Faster Gaussian Sampling for Trapdoor Lattices with Arbitrary Modulus *

Nicholas Genise
ngenise@eng.ucsd.edu
UCSD

Daniele Micciancio
daniele@cs.ucsd.edu
UCSD

September 19, 2017

Abstract

We present improved algorithms for gaussian preimage sampling using the lattice trapdoors of (Micciancio and Peikert, CRYPTO 2012). The MP12 work only offered a highly optimized algorithm for the on-line stage of the computation in the special case when the lattice modulus q is a power of two. For arbitrary modulus q, the MP12 preimage sampling procedure resorted to general lattice algorithms with complexity cubic in the bitsize of the modulus (or quadratic, but with substantial preprocessing and storage overheads). Our new preimage sampling algorithm (for any modulus q) achieves linear complexity with very modest storage requirements, and experimentally outperforms the generic method of MP12 already for small values of q. As an additional contribution, we give a new, quasi-linear time algorithm for the off-line perturbation sampling phase of MP12 in the ring setting. Our algorithm is based on a variant of the Fast Fourier Orthogonalization (FFO) algorithm of (Ducas and Prest, ISSAC 2016), but avoids the need to precompute and store the FFO matrix by a careful rearrangement of the operations. All our algorithms are fairly simple, with small hidden constants, and offer a practical alternative to use the MP12 trapdoor lattices in a broad range of cryptographic applications.

1 Introduction

Lattice cryptography provides powerful techniques to build a wide range of advanced cryptographic primitives, like identity based encryption [32, 24, 11, 4, 2, 3], attribute based encryption [17, 16, 19, 34, 14], some types of fully homomorphic encryption and signatures [13, 12, 33, 36, 25], group signatures [37, 21, 44, 45, 54] and much more (e.g., see [56, 51, 10, 57, 60, 6, 46, 35]). Most of the advanced applications of lattice cryptography rely on a notion of strong lattice trapdoor, introduced in [32], which allows to sample points from an n-dimensional lattice L with a gaussian-like distribution. This gaussian sampling operation is often the main bottleneck in the implementation of advanced cryptographic functions that make use of strong lattice trapdoors, and improving the methods to generate and use lattice trapdoors has been the subject of several investigations [5, 32, 7, 55].

The current state of the art in lattice trapdoor generation and sampling is given by the work of Micciancio and Peiker [51], which introduces a new notion of lattice trapdoor, specialized to the type of q-ary lattices used in cryptography, i.e., integer lattices $L \subseteq \mathbb{Z}^n$ that are periodic modulo $q \cdot \mathbb{Z}^n$. The trapdoor is then used to efficiently sample lattice points with gaussian distribution around a given target. Building on techniques from [55], the sampling algorithm of [51] includes both an on-line and an off-line stage, and [51] focuses on improving the complexity of the on-line stage, which is far more critical in applications. Unfortunately, the most efficient algorithms proposed in [51] for (the on-line stage of) preimage sampling only apply to lattices with modulus $q = 2^k$ equal to a power of 2 (or, more generally, the power $q = p^k$ of a small prime p,) which is

*Research supported in part by the DARPA SafeWare program. Opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of DARPA.

	MP12	MP12	**This work**
modulus q	2^k	any	any
G-Sampling precomp.	—	$O(\log^3 q)$	—
G-Sampling space	$O(\log q)$	$O(\log^2 q)$	$O(\log q)$
G-Sampling time	$O(n \log q)$	$O(n \log^2 q)$	$O(n \log q)$

Table 1: Running time and storage of the (G-sampling) algorithm. G-Sampling running times are scaled by a factor n to take into account that each sample requires n independent calls to the underlying G-sampling operation.

not compatible with the functional or efficiency requirements of many applications. Moreover, only the on-line stage of [51] takes full advantage of the structure of algebraic lattices [50, 48, 49] typically employed in the efficient instantiation of lattice cryptography, and essential to reduce the running time of lattice operations from quadratic (in the lattice dimension) to quasi-linear. A straightforward implementation of the off-line stage (e.g., using a generic Cholesky decomposition algorithm) completely destroys the algebraic structure, and degrades the running time of the (off-line) algorithm from quasi-linear to quadratic or worse. For lattices over "power-of-two" cyclotomic rings (the most popular class of algebraic lattices used in cryptography), a much faster algorithm for the off-line stage was proposed by Ducas and Nguyen in [28, Section 6], and subsequently simplified, improved and extended to a more general class of cyclotomic rings by the *Fast Fourier Orthogonalization (FFO)* of Ducas and Prest [29].

Our Contribution We present improved algorithms for gaussian preimage sampling using the lattice trapdoors of [51]. Specifically, we present a new algorithm (for the on-line stage) capable of handling any modulus q (including the large prime moduli required by some applications) and still achieve the same level of performance of the specialized algorithm of [51] for power-of-two modulus $q = 2^k$. This improves the running time of [51] for arbitrary modulus from cubic $\log^3 q$ (or quadratic $\log^2 q$, using precomputation and a substantial amount of storage) to just linear in $\log q$ and with minimal storage requirements.

As an additional contribution, we present an improved algorithm for the off-line perturbation generation problem which takes full advantage of the algebraic structure of ring lattices. We remark that this problem can already be solved (in quasilinear time $\tilde{O}(n)$) using the (FFO) algorithm of [29], which first produces a compact representation of the orthogonalized lattice basis (or covariance matrix), and then uses it to quickly generate lattice samples. We improve on the algorithm of [29] on two fronts. First, the FFO algorithm is quasi-linear in the ring dimension, but quadratic in the module dimension (which, in our application, is $\log q$). We combine [29] with the "sparse matrix" optimization of [9] to yield an algorithm that is linear both in the ring dimension and $\log q$. Moreover, we provide a variant of the FFO algorithm that performs essentially the same operations as [29], but without requiring the precomputation and storage of the FFO (structured) matrix, thereby simplifying the implementation and improving the space complexity of [29].

The G-sampling improvements are summarized in Table 1. The improvements are not just asymptotic: our new algorithms are fairly simple, with small hidden constants, and include a careful choice of the parameters that allows to implement most steps using only integer arithmetic on very small numbers. In Section 3.3, we provide an experimental comparison showing that the new algorithm outperforms the generic method of [51] already for small values of the moduli, making it an attractive choice for implementations even in applications where the modulus $q = n^{O(1)}$ has logarithmic bit-size. For applications using an exponentially large $q = \exp(n)$, the projected performance improvements are dramatic. The concrete efficiency of our algorithms in the context of full blown cryptographic applications, has been recently confirmed by independent implementation efforts [39, 26, 38].

Technical details In order to describe our techniques, we need first to provide more details on the lattice trapdoor sampling problem. Given a lattice L and a target point $\mathbf{t}$, the lattice gaussian sampling problem asks to generate (possibly with the help of some trapdoor information) a random lattice point $\mathbf{v} \in L$ with

probability proportional to $\exp(-c\|\mathbf{v}-\mathbf{t}\|^2)$. Building on techniques from [55], this problem is solved in [51] by mapping L to a fixed (key independent) lattice G^n, generating a gaussian sample in G^n, and then mapping the result back to L. (The linear function T mapping G^n to L serves as the trapdoor.) Without further adjustments, this produces a lattice point in L with ellipsoidal gaussian distribution, with covariance which depends on the linear transformation T. In order to produce spherical samples (as required[1] by applications), [51] employs a perturbation technique of Peikert [55] which adds some noise (with complementary covariance) to the target $\mathbf{t}$, before using it as a center for the G^n-lattice sampling operation. In summary, the sampling algorithm of [55, 51] consists of two stages:

- an off-line (target independent) stage, which generates perturbation vectors with covariance matrix defined by the trapdoor transformation T, and
- an on-line (target dependent) stage which generates gaussian samples from an (easy to sample) lattice G^n.

Not much attention is paid in [51] to the perturbation generation, as it does not depend on the target vector $\mathbf{t}$, and it is far less time critical in applications.[2] As for the on-line stage, one of the properties that make the lattice G^n easy to sample is that it is the orthogonal sum of n copies of a $(\log q)$-dimensional lattice G. So, even using generic algorithms with quadratic running time, G sampling takes a total of $O(n \log^2 q)$ operations. For moduli $q = n^{O(1)}$ polynomial in the lattice dimension n, this results in quasilinear running time $O(n \log^2 n)$. However, since the G-sampling operation directly affects the on-line running time of the signing algorithm, even a polylogarithmic term $\log^2 q$ can be highly undesirable. To this end, [51] gives a particularly efficient (and easy to implement) algorithm for G-lattice sampling when the lattice modulus $q = 2^k$ is a power of 2 (or more generally, a power $q = p^k$ of a small prime p.) The running time of this specialized G-sampling algorithm is $\log q$, just linear in the lattice dimension, and has minimal (constant) storage requirements. Thanks to its simplicity and efficiency, this algorithm has quickly found its way in concrete implementations of lattice based cryptographic primitives (e.g., see [9]), largely solving the problem of efficient lattice sampling for $q = 2^k$. However, setting q to a power of 2 (or more generally, the power of a small prime), may be incompatible with applications and other techniques used in lattice cryptography, like *attribute based encryption (ABE)* schemes [14] and fast implementation via the *number theoretic transform* [47, 49]. For arbitrary modulus q, [51] falls back to generic algorithms (for arbitrary lattices) with quadratic complexity. This may still be acceptable when the modulus q is relatively small. But it is nevertheless undesirable, as even polylogarithmic factors have a significant impact on the practical performance of cryptographic functions (easily increasing running times by an order of magnitude), and can make applications completely unusable when the modulus $q = \exp(n)$ is exponentially large. The concrete example best well illustrates the limitations of [51] is the recent conjunction obfuscator of [20], which requires the modulus q to be prime with bitsize $\log(q) = O(n)$ linear in the security parameter. In this setting, the specialized algorithm of [51] (for $q = 2^k$) is not applicable, and using a generic algorithm slows down the on-line stage by a factor $O(n)$, or, more concretely, various orders of magnitude for typical parameter settings. Another, less drastic, example is the arithmetic circuit ABE scheme of [14] where q is $O(2^{n^\epsilon})$ for some fixed $0 < \epsilon < 1/2$. Here the slow down is asymptotically smaller, n^ϵ, but still polynomial in the security parameter n.

Unfortunately, the specialized algorithm from [51] makes critical use of the structure of the G-basis when $q = 2^k$, and is not easily adapted to other moduli. (See Section 3 for details.) In order to solve this problem we resort to the same approach used in [55, 51] to generate samples from arbitrary lattices: we map G to an even simpler lattice D using an easy to compute linear transformation T', perform the gaussian sampling in D, and map the result back to G. As usual, the error shape is corrected by including a perturbation term with appropriate covariance matrix. The main technical problem to be solved is to find a suitable linear transformation T' such that D can be efficiently sampled and perturbation terms can be easily

[1] More generally, applications require samples to be generated according to a distribution that does not depend on the trapdoor/secret key.

[2] E.g., in lattice based digital signature schemes [32, 51], the off-line computation depends only on the secret key, and can be performed in advance without knowing the message to be signed.

generated. In Section 3 we demonstrate a choice of transformation T' with all these desirable properties. In particular, using a carefully chosen transformation T', we obtain lattices D and perturbation matrices that are triangular, sparse, and whose entries admit a simple (and efficiently computable) closed formula expression. So, there is not even a need to store these sparse matrices explicitly, as their entries can be easily computed on the fly. This results in a G-sampling algorithm with linear running time, and minimal (constant) space requirements, beyond the space necessary to store the input, output and randomness of the algorithm.

Next, in Section 4, we turn to the problem of efficiently generating the perturbations of the off-line stage. Notice that generating these perturbations is a much harder problem than the one faced when mapping G to D (via T'). The difference is that while G, D, T' are fixed (sparse, carefully designed) matrices, the transformation T is a randomly chosen matrix that is used as secret key. In this setting, there is no hope to reduce the computation time to linear in the lattice dimension, because even reading/writing the matrix T can in general take quadratic time. Still, when using algebraic lattices, matrix T admits a compact (linear size) representation, and one can reasonably hope for faster perturbation generation algorithms. As already noted, this can be achieved using the Fast Fourier Orthogonalization algorithm of Ducas and Prest [29], which has running time quasilinear in the ring dimension, but quadratic in the dimension (over the ring) of the matrix T, which is $O(\log q)$ in our setting. As usual, while for polynomial moduli $q = n^{O(1)}$, this is only a polylogarithmic slow down, it can be quite significant in practice [9]. We improve on a direct application of the FFO algorithm by first employing an optimization of Bansarkhani and Buchmann [9] to exploint the sparsity of T. (This corresponds to the top level function SamplePz in Figure 4.) This optimization makes the computation linear in the dimension of T ($\log q$ in our setting), while keeping the quasilinear dependency on the ring dimension n from [29]. We further improve this combined algorithm by presenting a variant of FFO (described by the two mutually recursive functions SampleFz/Sample2z in Figure 4) that does not require the precomputation and storage of the FFO matrix.

Comparison with FFO Since our SamplePz function (Figure 4) uses a subprocedure SampleFz which is closely related to the FFO algorithm [29], we provide a detailed comparison between the two. We recall that FFO works by first computing a *binary tree* data structure [29, Algorithm 3], where the root node is labeled by an n-dimensional vector, its two children are labeled by $(n/2)$-dimensional vectors, and so on, all the way down to n leaves which are labeled with 1-dimensinal vectors. Then, [29, Algorithm 4] uses this binary tree data structure within a block/recursive variant of Babai's nearest plane algorithm.[3] Our SampleFz is based on the observation that one can blend/interleave the computation of [29, Algorithm 3] and [29, Algorithm 4], leading to a substantial (asymptotic) memory saving. Specifically, combining the two algorithms avoids the need to precompute and store the FFO binary tree data structure altogether, which is now implicitly generated, on the fly, one node/vector at a time, and discarding each node/vector as soon as possible in a way similar to a depth-first tree traversal. The resulting reduction in space complexity is easily estimated. The original FFO builds a tree with $\log n$ levels, where level l stores 2^l vectors in dimension $n/2^l$. So, the total storage requirement for each level is n, giving an overall space complexity of $n \log n$. Our FFO variant only stores one node/vector per level, and has space complexity $\sum_l (n/2^l) = 2n$, a $O(\log n)$ improvement over the space of original FFO algorithm. Moreover, the nodes/vectors are stored implicitly in the execution stack of the program, rather than an explicitly constructed binary tree data structure, yielding lower overhead and an algorithm that is easier to implement. For simplicity we specialized our presentation to power-of-two cyclotomics, which are the most commonly used in lattice cryptography, but everything works equally well for the larger class of cyclotomic rings considered in [29].

[3] Technically, [29, Algorithm 4] deterministically rounds a target point to a point in the lattice, rather than producing a probability distribution. But, as observed in [29], the algorithm is easily adapted to perform gaussian sampling by replacing deterministic rounding operations with probabilistic gaussian rounding.

2 Preliminaries

We denote the complex numbers as $\mathbb{C}$, the real numbers as $\mathbb{R}$, the rational numbers as $\mathbb{Q}$, and the integers as $\mathbb{Z}$. A number is denoted by a lower case letter, $n \in \mathbb{Z}$ for example. We denote the conjugate of a complex number y as y^*. When q is a positive integer, $\log q$ is short for its rounded up logarithm in base two, $\lceil \log_2 q \rceil$. A floating point number with mantissa length m representing $x \in \mathbb{R}$ is denoted as $\bar{x}$. The index set of the first n natural numbers is $[n] = \{1, \ldots, n\}$. Vectors are denoted by bold lower case letters, $\mathbf{v}$, and are in column form ($\mathbf{v}^T$ is a row vector) unless stated otherwise. The inner product of two vectors is $\langle \mathbf{x}, \mathbf{y} \rangle = \mathbf{x}^T \mathbf{y}$. We denote matrices with bold upper case letters $\mathbf{B}$ or with upper case Greek letters (for positive-definite matrices). In addition, we denote the transpose of a matrix as $\mathbf{B}^T$, and its Hermitian transpose as $\mathbf{B}^\dagger$. The entry of $\mathbf{B}$ in row i and column j is denoted $B_{i,j}$. Unless otherwise stated, the norm of a vector is the ℓ_2 norm. The norm of a matrix $\|\mathbf{B}\| = \max_i \|\mathbf{b}_i\|$ is the maximum norm of its column vectors. Given two probability distributions over a countable domain D, the statistical distance between them is $\Delta_{\text{SD}}(X, Y) = \frac{1}{2} \sum_{\omega \in D} |X(\omega) - Y(\omega)|$. In order to avoid tracing irrelevant terms in our statistical distance computations, we define $\hat{\epsilon} = \epsilon + O(\epsilon^2)$.

We denote a random variable x sampled from a distribution $\mathcal{D}$ as $x \leftarrow \mathcal{D}$. A random variable distributed as $\mathcal{D}$ is denoted $x \sim \mathcal{D}$. We denote an algorithm $\mathcal{A}$ with oracle access to another algorithm $\mathcal{B}$ (distribution $\mathcal{D}$) as $\mathcal{A}^{\mathcal{B}}$ ($\mathcal{A}^{\mathcal{D}}$).

The max-log, or ML, distance between two distributions was recently introduced by [53] in order to prove tighter bounds for concrete security. The *ML distance* between two discrete distributions over the same support, S, as

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

Let $\mathcal{P}, \mathcal{Q}$ be distributions over a countable domain again and let S be the support of $\mathcal{P}$. The *Rényi divergence* of order infinity of $\mathcal{Q}$ from $\mathcal{P}$ is

$$R_\infty(\mathcal{P} || \mathcal{Q}) = \max_{x \in S} \frac{\mathcal{P}(x)}{\mathcal{Q}(x)}.$$

Rényi divergence is used in [8] to yield a tighter security analysis than one using statistical distance.

2.1 Linear Algebra

The *Gram-Schmidt orthogonalization* of an ordered set of linearly independent vectors $\mathbf{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_k\}$ is $\widetilde{\mathbf{B}} = \{\widetilde{\mathbf{b}}_1, \ldots, \widetilde{\mathbf{b}}_k\}$ where each $\widetilde{\mathbf{b}}_i$ is the component of $\mathbf{b}_i$ orthogonal to $\text{span}(\mathbf{b}_1, \ldots, \mathbf{b}_{i-1})$. An *anti-cylic* matrix is an $n \times n$ matrix of the form

$$\begin{bmatrix} a_0 & -a_{n-1} & \ldots & -a_1 \\ a_1 & a_0 & \ldots & -a_2 \\ \vdots & \vdots & \ddots & \vdots \\ a_{n-1} & a_{n-2} & \ldots & a_0 \end{bmatrix}.$$

For any two (symmetric) matrices $\Sigma, \Gamma \in \mathbb{R}^{n \times n}$, we write $\Sigma \succeq \Gamma$ if $\mathbf{x}^T(\Sigma - \Gamma)\mathbf{x} \geq 0$ for all (nonzero) vectors $\mathbf{x} \in \mathbb{R}^n$, and $\Sigma \succ \Gamma$ if $\mathbf{x}^T(\Sigma - \Gamma)\mathbf{x} > 0$. It is easy to check that $\succeq$ is a partial order relation. Relations $\preceq$ and $\prec$ are defined symmetrically. When one of the two matrices $\Gamma = s\mathbf{I}$ is scalar, we simply write $\Sigma \succeq s$ or $\Sigma \preceq s$. A symmetric matrix $\Sigma \in \mathbb{R}^{n \times n}$ is called *positive definite* if $\Sigma \succ 0$, and *positive semidefinite* if $\Sigma \succeq 0$. Equivalently, Σ is positive semidefinite if and only if it can be written as $\Sigma = \mathbf{B}\mathbf{B}^T$ for some (square) matrix $\mathbf{B}$, called a *square root* of Σ and denoted $\mathbf{B} = \sqrt{\Sigma}$. (Notice that any $\Sigma \succ 0$ has infinitely many square roots $\mathbf{B} = \sqrt{\Sigma}$.) Σ is positive definite if and only if its square root $\mathbf{B}$ is a square nonsingular matrix. When $\mathbf{B}$ is upper (resp. lower) triangular, the factorization $\Sigma = \mathbf{B}\mathbf{B}^T$ is called the upper (resp. lower) triangular *Cholesky decomposition* of Σ. The Cholesky decomposition of any positive definite $\Sigma \in \mathbb{R}^{n \times n}$ can be computed with $O(n^3)$ floating point arithmetic operations. For any scalar s, $\Sigma \succ s$ if all eigenvalues of Σ are strictly greater than s. In particular, positive definite matrices are nonsingular.

For any $n \times n$ matrix $\mathbf{S}$ and non-empty index sets $I, J \subseteq \{1, \ldots, n\}$, we write $\mathbf{S}[I, J]$ for the $|I| \times |J|$ matrix obtained by selecting the elements at positions $(i, j) \in I \times J$ from $\mathbf{S}$. When $I = J$, we write $\mathbf{S}[I]$ as a shorthand for $\mathbf{S}[I, I]$. For any nonsingular matrix $\mathbf{S} \in \mathbb{R}^{n \times n}$ and index partition $I \cup \bar{I} = \{1, \ldots, n\}$, $I \cap \bar{I} = \emptyset$, the $I \times I$ matrix

$$\mathbf{S}/I = \mathbf{S}[I] - \mathbf{S}[I, \bar{I}] \cdot \mathbf{S}[\bar{I}]^{-1} \cdot \mathbf{S}[\bar{I}, I]$$

is called the *Schur complement* of $\mathbf{S}[\bar{I}]$, often denoted by $\mathbf{S}/\mathbf{S}[\bar{I}] = \mathbf{S}/I$. In particular, if $\mathbf{S} = \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{B}^T & \mathbf{D} \end{bmatrix}$ then the Schur complement of $\mathbf{A}$ is the matrix $\mathbf{S}/\mathbf{A} = \mathbf{D} - \mathbf{B}^T\mathbf{A}^{-1}\mathbf{B}$. For any index set I, a symmetric matrix $\mathbf{S}$ is positive definite if and only if both $\mathbf{S}[I]$ and its Schur's complement $\mathbf{S}/\mathbf{S}[I]$ are positive definite.

Let $\Sigma = \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{B}^T & \mathbf{D} \end{bmatrix} \succ 0$. We can factor Σ in terms of a principal submatrix, say $\mathbf{D}$, and its Schur complement, $\Sigma/\mathbf{D} = \mathbf{A} - \mathbf{B}\mathbf{D}^{-1}\mathbf{B}^T$, as follows:

$$\Sigma = \begin{bmatrix} \mathbf{I} & \mathbf{B}\mathbf{D}^{-1} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \Sigma/\mathbf{D} & \mathbf{0} \\ \mathbf{0} & \mathbf{D} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{D}^{-1}\mathbf{B}^T & \mathbf{I} \end{bmatrix}.$$

The next two theorems regarding the spectra of principal submatrices and Schur complements of positive definite matrices are used in Section 4. In both theorems, λ_i is the ith (in non-increasing order, with multiplicity) eigenvalue of a symmetric matrix.

Theorem 2.1 (Cauchy) *For any symmetric matrix $\mathbf{S} \in \mathbb{R}^{n \times n}$, $I \subseteq \{1, \ldots, n\}$ and $1 \leq i \leq |I|$*

$$\lambda_i(\mathbf{S}) \geq \lambda_i(\mathbf{S}[I]) \geq \lambda_{i+n-|I|}(\mathbf{S}).$$

Theorem 2.2 ([61, Corollary 2.3]) *For any positive definite $\Sigma \in \mathbb{R}^{n \times n}$, $I \subseteq \{1, \ldots, n\}$ and $1 \leq i \leq |I|$*

$$\lambda_i(\Sigma) \geq \lambda_i(\Sigma/I) \geq \lambda_{i+n-|I|}(\Sigma).$$

In other words, the eigenvalues of principal submatrices and Schur complements of a positive definite matrix are bounded from below and above by the smallest and largest eigenvalues of the original matrix, respectively.

2.2 Gaussians and Lattices

A *lattice* $\Lambda \subset \mathbb{R}^n$ is a discrete subgroup of $\mathbb{R}^n$. Specifically, a lattice of *rank* k is the integer span $\mathcal{L}(\mathbf{B}) = \{z_1\mathbf{b}_1 + \cdots + z_k\mathbf{b}_k \mid z_i \in \mathbb{Z}\}$ of a basis $\mathbf{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_k\} \subset \mathbb{R}^n$ ($k \leq n$). There are infinitely many bases for a given lattice since right multiplying a basis by a unimodular transformation gives another basis. The *dual lattice* of Λ, denoted by Λ^*, is the lattice $\{\mathbf{x} \in \text{span}(\Lambda) | \langle \mathbf{x}, \Lambda \rangle \subseteq \mathbb{Z}\}$. It is easy to see that $\mathbf{B}^{-T}$ is a basis for $\mathcal{L}(\mathbf{B})^*$ for a full rank lattice ($n = k$).

The n-dimensional *gaussian* function $\rho : \mathbb{R}^n \to (0, 1]$ is defined as $\rho(\mathbf{x}) := \exp(-\pi\|\mathbf{x}\|^2)$. Applying an invertible linear transformation $\mathbf{B}$ to the gaussian function yields

$$\rho_{\mathbf{B}}(\mathbf{x}) = \rho(\mathbf{B}^{-1}\mathbf{x}) = \exp(-\pi \cdot \mathbf{x}^T\Sigma^{-1}\mathbf{x})$$

with $\Sigma = \mathbf{B}\mathbf{B}^T \succ 0$. For any $\mathbf{c} \in \text{span}(\mathbf{B}) = \text{span}(\Sigma)$, we also define the shifted gaussian function (centered at $\mathbf{c}$) as $\rho_{\sqrt{\Sigma},\mathbf{c}}(\mathbf{x}) = \rho_{\sqrt{\Sigma}}(\mathbf{x} - \mathbf{c})$. Normalizing the function $\rho_{\mathbf{B},\mathbf{c}}(\mathbf{x})$ by the measure of $\rho_{\mathbf{B},\mathbf{c}}$ over the span of $\mathbf{B}$ gives the *continuous gaussian distribution* with covariance $\Sigma/(2\pi)$, denoted by $D_{\sqrt{\Sigma},\mathbf{c}}$. Let $S \subset \mathbb{R}^n$ be any discrete set in $\mathbb{R}^n$, then $\rho_{\sqrt{\Sigma}}(S) = \sum_{\mathbf{s} \in S} \rho_{\sqrt{\Sigma}}(\mathbf{s})$. The *discrete gaussian* distribution over a lattice Λ, denoted by $D_{\Lambda,\sqrt{\Sigma},\mathbf{c}}$, is defined by restricting the support of the distribution to Λ. Specifically, a sample $\mathbf{y} \leftarrow D_{\Lambda,\sqrt{\Sigma},\mathbf{c}}$ has probability mass function $\rho_{\sqrt{\Sigma},\mathbf{c}}(\mathbf{x})/\rho_{\sqrt{\Sigma},\mathbf{c}}(\Lambda)$ for all $\mathbf{x} \in \Lambda$. Discrete gaussians on lattice cosets $\Lambda + \mathbf{c}$, for $\mathbf{c} \in \text{span}(\Lambda)$, are defined similarly setting $\Pr\{\mathbf{y} \leftarrow D_{\Lambda+\mathbf{c},\sqrt{\Sigma},\mathbf{p}}\} = \rho_{\sqrt{\Sigma},\mathbf{p}}(\mathbf{y})/\rho_{\sqrt{\Sigma},\mathbf{p}}(\Lambda + \mathbf{c})$ for all $\mathbf{y} \in \Lambda + \mathbf{c}$. For brevity we let $D_{\Lambda+\mathbf{c},\sqrt{\Sigma},\mathbf{p}}(\mathbf{y}) := \Pr\{\mathbf{y} \leftarrow D_{\Lambda+\mathbf{c},\sqrt{\Sigma},\mathbf{p}}\}$.

For a lattice Λ and any (typically small) positive $\epsilon > 0$, the *smoothing parameter* $\eta_\epsilon(\Lambda)$ [52] is the smallest $s > 0$ such that $\rho(s \cdot \Lambda^*) \leq 1 + \epsilon$. A one-dimensional discrete gaussian with a tail-cut, t, is a discrete gaussian

$D_{\mathbb{Z},c,s}$ restricted to a support of $\mathbb{Z} \cap [c - t \cdot s, c + t \cdot s]$. We denote this truncated discrete gaussian as $D^t_{\mathbb{Z},c,s}$. In order to use the ML distance in Section 3, we will restrict all tail-cut discrete gaussians to a universal support of $\mathbb{Z} \cap [c - t \cdot s_{max}, c + t \cdot s_{max}]$ for some s_{max}.

Lemma 2.1 ([32, Lemma 4.2]) *For any $\epsilon > 0$, any $s \geq \eta_\epsilon(\mathbb{Z})$, and any $t > 0$,*

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

More generally, for any positive definite matrix Σ and lattice $\Lambda \subset \text{span}(\Sigma)$, we write $\sqrt{\Sigma} \geq \eta_\epsilon(\Lambda)$, or $\Sigma \succeq \eta^2_\epsilon(\Lambda)$, if $\rho(\sqrt{\Sigma}^T \cdot \Lambda^*) \leq 1 + \epsilon$. The reader is referred to [52, 32, 55] for additional information on the smoothing parameter.

Here we recall two bounds and a discrete gaussian convolution theorem to be used later.

Lemma 2.2 ([32, Lemma 3.1]) *Let $\Lambda \subset \mathbb{R}^n$ be a lattice with basis $\mathbf{B}$, and let $\epsilon > 0$. Then,*

$$\eta_\epsilon(\Lambda) \leq \|\widetilde{\mathbf{B}}\| \sqrt{\log(2n(1 + 1/\epsilon))/\pi}.$$

Lemma 2.3 ([55, Lemma 2.5]) *For any full rank n-dimensional lattice Λ, vector $\mathbf{c} \in \mathbb{R}^n$, real $\epsilon \in (0, 1)$, and positive definite $\Sigma \succeq \eta^2_\epsilon(\Lambda)$,*

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

Theorem 2.3 ([55, Theorem 3.1]) *For any vectors $\mathbf{c}_1, \mathbf{c}_2 \in \mathbb{R}^n$, lattices $\Lambda_1, \Lambda_2 \subset \mathbb{R}^n$, and positive definite matrices $\Sigma_1, \Sigma_2 \succ 0$, $\Sigma = \Sigma_1 + \Sigma_2 \succ 0$, $\Sigma_3^{-1} = \Sigma_1^{-1} + \Sigma_2^{-1} \succ 0$, if $\sqrt{\Sigma_1} \succeq \eta_\epsilon(\Lambda_1)$ and $\sqrt{\Sigma_3} \succeq \eta_\epsilon(\Lambda_2)$ for some $0 < \epsilon \leq 1/2$, then the distribution*

$$X = \{\mathbf{x} \mid \mathbf{p} \leftarrow D_{\Lambda_2 + \mathbf{c}_2, \sqrt{\Sigma_2}}, \mathbf{x} \leftarrow D_{\Lambda_1 + \mathbf{c}_1, \sqrt{\Sigma_1}, \mathbf{p}}\}$$

is within statistical distance $\Delta(X, Y) \leq 8\epsilon$ from the discrete gaussian $Y = D_{\Lambda_1 + \mathbf{c}_1, \sqrt{\Sigma}}$.

Below, we have the correctness theorem for the standard, randomized version of Babai's nearest plane algorithm. The term *statistically close* is the standard cryptographic notion of negligible statistical distance. Precisely, a function $f : \mathbb{N} \to \mathbb{R}_{\geq 0}$ is negligible if for every $c > 1$ there exists an N such that for all $n > N$, $f(n) < n^{-c}$. We emphasize that the algorithm reduces to sampling $D_{\mathbb{Z},s,c}$.

Theorem 2.4 ([32, Theorem 4.1]) *Given a full-rank lattice basis $\mathbf{B} \in \mathbb{R}^{n \times n}$, a parameter $s \geq \|\tilde{\mathbf{B}}\| \omega(\sqrt{\log n})$, and a center $\mathbf{c} \in \mathbb{R}^n$, there is an $O(n^2)$-time, with a $O(n^3)$-time preprocessing, probabilistic algorithm whose output is statistically close to $D_{\mathcal{L}(\mathbf{B}),s,\mathbf{c}}$.*

2.3 Cyclotomic Fields

Let n be a positive integer. The *n-th cyclotomic field over $\mathbb{Q}$* is the number field $\mathcal{K}_n = \mathbb{Q}[x]/(\Phi_n(x)) \cong \mathbb{Q}(\zeta)$ where ζ is an *n-th primitive root of unity* and $\Phi_n(x)$ is the minimal polynomial of ζ over $\mathbb{Q}$. The nth *cyclotomic ring* is $\mathcal{O}_n = \mathbb{Z}[x]/(\Phi_n(x))$. Let $\varphi(n)$ be Euler's totient function. $\mathcal{K}_n$ is a $\varphi(n)$-dimensional $\mathbb{Q}$-vector space, and we can view $\mathcal{K}_n$ as a subset of $\mathbb{C}$ by viewing ζ as a complex primitive n-th root of unity.

Multiplication by a fixed element f, $g \mapsto f \cdot g$, is a linear transformation on $\mathcal{K}_n$ as a $\mathbb{Q}$-vector space. We will often view field elements as $\varphi(n)$-dimensional rational vectors via the *coefficient embedding*. This is defined by $f(x) = \sum_{i=0}^{\varphi(n)-1} f_i x^i \mapsto (f_0, \cdots, f_{\varphi(n)-1})^T$ mapping a field element to its vector of coefficients under the *power basis* $\{1, x, \cdots, x^{\varphi(n)-1}\}$ (or equivalently $\{1, \zeta, \cdots, \zeta^{\varphi(n)-1}\}$). We can represent a field element as the matrix in $\mathbb{Q}^{\varphi(n) \times \varphi(n)}$ that represents the linear transformation by its multiplication in the coefficient embedding. This matrix is called a field element's coefficient *multiplication* matrix. When n is a power of two, an element's coefficient multiplication matrix is anti-cyclic.

$\textsc{SampleG}(q = b^k, s, u)$
  for $i = 0, \ldots, k-1$:
    $x_i \leftarrow \mathcal{D}_{b\mathbb{Z}+u,s}$
    $u := (u - x_i)/b \in \mathbb{Z}$.
  return $(x_0, \ldots, x_{k-1})$.

Figure 1: A sampling algorithm for G-lattices when the modulus q is a perfect power of the base b. The algorithm is implicitly parametrized by a base b and dimension k.

An *isomorphism* from the field F to the field K is a bijection $\theta : F \to K$ such that $\theta(fg) = \theta(f)\theta(g)$, and $\theta(f + g) = \theta(f) + \theta(g)$ for all $f, g \in F$. An *automorphism* is an isomorphism from a field to itself. For example, if we view the cyclotomic field $\mathcal{K}_n$ as a subset of the complex numbers, then the *conjugation* map $f(\zeta) \mapsto f(\zeta)^* = f(\zeta^*)$ is an automorphism and can be computed in linear time $O(n)$. In power-of-two cyclotomic fields, the conjugation of a field element corresponds to the matrix transpose of an element's anti-cyclic multiplication matrix.

Another embedding is the *canonical* embedding which maps an element $f \in \mathcal{K}_n$ to the vector of evaluations of f, as a polynomial, at each root of $\Phi_n(x)$. When n is a power of two, the linear transformation between the coefficient embedding and the canonical embedding is a scaled isometry.

Let n be a power of two, then the field $\mathcal{K}_{2n}$ is a two-dimensional $\mathcal{K}_n$-vector space as see by splitting a polynomial $f(x) \in \mathcal{K}_{2n}$ into $f(x) = f_0(x^2) + x \cdot f_1(x^2)$ for $f_i \in \mathcal{K}_n$. Now, we can view the linear transformation given by multiplication by some $f \in \mathcal{K}_{2n}$ as a linear transformation over $\mathcal{K}_n \oplus \mathcal{K}_n \cong \mathcal{K}_{2n}$. Let $\phi_{2n} : \mathcal{K}_{2n} \to \mathbb{Q}^{n \times n}$ be the injective ring homomorphism from the field to an element's anti-cyclic matrix. Then, we have the following relationship where $\mathbf{P}$ below is a simple re-indexing matrix known as a stride permutation (increasing evens followed by increasing odds in $\{0, 1, \ldots, n-1\}$),

$$\mathbf{P}\phi_n(f)\mathbf{P}^T = \begin{bmatrix} \phi_{n/2}(f_0) & \phi_{n/2}(x \cdot f_1) \\ \phi_{n/2}(f_1) & \phi_{n/2}(f_0) \end{bmatrix}.$$

3 Sampling G-lattices

For any positive integers $b \geq 2$, $k \geq 1$ and non-negative integer $u < b^k$, we write $[u]_b^k$ for the base-b expansion of u, i.e., the unique vector $(u_0, \ldots, u_{k-1})$ with entries $0 \leq u_i < b$ such that $u = \sum_i u_i b^i$. Typically, $b = 2$ and $[u]_2^k$ is just the k-digits binary representation of u, but larger values of b may be used to obtain interesting efficiency trade-offs. Throughout this section, we consider the values of b and k as fixed, and all definitions and algorithms are implicitly parametrized by them.

In this section we study the so-called G-lattice sampling problem, i.e., the problem of sampling the discrete Gaussian distribution on a lattice coset

$$\Lambda_u^\perp(\mathbf{g}^T) = \{\mathbf{z} \in \mathbb{Z}^k : \mathbf{g}^T\mathbf{z} = u \bmod q\}$$

where $q \leq b^k$, $u \in \mathbb{Z}_q$, $k = \lceil \log_b q \rceil$, and $\mathbf{g} = (1, b, \ldots, b^{k-1})$. G-lattice sampling is used in many lattice schemes employing a trapdoor. Both schemes with practical (polynomial) modulus, like IBE [18, 11, 4, 2], group signatures [44, 54, 45, 37], and others (double authentication preventing and predicate authentication preventing signuatures, constraint-hiding PRFs) [15, 22], and schemes with super-polynomial modulus [19, 20, 34, 17, 42, 36, 1] (ABE, obfuscation, watermarking, etc.).

A very efficient algorithm to solve this problem is given in [51] for the special case when $q = b^k$ is a power of the base b. The algorithm, shown in Figure 1, is very simple. This algorithm reduces the problem of sampling the k-dimensional lattice coset $\Lambda_u^\perp(\mathbf{g}^T)$ for $u \in \mathbb{Z}_q$ to the much simpler problem of sampling the *one-dimensional* lattice cosets $u + b\mathbb{Z}$ for $u \in \mathbb{Z}_b$. The simplicity of the algorithm is due to the fact that,

when $q = b^k$ is an exact power of b, the lattice $\Lambda^\perp(\mathbf{g}^T)$ has a very special basis

$$\mathbf{B}_{b^k} = \begin{bmatrix} b & & & & \\ -1 & b & & & \\ & -1 & \ddots & & \\ & & \ddots & b & \\ & & & -1 & b \end{bmatrix}$$

which is sparse, triangular, and with small integer entries. (In particular, its Gram-Schmidt orthogonalization $\widetilde{\mathbf{B}}_{b^k} = b\mathbf{I}$ is a scalar matrix.) As a result, the general lattice sampling algorithm of [43, 32] (which typically requires $O(k^3)$-time preprocessing, and $O(k^2)$ storage and online running time) can be specialized to the much simpler algorithm in Figure 1 that runs in linear time $O(k)$, with minimal memory requirements and no preprocessing at all.

We give a specialized algorithm to solve the same sampling problem when $q < b^k$ is an arbitrary modulus. This is needed in many cryptographic applications where the modulus q is typically a prime. As already observed in [51] the lattice $\Lambda^\perp(\mathbf{g}^T)$ still has a fairly simple and sparse basis matrix

$$\mathbf{B}_q = \begin{bmatrix} b & & & & q_0 \\ -1 & b & & & q_1 \\ & -1 & \ddots & & \vdots \\ & & \ddots & b & q_{k-2} \\ & & & -1 & q_{k-1} \end{bmatrix}$$

where $(q_0, \ldots, q_{k-1}) = [q]_b^k = \mathbf{q}$ is the base-b representation of the modulus q. This basis still has good geometric properties, as all vectors in its (left-to-right) Gram-Schmidt orthogonalization have length at most $O(b)$. So, it can be used with the algorithm of [43, 32] to generate good-quality gaussian samples on the lattice cosets with small standard deviation. However, since the basis is no longer triangular, its Gram-Schmidt orthogonalization is not sparse anymore, and the algorithm of [43, 32] can no longer be optimized to run in linear time as in Figure 1. In applications where $q = n^{O(1)}$ is polynomial in the security parameter n, the matrix dimension $k = O(\log n)$ is relatively small, and the general sampling algorithm (with $O(k^2)$ storage and running time) can still be used with an acceptable (albeit significant) performance degradation. However, for larger q this becomes prohibitive in practice. Moreover, even for small q, it would be nice to have an optimal sampling algorithm with $O(k)$ running time, linear in the matrix dimension, as for the exact power case. Here we give such an algorithm, based on the convolution methods of [55], but specialized with a number of concrete technical choices that result in a simple and very fast implementation, comparable to the specialized algorithm of [51] for the exact power case.

The reader may notice that the alternating columns of $\mathbf{B}_q$, $\mathbf{b}_1, \mathbf{b}_3, \ldots$ and $\mathbf{b}_2, \mathbf{b}_4, \ldots$, are pair-wise orthogonal. Let us call these sets $\mathbf{B}_1$ and $\mathbf{B}_2$, respectively. Then, another basis for $\Lambda^\perp(\mathbf{g}^T)$ is $(\mathbf{B}_1, \mathbf{B}_2, \mathbf{q})$ and this might suggest that the GSO of this basis is sparse. Unfortunately, this leads to a GSO of $(\mathbf{B}_1, \mathbf{B}_2^*, \mathbf{q}^*)$ where $\mathbf{B}_2^*$ is a dense, upper triangular block. Let $\mathbf{b}$ be the $i-th$ vector in $\mathbf{B}_2$. Then, there are $2 + i - 1$ non-orthogonal vectors to $\mathbf{b}$ preceding it in $\mathbf{B}_1$ and $\mathbf{B}_2^*$, filling in the upper portion of $\tilde{\mathbf{b}}$.

Regarding the MP12 sampler as a whole, another intuitive attempt would be to use Peikert's randomized rounder without perturbation [55] and simply sample $\Lambda^\perp(\mathbf{g}^T)$ with a skewed distribution (where we would simply change Σ_p in Section 4 to account for this). The goal is a linear algorithm since $\mathbf{B}_q$ is sparse. This intuition is correct since we can use back substitution twice from the decomposition of $\mathbf{B}_q$ into the product of two triangular matrices given in the next subsection. This decomposition of $\mathbf{B}_q$ is needed for this method to yield a linear time algorithm.

Overview The idea is the following. Instead of sampling $\Lambda_u^\perp(\mathbf{g}^T)$ directly, we express the lattice basis $\mathbf{B}_q = \mathbf{TD}$ as the image (under a linear transformation $\mathbf{T}$) of some other matrix $\mathbf{D}$ with very simple (sparse,

triangular) structure. Next, we sample the discrete gaussian distribution (say, with variance σ^2) on an appropriate coset of $\mathcal{L}(\mathbf{D})$. Finally, we map the result back to the original lattice applying the linear transformation $\mathbf{T}$ to it. Notice that, even if $\mathcal{L}(\mathbf{D})$ is sampled according to a spherical gaussian distribution, the resulting distribution is no longer spherical. Rather, it follows an ellipsoidal gaussian distribution with (scaled) covariance $\sigma^2\mathbf{T}\mathbf{T}^T$. This problem is solved using the convolution method of [55], i.e., initially adding a perturbation with complementary covariance $s^2\mathbf{I} - \sigma^2\mathbf{T}\mathbf{T}^T$ to the target, so that the final output has covariance $\sigma^2\mathbf{T}\mathbf{T}^T + (s^2\mathbf{I} - \sigma^2\mathbf{T}\mathbf{T}^T) = s^2\mathbf{I}$. In summary, at a very high level, the algorithm performs (at least implicitly) the following steps:

1. Compute the covariance matrix $\Sigma_1 = \mathbf{T}\mathbf{T}^T$ and an upper bound r on the spectral norm of $\mathbf{T}\mathbf{T}^T$
2. Compute the complementary covariance matrix $\Sigma_2 = r^2\mathbf{I} - \Sigma_1$
3. Sample $\mathbf{p} \leftarrow D_{\Lambda_1,\sigma\sqrt{\Sigma_2}}$, from some convenient lattice Λ_1 using the Cholesky decomposition of Σ_2
4. Compute the preimage $\mathbf{c} = \mathbf{T}^{-1}(\mathbf{u} - \mathbf{p})$
5. Sample $\mathbf{z} \leftarrow D_{\mathcal{L}(\mathbf{D}),-\mathbf{c},\sigma}$
6. Output $\mathbf{u} + \mathbf{T}\mathbf{z}$

The technical challenge is to find appropriate matrices $\mathbf{T}$ and $\mathbf{D}$ that lead to a very efficient implementation of all the steps. In particular, we would like $\mathbf{T}$ to be a very simple matrix (say, sparse, triangular, and with small integer entries) so that $\mathbf{T}$ has small spectral norm, and both linear transformations $\mathbf{T}$ and $\mathbf{T}^{-1}$ can be computed efficiently. The matrix $\mathbf{D}$ (which is uniquely determined by $\mathbf{B}$ and $\mathbf{T}$) should also be sparse and triangular, so that the discrete gaussian distribution on the cosets of $\mathcal{L}(\mathbf{D})$ can be efficiently sampled. Finally (and this is the trickiest part in obtaining an efficient instantiation) the complementary covariance matrix $\Sigma_2 = r^2\mathbf{I} - \Sigma_1$ should also have a simple Cholesky decoposition $\Sigma_2 = \mathbf{L}\mathbf{L}^T$ where $\mathbf{L}$ is triangular, sparse and with small entries, so that perturbations can be generated efficiently. Ideally, all matrices should also have a simple, regular structure, so that they do not need to be stored explicitly, and can be computed on the fly with minimal overhead.

In the next subsection we provide an instantiation that satisfies all of these properties. Next, in Subsection 3.2 we describe the specialized sampling algorithm resulting from the instantiation, and analyze its correctness and efficiency properties.

3.1 Instantiation

In this subsection, we describe a specific choice of linear transformations and matrix decompositions that satisfies all our desiderata, and results in a very efficient instantiation of the convolution sampling algorithm on G-lattices.

A tempting idea may be to map the lattice basis $\mathbf{B}_q$ to the basis $\mathbf{B}_{b^k}$, and then use the efficient sampling algorithm from Figure 1. However, this does not quite work because it results in a pretty bad transformation $\mathbf{T}$ which has both poor geometrical properties and a dense matrix representation. It turns out that a very good choice for a linear transformation $\mathbf{T}$ is given precisely by the matrix $\mathbf{T} = \mathbf{B}_{b^k}$ describing the basis when q is a power of b. We remark that $\mathbf{T}$ is used as a linear transformation, rather than a lattice basis. So, the fact that it equals $\mathbf{B}_{b^k}$ does not seem to carry any special geometric meaning, it just works! In particular, what we do here should not be confused with mapping $\mathbf{B}_q$ to $\mathbf{B}_{b^k}$. The resulting factorization is

$$\mathbf{B}_q = \begin{bmatrix} b & & & & q_0 \\ -1 & b & & & q_1 \\ & -1 & \ddots & & \vdots \\ & & \ddots & b & q_{k-2} \\ & & & -1 & q_{k-1} \end{bmatrix} = \begin{bmatrix} b & & & & \\ -1 & b & & & \\ & -1 & \ddots & & \\ & & \ddots & b & \\ & & & -1 & b \end{bmatrix} \begin{bmatrix} 1 & & & & d_0 \\ & 1 & & & d_1 \\ & & \ddots & & \vdots \\ & & & 1 & d_{k-2} \\ & & & & d_{k-1} \end{bmatrix} = \mathbf{B}_{b^k}\mathbf{D}$$

where the entries of the last column of $\mathbf{D}$ are defined by the recurrence $d_i = \frac{d_{i-1}+q_i}{b}$ with initial condition $d_{-1} = 0$. Notice that all the d_i are in the range $[0, 1)$, and $b^{i+1} \cdot d_i$ is always an integer. In some sense, sampling from $\mathcal{L}(\mathbf{D})$ is even easier than sampling from $\mathcal{L}(\mathbf{B}_{b^k})$ because the first $k-1$ columns of $\mathbf{D}$ are orthogonal and the corresponding coordinates can be sampled independently in parallel. (This should be contrasted with the sequential algorithm in Figure 1.)

We now look at the geometry and algorithmic complexity of generating perturbations. The covariance matrix of $\mathbf{T} = \mathbf{B}_{b^k}$ is given by

$$\Sigma_1 = \mathbf{B}_{b^k}\mathbf{B}_{b^k}^T = \begin{bmatrix} b^2 & -b & & & & \\ -b & (b^2+1) & -b & & & \\ & \ddots & \ddots & \ddots & & \\ & & -b & (b^2+1) & -b & \\ & & & -b & (b^2+1) \end{bmatrix}.$$

The next step is to find an upper bound r^2 on the spectral norm of Σ_2, and compute the Cholesky decomposition $\mathbf{L}\mathbf{L}^T$ of the complementary covariance matrix $\Sigma_2 = r^2\mathbf{I} - \Sigma_1$. By the Gershgorin circle theorem, all eigenvalues of Σ_1 are in the range $(b \pm 1)^2$. So, we may set $r = b + 1$. Numerical computations also suggest that this choice of r is optimal, in the sense that the spectral norm of Σ_1 approaches $b + 1$ as k tends to infinity. The Cholesky decomposition is customarily defined by taking $\mathbf{L}$ to be a *lower* triangular matrix. However, for sampling purposes, an upper triangular $\mathbf{L}$ works just as well. It turns out that using an upper triangular $\mathbf{L}$ in the decomposition process leads to a much simpler solution, where all (squared) entries have a simple, closed form expression, and can be easily computed on-line without requiring any preprocessing computation or storage. (By contrast, numerical computations suggest that the standard Cholesky decomposition with lower triangular $\mathbf{L}$ is far less regular, and even precomputing it requires exponentially higher precision arithmetic than our upper triangular solution.) So, we let $\mathbf{L}$ be an upper triangular matrix, and set $r = b + 1$.

For any r, the perturbation's covariance matrix $\Sigma_2 = r^2\mathbf{I} - \Sigma_1$ has Cholesky decomposition $\Sigma_2 = \mathbf{L} \cdot \mathbf{L}^T$ where $\mathbf{L}$ is the sparse upper triangular matrix defined by the following equations:

$$\mathbf{L} = \begin{bmatrix} l_0 & h_1 & & & \\ & l_1 & h_2 & & \\ & & \ddots & \ddots & \\ & & & & h_{k-1} \\ & & & & l_{k-1} \end{bmatrix} \quad \text{where} \quad \begin{array}{rcll} l_0^2 + h_1^2 & = & r^2 - b^2 & \\ l_i^2 + h_{i+1}^2 & = & r^2 - (b^2+1) & (i = 1, \ldots, k-2) \\ l_{k-1}^2 & = & r^2 - (b^2+1) & \\ l_i h_i & = & b & (i = 1, \ldots, k-1) \end{array}$$

It can be easily verified that these equations have the following simple closed form solution:

$$r = b + 1, \quad l_0^2 = b\left(1 + \frac{1}{k}\right) + 1, \quad l_i^2 = b\left(1 + \frac{1}{k-i}\right), \quad h_{i+1}^2 = b\left(1 - \frac{1}{k-i}\right) \tag{1}$$

We observe that also the inverse transformation $\mathbf{B}_{b^k}^{-1}$ has a simple, closed-form solution: the ith column of $\mathbf{B}_{b^k}^{-1}$ equals $(0, \cdots, 0, \frac{1}{b}, \ldots, (\frac{1}{b})^{k-i})$. Notice that this matrix is not sparse, as it has $O(k^2)$ nonzero entries. However, there is no need to store it and the associated transformation can still be computed in linear time by solving the sparse triangular system $\mathbf{T}\mathbf{x} = \mathbf{b}$ by back-substitution.

3.2 The Algorithm

The sampling algorithm, SampleG, is shown in Figure 2. It takes as input a modulus q, an integer variance s, a coset u of $\Lambda^{\perp}(\mathbf{g}^T)$, and outputs a sample statistically close to $D_{\Lambda_u^{\perp}(\mathbf{g}^T),s}$. SampleG relies on subroutines Perturb and SampleD where Perturb(σ) returns a perturbation, $\mathbf{p}$, statistically close to $D_{\mathcal{L}(\Sigma_2),\sigma\cdot\sqrt{\Sigma_2}}$, and SampleD($\sigma, \mathbf{c}$) returns a sample $\mathbf{z}$ such that $\mathbf{D}\mathbf{z}$ is statistically close to $D_{\mathcal{L}(\mathbf{D}),-\mathbf{c},\sigma}$.

SampleG($s, \mathbf{u} = [u]_b^k, \mathbf{q} = [q]_b^k$)
  $\sigma := s/(b+1)$
  $\mathbf{p} \leftarrow$ Perturb(σ)
  for $i = 0, \ldots, k-1$:
    $c_i := (c_{i-1} + u_i - p_i)/b$
  $\mathbf{z} \leftarrow$ SampleD($\sigma, \mathbf{c}$)
  for $i = 0, \ldots, k-2$:
    $t_i := b \cdot z_i - z_{i-1} + q_i \cdot z_{k-1} + u_i$
  $t_{k-1} := q_{k-1} \cdot z_{k-1} - z_{k-2} + u_{k-1}$
  return t

Perturb(σ)
  $\beta := 0$
  for $i = 0, \ldots, k-1$:
    $c_i := \beta/l_i$, and $\sigma_i := \sigma/l_i$
    $z_i \leftarrow \lfloor c_i \rfloor +$ SampleZ$_t(\sigma_i, \lfloor c_i \rceil_{[0,1)}, s)$
    $\beta := -z_i h_i$
  $p_0 := (2b+1)z_0 + bz_1$
  for $i := 1, \ldots, k-1$:
    $p_i := b(z_{i-1} + 2z_i + z_{i+1})$
  return p

SampleD($\sigma, \mathbf{c}$)
  $z_{k-1} \leftarrow \lfloor -c_{k-1}/d_{k-1} \rfloor$
  $z_{k-1} \leftarrow z_{k-1} +$ SampleZ$_t(\sigma/d_{k-1}, \lfloor -c_{k-1}/d_{k-1} \rceil_{[0,1)}, s)$
  $\mathbf{c} := \mathbf{c} - z_{k-1}\mathbf{d}$
  for $i \in \{0, \ldots, k-2\}$:
    $z_i \leftarrow \lfloor -c_i \rfloor +$ SampleZ$_t(\sigma, \lfloor -c_i \rceil_{[0,1)}, s)$
  return z

Figure 2: Sampling algorithm for G-lattices for any modulus $q < b^k$. The algorithms take b and k as implicit parameters, and SampleG outputs a sample with distribution statistically close to $D_{\Lambda_u^\perp(\mathbf{g}^T), s}$. Any scalar with an index out of range is 0, i.e. $c_{-1} = z_{-1} = z_k = 0$. SampleZ$_t(\sigma, c, \sigma_{max})$ is any algorithm that samples from a discrete gaussian over $\mathbb{Z}$ exactly or approximately with centers in $[0, 1)$ and a fixed truncated support $\mathbb{Z} \cap [c - t \cdot \sigma_{max}, c + t \cdot \sigma_{max}]$ (t is the tail-cut parameter). We denote $x - \lfloor x \rfloor$ as $\lfloor x \rceil_{[0,1)}$.

Both Perturb and SampleD are instantiations of the randomized nearest plane algorithm [43, 32]. Consequently, both algorithms rely on a subroutine SampleZ$_t(\sigma, c, \sigma_{max})$ which returns a sample statistically close to one-dimensional discrete gaussian with it a tail-cut t, $D^t_{\mathbb{Z},\sigma,c}$ over the *fixed* support of $\mathbb{Z} \cap [c - t \cdot \sigma_{max}, c + t \cdot \sigma_{max}]$. We fix the support of all one dimensional discrete gaussians for compatibility with ML distance. In addition, we only feed SampleZ centers $c \in [0, 1)$ since we can always shift by an integer.

Time Complexity Assuming constant time sampling for SampleZ and scalar arithmetic, SampleG runs in time $O(k)$. Now let us consider all operations: there are $10k$ integer additions/subtractions, $3k$ integer multiplications, $3k + 2$ floating point divisions, $2k$ floating point multiplications, and $2k$ floating point additions. Let us assume we use floating point numbers with length t mantissa and N-bit integers. Then, SampleG is time $k(3t^2 + 2t \log t + 2t + 10N + 3N \log N) + 2t^2$. Adding in the time to sample a $\mathbb{Z}$ ($t_{\mathbb{Z}}$), we have $k(3t^2 + 2t \log t + 10N + 2N \log N + 2t_{\mathbb{Z}}) + 2t^2$ compared to $k^2(t \log t + 2N \log N) + k(2t^2 + t + t_{\mathbb{Z}})$ for the generic randomized nearest plane. The analysis below shows we can use double precision floating point numbers for most applications.

Storage The scalars c_i in SampleG, representing $\mathbf{c} = \mathbf{B}_{b^k}^{-1}(\mathbf{u} - \mathbf{p})$, and d_i in SampleD, representing the last column of $\mathbf{D}$, are rational numbers of the form x/b^i for a small integer x and $i \in [k]$. The numbers l_i, h_i are positive numbers of magnitude less than $\sqrt{2b+1}$.

The algorithms store floating point numbers c_i, d_i, h_i, and l_i for a total storage of $4k$ floating point numbers, but can be adapted to constant time storage since they are determined by simple recurrence relations (c_i, d_i) or simple formulas (h_i, l_i).

Statistical Analysis and Floating Point Precision We now perform a statistical analysis on SampleG with a perfect one-dimensional sampler (and no tail-bound), then with an imperfect sampler in terms of ML

distance. This allows us to measure loss in concrete security. We direct the reader to [53, Section 3] for more details on the ML distance and a complete concrete security analysis.

The statistical distance bound of $\Theta(k\hat{\epsilon})$ results in about a loss of $\log\log q$ bits in security if $\epsilon = 2^{-\kappa}$ for a security parameter κ by [53, Lemma 3.1]. (The multiplicative factor of k comes from the randomized nearest plane algorithm's analysis: see [32, Theorem 4.1].)

Corollary 3.1 *Fix $0 < \epsilon \leq 1/2$ and let $s \geq (b+1) \cdot \eta_\epsilon(\mathcal{L}(\mathbf{D}))$ and $\sqrt{\Sigma}_3 \geq \eta_\epsilon(\mathcal{L}(\Sigma_2))$ where $\Sigma_3^{-1} = \frac{(b+1)^2}{s^2}[\Sigma_1^{-1} + [(b+1)^2\mathbf{I} - \Sigma_1]^{-1}]$. Then,* SampleG *returns a perturbation within a statistical distance $\Theta(k\hat{\epsilon})$ from $D_{\Lambda_u^\perp(\mathbf{g}^T),s}$ for any $q < b^k$ when* Perturb *and* SampleD *use a perfect one-dimensional sampler,* SampleZ*. In addition, the Rényi divergence of order infinity of $D_{\Lambda_u^\perp(\mathbf{g}^T),s}$ from* SampleG *with a perfect one-dimensional sampler is less than or equal to $1 + \Theta(k\hat{\epsilon})$.*

Note, the lower bound on how small s can be depends quadratically on the base b. The condition on Σ_3 above translates to a bound of about $s \geq 4b^2 C_{\epsilon,k}$, where $C_{\epsilon,k} = \sqrt{\log(2k(1+1/\epsilon))/\pi}$. This can be seen by using the Gershgorin circle theorem on the involved covariance matrices yielding $s \geq b \cdot \sqrt{\frac{b+1}{b-1}} \cdot \eta_\epsilon(\mathcal{L}(\Sigma_2))$. The lattice $\mathcal{L}(\Sigma_2)$ will have vectors no shorter than length $2b$. If one were inclined to use a large base b, then we suggest Perturb(σ) be changed to the continuous version, sample $\mathbf{p} \leftarrow D_\sigma^k$ and return $\mathbf{L} \cdot \mathbf{p} \sim D_{\sigma \cdot \Sigma_2}$. This is efficient since $\mathbf{L}$ is sparse and s can be as small as $(b+1) \cdot C_{\epsilon,k}$ since we no longer require the condition on Σ_3.

Next, we turn to the ML distance for a tighter analysis on the bits of security lost in using SampleG with an imperfect one-dimensional sampler. Since the centers, c, and variances, s, given to SampleZ are computed from two or three floating point computations, we assume both $\bar{c}$ and $\bar{s}$ are within a relative error of 2^{-m} of c and s, respectively.

Proposition 3.1 *Fix an $\epsilon > 0$ and let $s \geq (b+1) \cdot \eta_\epsilon(\mathbb{Z})$. For any one-dimensional sampler* SampleZ$_t(\bar{\sigma}, \bar{c}, s)$ *that takes as inputs approximated centers $\bar{c} \in [0,1)$ and variances $\bar{\sigma} \in [s/(b+1), s \cdot b/(b+1)]$ represented as floating point numbers with mantissa length m, $\Delta_{\text{ML}}($SampleG$^{D^t_{\mathbb{Z},\sigma,c}},$ SampleG$^{\text{SampleZ}_t(\bar{\sigma},\bar{c})}) \leq$ $2k[O(t^2 2^{-m}) + \max_{\bar{\sigma},\bar{c}} \Delta_{\text{ML}}($SampleZ$_t(\bar{\sigma}, \bar{c}, s), D^t_{\mathbb{Z},\bar{\sigma},\bar{c}})]$.*

Assuming a cryptosystem using a perfect sampler for $D_{\Lambda_u^\perp(g^T),s}$ has κ bits of security, we can combine the results of Corollary 3.1, Proposition 3.1, and [53, Lemma 3.3] to conclude that swapping $D_{\Lambda_u^\perp(g^T),s}$ with SampleG yields $\kappa - 2\log t - 1.5\log\log q - 4$ bits of security when $m = \kappa/2$, $\Delta_{\text{ML}}($SampleZ$_t(\bar{s}, \bar{c}), D^t_{\mathbb{Z},\bar{s},\bar{c}})$ $< 2^{-\kappa/2}$, and $\epsilon = 2^{-\kappa}$.

3.3 Implementation and Comparison

In this subsection, we compare simple implementations of both SampleG and the generic randomized nearest plane algorithm [32, Section 4] used in the G-lattice setting. The implementations were carried out in C++ with double precision floating point numbers for non-integers on an Intel i7-2600 3.4 GHz CPU. Clock cycles were measured with the "time.h" library and the results are charted in Figure 3.3.

The one-dimensional sampler, SampleZ, was an instantiation of a discrete version of Karney's sampler [41], which is a modified rejection sampler. The moduli q were chosen from the common parameters subsection of [40, Section 4.2], in addition to an arbitrary 60-bit modulus. Most practical schemes require no more than a 30-bit modulus [9] for lattice dimension (n) up to 1024. More advanced schemes however, like ABE-encryption [14, 19], predicate encryption [35], and obfuscation [20, 27], require a super-polynomial modulus often 90 or more bits (assuming the circuits in the ABE and predicate schemes are of log-depth).

For the generic, randomized nearest plane sampler, we pre-computed and stored the Gram-Schmidt orthogonalization of the basis $\mathbf{B}_q$ and we only counted the clock cycles to run the algorithm thereafter. We had two versions of SampleG: the first was the algorithm as-is, and the second would store pre-computed perturbations from Perturb(σ), one for each G-lattice sample. This version of SampleG with pre-computation saved about a factor of two in clock cycles.

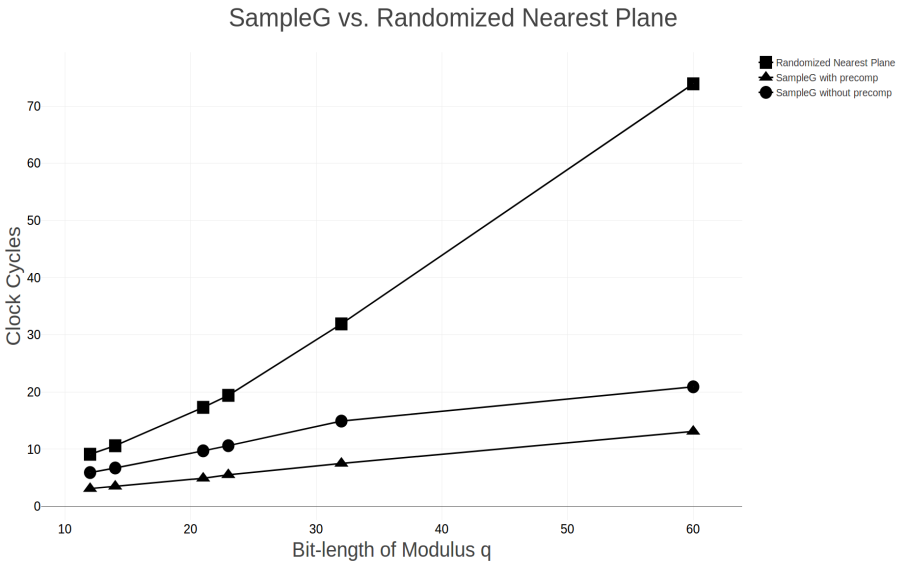

Figure 3: Measured clock cycles with $q = \{4093, 12289, 1676083, 8383498, 4295967357, \approx 9 \cdot 10^{18}\}$ and $s = 100$ averaged over 100,000 runs. The clock cycles for the last three moduli are $\{19.4, 31.9, 73.9\}$ for GPV and $\{5.5, 7.5, 13.1\}$ for SampleG with pre-computation.

4 Perturbation Sampling in Cyclotomic Rings

The lattice preimage sampling algorithm of [51] requires the generation of $n(2 + \log q)$-dimensional gaussian perturbation vectors $\mathbf{p}$ with covariance $\Sigma_p = s^2 \cdot \mathbf{I} - \alpha^2 \mathbf{T} \cdot \mathbf{T}^T$ where $\mathbf{T} \in \mathbb{Z}^{(2+\log q)n \times n \log q}$ is a matrix with small entries serving as a lattice trapdoor, α is a small constant factor and s is an upper bound on the spectral norm of $\alpha\mathbf{T}$. In [51] this is accomplished using the Cholesky factorization of Σ_p, which takes $O(n \log q)^3$ precomputation and $O(n \log q)^2$ storage and running time.

The trapdoor matrix $\mathbf{T}$ of [51] has some additional structure: $\mathbf{T}^T = [\bar{\mathbf{T}}^T, \mathbf{I}]$ for some $\bar{\mathbf{T}} \in \mathbb{Z}^{2n \times n \log q}$. Moreover, when working with algebraic lattices, $\bar{\mathbf{T}} = \phi_n(\tilde{\mathbf{T}})$ is the image (under a ring embedding $\phi_n \colon R_n \to \mathbb{Z}^{n \times n}$) of some matrix $\tilde{\mathbf{T}} \in R_n^{2 \times \log q}$ with entries in a ring R_n of rank n. (Most commonly, $R_n = \mathcal{O}_{2n} = \mathbb{Z}[x]/(x^n + 1)$ is the ring of integers of the $(2n)$th cyclotomic field $\mathcal{K}_{2n}$ for $n = 2^k$ a power of two.) In [9] it is observed that, using the sparsity of Σ_p, the preprocessing storage and on-line computation cost of noise perturbation reduce to $O(n^2 \log q)$.[4] This is a factor $\log q$ improvement over a generic implementation, but it is still quadratic in the main security parameter n. This can be a significant improvement in practice, but the overall cost of the algorithm remains substantial. When using generic trapdoors $\bar{\mathbf{T}} \in \mathbb{Z}^{2n \times n \log q}$, there is little hope to improve the running time below $O(n^2 \log q)$, because just reading the matrix $\bar{\mathbf{T}}$ takes this much time. However, when using algebraic lattices, the trapdoor $\bar{\mathbf{T}} = \phi_n(\tilde{\mathbf{T}})$ admits a compact representation $\tilde{\mathbf{T}}$ consisting of only $2n \log q$ integers, so one may hope to reduce the running time to linear or quasi-linear in n.

In this section we give an alternative algorithm to generate integer perturbation vectors $\mathbf{p}$ with covariance Σ_p when $\bar{\mathbf{T}} = \phi_n(\tilde{\mathbf{T}})$. Our algorithm takes full advantage of the ring structure of R_n, compactly representing Σ_p and all other matrices generated during the execution of the algorithm as the image of matrices with entries in the ring R_n. In particular, similarly to [28, 29], our algorithm has time and space complexity quasi-linear in n, but does not require any preprocessing/storage. The algorithm can be expressed in a modular way as the combination of three steps:

1. First, the problem of sampling a $O(n \log q)$-dimensional integer vectors $\mathbf{p}$ with covariance Σ_p is reduced

[4] Sparsity also reduces the preprocessing running time to $O(\log q \cdot n^2 + n^3) = O(n^3)$, but still cubic in n.

SamplePz$(n, q, s, \alpha, \tilde{\mathbf{T}}, \Sigma_2, z)$
  for $i = 0, \ldots, (n \log q - 1)$:
    $q_i \leftarrow$ SampleZ$(s^2 - \alpha^2)$
  $(c_0, c_1) := -\frac{\alpha^2}{s^2 - \alpha^2} \tilde{\mathbf{T}}\mathbf{q}$
  $c'(x) := c_0(x^2) + x \cdot c_1(x^2))$
  $\mathbf{p} \leftarrow$ Sample2z(a, b, d, c')
  return $(\mathbf{p}, \mathbf{q})$

Sample2z(a, b, d, c)
  let $c(x) = c_0(x^2) + x \cdot c_1(x^2)$
  $q_1 \leftarrow$ SampleFz(d, c_1)
  $c_0 := c_0 + bd^{-1}(q_1 - c_1)$
  $q_0 \leftarrow$ SampleFz$(a - bd^{-1}b^*, c_0)$
  return (q_0, q_1)

SampleFz(f, c)
  if $\dim(f) = 1$ **return** SampleZ(f, c)
  else let $f(x) = f_0(x^2) + x \cdot f_1(x^2)$
    $(q_0, q_1) \leftarrow$ Sample2z(f_0, f_1, f_0, c)
    let $q(x) = q_0(x^2) + x \cdot q_1(x^2)$
    return q

Figure 4: Sampling algorithm SamplePz for integer perturbations where $\mathbf{T} = \phi_n(\tilde{\mathbf{T}})$ is a compact trapdoor over a power of two cyclotomic ring. Note, $\tilde{\mathbf{T}}_i$ is a *row* vector over R_n for each $i \in \{0, 1\}$. The algorithm uses a subroutine SampleZ(σ^2, t) which samples a discrete gaussian over $\mathbb{Z}$ with variance σ^2 centered at t. The scalar $z = (\alpha^{-2} - s^{-2})^{-1}$.

to the problem of sampling a $2n$-dimensional integer vector with covariance expressed by a 2×2 matrix over R_n.

2. Next, the problem of sampling with covariance in $R_n^{2\times 2}$ is reduced to sampling two n-dimensional vectors, each with a covariance expressed by a single ring element in R_n.

3. Finally, if $n > 1$, the sampling problem with covariance in R_n is reduced to sampling an n-dimensional perturbation with covariance expressed by a 2×2 matrix over the smaller ring $R_{n/2}$.

Iterating the last two steps $\log n$ times reduces the original problem to sampling in $R_1 = \mathbb{Z}$. Details about each step are given in the next subsections. We remark that the algorithm is described as a recursive procedure only for simplicity of presentation and analysis, and it can be implemented just as easily using a simple nested loop, similarly to many FFT-like algorithms.

The discrete version of the algorithm ,which directly generates perturbations from discrete gaussians over $\mathbb{Z}^{n(2+\log q)}$, is in the following subsection.

4.1 Discrete Perturbation Algorithm for Power of Two Cyclotomics

In this subsection we present the perturbation algorithm algorithm which produces $n(2 + \log q)$-dimensional perturbations from a discrete gaussian on $\mathbb{Z}^{n(2+\log q)}$ in time $\tilde{O}(n \log q)$.

The entry point of the algorithm is the SamplePz procedure, which takes as input two integer parameters n, q, matrices $\tilde{\mathbf{T}} \in R_n^{2\times \log q}$, $\Sigma_2 \in R_n^{2\times 2}$, and three positive real numbers s^2, α^2, $z = (\alpha^{-2} - s^{-2})^{-1}$, and is expected to produce an $n(2 + \log q)$-dimensional vector $\mathbf{p}$ with (non-spherical) discrete gaussian distribution $D_{\mathbb{Z}^{n(2+\log q)}, \sqrt{\Sigma_p}}$ of covariance

$$\begin{aligned}\Sigma_p &= s^2 \cdot \mathbf{I} - \alpha^2 \begin{bmatrix} \phi_n(\tilde{\mathbf{T}}) \\ \mathbf{I} \end{bmatrix} \cdot \begin{bmatrix} \phi_n(\tilde{\mathbf{T}})^T & \mathbf{I} \end{bmatrix} \\ &= \begin{bmatrix} \Sigma_2 & -\alpha^2 \phi_n(\tilde{\mathbf{T}}) \\ -\alpha^2 \phi_n(\tilde{\mathbf{T}})^T & (s^2 - \alpha^2)\mathbf{I} \end{bmatrix}.\end{aligned}$$

The algorithm calls two subroutines:

- SampleZ($s^2 - \alpha^2$) which samples a one-dimensional discrete gaussian variable of variance $s^2 - \alpha^2$ centered at 0, and can be implemented using any standard technique, and
- Sample2z(a, b, d), which, on input three ring elements a, b, d compactly describing a positive definite matrix

$$\Sigma_2 = \begin{bmatrix} \phi_n(a) & \phi_n(b) \\ \phi_n(b)^T & \phi_n(d) \end{bmatrix},$$

is expected to sample a $(2n)$-dimensional vector $p \leftarrow D_{\mathbb{Z}^{2n}, \sqrt{\Sigma_2}}$.

In turn, Sample2z (also described in Figure 4) makes use of a procedure SampleFz(f) which on input a ring element f with positive definite $\phi_n(f)$, returns a sample $p \leftarrow D_{\mathbb{Z}^n, \sqrt{\phi_n(f)}}$.

Efficiency Multiplications are done in the field $\mathcal{K}_i$, for an element's dimension $i \in \{1, 2, \ldots, 2n\}$, in time $\Theta(i \log i)$ by using the Chinese remainder transform (CRT) [49].

By treating scalar arithmetic as constant time, SamplePz has a time complexity of $\Theta(n \log n \log q)$ because the transformation by $\tilde{\mathbf{T}}$ is $\Theta(n \log n \log q)$ and SampleFz has complexity $\Theta(n \log^2 n)$ (represented by the recurrence $R(n) = 2R(n/2) + 2 \log n/2 + 4.5n$). The algorithm requires $2n \log q$ scalar storage for the trapdoor $\tilde{\mathbf{T}}$.

Note, SampleFz is even more efficient, $\Theta(n \log n)$, if one were to store the polynomials in $\mathcal{K}_i$ in the canonical embedding (Fourier domain). One would change SamplePz to give Sample2z the Fourier/canonical representations of a, b, d, c_0, c_1 and perform an inverse CRT/FFT on $\mathbf{p} = (\mathbf{p}_0, \mathbf{p}_1)$. This allows us to use the FFT's butterfly transformation to convert to the Fourier representation of $f(x) = f_0(x^2) + x f_1(x^2) \in \mathcal{K}_{2n}$ to the Fourier representation of $f_0, f_1 \in \mathcal{K}_n$ and multiplication/inversion is now linear time (we would only invert the non-zero entries in the Fourier domain since this corresponds to pulling back to the field, inverting, then pushing forward to the cyclic ring via the embedding given by the Chinese remainder theorem) [29, Lemma 1]. (Moving from the canonical embedding to the FFT domain is linear time since we place zeros for the non-primitive roots of unity [29, Section A.2].) This, however, does not change the asymptotic time complexity of SamplePz since generating $\mathbf{q}$ in the canonical embedding is now $\Theta(n \log n \log q)$.

Correctness One would use Peikert's convolution theorem, Theorem 2.3, in an initial attempt to prove the correctness of the algorithms in Figure 4. However, this would only ensure the correctness of the marginal distributions of $\mathbf{p}$ in SamplePz and q_0 in Sample2z and not their respective joint distributions, $(\mathbf{p}, \mathbf{q})$ and (q_0, q_1). Even if it were enough, tracking the Σ_3 condition in Theorem 2.3 through the recursive calls of the algorithms above is tedious. Instead, we derive a convolution lemma without a Σ_3 condition for the joint distribution of our discrete gaussian convolutions on the simple lattice $\mathbb{Z}^n$.

First, we show the gaussian function $\rho_{\sqrt{\Sigma}}(\cdot)$ factors in a useful manner with respect to a Schur complement decomposition.

Lemma 4.1 *Let* $\Sigma = \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{B}^T & \mathbf{D} \end{bmatrix} \succ \mathbf{0}$ *be a positive definite with* $\mathbf{A} \in \mathbb{R}^{n \times n}$ *and* $\mathbf{D} \in \mathbb{R}^{m \times m}$ *and* $\Sigma/\mathbf{D} = \mathbf{A} - \mathbf{B}\mathbf{D}^{-1}\mathbf{B}^T$ *is* $\mathbf{D}$*'s Schur complement, and let* $\mathbf{x}_1 \in \mathbb{R}^n$ *and* $\mathbf{x}_2 \in \mathbb{R}^m$ *be arbitrary. Then, the gaussian function* $\rho_{\sqrt{\Sigma}}(\mathbf{x})$ *factors as* $\rho_{\sqrt{\Sigma/\mathbf{D}}}(\mathbf{x}_1 - \mathbf{B}\mathbf{D}^{-1}\mathbf{x}_2) \cdot \rho_{\sqrt{\mathbf{D}}}(\mathbf{x}_2) = \rho_{\sqrt{\Sigma}}(\mathbf{x})$ *where* $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2) \in \mathbb{R}^{n+m}$.

Proof:(Sketch) This is seen through defining the inverse of Σ in terms of $\Sigma/\mathbf{D}$ and writing out $\rho_{\sqrt{\Sigma}}(\mathbf{x})$ in terms of $\Sigma/\mathbf{D}$. The matrix factorization

$$\Sigma = \begin{bmatrix} \mathbf{I} & \mathbf{B}\mathbf{D}^{-1} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \Sigma/\mathbf{D} & \mathbf{0} \\ \mathbf{0} & \mathbf{D} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{D}^{-1}\mathbf{B}^T & \mathbf{I} \end{bmatrix}$$

yields the formula for Σ^{-1} needed to show the result. □

A consequence of the above lemma is that the gaussian sum $\rho_{\sqrt{\Sigma}}(\mathbb{Z}^{n+m})$ expands in terms of the gaussian functions $\rho_{\sqrt{\mathbf{D}}}(\cdot)$ and $\rho_{\sqrt{\Sigma/\mathbf{D}}}(\cdot)$,

$$\rho_{\sqrt{\Sigma}}(\mathbb{Z}^{n+m}) = \sum_{\mathbf{y}_2 \in \mathbb{Z}^m} \rho_{\sqrt{\mathbf{D}}}(\mathbf{y}_2) \cdot \rho_{\sqrt{\Sigma/\mathbf{D}}}(\mathbb{Z}^n - \mathbf{B}\mathbf{D}^{-1}\mathbf{y}_2).$$

We will use the following lemma for the correctness proof. It states that if a discrete gaussian on the integer lattice is wide enough in its slimmest direction, then the lower dimensional discrete gaussians with covariance shaped with principal submatrices of the original are wide enough on their respective $\mathbb{Z}^{n'}$s.

Lemma 4.2 *Let $\epsilon > 0$, $\Sigma \succ 0$ be a positive definite matrix in $\mathbb{R}^{n\times n}$, and let $I_0 \subset [n]$ be an arbitrary, non-empty subset. If $\Sigma \succeq \eta_\epsilon^2(\mathbb{Z}^n)$, then $\Sigma[I_0] \succeq \eta_\epsilon^2(\mathbb{Z}^{|I_0|})$ and $\Sigma/\bar{I}_0 \succeq \eta_\epsilon^2(\mathbb{Z}^{n-|I_0|})$ for any principal submatrix - Schur complement pair, $(\Sigma[I_0], \Sigma/\bar{I}_0)$, of Σ.*

Proof: Note, a consequence of $\Sigma \succeq \eta_\epsilon^2(\mathbb{Z}^n)$ is that Σ's minimum eigenvalue, $\lambda_{min}(\Sigma)$, is greater than $\eta_\epsilon^2(\mathbb{Z}^n)$. Let $\mathbf{M} := \Sigma[I_0] \in \mathbb{R}^{n_0\times n_0}$ for $n_0 = |I_0|$. $\mathbf{M}$ is diagonalizable so let $\mathbf{M} = \mathbf{Q}^T\mathbf{\Lambda}\mathbf{Q}$ be its diagonalization. Notice, we have the following inequality from the interlacing theorems which imply $\lambda_{min}(\mathbf{M}) \geq \lambda_{min}(\Sigma)$,

$$\mathbf{x}^T\mathbf{M}\mathbf{x} = \mathbf{x}^T\mathbf{Q}^T\mathbf{\Lambda}\mathbf{Q}\mathbf{x} = \mathbf{y}^T\mathbf{\Lambda}\mathbf{y} = \sum_{i\in[n_0]} \lambda_i y_i^2 \geq \lambda_{min}(\Sigma)\|\mathbf{y}\|^2 = \lambda_{min}(\Sigma)\|\mathbf{x}\|^2.$$

Next, we can bound the quantity $\rho_{\sqrt{\mathbf{M}^{-1}}}((\mathbb{Z}^{n_0})^*) = \rho_{\sqrt{\mathbf{M}^{-1}}}(\mathbb{Z}^{n_0})$ by $1+\epsilon$:

$$\rho_{\sqrt{\mathbf{M}^{-1}}}(\mathbb{Z}^{n_0}) = \sum_{\mathbf{x}\in\mathbb{Z}^{n_0}} e^{-\pi\mathbf{x}^T\mathbf{M}\mathbf{x}} \leq \sum_{\mathbf{x}\in\mathbb{Z}^{n_0}} e^{-\pi\lambda_{min}(\Sigma)\|\mathbf{x}\|^2}$$

$$\leq \sum_{\mathbf{x}\in\mathbb{Z}^{n}} e^{-\pi\lambda_{min}(\Sigma)\|\mathbf{x}\|^2} \leq 1+\epsilon.$$

The jump from $\mathbb{Z}^{n_0}$ to $\mathbb{Z}^n$ comes from the relation $\mathbb{Z}^{n_0} \subset \mathbb{Z}^n$. The proof for the Schur complement is identical. □

Next, we state and prove our main convolution lemma.

Lemma 4.3 *For any real $0 < \epsilon \leq 1/2$, positive integers n, m, vector $\mathbf{c} = (\mathbf{c}_1, \mathbf{c}_2) \in \mathbb{R}^{n+m}$, and positive definite matrix $\Sigma = \begin{bmatrix}\mathbf{A} & \mathbf{B}\\ \mathbf{B}^T & \mathbf{D}\end{bmatrix} \succeq \eta_\epsilon^2(\mathbb{Z}^{n+m})$, $\mathbf{A} \in \mathbb{Z}^{n\times n}$, $\mathbf{B} \in \mathbb{Z}^{n\times m}$, and $\mathbf{D} \in \mathbb{Z}^{m\times m}$ (where $\Sigma/\mathbf{D} = \mathbf{A} - \mathbf{B}\mathbf{D}^{-1}\mathbf{B}^T$ is the Schur complement of $\mathbf{D}$) the random process*

- $\mathbf{x}_2 \leftarrow D_{\mathbb{Z}^m, \sqrt{\mathbf{D}}, \mathbf{c}_2}$.
- $\mathbf{x}_1 \leftarrow D_{\mathbb{Z}^n, \sqrt{\Sigma/\mathbf{D}}, \mathbf{c}_1 + \mathbf{B}\mathbf{D}^{-1}(\mathbf{x}_2 - \mathbf{c}_2)}$.

produces a vector $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2) \in \mathbb{Z}^{n+m}$ such that the Rényi divergence of order infinity of $D_{\mathbb{Z}^{n+m}, \sqrt{\Sigma}, \mathbf{c}}$ from $\mathbf{x}$ is less than or equal to $1 + 4\epsilon$.

Proof:

First, we write out the probability and use Lemma 4.1 to simplify the numerator. Let $\mathbf{x}' = (\mathbf{x}'_1, \mathbf{x}'_2)$ below.

$$\Pr[\mathbf{x}_1 = \mathbf{x}'_1, \mathbf{x}_2 = \mathbf{x}'_2] = \frac{\rho_{\sqrt{\Sigma/\mathbf{D}}}(\mathbf{x}'_1 - \mathbf{c}_1 - \mathbf{B}\mathbf{D}^{-1}(\mathbf{x}'_2 - \mathbf{c}_2)) \cdot \rho_{\sqrt{\mathbf{D}}}(\mathbf{x}'_2 - \mathbf{c}_2)}{\rho_{\sqrt{\Sigma/\mathbf{D}}}(\mathbb{Z}^n - \mathbf{c}_1 - \mathbf{B}\mathbf{D}^{-1}(\mathbf{x}'_2 - \mathbf{c}_2)) \cdot \rho_{\sqrt{\mathbf{D}}}(\mathbb{Z}^m - \mathbf{c}_2)}$$

$$= \frac{\rho_{\sqrt{\Sigma}}(\mathbf{x}' - \mathbf{c})}{\rho_{\sqrt{\Sigma/\mathbf{D}}}(\mathbb{Z}^n - \mathbf{c}_1 - \mathbf{B}\mathbf{D}^{-1}(\mathbf{x}'_2 - \mathbf{c}_2)) \cdot \rho_{\sqrt{\mathbf{D}}}(\mathbb{Z}^m - \mathbf{c}_2)}$$

Regarding the denominator, we use Lemma 4.2 to see that $\mathbf{\Sigma}/\mathbf{D} \succeq \eta_\epsilon^2(\mathbb{Z}^n)$ since $\Sigma \succeq \eta_\epsilon^2(\mathbb{Z}^{n+m})$. Now, we can use Lemma 2.3 for the first gaussian sum (dependent on $\mathbf{x'}_2$) in the denominator to see,

$$\Pr[\mathbf{x}_1 = \mathbf{x'}_1, \mathbf{x}_2 = \mathbf{x'}_2] \in \alpha \cdot D_{\mathbb{Z}^{n+m}, \sqrt{\Sigma}, \mathbf{c}}(\mathbf{x'}) \cdot \left[\left(\frac{1-\epsilon}{1+\epsilon}\right), 1\right]^{-1}$$

where $\alpha = \frac{\rho_{\sqrt{\Sigma}}(\mathbb{Z}^{n+m} - \mathbf{c})}{\rho_{\sqrt{\Sigma/\mathbf{D}}}(\mathbb{Z}^n) \cdot \rho_{\sqrt{\mathbf{D}}}(\mathbb{Z}^m - \mathbf{c}_2)}$.

Next, we show $\alpha \approx 1$. Using Lemma 4.1 we expand

$$\rho_{\sqrt{\Sigma}}(\mathbb{Z}^{n+m} - \mathbf{c}) = \sum_{\mathbf{y}_2 \in \mathbb{Z}^m} \rho_{\sqrt{\mathbf{D}}}(\mathbf{y}_2 - \mathbf{c}_2) \cdot \rho_{\sqrt{\Sigma/\mathbf{D}}}(\mathbb{Z}^n - \mathbf{c}_1 - \mathbf{B}\mathbf{D}^{-1}(\mathbf{y}_2 - \mathbf{c}_2)).$$

The sum $\rho_{\sqrt{\Sigma/\mathbf{D}}}(\mathbb{Z}^n - \mathbf{c}_1 - \mathbf{B}\mathbf{D}^{-1}(\mathbf{y}_2 - \mathbf{c}_2))$ is approximately $\rho_{\sqrt{\Sigma/\mathbf{D}}}(\mathbb{Z}^n)$ because $\Sigma/\mathbf{D} \succeq \eta_\epsilon^2(\mathbb{Z}^n)$ as a consequence of Lemma 4.2 and $\Sigma \succeq \eta_\epsilon^2(\mathbb{Z}^{n+m})$. In other words,

$$\rho_{\sqrt{\Sigma/\mathbf{D}}}(\mathbb{Z}^n - \mathbf{c}_1 - \mathbf{B}\mathbf{D}^{-1}(\mathbf{y}_2 - \mathbf{c}_2)) \in \left[\frac{1-\epsilon}{1+\epsilon}, 1\right] \cdot \rho_{\sqrt{\Sigma/\mathbf{D}}}(\mathbb{Z}^n)$$

and $\alpha \in \left[\left(\frac{1-\epsilon}{1+\epsilon}\right), 1\right]$.

Finally, we have the approximation

$$\Pr[\mathbf{x}_1 = \mathbf{x'}_1, \mathbf{x}_2 = \mathbf{x'}_2] \in \left[\left(\frac{1-\epsilon}{1+\epsilon}\right), \left(\frac{1+\epsilon}{1-\epsilon}\right)\right] \cdot D_{\mathbb{Z}^{n+m}, \sqrt{\Sigma}, \mathbf{c}}(\mathbf{x'}).$$

Given the restriction on $\epsilon \in (0, 1/2]$, we have the relation we desire

$$\Pr[\mathbf{x}_1 = \mathbf{x'}_1, \mathbf{x}_2 = \mathbf{x'}_2] \in [1-4\epsilon, 1+4\epsilon] \cdot D_{\mathbb{Z}^{n+m}, \sqrt{\Sigma}, \mathbf{c}}(\mathbf{x'}).$$

□

Next, we bound the Rényi divergence of order infinity between the output of SamplePz and the desired distribution. We need to ensure each discrete gaussian convolution in the algorithm is non-degenerate. We do not analyze the statistical loss from the floating point computations. As shown in Lemma 4.3, we need $\Sigma/\mathbf{D} \succeq \eta_\epsilon^2(\mathbb{Z}^{n_0})$ and $\mathbf{D} \succeq \eta_\epsilon^2(\mathbb{Z}^{n_1})$ at *each* of the n discrete gaussian convolutions. Thankfully, this is met through a simple condition on Σ_p as hinted to in Lemma 4.2.

Theorem 4.1 *Let $0 < \epsilon \leq 1/2$. If $\Sigma_p \succeq \eta_\epsilon^2(\mathbb{Z}^{n(2+\log q)})$, then* SamplePz *returns a perturbation with a Rényi divergence of order infinity $R_\infty(D_{\mathbb{Z}^{n(2+\log q)}, \sqrt{\Sigma_p}} || \text{SamplePz}) \leq 1 + 12n\hat{\epsilon}$.*

Proof: Since each covariance given to SampleFz is a Schur complement or a principal submatrix of a Schur complement of Σ_p, Lemma 4.2 and the interlacing theorems (Theorem 2.1 and Theorem2.2) imply the conditions for Lemma 4.3 are met. As there are $n-1$ convolutions (inner nodes of a full binary tree of depth $\log n$), a quick induction argument shows the probability distribution of the output of SamplePz is in the interval $[(1-4\epsilon)^{3(n-1)}, (1+4\epsilon)^{3(n-1)}] \cdot D_{\mathbb{Z}^{n(2+\log q)}, \sqrt{\Sigma_p}}(\mathbf{x})$. Then, we have $R_\infty(D_{\mathbb{Z}^{n(2+\log q)}, \sqrt{\Sigma_p}} || \text{SamplePz}) \leq (1+4\epsilon)^{3(n-1)} \approx 1 + 12n\hat{\epsilon}$. □

For common parameters $\epsilon = 2^{-128}$ and $n = 1024$, we have $1 - (1+4\epsilon)^{3(n-1)} \approx 2^{-114}$.

Acknowledgment

We thank Léo Ducas, Yuriy Polyakov, Kurt Rohloff, and Michael Walter for their helpful discussions as well as the anonymous reviewers for their helpful feedback and suggestions.

References

[1] S. Agrawal. Stronger security for reusable garbled circuits, general definitions and attacks. In *CRYPTO 2017*, pages 3–35, 2017.

[2] S. Agrawal, D. Boneh, and X. Boyen. Efficient lattice (H)IBE in the standard model. In H. Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 553–572. Springer, 2010.

[3] S. Agrawal, D. Boneh, and X. Boyen. Lattice basis delegation in fixed dimension and shorter-ciphertext hierarchical IBE. In Rabin [59], pages 98–115.

[4] S. Agrawal, X. Boyen, V. Vaikuntanathan, P. Voulgaris, and H. Wee. Functional encryption for threshold functions (or fuzzy IBE) from lattices. In Fischlin et al. [30], pages 280–297.

[5] M. Ajtai. Generating hard instances of the short basis problem. In *Automata, Languages and Programming, 26th International Colloquium, ICALP'99*, pages 1–9.

[6] J. Alperin-Sheriff and C. Peikert. Circular and KDM security for identity-based encryption. In Fischlin et al. [30], pages 334–352.

[7] J. Alwen and C. Peikert. Generating shorter bases for hard random lattices. *Theory Comput. Syst.*, 48(3):535–553, 2011.

[8] S. Bai, A. Langlois, T. Lepoint, D. Stehlé, and R. Steinfeld. Improved security proofs in lattice-based cryptography: Using the rényi divergence rather than the statistical distance. In *ASIACRYPT 2015*, pages 3–24, 2015.

[9] R. E. Bansarkhani and J. A. Buchmann. Improvement and efficient implementation of a lattice-based signature scheme. In *Selected Areas in Cryptography - SAC 2013*, pages 48–67.

[10] M. Bellare, E. Kiltz, C. Peikert, and B. Waters. Identity-based (lossy) trapdoor functions and applications. In Pointcheval and Johansson [58], pages 228–245.

[11] R. Bendlin, S. Krehbiel, and C. Peikert. How to share a lattice trapdoor: Threshold protocols for signatures and (H)IBE. In *Applied Cryptography and Network Security, ACNS 2013*, pages 218–236.

[12] D. Boneh and D. M. Freeman. Homomorphic signatures for polynomial functions. In K. G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *Lecture Notes in Computer Science*, pages 149–168. Springer, 2011.

[13] D. Boneh and D. M. Freeman. Linearly homomorphic signatures over binary fields and new tools for lattice-based signatures. In D. Catalano, N. Fazio, R. Gennaro, and A. Nicolosi, editors, *Public Key Cryptography - PKC 2011*, volume 6571 of *Lecture Notes in Computer Science*, pages 1–16. Springer, 2011.

[14] D. Boneh, C. Gentry, S. Gorbunov, S. Halevi, V. Nikolaenko, G. Segev, V. Vaikuntanathan, and D. Vinayagamurthy. Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In P. Q. Nguyen and E. Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *Lecture Notes in Computer Science*, pages 533–556. Springer, 2014.

[15] D. Boneh, S. Kim, and V. Nikolaenko. Lattice-based DAPS and generalizations: Self-enforcement in signature schemes. In *Applied Cryptography and Network Securitym ACNS 2017*, pages 457–477, 2017.

[16] X. Boyen. Attribute-based functional encryption on lattices. In *TCC*, pages 122–142, 2013.

[17] X. Boyen and Q. Li. Attribute-based encryption for finite automata from LWE. In M. H. Au and A. Miyaji, editors, *Provable Security, ProvSec 2015*, volume 9451 of *Lecture Notes in Computer Science*, pages 247–267. Springer, 2015.

[18] X. Boyen and Q. Li. Towards tightly secure lattice short signature and id-based encryption. In *ASIACRYPT 2016*, pages 404–434, 2016.

[19] Z. Brakerski and V. Vaikuntanathan. Circuit-abe from LWE: unbounded attributes and semi-adaptive security. In M. Robshaw and J. Katz, editors, *CRYPTO 2016*, volume 9816 of *Lecture Notes in Computer Science*, pages 363–384. Springer, 2016.

[20] Z. Brakerski, V. Vaikuntanathan, H. Wee, and D. Wichs. Obfuscating conjunctions under entropic ring LWE. In M. Sudan, editor, *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science*, pages 147–156. ACM, 2016.

[21] J. Camenisch, G. Neven, and M. Rückert. Fully anonymous attribute tokens from lattices. In I. Visconti and R. D. Prisco, editors, *Security and Cryptography for Networks, SCN 2012*, volume 7485 of *Lecture Notes in Computer Science*, pages 57–75. Springer, 2012.

[22] R. Canetti and Y. Chen. Constraint-hiding constrained prfs for nc^1 from LWE. In *EUROCRYPT 2017*, pages 446–476, 2017.

[23] R. Canetti and J. A. Garay, editors. *CRYPTO 2013*, volume 8042 of *Lecture Notes in Computer Science*. Springer, 2013.

[24] D. Cash, D. Hofheinz, E. Kiltz, and C. Peikert. Bonsai trees, or how to delegate a lattice basis. *J. Cryptology*, 25(4):601–639, 2012.

[25] M. Clear and C. McGoldrick. Multi-identity and multi-key leveled FHE from learning with errors. In Gennaro and Robshaw [31], pages 630–656.

[26] W. Dai, Y. Doröz, Y. Polyakov, K. Rohloff, H. Sajjadpour, E. Savas, and B. Sunar. Implementation and evaluation of a lattice-based key-policy ABE scheme. *IACR Cryptology ePrint Archive*, 2017:601, 2017.

[27] A. Davidson. Obfuscation of bloom filter queries from ring-lwe. *IACR Cryptology ePrint Archive*, 2017:448, 2017.

[28] L. Ducas and P. Q. Nguyen. Faster gaussian lattice sampling using lazy floating-point arithmetic. In X. Wang and K. Sako, editors, *ASIACRYPT 2012*, volume 7658 of *Lecture Notes in Computer Science*, pages 415–432. Springer, 2012.

[29] L. Ducas and T. Prest. Fast fourier orthogonalization. In S. A. Abramov, E. V. Zima, and X. Gao, editors, *Proceedings of the ACM on International Symposium on Symbolic and Algebraic Computation, ISSAC 2016*, pages 191–198. ACM, 2016.

[30] M. Fischlin, J. A. Buchmann, and M. Manulis, editors. *Public Key Cryptography - PKC 2012*, volume 7293 of *Lecture Notes in Computer Science*. Springer, 2012.

[31] R. Gennaro and M. Robshaw, editors. *CRYPTO 2015*, volume 9216 of *Lecture Notes in Computer Science*. Springer, 2015.

[32] C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In C. Dwork, editor, *Proceedings of the 40th Annual ACM Symposium on Theory of Computing*, pages 197–206. ACM, 2008.

[33] C. Gentry, A. Sahai, and B. Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In Canetti and Garay [23], pages 75–92.

[34] S. Gorbunov, V. Vaikuntanathan, and H. Wee. Attribute-based encryption for circuits. *J. ACM*, 62(6):45:1–45:33, 2015. Prelim. version in STOC 2013.

[35] S. Gorbunov, V. Vaikuntanathan, and H. Wee. Predicate encryption for circuits from LWE. In Gennaro and Robshaw [31], pages 503–523.

[36] S. Gorbunov, V. Vaikuntanathan, and D. Wichs. Leveled fully homomorphic signatures from standard lattices. In R. A. Servedio and R. Rubinfeld, editors, *Proceedings ACM on Symposium on Theory of Computing, STOC 2015*, pages 469–477. ACM, 2015.

[37] S. D. Gordon, J. Katz, and V. Vaikuntanathan. A group signature scheme from lattice assumptions. In M. Abe, editor, *ASIACRYPT 2010*, volume 6477 of *Lecture Notes in Computer Science*, pages 395–412. Springer, 2010.

[38] K. D. Gür, Y. Polyakov, K. Rohloff, G. W. Ryan, E. Savaş, and H. Sajjadpour. Efficient implementation of gaussian sampling for trapdoor lattices and its applications. In preparation, (Personal communication), 2017.

[39] K. D. Gur, Y. Polyakov, K. Rohloff, G. W. Ryan, and E. Savas. Implementation and evaluation of improved gaussian sampling for lattice trapdoors. *IACR Cryptology ePrint Archive*, 2017:285, 2017.

[40] J. Howe, T. Pöppelmann, M. O'Neill, E. O'Sullivan, and T. Güneysu. Practical lattice-based digital signature schemes. *ACM Trans. Embedded Comput. Syst.*

[41] C. F. F. Karney. Sampling exactly from the normal distribution. *ACM Trans. Math. Softw.*, 42(1):3:1–3:14, Jan. 2016.

[42] S. Kim and D. J. Wu. Watermarking cryptographic functionalities from standard lattice assumptions. In *CRYPTO 2017*, pages 503–536, 2017.

[43] P. N. Klein. Finding the closest lattice vector when it's unusually close. In D. B. Shmoys, editor, *Proceedings of ACM-SIAM Symposium on Discrete Algorithms*, pages 937–941. ACM/SIAM, 2000.

[44] F. Laguillaumie, A. Langlois, B. Libert, and D. Stehlé. Lattice-based group signatures with logarithmic signature size. In K. Sako and P. Sarkar, editors, *ASIACRYPT 2013*, volume 8270 of *Lecture Notes in Computer Science*, pages 41–61. Springer, 2013.

[45] A. Langlois, S. Ling, K. Nguyen, and H. Wang. Lattice-based group signature scheme with verifier-local revocation. In H. Krawczyk, editor, *Public-Key Cryptography - PKC 2014*, volume 8383 of *Lecture Notes in Computer Science*, pages 345–361. Springer, 2014.

[46] S. Ling, K. Nguyen, D. Stehlé, and H. Wang. Improved zero-knowledge proofs of knowledge for the ISIS problem, and applications. In K. Kurosawa and G. Hanaoka, editors, *Public-Key Cryptography - PKC 2013*, volume 7778 of *Lecture Notes in Computer Science*, pages 107–124. Springer, 2013.

[47] V. Lyubashevsky, D. Micciancio, C. Peikert, and A. Rosen. SWIFFT: A modest proposal for FFT hashing. In K. Nyberg, editor, *Fast Software Encryption, 15th International Workshop, FSE 2008*, volume 5086 of *Lecture Notes in Computer Science*, pages 54–72. Springer, 2008.

[48] V. Lyubashevsky, C. Peikert, and O. Regev. On ideal lattices and learning with errors over rings. *J. ACM*, 60(6):43, 2013.

[49] V. Lyubashevsky, C. Peikert, and O. Regev. A toolkit for ring-lwe cryptography. In T. Johansson and P. Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *Lecture Notes in Computer Science*, pages 35–54. Springer, 2013.

[50] D. Micciancio. Generalized compact knapsacks, cyclic lattices, and efficient one-way functions. *Computational Complexity*, 16(4):365–411, 2007.

[51] D. Micciancio and C. Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In Pointcheval and Johansson [58], pages 700–718.

[52] D. Micciancio and O. Regev. Worst-case to average-case reductions based on gaussian measures. *SIAM J. Comput.*, 37(1):267–302, 2007.

[53] D. Micciancio and M. Walter. Gaussian sampling over the integers: Efficient, generic, constant-time. In *CRYPTO 2017*, pages 455–485, 2017.

[54] P. Q. Nguyen, J. Zhang, and Z. Zhang. Simpler efficient group signatures from lattices. In J. Katz, editor, *Public-Key Cryptography - PKC 2015 - 18th IACR International Conference on Practice and Theory in Public-Key Cryptography, Gaithersburg, MD, USA, March 30 - April 1, 2015, Proceedings*, volume 9020 of *Lecture Notes in Computer Science*, pages 401–426. Springer, 2015.

[55] C. Peikert. An efficient and parallel gaussian sampler for lattices. In Rabin [59], pages 80–97.

[56] C. Peikert. A decade of lattice cryptography. *Foundations and Trends in Theoretical Computer Science*, 10(4):283–424, 2016.

[57] C. Peikert, V. Vaikuntanathan, and B. Waters. A framework for efficient and composable oblivious transfer. In D. Wagner, editor, *CRYPTO 2008*, volume 5157 of *Lecture Notes in Computer Science*, pages 554–571. Springer, 2008.

[58] D. Pointcheval and T. Johansson, editors. *Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings*, volume 7237 of *Lecture Notes in Computer Science*. Springer, 2012.

[59] T. Rabin, editor. *Advances in Cryptology - CRYPTO 2010, 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings*, volume 6223 of *Lecture Notes in Computer Science*. Springer, 2010.

[60] H. Wee. Public key encryption against related key attacks. In Fischlin et al. [30], pages 262–279.

[61] F. Zhang. *The Schur Complement and Its Applications*, volume 4. Springer Science, 2006.

A ML Analysis of SampleG

Here we give the proof of Proposition 3.1. We restate the proposition for convenience.

Proposition A.1 *Fix an $\epsilon > 0$ and let $s \geq (b+1)\cdot\eta_\epsilon(\mathbb{Z})$. For any one-dimensional sampler $\textsc{SampleZ}_t(\bar{\sigma}, \bar{c}, s)$ that takes as inputs approximated centers $\bar{c} \in [0,1)$ and variances $\bar{\sigma} \in [s/(b+1), s\cdot b/(b+1)]$ represented as floating point numbers with mantissa length m,*
$\Delta_{\text{ML}}(\textsc{SampleG}^{D^t_{\mathbb{Z},\sigma,c}}, \textsc{SampleG}^{\textsc{SampleZ}_t(\bar{\sigma},\bar{c})}) \leq$
$2k[O(t^2 2^{-m}) + \max_{\bar{\sigma},\bar{c}} \Delta_{\text{ML}}(\textsc{SampleZ}_t(\bar{\sigma},\bar{c},s), D^t_{\mathbb{Z},\bar{\sigma},\bar{c}})]$.

Before we begin the proof, we note that $d_{k-1} = q/b^k \in [1/b, 1]$ since $k = \lceil \log_b q \rceil$. This implies that every variance fed to SampleZ is in the range $[s/(b+1), s\cdot b/(b+1)] \subseteq [s/(b+1), s]$. We restrict all truncated one-dimensional discrete gaussians to $\mathbb{Z} \cap [c - t\cdot s, c + t\cdot s]$ since it is unclear when $\mathbb{Z} \cap [c - t\cdot\sigma, c + t\cdot\sigma] = \mathbb{Z} \cap [c - t\cdot\bar{\sigma}, c + t\cdot\bar{\sigma}]$ when using floating point variances $\bar{\sigma}$. The ML distance is undefined when these two sets are not equal.

Proof: First, we use the triangle inequality on ML distance in order to pair together terms for an easier analysis.

$\Delta_{\text{ML}}(\textsc{SampleG}^{D^t_{\mathbb{Z},\sigma,c}}, \textsc{SampleG}^{\textsc{SampleZ}_t(\bar{\sigma},\bar{c},s)}) \leq$
$\Delta_{\text{ML}}(\textsc{SampleG}^{D^t_{\mathbb{Z},\sigma,c}}, \textsc{SampleG}^{D^t_{\mathbb{Z},\bar{\sigma},c}}) + \Delta_{\text{ML}}(\textsc{SampleG}^{D^t_{\mathbb{Z},\bar{\sigma},c}}, \textsc{SampleG}^{D^t_{\mathbb{Z},\bar{\sigma},\bar{c}}}) +$

$\Delta_{\text{ML}}(\text{SAMPLEG}^{D^t_{\mathbb{Z},\bar{\sigma},\bar{c}}}, \text{SAMPLEG}^{\text{SAMPLEZ}_t(\bar{\sigma},\bar{c},s)})$.

Next, we use the data processing inequality on ML distance where we treat SAMPLEG as a function of $2k$ correlated samples from a one-dimensional discrete gaussian sampler. From [Lemma 3.2, MW17], we get the following inequality:

$\Delta_{\text{ML}}(\text{SAMPLEG}^{D^t_{\mathbb{Z},\sigma,c}}, \text{SAMPLEG}^{\text{SAMPLEZ}_t(\bar{\sigma},\bar{c},s)}) \leq$
$2k \cdot \max_{\sigma_i,c_i}[\Delta_{\text{ML}}(D^t_{\mathbb{Z},\sigma_1,c_1}, D^t_{\mathbb{Z},\bar{\sigma}_1,c_1}) + \Delta_{\text{ML}}(D^t_{\mathbb{Z},\bar{\sigma}_2,c_2}, D^t_{\mathbb{Z},\bar{\sigma}_2,\bar{c}_2}) +$
$\Delta_{\text{ML}}(D^t_{\mathbb{Z},\bar{\sigma}_3,\bar{c}_3}, \text{SAMPLEZ}_t(\bar{\sigma}_3,\bar{c}_3,s))]$.

The maximum is taken over all $c_i \in [0,1)$ and $\sigma_i \in [s/(b+1), s\cdot b/(b+1)]$. Let $\mathbb{Z}^t = \mathbb{Z} \cap [c - t\cdot s, c + t\cdot s]$. We bound $\max_{\sigma_1,c_1}\ \Delta_{\text{ML}}(D^t_{\mathbb{Z},\sigma_1,c_1}, D^t_{\mathbb{Z},\bar{\sigma}_1,c_1})$ as follows:

$$\begin{aligned}\max_{\sigma_1,c_1}\ \Delta_{\text{ML}}(D^t_{\mathbb{Z},\sigma_1,c_1}, D^t_{\mathbb{Z},\bar{\sigma}_1,c_1}) &= \max_{\sigma_1,c_1,x\in\mathbb{Z}^t} |\ln D^t_{\mathbb{Z},\sigma_1,c_1}(x) - \ln D^t_{\mathbb{Z},\bar{\sigma}_1,c_1}(x)| \\ &= \max_{\sigma_1,c_1,x\in\mathbb{Z}^t} \left|\pi(x-c)^2\left[\frac{1}{\sigma_1^2} - \frac{1}{\bar{\sigma}_1^2}\right] + \ln\frac{\rho_{\bar{\sigma}_1,c_1}(\mathbb{Z})}{\rho_{\sigma_1,c_1}(\mathbb{Z})}\right|.\end{aligned}$$

Since $\sigma_1, \bar{\sigma}_1 \geq \eta_\epsilon(\mathbb{Z})$, we can approximate $\rho_{\sigma_1,c}(\mathbb{Z}) \in [(1-\epsilon)^2, (1+\epsilon)^2]\cdot\sigma$ and $\rho_{\bar{\sigma}_1,c}(\mathbb{Z}) \in [(1-\epsilon)^2, (1+\epsilon)^2]\cdot\bar{\sigma}$. Using the bound on the relative error of $\bar{\sigma}_1$ ($\bar{\sigma}_1 \in [1-2^{-m}, 1+2^{-m}]\cdot\sigma_1$), we can bound the expression with a simplified form below.

$$\begin{aligned}\max_{\sigma_1,c_1}\ \Delta_{\text{ML}}(D^t_{\mathbb{Z},\sigma_1,c_1}, D^t_{\mathbb{Z},\bar{\sigma}_1,c_1}) \leq & \\ \max_{\sigma_1}\left|\pi\frac{t^2s^2}{\sigma_1^2}\cdot\frac{\bar{\sigma}_1^2-\sigma_1^2}{\sigma_1^2} + 2\hat{\epsilon} + 2^{\hat{-m}}\right| \leq & \\ \pi t^2(b+1)^2(2^{-m+1}+2^{-2m}) + \hat{\epsilon} + 2^{\hat{-m}}. &\end{aligned}$$

The proof for $\Delta_{\text{ML}}(D^t_{\mathbb{Z},\bar{\sigma}_2,c_2}, D^t_{\mathbb{Z},\bar{\sigma}_2,\bar{c}_2})$ is nearly identical except we get a term linear in t, yielding a bound of $O(t\cdot 2^{-m})$. □

B QR Factorization for the Basis $\mathbf{B}_q$

Here we show that despite $\mathbf{B}_q$ having a sparse $\mathbf{R}$ matrix in its QR-factorization, this does not lead to an alternative $\Theta(\log q)$-time sampling algorithm for the applications we are concerned with. The QR-factorization of a non-singular matrix $\mathbf{S}$ is $\mathbf{S} = \mathbf{QR}$ where $\mathbf{Q}$ is orthogonal and $\mathbf{R}$ is upper-triangular.

The motivation for such an algorithm comes from generic lattice algorithms, like BKZ lattice reduction, where we view the vector space holding our lattice in the basis given by the upper-triangular $\mathbf{R}$ since $\mathbf{Q}$ is orthogonal. The sparsity of $\mathbf{R}$ yields clear computational advantages.

In the G-lattice setting, the basis $\mathbf{B}_q = \mathbf{QR}$ *always* has a sparse $\mathbf{R}$ matrix (though $\mathbf{Q}$ is not sparse). This leads to a linear time algorithm to sample $D_{\mathcal{L}(\mathbf{R}),s}$ by using the canonical randomized nearest plane algorithm and a linear time algorithm for applications *if* we can view the ambient vector space in terms of $\mathbf{R}$ as a basis. Unfortunately, we cannot do this in the G-lattice setting.

Recall the general G-lattice paradigm: we have a secret trapdoor matrix $\mathbf{T}$ with small integer entries, a public psuedo-random matrix $\mathbf{A} = [\hat{\mathbf{A}}|\mathbf{G} - \hat{\mathbf{A}}\cdot\mathbf{T}]$, and we want to return a short vector in $\Lambda^\perp_{\mathbf{u}}(\mathbf{A}) = \{\mathbf{x} \in \mathbb{Z}^m : \mathbf{A}\cdot\mathbf{x} = \mathbf{u} \mod q\}$. The way we sample $\Lambda^\perp_{\mathbf{u}}(\mathbf{A})$ is as follows:

1. sample the perturbation $\mathbf{p} \sim D_{\mathbb{Z}^{n(2+\log q)}, \sqrt{s^2\mathbf{I} - \mathbf{M}\mathbf{M}^T}}$ where $\mathbf{M} = \begin{bmatrix}\mathbf{T}\\ \mathbf{I}\end{bmatrix}$
2. set the new coset $\mathbf{v} := \mathbf{u} - \mathbf{A}\mathbf{p} \mod q$

3. sample the G-lattice $\mathbf{y} \sim D_{\Lambda^\perp(\mathbf{G})_\mathbf{v},s} = D_{\mathcal{L}(\mathbf{B}_q)+\mathbf{v},s}$ where $\mathbf{G} = \mathbf{I}_n \otimes \mathbf{g}^T$
4. return $\mathbf{p} + \begin{bmatrix}\mathbf{T}\\ \mathbf{I}\end{bmatrix}\mathbf{y}$.

Next, we only consider the zero-coset of $\Lambda^\perp(\mathbf{G})$ for simplicity. Usually $\mathbf{y} = (\mathbf{I}_n \otimes \mathbf{B}_q)\mathbf{z}$ for $\mathbf{z} \in \mathbb{Z}^{n \log q}$. But if we were to use the sparsity of $\mathbf{R}$, then

$$\mathbf{y} = (\mathbf{I}_n \otimes \mathbf{R})\mathbf{z} = (\mathbf{I}_n \otimes \mathbf{Q}^T\mathbf{B}_q)\mathbf{z}.$$

Therefore, we would have to apply $\mathbf{Q}' := \mathbf{I}_n \otimes \mathbf{Q}$ as a linear transformation to $\mathbf{y}$ ($\Theta(n \log^2 q)$ time) yielding a quadratic increase (in $\log q$) in the last step as well as a quadratic increase in storage.