampling algorithms, characterizes the product of two Gaussian functions.

Fact 1. *Let $\Sigma_1, \Sigma_2 > \mathbf{0}$ be positive definite matrices, let $\Sigma_0 = \Sigma_1 + \Sigma_2 > \mathbf{0}$ and $\Sigma_3^{-1} = \Sigma_1^{-1} + \Sigma_2^{-1} > \mathbf{0}$, let $\mathbf{x}, \mathbf{c}_1, \mathbf{c}_2 \in \mathbb{R}^n$ be arbitrary, and let $\mathbf{c}_3 \in \mathbb{R}^n$ be such that $\Sigma_3^{-1}\mathbf{c}_3 = \Sigma_1^{-1}\mathbf{c}_1 + \Sigma_2^{-1}\mathbf{c}_2$. Then*

$$\rho_{\sqrt{\Sigma_1}}(\mathbf{x} - \mathbf{c}_1) \cdot \rho_{\sqrt{\Sigma_2}}(\mathbf{x} - \mathbf{c}_2) = \rho_{\sqrt{\Sigma_0}}(\mathbf{c}_1 - \mathbf{c}_2) \cdot \rho_{\sqrt{\Sigma_3}}(\mathbf{x} - \mathbf{c}_3).$$

2.4 Gaussians on Lattices

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 nonsingular *basis* $\mathbf{B} \in \mathbb{R}^{n \times n}$, as the set $\Lambda = \mathbf{B} \cdot \mathbb{Z}^n = \{\mathbf{Bz} : \mathbf{z} \in \mathbb{Z}^n\}$. When $n \geq 2$, every lattice has infinitely many bases, which are related by unimodular transformations: $\mathbf{B'}$ and $\mathbf{B}$ generate the same lattice if and only if $\mathbf{B'} = \mathbf{BU}$ for some unimodular $\mathbf{U} \in \mathbb{Z}^{n \times n}$. The *dual* lattice of Λ is defined as $\Lambda^* = \{\mathbf{w} \in \mathbb{R}^n : \langle \mathbf{x}, \mathbf{w} \rangle \in \mathbb{Z} \; \forall \; \mathbf{x} \in \Lambda\}$. (We only need this notion for defining the smoothing parameter of a lattice; see below.)

Let $\Lambda \subset \mathbb{R}^n$ be a lattice, let $\mathbf{c} \in \mathbb{R}^n$, and let $\Sigma > \mathbf{0}$ be a positive definite matrix. The *discrete Gaussian distribution* $D_{\Lambda + \mathbf{c}, \sqrt{\Sigma}}$ is simply the Gaussian distribution restricted so that its support is the coset $\Lambda + \mathbf{c}$. That is, for all $\mathbf{x} \in \Lambda + \mathbf{c}$,

$$D_{\Lambda + \mathbf{c}, \sqrt{\Sigma}}(\mathbf{x}) = \frac{\rho_{\sqrt{\Sigma}}(\mathbf{x})}{\rho_{\sqrt{\Sigma}}(\Lambda + \mathbf{c})} \propto \rho_{\sqrt{\Sigma}}(\mathbf{x}).$$

We recall the definition of the *smoothing parameter* from [17].

Definition 1. *For a lattice Λ and positive real $\epsilon > 0$, the smoothing parameter $\eta_\epsilon(\Lambda)$ is the smallest real $s > 0$ such that $\rho_{1/s}(\Lambda^* \backslash \{\mathbf{0}\}) \leq \epsilon$.*

Observe that if Λ_1 is a sublattice of a lattice Λ_0, then $\eta_\epsilon(\Lambda_1) \geq \eta_\epsilon(\Lambda_0)$ for any $\epsilon > 0$, because $\Lambda_0^* \subseteq \Lambda_1^*$ and hence $\rho_{1/s}(\Lambda_0^* \setminus \{\mathbf{0}\}) \leq \rho_{1/s}(\Lambda_1^* \setminus \{\mathbf{0}\})$ by positivity of $\rho_{1/s}$.

Note that the smoothing parameter as defined above is a scalar; in this work we need to extend the notion to positive definite matrices.

Definition 2. *Let $\Sigma > \mathbf{0}$ be any positive definite matrix. We say that $\sqrt{\Sigma} \geq \eta_\epsilon(\Lambda)$ if $\rho_{\sqrt{\Sigma^{-1}}}(\Lambda^* \backslash \{\mathbf{0}\}) = \rho(\sqrt{\Sigma} \cdot \Lambda^* \backslash \{\mathbf{0}\}) \leq \epsilon$, i.e., if $\eta_\epsilon(\sqrt{\Sigma^{-1}} \cdot \Lambda) \leq 1$.*

Lemma 1 (Corollary of [17, Lemma 4.4]). *Let Λ be any n-dimensional lattice. For any $\epsilon \in (0,1)$, $\Sigma > \mathbf{0}$ such that $\sqrt{\Sigma} \geq \eta_\epsilon(\Lambda)$, and any $\mathbf{c} \in \mathbb{R}^n$,*

$$\rho_{\sqrt{\Sigma}}(\Lambda + \mathbf{c}) \in [\tfrac{1-\epsilon}{1+\epsilon}, 1] \cdot \rho_{\sqrt{\Sigma}}(\Lambda).$$

Proof. Follows directly by applying $\sqrt{\Sigma^{-1}}$ as a linear transform to Λ, and by $\eta_\epsilon(\sqrt{\Sigma^{-1}} \cdot \Lambda) \leq 1$.

Lemma 2 (Special case of [17, Lemma 3.3]). *For any $\epsilon > 0$,*

$$\eta_\epsilon(\mathbb{Z}^n) \leq \sqrt{\ln(2n(1+1/\epsilon))/\pi}.$$

In particular, for any $\omega(\sqrt{\log n})$ function, there is a negligible $\epsilon = \epsilon(n)$ such that $\eta_\epsilon(\mathbb{Z}^n) \leq \omega(\sqrt{\log n})$.

3 Analysis of 'Convolved' Discrete Gaussians

In this section we prove some general facts about 'convolutions' of (possibly non-spherical) discrete Gaussian distributions, which are important for the conception and analysis of our sampling algorithm; we expect these facts to have other applications as well. (Strictly speaking, the probabilistic experiments that we analyze are not true convolutions, because we are adding random variables that are not formally independent. However, the spirit of the experiment and its outcome are entirely 'convolution-like.')

Because the proof of the theorem is rather technical, the reader who is interested in applications may wish to skip ahead to the next section after understanding the theorem statement.

Theorem 1. *Let $\Sigma_1, \Sigma_2 > \mathbf{0}$ be positive definite matrices, with $\Sigma = \Sigma_1 + \Sigma_2 > \mathbf{0}$ and $\Sigma_3^{-1} = \Sigma_1^{-1} + \Sigma_2^{-2} > \mathbf{0}$. 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 $\mathbf{c}_1, \mathbf{c}_2 \in \mathbb{R}^n$ be arbitrary. Consider the following probabilistic experiment:*

$$\textit{Choose } \mathbf{x}_2 \leftarrow D_{\Lambda_2 + \mathbf{c}_2, \sqrt{\Sigma_2}}, \textit{ then choose } \mathbf{x}_1 \leftarrow \mathbf{x}_2 + D_{\Lambda_1 + \mathbf{c}_1 - \mathbf{x}_2, \sqrt{\Sigma_1}}.$$

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

If $\mathbf{x}_2$ is instead chosen from the continuous Gaussian distribution $D_{\sqrt{\Sigma_2}}$ over $\mathbb{R}^n$, the marginal distribution of $\mathbf{x}_1$ is as above, and the conditional distribution of $\mathbf{x}_2 \in \mathbb{R}^n$ given $\mathbf{x}_1 = \bar{\mathbf{x}}_1 \in \Lambda_1 + \mathbf{c}_1$ is within statistical distance 2ϵ of $\mathbf{c}_3 + D_{\sqrt{\Sigma_3}}$. (In this setting, the lattice Λ_2 and the hypothesis $\sqrt{\Sigma_3} \geq \eta_\epsilon(\Lambda_2)$ are unneeded.)

Proof. We start by analyzing the joint distribution of $\mathbf{x}_1 \in \Lambda_1 + \mathbf{c}_1$ and $\mathbf{x}_2 \in \Lambda_2 + \mathbf{c}_2$. Let $\bar{\mathbf{x}}_1 \in \Lambda_1 + \mathbf{c}_1$ and $\bar{\mathbf{x}}_2 \in \Lambda_2 + \mathbf{c}_2$ be arbitrary, and let $\Sigma_3^{-1}\mathbf{c}_3 = \Sigma_1^{-1}\bar{\mathbf{x}}_1$.

Then we have

$$\begin{aligned}
&\Pr[\mathbf{x}_1 = \bar{\mathbf{x}}_1 \wedge \mathbf{x}_2 = \bar{\mathbf{x}}_2] \\
&\quad = D_{\Lambda_1+\mathbf{c}_1-\bar{\mathbf{x}}_2,\sqrt{\Sigma_1}}(\bar{\mathbf{x}}_1 - \bar{\mathbf{x}}_2) \cdot D_{\Lambda_2+\mathbf{c}_2,\sqrt{\Sigma_2}}(\bar{\mathbf{x}}_2) && (1)\\
&\quad = \frac{\rho_{\sqrt{\Sigma_1}}(\bar{\mathbf{x}}_2 - \bar{\mathbf{x}}_1) \cdot \rho_{\sqrt{\Sigma_2}}(\bar{\mathbf{x}}_2)}{\rho_{\sqrt{\Sigma_1}}(\Lambda_1 + \mathbf{c}_1 - \bar{\mathbf{x}}_2) \cdot \rho_{\sqrt{\Sigma_2}}(\Lambda_2 + \mathbf{c}_2)} && (2)\\
&\quad \propto \frac{\rho_{\sqrt{\Sigma}}(\bar{\mathbf{x}}_1) \cdot \rho_{\sqrt{\Sigma_3}}(\bar{\mathbf{x}}_2 - \mathbf{c}_3)}{\rho_{\sqrt{\Sigma_1}}(\Lambda_1 + \mathbf{c}_1 - \bar{\mathbf{x}}_2)}. && (3)
\end{aligned}$$

Equation (1) is by construction; Equation (2) is by definition of $D_{\Lambda+\mathbf{c}}$ and by symmetry of $\rho_{\sqrt{\Sigma_1}}$; Equation (3) is by Fact 1. (The $\rho_{\sqrt{\Sigma_2}}(\Lambda_2 + \mathbf{c}_2)$ term in the denominator of (2) is the same for all $\bar{\mathbf{x}}_1, \bar{\mathbf{x}}_2$, so we can treat it as a constant of proportionality.)

We now analyze the marginal distribution of $\mathbf{x}_1$. For any $\bar{\mathbf{x}}_1 \in \Lambda_1 + \mathbf{c}_1$, let $\Sigma_3^{-1}\mathbf{c}_3 = \Sigma_1^{-1}\bar{\mathbf{x}}_1$ as above; then we have

$$\begin{aligned}
&\Pr[\mathbf{x}_1 = \bar{\mathbf{x}}_1] \\
&\quad = \sum_{\bar{\mathbf{x}}_2 \in \Lambda_2+\mathbf{c}_2} \Pr[\mathbf{x}_1 = \bar{\mathbf{x}}_1 \wedge \mathbf{x}_2 = \bar{\mathbf{x}}_2] && \text{(by construction)}\\
&\quad \propto \rho_{\sqrt{\Sigma}}(\bar{\mathbf{x}}_1) \cdot \sum_{\bar{\mathbf{x}}_2 \in \Lambda_2+\mathbf{c}_2} \frac{\rho_{\sqrt{\Sigma_3}}(\bar{\mathbf{x}}_2 - \mathbf{c}_3)}{\rho_{\sqrt{\Sigma_1}}(\Lambda_1 + \mathbf{c}_1 - \bar{\mathbf{x}}_2)} && \text{(Equation (3))}\\
&\quad \in \rho_{\sqrt{\Sigma}}(\bar{\mathbf{x}}_1) \cdot [1, \tfrac{1+\epsilon}{1-\epsilon}] \cdot \frac{\rho_{\sqrt{\Sigma_3}}(\Lambda_2 + \mathbf{c}_2 - \mathbf{c}_3)}{\rho_{\sqrt{\Sigma_1}}(\Lambda_1)} && (\sqrt{\Sigma_1} \geq \eta_\epsilon(\Lambda_1), \text{ Lemma 1})\\
&\quad \subseteq \rho_{\sqrt{\Sigma}}(\bar{\mathbf{x}}_1) \cdot [\tfrac{1-\epsilon}{1+\epsilon}, \tfrac{1+\epsilon}{1-\epsilon}] \cdot \frac{\rho_{\sqrt{\Sigma_3}}(\Lambda_2)}{\rho_{\sqrt{\Sigma_1}}(\Lambda_1)} && (\sqrt{\Sigma_3} \geq \eta_\epsilon(\Lambda_2), \text{ Lemma 1})\\
&\quad \propto \rho_{\sqrt{\Sigma}}(\bar{\mathbf{x}}_1) \cdot [\tfrac{1-\epsilon}{1+\epsilon}, \tfrac{1+\epsilon}{1-\epsilon}].
\end{aligned}$$

It follows that

$$\Pr[\mathbf{x}_1 = \bar{\mathbf{x}}_1] \in [(\tfrac{1-\epsilon}{1+\epsilon})^2, (\tfrac{1+\epsilon}{1-\epsilon})^2] \cdot \frac{\rho_{\sqrt{\Sigma}}(\bar{\mathbf{x}}_1)}{\rho_{\sqrt{\Sigma}}(\Lambda_1 + \mathbf{c}_1)} \subseteq [1-16\epsilon, 1+16\epsilon] \cdot D_{\Lambda_1+\mathbf{c}_1,\sqrt{\Sigma}}(\bar{\mathbf{x}}_1),$$

because $\epsilon \leq 1/2$. The claim on the marginal distribution of $\mathbf{x}_1$ follows by definition of statistical distance.

When $\mathbf{x}_2$ is chosen from the continuous Gaussian $D_{\sqrt{\Sigma_2}}$, the analysis is almost identical: we simply replace the summation over $\bar{\mathbf{x}}_2 \in \Lambda_2 + \mathbf{c}_2$ with integration over $\bar{\mathbf{x}}_2 \in \mathbb{R}^n$. Because $\int_{\bar{\mathbf{x}}_2} \rho_{\sqrt{\Sigma_3}}(\bar{\mathbf{x}}_2 - \mathbf{c}_3)\, d\bar{\mathbf{x}}_2 = \sqrt{\det \Sigma_3}$ is independent of $\mathbf{c}_3$, there is no need to invoke Lemma 1 a second time.

The analysis of the conditional distribution of $\mathbf{x}_2 \in \Lambda_2 + \mathbf{c}_2$ proceeds similarly; due to space restrictions, we omit here it an refer the reader to the full version.

4 Discrete Gaussian Sampling Algorithms

In this section we define and analyze some new discrete Gaussian sampling algorithms. We start in Section 4.1 by defining and analyzing some important

randomized-rounding subroutines. In Section 4.2 we describe a general-purpose (but unoptimized) sampling algorithm, then in Section 4.3 we describe a highly optimized sampling algorithm for q-ary lattices.

4.1 Randomized Rounding

We first need to define and analyze some simple 'randomized-rounding' operations from the reals to lattices, which are an important component of our sampling algorithms.

We start with a basic rounding operation from $\mathbb{R}$ to the integers $\mathbb{Z}$, denoted $\lfloor v \rceil_r$ for $v \in \mathbb{R}$ and some positive real 'rounding parameter' $r > 0$. The output of this operation is a random variable over $\mathbb{Z}$ having distribution $v + D_{\mathbb{Z}-v,r}$. Observe that for any integer $z \in \mathbb{Z}$, the random variables $\lfloor z + v \rceil_r$ and $z + \lfloor v \rceil_r$ are identically distributed; therefore, we sometimes assume that $v \in [0, 1)$ without loss of generality. We extend the rounding operation coordinate-wise to vectors $\mathbf{v} \in \mathbb{R}^n$, where each entry is rounded independently. It follows that for any $\mathbf{v} \in \mathbb{R}^n$ and $\bar{\mathbf{z}} \in \mathbb{Z}^n$,

$$\Pr\left[\lfloor \mathbf{v} \rceil_r = \bar{\mathbf{z}}\right] \propto \prod_{i \in [n]} \rho_r(\bar{z}_i - v_i) = \exp(-\pi \|\bar{\mathbf{z}} - \mathbf{v}\|^2 / r^2) = \rho_r(\bar{\mathbf{z}} - \mathbf{v}).$$

That is, $\lfloor \mathbf{v} \rceil_r$ has distribution $\mathbf{v} + D_{\mathbb{Z}^n - \mathbf{v}, r}$, because the standard basis for $\mathbb{Z}^n$ is orthonormal.

The next lemma characterizes the distribution obtained by randomized rounding to an arbitrary lattice, using an arbitrary (possibly non-orthonormal) basis.

Lemma 3. *Let $\mathbf{B}$ be a basis of a lattice $\Lambda = \mathcal{L}(\mathbf{B})$, let $\Sigma = r^2 \cdot \mathbf{B}\mathbf{B}^t$ for some real $r > 0$, and let $\mathbf{t} \in \mathbb{R}^n$ be arbitrary. The random variable $\mathbf{x} = \mathbf{t} - \mathbf{B}\lfloor \mathbf{B}^{-1}\mathbf{t} \rceil_r$ has distribution $D_{\Lambda + \mathbf{t}, \sqrt{\Sigma}}$.*

Proof. Let $\mathbf{v} = \mathbf{B}^{-1}\mathbf{t}$. The support of $\lfloor \mathbf{v} \rceil_r$ is $\mathbb{Z}^n$, so the support of $\mathbf{x}$ is $\mathbf{t} - \mathbf{B} \cdot \mathbb{Z}^n = \Lambda + \mathbf{t}$. Now for any $\bar{\mathbf{x}} = \mathbf{t} - \mathbf{B}\bar{\mathbf{z}}$ where $\bar{\mathbf{z}} \in \mathbb{Z}^n$, we have $\mathbf{x} = \bar{\mathbf{x}}$ if and only if $\lfloor \mathbf{v} \rceil_r = \bar{\mathbf{z}}$. As desired, this event occurs with probability proportional to

$$\rho_r(\bar{\mathbf{z}} - \mathbf{v}) = \rho_r(\mathbf{B}^{-1}(\mathbf{t} - \bar{\mathbf{x}}) - \mathbf{B}^{-1}\mathbf{t}) = \rho_r(-\mathbf{B}^{-1}\bar{\mathbf{x}}) = \rho_{r\mathbf{B}}(\bar{\mathbf{x}}) = \rho_{\sqrt{\Sigma}}(\bar{\mathbf{x}}).$$

Efficient rounding. In [8] it is shown how to sample from $D_{\mathbb{Z}-v,r}$ for any $v \in \mathbb{R}$ and $r > 0$, by rejection sampling. While the algorithm requires only $\mathrm{poly}(\log n)$ iterations before terminating, its concrete running time and randomness complexity are not entirely suitable for practical implementations.

In this work, we can sample from $v + D_{\mathbb{Z}-v,r}$ more efficiently because r is always fixed, known in advance, and relatively small (about $\sqrt{\log n}$). Specifically, given r and $v \in \mathbb{R}$ we can (pre)compute a compact table of the approximate cumulative distribution function of $\lfloor v \rceil_r$, i.e., the probabilities

$$p_{\bar{z}} := \Pr[v + D_{\mathbb{Z}-v,r} \leq \bar{z}]$$

for each $\bar{z} \in \mathbb{Z}$ in an interval $[v - r \cdot \omega(\sqrt{\log n}), v + r \cdot \omega(\sqrt{\log n})]$. (Outside of that interval are the tails of the distribution, which carry negligible probability mass.) Then we can sample directly from $v + D_{\mathbb{Z}-v,r}$ by choosing a uniformly random $x \in [0, 1)$ and performing a binary search through the table for the $\bar{z} \in \mathbb{Z}$ such that $x \in [p_{\bar{z}-1}, p_{\bar{z}})$. Moreover, the bits of x may be chosen 'lazily,' from most- to least-significant, until $\bar{z}$ is determined uniquely. To sample within a $\mathrm{negl}(n)$ statistical distance of the desired distribution, these operations can all be implemented in time $\mathrm{poly}(\log n)$.

4.2 Generic Sampling Algorithm

Here we apply Theorem 1 to sample from a discrete Gaussian of any sufficiently large covariance, given a good enough basis of the lattice. This procedure, described in Algorithm 1, serves mainly as a 'proof of concept' and a warm-up for our main algorithm on q-ary lattices. As such, it is not optimized for runtime efficiency (because it uses arbitrary-precision real operations), though it is still fully parallelizable and offline/online.

Algorithm 1 Generic algorithm $\mathsf{SampleD}(\mathbf{B}_1, r, \Sigma, \mathbf{c})$ for sampling from a discrete Gaussian distribution.

Input:

Offline phase: Basis $\mathbf{B}_1$ of a lattice $\Lambda = \mathcal{L}(\mathbf{B}_1)$, rounding parameter $r = \omega(\sqrt{\log n})$, and positive definite covariance matrix $\Sigma > \Sigma_1 = r^2 \cdot \mathbf{B}_1\mathbf{B}_1^t$.

Online phase: a vector $\mathbf{c} \in \mathbb{R}^n$.

Output: A vector $\mathbf{x} \in \Lambda + \mathbf{c}$ drawn from a distribution within $\mathrm{negl}(n)$ statistical distance of $D_{\Lambda+\mathbf{c},\sqrt{\Sigma}}$.

Offline phase:

1: Let $\Sigma_2 = \Sigma - \Sigma_1 > \mathbf{0}$, and compute some $\mathbf{B}_2 = \sqrt{\Sigma_2}$.

2: Before each call to the online phase, choose a fresh $\mathbf{x}_2 \leftarrow D_{\sqrt{\Sigma_2}}$, as $\mathbf{x}_2 \leftarrow \mathbf{B}_2 \cdot D_1$.

Online phase:

3: **return** $\mathbf{x} \leftarrow \mathbf{c} - \mathbf{B}_1 \lfloor \mathbf{B}_1^{-1}(\mathbf{c} - \mathbf{x}_2) \rceil_r$.

Theorem 2. *Algorithm 1 is correct, and for any $P \in [1, n^2]$, its online phase can be executed in parallel by P processors that each perform $O(n^2/P)$ operations on real numbers (of sufficiently high precision).*

Proof. We first show correctness. Let $\Sigma, \Sigma_1, \Sigma_2$ be as in Algorithm 1. The output $\mathbf{x}$ is distributed as

$$\mathbf{x} = \mathbf{x}_2 + (\mathbf{c} - \mathbf{x}_2) - \mathbf{B}_1 \left\lfloor \mathbf{B}_1^{-1}(\mathbf{c} - \mathbf{x}_2) \right\rceil_r ,$$

where $\mathbf{x}_2$ has distribution $D_{\sqrt{\Sigma_2}}$. By Lemma 3 with $\mathbf{t} = (\mathbf{c} - \mathbf{x}_2)$, we see that $\mathbf{x}$ has distribution $\mathbf{x}_2 + D_{\Lambda+\mathbf{c}-\mathbf{x}_2,\sqrt{\Sigma_1}}$. Now because $\Lambda = \mathcal{L}(\mathbf{B}_1) = \mathbf{B}_1 \cdot \mathbb{Z}^n$, we have $\sqrt{\Sigma_1} = r\mathbf{B}_1 \geq \eta_\epsilon(\Lambda)$ for some negligible $\epsilon = \epsilon(n)$, by Definition 2 and Lemma 2.

Therefore, by the second part of Theorem 1, $\mathbf{x}$ has distribution within $\mathrm{negl}(n)$ statistical distance of $D_{\Lambda+\mathbf{c},\sqrt{\Sigma}}$.

To parallelize the algorithm, simply observe that $\mathbf{B}_1^{-1}$ can be precomputed in the offline phase, and that the matrix-vector products and randomized rounding can all be executed in parallel on P processors in the natural way.

4.3 Efficient Sampling Algorithm for q-ary Lattices

Algorithm 2 is an optimized sampling algorithm for q-ary (integral) lattices Λ, i.e., lattices for which $q\mathbb{Z}^n \subseteq \Lambda \subseteq \mathbb{Z}^n$ for some positive integer q. These include NTRU lattices [10], as well as the family of lattices for which Ajtai [3] first demonstrated worst-case hardness.

Algorithm 2 Efficient algorithm $\mathsf{SampleD}(\mathbf{B}_1, r, \Sigma, \mathbf{c})$ for sampling a discrete Gaussian over a q-ary lattice.

Input:

Offline phase: Basis $\mathbf{B}_1$ of a q-ary integer lattice $\Lambda = \mathcal{L}(\mathbf{B}_1)$, rounding parameter $r = \omega(\sqrt{\log n})$, and positive definite covariance matrix $\Sigma \geq r^2 \cdot (4\mathbf{B}_1\mathbf{B}_1^t + \mathbf{I})$.

Online phase: a vector $\mathbf{c} \in \mathbb{Z}^n$.

Output: A vector $\mathbf{x} \in \Lambda + \mathbf{c}$ drawn from a distribution within $\mathrm{negl}(n)$ statistical distance of $D_{\Lambda+\mathbf{c},\sqrt{\Sigma}}$.

Offline phase:

1: Compute $\mathbf{Z} = q \cdot \mathbf{B}_1^{-1} \in \mathbb{Z}^{n \times n}$.

2: Let $\Sigma_1 = 2r^2 \cdot \mathbf{B}_1\mathbf{B}_1^t$, let $\Sigma_2 = \Sigma - \Sigma_1 \geq r^2 \cdot (2\mathbf{B}_1\mathbf{B}_1^t + \mathbf{I})$, and compute some $\mathbf{B}_2 = \sqrt{\Sigma_2 - r^2}$.

3: Before each call to the online phase, choose a fresh $\mathbf{x}_2$ from $D_{\mathbb{Z}^n,\sqrt{\Sigma_2}}$ by letting $\mathbf{x}_2 \leftarrow \lfloor \mathbf{B}_2 \cdot D_1 \rceil_r$.

Online phase:

4: **return** $\mathbf{x} \leftarrow \mathbf{c} - \mathbf{B}_1 \left\lfloor \frac{\mathbf{Z}(\mathbf{c}-\mathbf{x}_2)}{q} \right\rceil_r$.

Note that Algorithm 2 samples from the coset $\Lambda + \mathbf{c}$ for a given *integral* vector $\mathbf{c} \in \mathbb{Z}^n$; as we shall see, this allows for certain optimizations. Fortunately, all known cryptographic applications of Gaussian sampling over q-ary lattices use an integral $\mathbf{c}$. Also note that the algorithm will typically be used to sample from a *spherical* discrete Gaussian, i.e., one for which the covariance matrix $\Sigma = s^2\mathbf{I}$ for some real $s > 0$. As long as s slightly exceeds the largest singular value of $\mathbf{B}_1$, i.e., $s \geq r \cdot (2s_1(\mathbf{B}_1) + 1)$ for some $r = \omega(\sqrt{\log n})$, then we have $\Sigma \geq r^2 \cdot (4\mathbf{B}_1\mathbf{B}_1^t + \mathbf{I})$ as required by the algorithm.

Theorem 3. *Algorithm 2 is correct, and for any $P \in [1, n^2]$, its online phase can be implemented in parallel by P processors that each perform at most $\lceil n/P \rceil$ randomized-rounding operations on rational numbers from the set $\{\frac{0}{q}, \frac{1}{q}, \ldots, \frac{q-1}{q}\}$, and $O(n^2/P)$ integer operations.*

When the width of the desired Gaussian distribution is much less than q, which is the case in all known cryptographic applications of the sampling algorithm, all the integer operations in the online phase may be performed modulo either q or q^2; see the discussion following the proof for details.

Proof. First observe that because $\Lambda = \mathcal{L}(\mathbf{B}_1)$ is q-ary, i.e., $q\mathbb{Z}^m \subseteq \Lambda$, there exists an integral matrix $\mathbf{Z} \in \mathbb{Z}^{n\times n}$ such that $\mathbf{B}_1\mathbf{Z} = q\cdot\mathbf{I}$. Therefore, $\mathbf{Z} = q\cdot\mathbf{B}_1^{-1} \in \mathbb{Z}^{n\times n}$ as stated in Step 1 of the algorithm. We also need to verify that $\mathbf{x}_2 \leftarrow \lfloor \mathbf{B}_2 \cdot D_1 \rceil_r$ has distribution $D_{\mathbb{Z}^n,\sqrt{\Sigma_2}}$ in Step 3. Let $\mathbf{w} \in \mathbb{R}^n$ have distribution $\mathbf{B}_2 \cdot D_1 = D_{\sqrt{\Sigma_2 - r^2}}$. Then $\mathbf{x}_2$ has distribution

$$\lfloor \mathbf{w} \rceil_r = \mathbf{w} + (-\mathbf{w} + \lfloor \mathbf{w} \rceil_r) = \mathbf{w} + D_{\mathbb{Z}^n - \mathbf{w}, r},$$

by Lemma 3 (using the standard basis for $\mathbb{Z}^n$). Then because $r \geq \eta_\epsilon(\mathbb{Z}^n)$ for some negligible $\epsilon = \epsilon(n)$ and $\mathbf{B}_2\mathbf{B}_2^t + r^2 = \Sigma_2$, by Theorem 1 we conclude that $\mathbf{x}_2$ has distribution $D_{\mathbb{Z}^n,\sqrt{\Sigma_2}}$ as desired.

We now analyze the online phase, and show correctness. Because $\mathbf{B}_1^{-1} = \mathbf{Z}/q$, the algorithm's output $\mathbf{x}$ is distributed as

$$\mathbf{x}_2 + (\mathbf{c} - \mathbf{x}_2) - \mathbf{B}_1 \lfloor \mathbf{B}_1^{-1}(\mathbf{c} - \mathbf{x}_2) \rceil_r.$$

We would like to apply Lemma 3 (with $\mathbf{t} = \mathbf{c} - \mathbf{x}_2$) and Theorem 1 (with $\Lambda_1 = \Lambda$, $\Lambda_2 = \mathbb{Z}^n$, $\mathbf{c}_1 = \mathbf{c}$, and $\mathbf{c}_2 = \mathbf{0}$) to conclude that $\mathbf{x}$ has distribution within $\mathrm{negl}(n)$ statistical distance of $D_{\Lambda+\mathbf{c},\sqrt{\Sigma}}$. To do so, we merely need to check that the hypotheses of Theorem 1 are satisfied, namely, that $\sqrt{\Sigma_1} \geq \eta_\epsilon(\Lambda)$ and $\sqrt{\Sigma_3} \geq \eta_\epsilon(\mathbb{Z}^n)$ for some negligible $\epsilon = \epsilon(n)$, where $\Sigma_3^{-1} = \Sigma_1^{-1} + \Sigma_2^{-1}$.

For the first hypothesis, we have $\sqrt{\Sigma_1} = 2r \cdot \mathbf{B}_1 \geq \eta_\epsilon(\Lambda)$ because $\Lambda = \mathbf{B}_1 \cdot \mathbb{Z}^n$, and by Definition 2 and Lemma 2. For the second hypothesis, we have

$$\Sigma_3^{-1} = \Sigma_1^{-1} + \Sigma_2^{-1} \leq 2 \cdot \left(2r^2\mathbf{B}_1\mathbf{B}_1^t\right)^{-1} = \left(r^2\mathbf{B}_1\mathbf{B}_1^t\right)^{-1}.$$

Therefore, $\sqrt{\Sigma_3} \geq \eta_\epsilon(\Lambda) \geq \eta_\epsilon(\mathbb{Z}^n)$, as desired. This completes the proof of correctness.

For the parallelism claim, observe that computing $\mathbf{Z}(\mathbf{c} - \mathbf{x}_2)$ and the final multiplication by $\mathbf{B}_1$ can be done in parallel using P processors that each perform $O(n^2/P)$ integer operations. (See below for a discussion of the sizes of the integers involved in these operations.) Moreover, the fractional parts of the n-dimensional vector $\frac{\mathbf{Z}(\mathbf{c}-\mathbf{x}_2)}{q}$ are all in the set $\{\frac{0}{q}, \ldots, \frac{q-1}{q}\}$, and rounding these n entries may be done independently in parallel.

Implementation notes. For a practical implementation, Algorithm 2 admits several additional optimizations, which we discuss briefly here.

In all cryptographic applications of Gaussian sampling on q-ary lattices, the length of the sampled vector is significantly shorter than q, i.e., its entries lie within a narrow interval around 0. Therefore, it suffices for the sampling algorithm to compute its output modulo q, using the integers $\{-\lfloor \frac{q}{2} \rfloor, \ldots, \lfloor \frac{q-1}{2} \rfloor\}$ as the set of canonical representatives. For this purpose, the final multiplication by the

input basis $\mathbf{B}_1$ need only be performed modulo q. Similarly, $\mathbf{Z}$ and $\mathbf{Z}(\mathbf{c} - \mathbf{x}_2)$ need only be computed modulo q^2, because we are only concerned with the value of $\frac{\mathbf{Z}(\mathbf{c}-\mathbf{x}_2)}{q}$ modulo q.

Because all the randomized-rounding steps are performed on rationals whose fractional parts are in $\{\frac{0}{q}, \ldots, \frac{q-1}{q}\}$, if q is reasonably small it may be worthwhile (for faster rounding) to precompute the tables of the cumulative distribution functions for all q possibilities. Alternatively (or in addition), during the offline phase the algorithm could precompute and cache a few rounded samples for each of the q possibilities, consuming them as needed in the online phase.

5 Singular Value Bounds

In this section we give bounds on the largest singular value of a basis $\mathbf{B}$ and relate them to other geometric quantities that are relevant to the prior sampling algorithm of [8].

5.1 General Bounds

The *Gram-Schmidt orthogonalization* of a nonsingular matrix $\mathbf{B}$ is $\mathbf{B} = \mathbf{QG}$, where $\mathbf{Q}$ is an orthogonal matrix and $\mathbf{G}$ is right-triangular, with positive diagonal entries $g_{i,i} > 0$ (without loss of generality). The Gram-Schmidt vectors for $\mathbf{B}$ are $\widetilde{\mathbf{b}_i} = g_{i,i} \cdot \mathbf{q}_i$. That is, $\widetilde{\mathbf{b}_1} = \mathbf{b}_1$, and $\widetilde{\mathbf{b}_i}$ is the component of $\mathbf{b}_i$ orthogonal to the linear span of $\mathbf{b}_1, \ldots, \mathbf{b}_{i-1}$. The Gram-Schmidt orthogonalization can be computed efficiently in a corresponding iterative manner.

Let $\mathbf{B} = \mathbf{QG}$ be the Gram-Schmidt orthogonalization of $\mathbf{B}$. Without loss of generality we can assume that $\mathbf{B}$ is *size-reduced*, i.e., that $|g_{i,j}| \leq g_{i,i}/2$ for every $i < j$. This condition can be achieved efficiently by the following process: for each $j = 1, \ldots, n$, and for each $i = j - 1, \ldots, 1$, replace $\mathbf{b}_j$ by $\mathbf{b}_j - \lfloor g_{i,j}/g_{i,i} \rceil \cdot \mathbf{b}_i$. Note that the size reduction process leaves the lattice $\mathcal{L}(\mathbf{B})$ and Gram-Schmidt vectors $\widetilde{\mathbf{b}_i} = g_{i,i} \cdot \mathbf{q}_i$ unchanged. Note also that $\|\mathbf{g}_i\| \leq \sqrt{n} \cdot \max_i g_{i,i}$, by the Pythagorean theorem.

Lemma 4. *Let $\mathbf{B} \in \mathbb{R}^{n \times n}$ be a size-reduced nonsingular matrix. We have*

$$s_1(\mathbf{B}) \leq \sqrt{n} \cdot \sqrt{\sum_{i \in [n]} \|\widetilde{\mathbf{b}_i}\|^2} \leq n \cdot \|\widetilde{\mathbf{B}}\|.$$

The lemma is tight up to a constant factor, which may be seen by considering the right-triangular matrix with 1s on the diagonal and 1/2 in every entry above the diagonal.

Proof. Let $\mathbf{B}$ have Gram-Schmidt orthogonalization $\mathbf{B} = \mathbf{QG}$. We have

$$s_1(\mathbf{B}) = \max_{\mathbf{x}} \|\mathbf{B}^t \mathbf{x}\| = \max_{\mathbf{x}} \|\mathbf{G}^t \mathbf{x}\| \leq \sqrt{\sum_{i \in [n]} (\sqrt{n} \cdot g_{i,i})^2} = \sqrt{n} \cdot \sqrt{\sum_{i \in [n]} g_{i,i}^2},$$

where the maxima are taken over all unit vectors $\mathbf{x} \in \mathbb{R}^n$, the second equality uses the fact that $\mathbf{Q}$ is orthogonal, and the first inequality is by Cauchy-Schwarz.

5.2 Bases for Cryptographic Lattices

Ajtai [2] gave a procedure for generating a uniformly random q-ary lattice from a certain family of 'worst-case-hard' cryptographic lattices, together with a relatively short basis $\mathbf{B}$. Alwen and Peikert [4] recently improved and extended the construction to yield asymptotically optimal bounds on $\|\mathbf{B}\| = \max_i \|\mathbf{b}_i\|$ and $\|\widetilde{\mathbf{B}}\| = \max_i \|\widetilde{\mathbf{b}_i}\|$. Here we show that with a small modification, one of the constructions of [4] yields (with overwhelming probability) a basis whose largest singular value is within an $O(\sqrt{\log q})$ factor of $\|\widetilde{\mathbf{B}}\|$. It follows that our efficient Gaussian sampling algorithm is essentially as tight as the GPV algorithm on such bases. Due to space restrictions, we state the main result here; the proof may be found in the full version.

Lemma 5. *The (slightly modified) construction of [4, Section 3.2] outputs a basis* $\mathbf{B}$ *such that* $s_1(\mathbf{B}) = O(\sqrt{\log q}) \cdot \|\widetilde{\mathbf{B}}\|$ *with overwhelming probability.*

5.3 Gaussian-Distributed Bases

Here we show that for a lattice basis generated by choosing its vectors from a discrete Gaussian distribution over the lattice (following by some post-processing), the largest singular value $s_1(\mathbf{B})$ of the resulting basis is essentially the same as the maximal Gram-Schmidt length $\|\widetilde{\mathbf{B}}\|$ (with high probability). Such a bound is important because applications that use 'basis delegation,' such as the hierarchical ID-based encryption schemes of [6, 1], generate random bases in exactly the manner just described.

Due to space restrictions, we state the main theorem here; the proof may be found in the full version.

Theorem 4. *With overwhelming probability, the* RandBasis *algorithm of [6] outputs a basis* $\mathbf{T}$ *such that* $\|\widetilde{\mathbf{T}}\| \geq s \cdot \Omega(\sqrt{n})$*, and for any* $\omega(\sqrt{\log n})$ *function,* $s_1(\mathbf{T}) \leq s \cdot O(\sqrt{n}) \cdot \omega(\sqrt{\log n})$*. In particular,* $s_1(\mathbf{T})/\|\widetilde{\mathbf{T}}\| = \omega(\sqrt{\log n})$*.*

Acknowledgments. The author thanks Phong Nguyen and the anonymous CRYPTO'10 reviewers for helpful comments.

References

[1] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient lattice (H)IBE in the standard model. In *EUROCRYPT*, 2010. To appear.
[2] Miklós Ajtai. Generating hard instances of the short basis problem. In *ICALP*, pages 1–9, 1999.
[3] Miklós Ajtai. Generating hard instances of lattice problems. *Quaderni di Matematica*, 13:1–32, 2004. Preliminary version in STOC 1996.
[4] Joël Alwen and Chris Peikert. Generating shorter bases for hard random lattices. In *STACS*, pages 75–86, 2009.
[5] László Babai. On Lovász' lattice reduction and the nearest lattice point problem. *Combinatorica*, 6(1):1–13, 1986.

[6] David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai trees, or how to delegate a lattice basis. In *EUROCRYPT*, 2010. To appear.

[7] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *STOC*, pages 169–178, 2009.

[8] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *STOC*, pages 197–206, 2008.

[9] Jeffrey Hoffstein, Nick Howgrave-Graham, Jill Pipher, Joseph H. Silverman, and William Whyte. NTRUSIGN: Digital signatures using the NTRU lattice. In *CT-RSA*, pages 122–140, 2003.

[10] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. NTRU: A ring-based public key cryptosystem. In *ANTS*, pages 267–288, 1998.

[11] Philip N. Klein. Finding the closest lattice vector when it's unusually close. In *SODA*, pages 937–941, 2000.

[12] A. M. Kshirsagar. Bartlett decomposition and Wishart distribution. *The Annals of Mathematical Statistics*, 30(1):239–241, March 1959. Available at `http://www.jstor.org/stable/2237140`.

[13] Vadim Lyubashevsky and Daniele Micciancio. Asymptotically efficient lattice-based digital signatures. In *TCC*, pages 37–54, 2008.

[14] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In *EUROCRYPT*, 2010. To appear.

[15] Tal Malkin, Chris Peikert, Rocco A. Servedio, and Andrew Wan. Learning an overcomplete basis: Analysis of lattice-based signatures with perturbations. Manuscript, 2009.

[16] Daniele Micciancio. Generalized compact knapsacks, cyclic lattices, and efficient one-way functions. *Computational Complexity*, 16(4):365–411, 2007. Preliminary version in FOCS 2002.

[17] Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on Gaussian measures. *SIAM J. Comput.*, 37(1):267–302, 2007. Preliminary version in FOCS 2004.

[18] Phong Q. Nguyen and Oded Regev. Learning a parallelepiped: Cryptanalysis of GGH and NTRU signatures. *J. Cryptology*, 22(2):139–160, 2009. Preliminary version in Eurocrypt 2006.

[19] Chris Peikert and Vinod Vaikuntanathan. Noninteractive statistical zero-knowledge proofs for lattice problems. In *CRYPTO*, pages 536–553, 2008.

[20] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM*, 56(6), 2009. Preliminary version in STOC 2005.

[21] Damien Stehlé, Ron Steinfeld, Keisuke Tanaka, and Keita Xagawa. Efficient public key encryption based on ideal lattices. In *ASIACRYPT*, pages 617–635, 2009.