View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by CWI's Institutional Repository

Integral Matrix Gram Root and Lattice Gaussian Sampling Without Floats

Léo Ducas[1], Steven Galbraith[2], Thomas Prest[3], and Yang Yu[4(✉)]

[1] Centrum Wiskunde en Informatica, Amsterdam, The Netherlands
ducas@cwi.nl
[2] Mathematics Department, University of Auckland, Auckland, New Zealand
s.galbraith@auckland.ac.nz
[3] PQShield Ltd, Oxford, UK
thomas.prest@pqshield.com
[4] Univ Rennes, CNRS, IRISA, Rennes, France
yang.yu0986@gmail.com

Abstract. Many advanced lattice based cryptosystems require to sample lattice points from Gaussian distributions. One challenge for this task is that all current algorithms resort to floating-point arithmetic (FPA) at some point, which has numerous drawbacks in practice: it requires numerical stability analysis, extra storage for high-precision, lazy/backtracking techniques for efficiency, and may suffer from weak determinism which can completely break certain schemes.

In this paper, we give techniques to implement Gaussian sampling over general lattices without using FPA. To this end, we revisit the approach of Peikert, using perturbation sampling. Peikert's approach uses continuous Gaussian sampling and some decomposition $\mathbf{\Sigma} = \mathbf{A}\mathbf{A}^t$ of the target covariance matrix $\mathbf{\Sigma}$. The suggested decomposition, e.g. the Cholesky decomposition, gives rise to a square matrix $\mathbf{A}$ with real (not integer) entries. Our idea, in a nutshell, is to replace this decomposition by an integral one. While there is in general no integer solution if we restrict $\mathbf{A}$ to being a square matrix, we show that such a decomposition can be efficiently found by allowing $\mathbf{A}$ to be wider (say $n \times 9n$). This can be viewed as an extension of Lagrange's four-square theorem to matrices. In addition, we adapt our integral decomposition algorithm to the ring setting: for power-of-2 cyclotomics, we can exploit the tower of rings structure for improved complexity and compactness.

1 Introduction

Lattice based cryptography is a promising post-quantum alternative to cryptography based on integer factorization and discrete logarithms. One of its attractive features is that lattices can be used to build various powerful cryptographic primitives including identity based encryption (IBE) [1,8,11,19], attribute based encryption (ABE) [5,20], functional encryption [2], group signatures [22,23,31]

© International Association for Cryptologic Research 2020
A. Canteaut and Y. Ishai (Eds.): EUROCRYPT 2020, LNCS 12106, pp. 608–637, 2020.
https://doi.org/10.1007/978-3-030-45724-2_21

and so on [7,18]. A core component of many advanced lattice based cryptosystems is sampling lattice points from discrete Gaussians, given a short basis (i.e. a trapdoor) [19,25,32].

Gaussian sampling is important to prevent leaking secret information. Indeed early lattice trapdoors have suffered from statistical attacks [14,30,37]. In 2008, Gentry, Peikert and Vaikuntanathan first showed that Gaussian distributions [19] can prevent such leaks, and that Klein's algorithm [21] could sample efficiently from a negligibly close distribution. This algorithm uses the Gram-Schmidt orthogonalization, which requires either arithmetic over the rationals with very large denominators, or floating-point approximations. An alternative algorithm was proposed by Peikert in [32], where most of the expensive computation, including floating-point arithmetic, can be done in an offline phase at the cost of somewhat increasing the width of the sampled Gaussian. This technique turned out to be particularly convenient in the lattice-trapdoor framework of Micciancio and Peikert [25].

We now explain Peikert's algorithm in more details. Let $\mathbf{B}$ be the input basis and the target distribution be a spherical discrete Gaussian of width s, center $\mathbf{c}$ and over the lattice $\mathcal{L}$ spanned by $\mathbf{B}$. Note that spherical Gaussian sampling over $\mathbb{Z}^n$ is easy, by applying $\mathbf{B}$ as a transformation, it is also easy to sample a Gaussian over $\mathcal{L}$ but of covariance $\mathbf{\Sigma} = \mathbf{B}\mathbf{B}^t$. To produce target samples, Peikert proposed to use convolution, that is adding some perturbation vector of covariance $\mathbf{\Sigma}_p = s^2\mathbf{I} - \mathbf{\Sigma}$ on the center $\mathbf{c}$. Indeed for continuous Gaussians, the resulting distribution is of covariance $s^2\mathbf{I}$. In [32], Peikert showed that this fact also holds for discrete Gaussians under some conditions. In summary, Peikert's approach consists of two phases:

- offline phase: one samples a perturbation vector of covariance $\mathbf{\Sigma}_p = s^2\mathbf{I} - \mathbf{\Sigma}$;
- online phase: one first samples a spherical Gaussian over $\mathbb{Z}^n$ and then applies the transformation of $\mathbf{B}$.

The online sampling can be rather efficient and fully performed over the integers [17,25,32]. By contrast, the offline sampling uses continuous Gaussian sampling and requires some matrix $\mathbf{A}$ such that $\mathbf{\Sigma}_p = \mathbf{A}\mathbf{A}^t$. The only suggested way to find such $\mathbf{A}$ is the Cholesky decomposition. Therefore high-precision floating-point arithmetic is still heavily used in the offline phase.

We now list some of the numerous drawbacks of high-precision FPA when it comes to practical efficiency and security in the wild.

- First, one needs to perform a *tedious numerical stability analysis* to determine what level of precision is admissible, and how much security is lost. Indeed, while such analysis may be reasonable when done asymptotically, doing a concrete and tight analysis requires significant effort [3,13,33,34], and may be considered too error-prone for deployed cryptography. Moreover, efficient numerical stability analysis for cryptosystems based on generic decision problems remain open.

- Second, for a security level of λ-bits, it incurs *significant storage overheads* as one requires at least a precision of $\lambda/2$ bits[1] for each matrix entry, while the trapdoor basis itself only needs $\log(s)$ bits per entry, where $s = \text{poly}(\lambda)$ in simple cryptosystems.
- Thirdly, the requirement for high-precision arithmetic would *significantly slow down* those sampling algorithms (may it be fix-point or floating-point arithmetic). While it has been shown in [13] that one can do most of the operations at lower precision, the proposed technique requires *complicated backtracking*, and high-precision arithmetic from time to time. While asymptotically interesting, it is unclear whether this technique is practical; in particular it has, to our knowledge never been implemented. It also seems particularly hard to protect against timing attacks.
- Finally we mention the intrinsic *weak determinism* of floating-point arithmetic. It is essential to de-randomize trapdoor sampling, as revealing two different vectors close to a single target instantly reveals a (secret) short vector of the lattice. Even with the same random stream, we need to assume that the rest of the algorithm is deterministic. In the case of high-precision arithmetic, one would for example have to assume that the `mpfr` library behaves exactly the same across different architectures and versions. But even at low precision, the use of native floats can be tricky despite deterministic IEEE standards. For example, while both ADD and MULTIPLY instructions are deterministically defined by the IEEE standard, the standard also allows the combined 'MULTIPLY-AND-ACCUMULATE' instruction to behave differently from applying both instructions sequentially, as long as the result is at least as precise [36]. As FPA addition is *not associative*, it is crucial to specify the order of operations for matrix-vector products as part of the scheme, and to not leave it as an implementation detail. Furthermore, compilers such as `gcc` do not guarantee determinism when considering code optimization over floating-point computation [29].

Our contribution. We present a new perturbation sampling algorithm in which no floating-point arithmetic is used. Compared with Peikert's algorithm [32], our new algorithm has the following features. A more detailed comparison is available in Sect. 5, with Tables 2 and 3.

- *Similar quality.* The final Gaussian width s achieved by our technique is only larger than its minimum by a factor of $1 + o(1)$: the parameters of the whole cryptosystems will be unaffected.
- *No need for FPA and less precision.* All operations are performed over integers of length about $\log s$, while previous algorithms required floating points with a mantissa of length at least $\lambda/2 + \log s$.
- *Less memory.* While the intermediate matrix, i.e. the Gram root of the covariance, is rectangular, it is integral and of a regular structure, requiring only $\approx n^2 \log s$, instead of $n^2(\lambda + \log s)/2$ bits for Cholesky decomposition [25,32].

[1] Unless one assumes strong bounds on the number of attackers' queries, as done in [34].

– *Simpler base sampling.* Only two kinds of base samplers are required: $D_{\mathbb{Z},Lr}$ and $D_{\mathbb{Z},r,c}$ with $c \in \frac{1}{L} \cdot \mathbb{Z}$, where L can be any integer larger than a polynomial bound; choosing L as a power of two is known to be particularly convenient [28].

In summary, not only do we get rid of FPA and its weak determinism, we also improve time and memory consumption; when $s = \text{poly}(\lambda)$ this improvement factor is quasilinear. In practice, it may allow to implement an HIBE or ABE with a few levels before having to resort to multi-precision arithmetic (note that the parameter s grows exponentially with the depth of such schemes). Compared to traditional samplers, we expect that the absence of floating-point arithmetic in our sampler will make it more amenable to side-channel countermeasures such as masking. We leave this for future work.

Techniques. Our main idea stems from the observation that, at least in the continuous case, sampling a Gaussian of covariance $\mathbf{\Sigma}$ can be done with using a matrix $\mathbf{A}$ such that $\mathbf{A}\mathbf{A}^t = \mathbf{\Sigma}$, may $\mathbf{A}$ not be a square matrix. This idea was already implicit in [28,32], and we push it further.

The first step is to prove that the above statement also holds in the discrete case. We show that when $\mathbf{A} \cdot \mathbb{Z}^m = \mathbb{Z}^n$, the distribution of $\mathbf{Ax}$ where $\mathbf{x}$ is drawn from $D_{\mathbb{Z}^m,r}$ is statistically close to $D_{\mathbb{Z}^n,r\sqrt{\mathbf{A}\mathbf{A}^t}}$ under some smoothness condition with respect to the orthogonal lattice $\Lambda^\perp(\mathbf{A})$.

Now the difficult step under study boils down to finding a Gram root of a given matrix (in the context of Peikert's algorithm, $\mathbf{A}\mathbf{A}^t = d\mathbf{I} - \mathbf{\Sigma}$). To avoid the FPA issues, we want this Gram root integral. Driven by this, we proceed to study the Integral Gram Decomposition Problem denoted by $\mathsf{IGDP}_{n,B,d,m}$ as follows: given an integral symmetric matrix $\mathbf{\Sigma} \in \mathbb{Z}^{n\times n}$ with $\|\mathbf{\Sigma}\|_2 \leq B$, find an integral matrix $\mathbf{A} \in \mathbb{Z}^{n\times m}$ such that $\mathbf{A}\mathbf{A}^t = d\mathbf{I}_n - \mathbf{\Sigma}$. For $n = 1$, Lagrange's 4-square theorem has provided a solution to IGDP. Our goal is finding an algorithmic generalization of such a decomposition to larger matrices, which will be bootstrapped from the case of $n = 1$. Aiming at $\mathsf{IGDP}_{n,B,d,m}$, our initial method is recursive, and can be summarized as the following reduction

$$\mathsf{IGDP}_{n,B,d,m} \to \mathsf{IGDP}_{n-1,B',d,m'}$$

where $B' \approx B$, $m = m' + \lceil \log_b B \rceil + 4$ and b is the base of the used gadget decomposition. The reduction is constructive: by gadget decomposition (also called b-ary decomposition), one first finds a matrix $\mathbf{T}$ such that $\mathbf{T}\mathbf{T}^t$ has the same first row and column as $d\mathbf{I}_n - \mathbf{\Sigma}$ except the diagonal element, and then clears out the remaining diagonal element by the 4-square decomposition given by Lagrange's theorem. However, this decomposition requires $d \gg B$, which significantly enlarges the width of corresponding Gaussian. To overcome this issue, we develop another tool called eigenvalue reduction, which can be viewed as the following reduction:

$$\mathsf{IGDP}_{n,B,d,m} \to \mathsf{IGDP}_{n,B',d-B,m-n}$$

with $B' \ll B$. By eigenvalue reduction, the final overhead on the Gaussian width is introduced during the decomposition on a small matrix, which becomes negligible compared with the original parameter. Combining the integral decomposition for $d \gg B$ and the eigenvalue reduction, we arrive at a solution to $\mathsf{IGDP}_{n,B,d,m}$ of a somewhat large B, say $B = \omega(n^4)$. This is the case of some advanced lattice based schemes, such as hierarchical IBE [1,8] and ABE [20]. Furthermore, if a few, say $O(\log n)$, bits of rational precision are permitted, we can find an almost integral Gram root for general positive definite matrices.

Techniques in the ring setting. The aforementioned algorithms apply to the ring setting, but the decompositions break the ring structure and thus lose the efficiency improvement provided by rings. To improve efficiency, we devised ring-based algorithms. The $\mathsf{IGDP}_{n,B,d,m}$ problem is naturally generalized to the ring setting by adding the underlying ring $\mathcal{R}$ as a parameter. To tackle the $\mathsf{IGDP}_{\mathcal{R},n,B,d,m}$ problem, we first study the special case $\mathsf{IGDP}_{\mathcal{R},1,B,d,m}$. We propose an analogue of 4-square decomposition in the power-of-2 cyclotomic ring, i.e. $\mathcal{R}_{2w} = \mathbb{Z}[x]/(x^w + 1)$ where $w = 2^l$. At a high level, our solution performs the reduction

$$\mathsf{IGDP}_{\mathcal{R}_{2w},1,B,d,m} \to \mathsf{IGDP}_{\mathcal{R}_w,1,B',d',m'},$$

where $B' \approx B$, $d = d' + \frac{b^{2k}-1}{b^2-1}$, $m = m' + k$ and b is the gadget base, $k = \lceil \log_b B \rceil$, which projects the problem onto a subring. To build this reduction, we make use of ring gadgets: given $f \in \mathcal{R}_{2w}$, the ring gadget computes a set of $a_i = b^{i-1} + xc_i(x^2) \in \mathcal{R}_{2w}$ such that $f + \sum_{i=1}^{k} a_i a_i^\star$ can be viewed as an element in the subring $\mathcal{R}_w$ and all c_i's are small. The resulting integral decomposition inherits the tower of rings structure and hence can be stored efficiently despite the output being wider by a factor of $O(\log w)$. Finally, this decomposition in the ring setting can be combined with the previous integer setting algorithm to yield an algorithm for solving $\mathsf{IGDP}_{\mathcal{R}_{2w},n,B,d,m}$.

Related work. While we are not aware of works on the Integral Gram Decomposition Problem, the rational version of this question arises as a natural mathematical and algorithmic question for the representation of quadratic forms. For example, Cassel [9] showed that a rational solution exists for $m = n + 3$; we are unaware of an efficient algorithm to find such a solution. Lenstra [24] proposed a polynomial time rational solution for $m = 4n$.

However, such rational solutions are not very satisfactory in our context, as the denominators can get as large as the determinant of the input lattice. In fact, if rational arithmetic with such large coefficient is deemed acceptable, then one could directly use an implementation of Klein-GPV [19,21] algorithm over the rationals.

In a concurrent work [10], Chen, Genise and Mukherjee proposed a notion of approximate trapdoor and adapted the Micciancio-Peikert trapdoor [25] to the approximate trapdoor setting. In the analysis of the preimage distribution, they used a similar linear transformation theorem for discrete Gaussians. Their

preimage sampling calls perturbation sampling as a "black-box" so that our technique is well compatible with [10].

Furthermore in [15], Ducas and Prest applied FFT techniques to improve the Klein-GPV algorithm in the ring setting. Similarly, Genise and Micciancio exploited the Schur complement and developed a discrete perturbation algorithm in [17]. Yet in practice these methods still resort to floating-point arithmetic.

Roadmap. We start in Sect. 2 with some preliminary material. Section 3 shows that rectangular Gram roots allow to sample according to the desired distribution. In Sect. 4, we introduce the Integral Gram Decomposition Problem and detail the algorithms to solve it. We provide a detailed comparison with Peikert's perturbation sampler in Sect. 5. Finally we propose a variant of our integral matrix decomposition geared to the ring setting in Sect. 6.

2 Preliminaries

2.1 Notations

We use log and ln to denote respectively the base 2 logarithm and the natural logarithm. Let $\epsilon > 0$ denote some very small number; we use the notational shortcut $\hat{\epsilon} = \epsilon + O(\epsilon^2)$. One can check that $\frac{1+\epsilon}{1-\epsilon} = 1 + 2\hat{\epsilon}$ and $\ln\left(\frac{1+\epsilon}{1-\epsilon}\right) = 2\hat{\epsilon}$.

For a distribution D over a countable set, we write $z \hookleftarrow D$ when the random variable z is sampled from D, and denote by $D(x)$ the probability of $z = x$. For a real-valued function f and a countable set S, we write $f(S) = \sum_{x \in S} f(x)$ assuming that this sum is absolutely convergent (which is always the case in this paper). Given two distributions D_1 and D_2 of common support E, the *max-log distance* between D_1 and D_2 is

$$\Delta_{\mathrm{ML}}(D_1, D_2) = \max_{x \in E} |\ln(D_1(x)) - \ln(D_2(x))|.$$

As shown in [28], it holds that $\Delta_{\mathrm{ML}}(D_1, D_2) \leq \Delta_{\mathrm{ML}}(D_1, D_3) + \Delta_{\mathrm{ML}}(D_2, D_3)$.

2.2 Linear Algebra

We use bold lower case letters to denote vectors, and bold upper case letters to denote matrices. By convention, vectors are in column form. For a matrix $\mathbf{A}$, we denote by $\mathbf{A}_{i,j}$ the element in the i-th row and j-th column, and by $\mathbf{A}_{i:j,k:l}$ the sub-block $(\mathbf{A}_{a,b})_{a \in \{i,\cdots,j\}, b \in \{k,\cdots,l\}}$. Let $\lfloor \mathbf{A} \rceil$ be the matrix obtained by rounding each entry of $\mathbf{A}$ to the nearest integer. Let $\mathbf{I}_n$ be the n-dimensional identity matrix.

Let $\mathbf{\Sigma} \in \mathbb{R}^{n \times n}$ be a symmetric matrix. We write $\mathbf{\Sigma} > 0$ when $\mathbf{\Sigma}$ is *positive definite*, i.e. $\mathbf{x}^t \mathbf{\Sigma} \mathbf{x} > 0$ for all non-zero $\mathbf{x} \in \mathbb{R}^n$. It is known that $\mathbf{\Sigma} > 0$ if and only if $\mathbf{\Sigma}^{-1} > 0$. We also write $\mathbf{\Sigma}_1 > \mathbf{\Sigma}_2$ when $\mathbf{\Sigma}_1 - \mathbf{\Sigma}_2 > 0$. It holds that $\mathbf{\Sigma}_1 > \mathbf{\Sigma}_2 > 0$ if and only if $\mathbf{\Sigma}_2^{-1} > \mathbf{\Sigma}_1^{-1} > 0$. Similarly, we write $\mathbf{\Sigma}_1 \geq \mathbf{\Sigma}_2$ or $\mathbf{\Sigma}_1 - \mathbf{\Sigma}_2 \geq 0$ to state that $\mathbf{\Sigma}_1 - \mathbf{\Sigma}_2$ is positive semi-definite. If $\mathbf{\Sigma} = \mathbf{A}\mathbf{A}^t$, we call

$\mathbf{A}$ a *Gram root* of $\mathbf{\Sigma}$. In particular, if a Gram root $\mathbf{A}$ is a square and invertible matrix, we call $\mathbf{A}$ a *square Gram root*[2]. When the context permits it, we denote $\sqrt{\mathbf{\Sigma}}$ for any square Gram root of $\mathbf{\Sigma}$.

For a positive definite matrix $\mathbf{\Sigma}$, let $e_1(\mathbf{\Sigma})$ be the largest eigenvalue of $\mathbf{\Sigma}$, then $e_1(\mathbf{\Sigma}) > 0$. Let $\mathbf{\Sigma}_1, \mathbf{\Sigma}_2$ be positive definite matrices and $\mathbf{\Sigma} = \mathbf{\Sigma}_1 + \mathbf{\Sigma}_2$, then $\mathbf{\Sigma} > 0$ and $e_1(\mathbf{\Sigma}) \leq e_1(\mathbf{\Sigma}_1) + e_1(\mathbf{\Sigma}_2)$. We recall the spectral norm $\|\mathbf{A}\|_2 = \max_{\mathbf{x}\neq\mathbf{0}} \frac{\|\mathbf{Ax}\|}{\|\mathbf{x}\|} = \sqrt{e_1(\mathbf{A}^t\mathbf{A})}$ and the Frobenius norm $\|\mathbf{A}\|_F = \sqrt{\sum_{i,j} \mathbf{A}_{i,j}^2}$. It is known that $\|\mathbf{A}^t\|_2 = \|\mathbf{A}\|_2$, $\|\mathbf{AB}\|_2 \leq \|\mathbf{A}\|_2\|\mathbf{B}\|_2$ and $\|\mathbf{A}\|_2 \leq \|\mathbf{A}\|_F$. We also write $\|\mathbf{A}\|_{\max} = \max_{i,j} |\mathbf{A}_{i,j}|$ and $\|\mathbf{A}\|_{\text{col}} = \max_j \sqrt{\sum_i \mathbf{A}_{i,j}^2}$.

2.3 Lattices

A lattice is a discrete additive subgroup of $\mathbb{R}^m$, and is the set of all integer linear combinations of linearly independent vectors $\mathbf{b}_1, \cdots, \mathbf{b}_n \in \mathbb{R}^m$. We call $\mathbf{B} = \begin{pmatrix} \mathbf{b}_1 & \cdots & \mathbf{b}_n \end{pmatrix}$ a basis and n the dimension of the lattice. If $n = m$, we call the lattice *full-rank*. We denote by $\mathcal{L}(\mathbf{B})$ the lattice generated by the basis $\mathbf{B}$. Let $\widehat{\mathcal{L}} = \{\mathbf{u} \in \text{span}(\mathcal{L}) \mid \forall \mathbf{v} \in \mathcal{L}, \langle \mathbf{u}, \mathbf{v} \rangle \in \mathbb{Z}\}$ be the dual lattice of $\mathcal{L}$. For $k \leq n$, the k-th minimum $\lambda_k(\mathcal{L})$ is the smallest value $r \in \mathbb{R}$ such that there are at least k linearly independent vectors in $\mathcal{L}$ whose lengths are not greater than r.

Given $\mathbf{A} \in \mathbb{Z}^{n\times m}$ with $m \geq n$, we denote the *orthogonal lattice*[3] defined by $\mathbf{A}$ by $\mathit{\Lambda}^\perp(\mathbf{A}) = \{\mathbf{v} \in \mathbb{Z}^m \mid \mathbf{Av} = \mathbf{0}\}$. When the rank of $\mathbf{A}$ is n, the dimension of $\mathit{\Lambda}^\perp(\mathbf{A})$ is $(m - n)$.

2.4 Gaussians

Let $\rho_{\mathbf{R},\mathbf{c}}(\mathbf{x}) = \exp\left(-\pi(\mathbf{x}-\mathbf{c})^t\mathbf{R}^{-t}\mathbf{R}^{-1}(\mathbf{x}-\mathbf{c})\right)$ be the n-dimensional Gaussian weight with center $\mathbf{c} \in \mathbb{R}^n$ and (scaled)[4] covariance matrix $\mathbf{\Sigma} = \mathbf{RR}^t$. Because $\rho_{\mathbf{R},\mathbf{c}}(\mathbf{x}) = \exp\left(-\pi(\mathbf{x}-\mathbf{c})^t\mathbf{\Sigma}^{-1}(\mathbf{x}-\mathbf{c})\right)$ is exactly determined by $\mathbf{\Sigma}$, we also write $\rho_{\mathbf{R},\mathbf{c}}$ as $\rho_{\sqrt{\mathbf{\Sigma}},\mathbf{c}}$. When $\mathbf{c} = \mathbf{0}$, the Gaussian function is written as $\rho_{\mathbf{R}}$ or $\rho_{\sqrt{\mathbf{\Sigma}}}$ and is called *centered*. When $\mathbf{\Sigma} = s^2\mathbf{I}_n$, we write the subscript $\sqrt{\mathbf{\Sigma}}$ as s directly, and call s the *width*.

The *discrete Gaussian distribution* over a lattice $\mathcal{L}$ with center $\mathbf{c}$ and covariance matrix $\mathbf{\Sigma}$ is defined by the probability function $D_{\mathcal{L},\sqrt{\mathbf{\Sigma}},\mathbf{c}}(\mathbf{x}) = \frac{\rho_{\sqrt{\mathbf{\Sigma}},\mathbf{c}}(\mathbf{x})}{\rho_{\sqrt{\mathbf{\Sigma}},\mathbf{c}}(\mathcal{L})}$ for any $\mathbf{x} \in \mathcal{L}$. We recall some notions related to the *smoothing parameter*.

Definition 1 ([27], Definition 3.1). *Given a lattice $\mathcal{L}$ and $\epsilon > 0$, the ϵ-smoothing parameter of $\mathcal{L}$ is* $\eta_\epsilon(\mathcal{L}) = \min\left\{s \mid \rho_{1/s}\left(\widehat{\mathcal{L}}\right) \leq 1 + \epsilon\right\}$.

[2] When $n \geq 2$, any $\mathbf{\Sigma} > 0$ has infinitely many square Gram roots.

[3] Take note that we are here not considering the "q-ary orthogonal lattice" $\mathit{\Lambda}_q^\perp(\mathbf{A}) = \{\mathbf{v} \in \mathbb{Z}^m \mid \mathbf{Av} = \mathbf{0} \bmod q\}$.

[4] The scaling factor is 2π and we omit it in this paper for convenience.

Definition 2 ([32], Definition 2.3). *Given a full-rank lattice $\mathcal{L}$, $\epsilon > 0$ and a positive definite matrix $\mathbf{\Sigma}$, we write $\sqrt{\mathbf{\Sigma}} \geq \eta_\epsilon(\mathcal{L})$ if $\eta_\epsilon\left(\sqrt{\mathbf{\Sigma}}^{-1} \cdot \mathcal{L}\right) \leq 1$ i.e. $\rho_{\sqrt{\mathbf{\Sigma}^{-1}}}\left(\widehat{\mathcal{L}}\right) \leq 1 + \epsilon$.*

We define $\eta_\epsilon^{\leq}(\mathbb{Z}^n) = \sqrt{\frac{\ln(2n(1+1/\epsilon))}{\pi}}$. We will use the following results later.

Proposition 1. *Given a lattice $\mathcal{L}$ and $\epsilon > 0$, then $\eta_\epsilon(r\mathcal{L}) = r \cdot \eta_\epsilon(\mathcal{L})$ for arbitrary $r > 0$.*

Proposition 2. *Let $\mathbf{\Sigma}_1 \geq \mathbf{\Sigma}_2 > 0$ be two positive definite matrices. Let $\mathcal{L}$ be a full-rank lattice and $\epsilon \in (0, 1)$. If $\sqrt{\mathbf{\Sigma}_2} \geq \eta_\epsilon(\mathcal{L})$, then $\sqrt{\mathbf{\Sigma}_1} \geq \eta_\epsilon(\mathcal{L})$.*

Proof. Notice that $\rho_{\sqrt{\mathbf{\Sigma}_1^{-1}}}(\mathbf{x}) = \exp(-\pi \mathbf{x}^t \mathbf{\Sigma}_1 \mathbf{x}) \leq \exp(-\pi \mathbf{x}^t \mathbf{\Sigma}_2 \mathbf{x}) = \rho_{\sqrt{\mathbf{\Sigma}_2^{-1}}}(\mathbf{x})$, hence $\rho_{\sqrt{\mathbf{\Sigma}_1^{-1}}}\left(\widehat{\mathcal{L}}\right) \leq \rho_{\sqrt{\mathbf{\Sigma}_2^{-1}}}\left(\widehat{\mathcal{L}}\right)$. By Definition 2, we complete the proof. □

Lemma 1 ([27], Lemma 3.3). *Let $\mathcal{L}$ be an n-dimensional lattice and $\epsilon \in (0, 1)$. Then $\eta_\epsilon(\mathcal{L}) \leq \eta_\epsilon^{\leq}(\mathbb{Z}^n) \cdot \lambda_n(\mathcal{L})$. In particular, for any $\omega(\sqrt{\log n})$ function, there is a negligible ϵ such that $\eta_\epsilon(\mathbb{Z}^n) \leq \omega(\sqrt{\log n})$.*

Lemma 2 ([27], implicit in Lemma 4.4). *Let $\mathcal{L}$ be a lattice and $\epsilon \in (0, 1)$. If $r \geq \eta_\epsilon(\mathcal{L})$, then $\rho_r(\mathbf{c} + \mathcal{L}) \in [\frac{1-\epsilon}{1+\epsilon}, 1]\rho_r(\mathcal{L})$ for any $\mathbf{c} \in \mathrm{span}(\mathcal{L})$.*

We recall the convolution theorem with respect to discrete Gaussians that was introduced in [32].

Theorem 1 (Adapted from Theorem 3.1 [32]). *Let $\mathbf{\Sigma}_1, \mathbf{\Sigma}_2 \in \mathbb{R}^{n \times n}$ be positive definite matrices. Let $\mathbf{\Sigma} = \mathbf{\Sigma}_1 + \mathbf{\Sigma}_2$ and let $\mathbf{\Sigma}_3 \in \mathbb{R}^{n \times n}$ be such that $\mathbf{\Sigma}_3^{-1} = \mathbf{\Sigma}_1^{-1} + \mathbf{\Sigma}_2^{-1}$. Let $\mathcal{L}_1, \mathcal{L}_2$ be two full-rank lattices in $\mathbb{R}^n$ such that $\sqrt{\mathbf{\Sigma}_1} \geq \eta_\epsilon(\mathcal{L}_1)$ and $\sqrt{\mathbf{\Sigma}_3} \geq \eta_\epsilon(\mathcal{L}_2)$ for $\epsilon \in (0, 1/2)$. Let $\mathbf{c}_1, \mathbf{c}_2 \in \mathbb{R}^n$. Then the distribution of $\mathbf{x}_1 \hookleftarrow D_{\mathcal{L}_1, \sqrt{\mathbf{\Sigma}_1}, \mathbf{x}_2 - \mathbf{c}_2 + \mathbf{c}_1}$ where $\mathbf{x}_2 \hookleftarrow D_{\mathcal{L}_2, \sqrt{\mathbf{\Sigma}_2}, \mathbf{c}_2}$ is within max-log distance $4\hat{\epsilon}$ of $D_{\mathcal{L}_1, \sqrt{\mathbf{\Sigma}}, \mathbf{c}_1}$.*

2.5 Integral Decompositions

Lagrange's four-square theorem states that every natural number can be represented as the sum of four integer squares. An efficient algorithm to find such a decomposition was given by Rabin and Shallit [35].

Theorem 2 (Rabin-Shallit algorithm [35]). *There is a randomized algorithm for expressing $N \in \mathbb{N}$ as a sum of four squares which requires an expected number of $O(\log^2 N \log\log N)$ operations with integers smaller than N.*

Another important integral decomposition for our work is the b-ary decomposition, more conveniently formalized with the *gadget vector* $\mathbf{g} = (1, b, \cdots, b^{k-1})^t$ in [25], hence called *gadget decomposition*. It says that for any $n \in (-b^k, b^k) \cap \mathbb{Z}$, there exists a vector $\mathbf{c} \in \mathbb{Z}^k$ such that $\langle \mathbf{c}, \mathbf{g} \rangle = n$ and $\|\mathbf{c}\|_\infty < b$. The cost of such a decomposition is dominated by $O(k)$ Euclidean divisions by b that are particularly efficient in practice when b is a power of 2. Note that $\|\mathbf{g}\|^2 = (b^{2k} - 1)/(b^2 - 1)$.

3 Gaussian Sampling with an Integral Gram Root

In Peikert's sampler [32], one samples perturbation vectors from a Gaussian with certain covariance, say $D_{\mathbb{Z}^n, r\sqrt{\mathbf{\Sigma}}}$ during the offline phase. Among existing perturbation samplers [17,25,32] this requires floating-point arithmetic in the linear algebraic steps.

To avoid FPA, our starting point is the following observation: given an integral Gram root of $\mathbf{\Sigma}-\mathbf{I}_n$, one can sample from $D_{\mathbb{Z}^n, r\sqrt{\mathbf{\Sigma}}}$ without resorting to FPA and in a quite simple manner. The main result of this section is the following theorem.

Theorem 3 (Sampling theorem). *Let $\mathbf{\Sigma} \in \mathbb{Z}^{n\times n}$ such that $\mathbf{\Sigma} - \mathbf{I}_n \geq \mathbf{I}_n$. Given $\mathbf{A} \in \mathbb{Z}^{n\times(n+m)}$ such that $\mathbf{A}\mathbf{A}^t = \mathbf{\Sigma}-\mathbf{I}_n$, $\mathbf{A}\cdot\mathbb{Z}^{n+m} = \mathbb{Z}^n$ and $\lambda_m(\Lambda^\perp(\mathbf{A})) \leq L$, let $\widetilde{D}_{\mathbf{A}}(L', r)$ denote the distribution of $D_{\mathbb{Z}^n, r, \frac{1}{L'}\cdot\mathbf{c}}$ where $\mathbf{c} = \mathbf{A}\mathbf{x}$ with $\mathbf{x} \hookleftarrow D_{\mathbb{Z}^{n+m}, L'r}$. For $\epsilon \in (0, 1/2)$, $r \geq \eta_\epsilon(\mathbb{Z}^n)$ and $L' \geq \max\{\sqrt{2}, (L/r)\cdot\eta^{\leq}_\epsilon(\mathbb{Z}^m)\}$, then*

$$\Delta_{\mathrm{ML}}\left(\widetilde{D}_{\mathbf{A}}(L', r), D_{\mathbb{Z}^n, r\sqrt{\mathbf{\Sigma}}}\right) \leq 8\hat{\epsilon}.$$

We give an algorithmic description in Algorithm 1.

Algorithm 1. New perturbation sampling algorithm $\mathsf{NewPert}(r, \mathbf{\Sigma})$

Input: a covariance matrix $\mathbf{\Sigma} \in \mathbb{Z}^{n\times n}$ and some $r \geq \eta_\epsilon(\mathbb{Z}^n)$.
Output: a sample $\mathbf{x}$ from a distribution within max-log distance $8\hat{\epsilon}$ of $D_{\mathbb{Z}^n, r\sqrt{\mathbf{\Sigma}}}$.

Precomputation:
1: compute $\mathbf{A} \in \mathbb{Z}^{n\times(n+m)}$ such that $\mathbf{A}\mathbf{A}^t = \mathbf{\Sigma} - \mathbf{I}_n$, $\mathbf{A}\cdot\mathbb{Z}^{n+m} = \mathbb{Z}^n$ and $\lambda_m(\Lambda^\perp(\mathbf{A})) \leq L$ (see Sect. 4 for details)
Sampling:
2: sample $\mathbf{x} \hookleftarrow D_{\mathbb{Z}^{n+m}, Lr}$ by base sampler $D_{\mathbb{Z}, Lr}$
3: $\mathbf{c} \leftarrow \mathbf{A}\mathbf{x}$
4: sample $\mathbf{y} \hookleftarrow D_{\mathbb{Z}^n, r, \frac{1}{L}\cdot\mathbf{c}}$ by base samplers $D_{\mathbb{Z}, r, \frac{c}{L}}$ with $c \in \{0, 1, \cdots, L-1\}$
5: return $\mathbf{y}$

To prove Theorem 3, we need the following linear transformation lemma for discrete Gaussians.

Lemma 3 (Linear Transformation Lemma). *Let $\mathbf{A} \in \mathbb{Z}^{n\times m}$ such that $\mathbf{A}\cdot\mathbb{Z}^m = \mathbb{Z}^n$. Let $\mathbf{\Sigma} = \mathbf{A}\mathbf{A}^t$. For $\epsilon \in (0, 1/2)$, if $r \geq \eta_\epsilon(\Lambda^\perp(\mathbf{A}))$, then the max-log distance between the distribution of $\mathbf{y} = \mathbf{A}\mathbf{x}$ where $\mathbf{x} \hookleftarrow D_{\mathbb{Z}^m, r}$ and $D_{\mathbb{Z}^n, r\sqrt{\mathbf{\Sigma}}}$ is at most $4\hat{\epsilon}$.*

Remark 1. Lemma 3 was used implicitly in [6,26]. Its proof is given in the full version [16]. Again, a general linear transformation theorem is stated in [10].

Remark 2. Lemma 3 implies similar bounds for other metrics, such as the Kullback-Leibler divergence [33] and the Rényi divergence [3,34].

Proof of Theorem 3. By Lemmata 3 and 1, the max-log distance between the distribution of $\mathbf{c} = \mathbf{Ax}$ and $D_{\mathbb{Z}^n, L'r\sqrt{\mathbf{\Sigma}-\mathbf{I}_n}}$ is at most $4\hat{\epsilon}$. By scaling, we have that the max-log distance between the distribution of $\frac{1}{L'} \cdot \mathbf{c}$ and $D_{\frac{1}{L'}\cdot\mathbb{Z}^n, r\sqrt{\mathbf{\Sigma}-\mathbf{I}_n}}$ is still at most $4\hat{\epsilon}$. It can be verified that

$$r\sqrt{\left((\mathbf{\Sigma}-\mathbf{I}_n)^{-1}+\mathbf{I}_n^{-1}\right)^{-1}} \geq \eta_\epsilon\left(\frac{1}{L'}\cdot\mathbb{Z}^n\right).$$

Combining Theorem 1, the proof follows. □

3.1 Reducing $\lambda_m(\Lambda^\perp(\mathbf{A}))$

As shown in Theorem 3, with an integral Gram root, the sampling of $D_{\mathbb{Z}^n, r\sqrt{\mathbf{\Sigma}}}$ is converted into two kinds of base samplings: $D_{\mathbb{Z}, L'r}$ and $D_{\mathbb{Z}, r, c}$ with $c \in \frac{1}{L'}\cdot\mathbb{Z}$. One sometimes may prefer to work with small L' whose size is mainly determined by $\lambda_m(\Lambda^\perp(\mathbf{A}))$. The following lemma suggests that given a matrix $\mathbf{A}$, one can construct an orthogonal lattice of relatively small successive minima by padding $\mathbf{A}$ with some gadget matrices $\left(\mathbf{I}_n\ b\mathbf{I}_n\ \cdots\ b^{k-1}\mathbf{I}_n\right)$.

Lemma 4. *Let $\mathbf{A} \in \mathbb{Z}^{n\times m}$ with $\|\mathbf{A}\|_2 \leq B$. For $b, k \in \mathbb{N}$ such that $b^k > B$, let $\mathbf{A}' = \left(\mathbf{I}_n\ b\mathbf{I}_n\ \cdots\ b^{k-1}\mathbf{I}_n\ \mathbf{A}\right)$, then $\lambda_{m+(k-1)n}(\Lambda^\perp(\mathbf{A}')) \leq \sqrt{nk(b-1)^2+1}$.*

The proof of Lemma 4 is given in the full version [16].

Remark 3. Lemma 4 provides a solution to reduce L' at the cost of more base samplings and some overhead on the final Gaussian width. In practice, it is optional to pad gadget matrices considering the tradeoff. In later discussions, we shall omit this trick and just focus on $\lambda_m(\Lambda^\perp(\mathbf{A}))$.

4 Integral Gram Decompositions

In Sect. 3, we have explicated how to sample perturbation vectors using no FPA with an integral Gram root. The computation of such an integral Gram root is developed in this section. Let us first formally define the *Integral Gram Decomposition Problem.*

Definition 3 ($\mathsf{IGDP}_{n,B,d,m}$)**.** *Let $n, B, d, m \in \mathbb{N}$. The Integral Gram Decomposition Problem, denoted by $\mathsf{IGDP}_{n,B,d,m}$, is defined as follows: given an integral symmetric matrix $\mathbf{\Sigma} \in \mathbb{Z}^{n\times n}$ with $\|\mathbf{\Sigma}\|_2 \leq B$, find an integral matrix $\mathbf{A} \in \mathbb{Z}^{n\times m}$ such that $\mathbf{AA}^t = d\mathbf{I}_n - \mathbf{\Sigma}$.*

Our final goal is to solve $\mathsf{IGDP}_{n,B,d,m}$ with fixed (n, B) while keeping $d = (1+o(1))B$ and m relatively small.

Our first approach (Sect. 4.1) only allows a decomposition of sufficiently diagonally dominant matrices, i.e. $d \gg B$, which implies a large overhead on the final width of the Gaussian. Fortunately, when the parameter B is somewhat large, say $\omega(n^4)$, this can be fixed by first resorting to some integral approximations of Cholesky Gram roots and then working on the left-over matrix of small norm. We call this procedure eigenvalue reduction and describe it in Sect. 4.2. Finally, we combine these two algorithms and give several example instances in Sect. 4.3.

4.1 Decomposition for Diagonally Dominant Matrices

We present an algorithm to compute an integral Gram root of $\mathbf{\Sigma}' = d\mathbf{I}_n - \mathbf{\Sigma}$ for a relatively large d. It is formally described in Algorithm 2.

The algorithm proceeds by induction, reducing $\mathsf{IGDP}_{n,B,d,m}$ to $\mathsf{IGDP}_{n-1,B',d,m'}$ where B' and m are slightly larger than B and m' respectively. To do so, one first constructs $\mathbf{T} \in \mathbb{Z}^{n\times k}$ such that $\mathbf{TT}^t$ and $\mathbf{\Sigma}'$ have the same first row and column, and then proceeds iteratively over $(\mathbf{\Sigma}' - \mathbf{TT}^t)_{2:n,2:n}$. In the construction of $\mathbf{T}$, to clear out the off-diagonal elements, we make use of a gadget decomposition $\langle \mathbf{c}_i, \mathbf{g}\rangle = \mathbf{\Sigma}'_{1,i}$ $(i > 1)$. The remaining diagonal element, namely $\mathbf{\Sigma}'_{1,1} - \|\mathbf{g}\|^2$, is then handled by the 4-square theorem.

To ensure the inductive construction goes through, Algorithm 2 requires a certain strongly positive definiteness condition. We need that $d - \mathbf{\Sigma}_{i,i} \geq \|\mathbf{g}\|^2$, but we also need to account for the perturbation $\mathbf{TT}^t$ subtracted from $\mathbf{\Sigma}'$ during the induction. The correctness and efficiency of this algorithm is given in Lemma 5.

Remark 4. For tighter parameters, we consider $\|\mathbf{\Sigma}\|_{\max}$ instead of direct $\|\mathbf{\Sigma}\|_2$ in Algorithm 2, which does not affect the main result, i.e. Corollary 1.

Lemma 5. *Algorithm 2 is correct. More precisely, let $\mathbf{\Sigma} \in \mathbb{Z}^{n\times n}$ be symmetric and $d, b, k \in \mathbb{Z}$ such that $b^k \geq \|\mathbf{\Sigma}\|_{\max} + k(n-1)b^2$ and $d \geq \frac{b^{2k}-1}{b^2-1} + b^k$. Then $\mathsf{DiagDomIGD}(\mathbf{\Sigma}, d, b, k)$ outputs $\mathbf{A} \in \mathbb{Z}^{n\times n(k+4)}$ such that $\mathbf{AA}^t = d\mathbf{I}_n - \mathbf{\Sigma}$.*

Moreover, $\mathsf{DiagDomIGD}(\mathbf{\Sigma}, d, b, k)$ performs $O(kn^3 + n\log^2 d \log\log d)$ arithmetic operations on integers of bitsize $O(\log d)$.

Proof. There are n calls to the Rabin-Shallit algorithm in Algorithm 2, and all input integers are at most $2d$. Thus the total cost of the Rabin-Shallit algorithm is $O(n\log^2 d \log\log d)$ operations on integers of bitsize $O(\log d)$. There are also $\frac{n(n-1)}{2}$ gadget decompositions, and the total cost is $O(n^2k)$ operations on integers of bitsize at most $k\log b \leq \log d$. For matrix multiplication, we follow the textbook algorithm, thus the total cost is $O(n^3k)$ operations on integers of bitsize at most $O(\log d)$. This yields the overall running time complexity.

We now prove the correctness. Since $d - \mathbf{\Sigma}_{1,1} \geq d - \|\mathbf{\Sigma}\|_{\max} \geq \|\mathbf{g}\|^2$, the existence of a 4-square decomposition $\mathbf{x}$ is ensured. For $\mathbf{\Sigma}_{1,j}$ with $j > 1$, we have $|\mathbf{\Sigma}_{1,j}| \leq \|\mathbf{\Sigma}\|_{\max} < b^k$, which implies the existence of $\mathbf{c}_j$ and $\|\mathbf{c}_j\|_\infty < b$. Then it can be verified that

$$d\mathbf{I}_n - \mathbf{\Sigma} - \mathbf{TT}^t = \begin{pmatrix} 0 & \mathbf{0}^t \\ \mathbf{0} & \mathbf{\Pi}' \end{pmatrix}$$

where $\mathbf{\Pi}' = d\mathbf{I}_{n-1} - \mathbf{\Pi} \in \mathbb{Z}^{(n-1)\times(n-1)}$ and $\mathbf{\Pi} = \mathbf{\Sigma}_{2:n,2:n} + \mathbf{\Xi}$ with $\mathbf{\Xi} = \mathbf{CC}^t$. Notice that $\|\mathbf{c}_j\| \leq b\sqrt{k}$, hence $\|\mathbf{\Pi}\|_{\max} \leq \|\mathbf{\Sigma}\|_{\max} + kb^2$. Further we have that

$$b^k \geq \|\mathbf{\Pi}\|_{\max} + k(n-2)b^2 \quad \text{and} \quad d \geq \frac{b^{2k}-1}{b^2-1} + b^k.$$

So far, all parameter conditions indeed hold for d and $\mathbf{\Pi}$ correspondingly. Therefore, the induction goes through and Algorithm 2 is correct. □

Algorithm 2. Integral matrix decomposition for a strongly diagonally dominant matrix $\mathsf{DiagDomIGD}(\mathbf{\Sigma}, d, b, k)$

Input: a symmetric matrix $\mathbf{\Sigma} \in \mathbb{Z}^{n\times n}$,
two integers $b, k \geq 2$ such that $b^k \geq \|\mathbf{\Sigma}\|_{\max} + k(n-1)b^2$,
an integer d such that $d \geq \frac{b^{2k}-1}{b^2-1} + b^k$.

Output: $\mathbf{A} = \begin{pmatrix}\mathbf{L}_1 \cdots \mathbf{L}_k\ \mathbf{D}_1 \cdots \mathbf{D}_4\end{pmatrix} \in \mathbb{Z}^{n\times n(k+4)}$ such that $\mathbf{A}\mathbf{A}^t = d\mathbf{I}_n - \mathbf{\Sigma}$ where $\mathbf{L}_i \in \mathbb{Z}^{n\times n}$ is a lower-triangular matrix whose diagonal elements are b^{i-1} and $\mathbf{D}_i \in \mathbb{Z}^{n\times n}$ is a diagonal matrix.

$\mathbf{g} \leftarrow (1, b, \cdots, b^{k-1})^t$
calculate $\mathbf{x} = (x_1, x_2, x_3, x_4)^t \in \mathbb{Z}^4$ such that $\|\mathbf{x}\|^2 = d - \mathbf{\Sigma}_{1,1} - \|\mathbf{g}\|^2$ using Rabin-Shallit algorithm (Theorem 2)
if $n = 1$ **then**
  return $(\mathbf{g}^t, \mathbf{x}^t)$
end if
for $j = 2, \cdots, n$ **do**
  calculate $\mathbf{c}_j \in \mathbb{Z}^k$ such that $\langle \mathbf{c}_j, \mathbf{g}\rangle = -\mathbf{\Sigma}_{1,j}$ by gadget decomposition
end for
$\mathbf{C} \leftarrow \begin{pmatrix}\mathbf{c}_2 \cdots \mathbf{c}_n\end{pmatrix}^t \in \mathbb{Z}^{(n-1)\times k}$, $\mathbf{T} \leftarrow \begin{pmatrix}\mathbf{g}^t\ \mathbf{x}^t \\ \mathbf{C}\end{pmatrix} \in \mathbb{Z}^{n\times(4+k)}$
$\mathbf{\Pi} \leftarrow \left(\mathbf{\Sigma} + \mathbf{T}\mathbf{T}^t\right)_{2:n,2:n}$
$\begin{pmatrix}\mathbf{L}'_1 \cdots \mathbf{L}'_k\ \mathbf{D}'_1 \cdots \mathbf{D}'_4\end{pmatrix} \leftarrow \mathsf{DiagDomIGD}(\mathbf{\Pi}, d, b, k)$ {Recursive call}
$\begin{pmatrix}\mathbf{v}'_1 \cdots \mathbf{v}'_k\end{pmatrix} \leftarrow \mathbf{C}$
$\mathbf{L}_i \leftarrow \begin{pmatrix} b^{i-1} & \\ \mathbf{v}'_i & \mathbf{L}'_i\end{pmatrix} \in \mathbb{Z}^{n\times n}$ for $i = 1, \cdots, k$
$\mathbf{D}_i \leftarrow \begin{pmatrix} x_i & \\ & \mathbf{D}'_i\end{pmatrix} \in \mathbb{Z}^{n\times n}$ for $i = 1, \cdots, 4$
return $\mathbf{A} = \begin{pmatrix}\mathbf{L}_1 \cdots \mathbf{L}_k\ \mathbf{D}_1 \cdots \mathbf{D}_4\end{pmatrix}$

Notice that $\|\mathbf{\Sigma}\|_{\max} \leq \|\mathbf{\Sigma}\|_2$, we immediately get the following result.

Corollary 1. *Let $n, B, d, b, k \in \mathbb{N}$ such that $b^k \geq B + k(n-1)b^2$ and $d \geq \frac{b^{2k}-1}{b^2-1} + b^k$. Then there exists a solution to $\mathsf{IGDP}_{n,B,d,n(k+4)}$ and it can be calculated by Algorithm 2.*

To use such a decomposition for perturbation sampling, we also need to control $\lambda_{m-n}(\Lambda^\perp(\mathbf{A}))$. Lemma 6 shows that this can easily be done by padding the output $\mathbf{A}$ with an identity matrix $\mathbf{I}_n$.

Lemma 6. *Let $\mathbf{A} = \mathsf{DiagDomIGD}(\mathbf{\Sigma}, d-1, b, k) \in \mathbb{Z}^{n\times m}$ where $m = n(k+4)$. Let $\mathbf{A}' = \begin{pmatrix}\mathbf{I}_n\ \mathbf{A}\end{pmatrix}$, then $\mathbf{A}' \cdot \mathbb{Z}^{n+m} = \mathbb{Z}^n$, the dimension of $\Lambda^\perp(\mathbf{A}')$ is m, and*

$$\lambda_m(\Lambda^\perp(\mathbf{A}')) \leq \max\left\{b^2\sqrt{n}, \sqrt{d - \frac{b^{2k}-1}{b^2-1} + \|\mathbf{\Sigma}\|_{\max}}\right\}.$$

Proof. Let $\mathbf{S} = \begin{pmatrix} 1 & -b & & \\ & 1 & -b & \\ & & \ddots & -b \\ & & & 1\end{pmatrix} \in \mathbb{Z}^{(k-1)\times(k-1)}$. We define $\mathbf{D} = \begin{pmatrix}\mathbf{D}_1 \cdots \mathbf{D}_4\end{pmatrix} \in \mathbb{Z}^{n\times 4n}$, $\mathbf{L} = \begin{pmatrix}\mathbf{L}_1 \cdots \mathbf{L}_k\end{pmatrix} \in \mathbb{Z}^{n\times kn}$ such that $\mathbf{A} = \begin{pmatrix}\mathbf{L}\ \mathbf{D}\end{pmatrix}$. We also define

$\overline{\mathbf{L}} = \mathbf{L} \cdot (\mathbf{S} \otimes \mathbf{I}_n) = \left(\mathbf{L}_1\ \mathbf{L}_2 - b\mathbf{L}_1 \cdots \mathbf{L}_k - b\mathbf{L}_{k-1}\right)$, then $\|\overline{\mathbf{L}}\|_{\max} < b^2$ and the diagonal elements of $\mathbf{L}_i - b\mathbf{L}_{i-1}$ are 0. Let

$$\mathbf{P} = \begin{pmatrix} \mathbf{A} \\ -\mathbf{I}_m \end{pmatrix} \cdot \begin{pmatrix} \mathbf{S} \otimes \mathbf{I}_n & \\ & \mathbf{I}_{4n} \end{pmatrix} = \begin{pmatrix} \overline{\mathbf{L}} & \mathbf{D} \\ -\mathbf{S} \otimes \mathbf{I}_n & \\ & -\mathbf{I}_{4n} \end{pmatrix},$$

then $\mathbf{P}$ contains m linearly independent vectors of $\Lambda^{\perp}(\mathbf{A}')$. A straightforward computation yields that

$$\|\mathbf{P}\|_{\text{col}} \leq \max\left\{ b^2\sqrt{n}, \sqrt{d - \frac{b^{2k}-1}{b^2-1} + \|\mathbf{\Sigma}\|_{\max}} \right\},$$

which implies the conclusion immediately. □

4.2 Eigenvalue Reduction

The parameter requirements of Corollary 1 are rather demanding. Indeed, the minimal d is at least $B + b^{2k-2} + k(n-1)b^2 > 2B\sqrt{k(n-1)}$, which results in costly overhead on the final Gaussian width, and therefore on all the parameters of the cryptosystem. Yet we claim that for some large B, say $\omega(n^4)$, one can overcome this issue with the help of some integral approximations of Cholesky decompositions. The case of large B is of interest in advanced lattice based schemes [1,8,20]. Note that by scaling, this constraint on B can even be removed if one accepts to include a few ($O(\log n)$) rational bits in the Gram decomposition.

This technique essentially can be summarized as a reduction from $\mathsf{IGDP}_{n,B,d,m}$ to $\mathsf{IGDP}_{n,B',d-B,m-n}$ in which $B' \ll B$. One first splits $d\mathbf{I}_n - \mathbf{\Sigma}$ into two parts: $\mathbf{\Sigma}' = B \cdot \mathbf{I}_n - \mathbf{\Sigma}$ and $(d-B) \cdot \mathbf{I}_n$. Exploiting an integral approximation of Cholesky decomposition, one decomposes $\mathbf{\Sigma}'$ as a Gram matrix $\mathbf{L}\mathbf{L}^t$ and a small matrix $\mathbf{\Sigma}''$. Then it suffices to decompose $(d-B)\mathbf{I}_n + \mathbf{\Sigma}''$, which implies the reduction. As $B' \ll B$, the overhead introduced by $\mathsf{IGDP}_{n,B',d-B,m-n}$ can be negligible compared with the original B. The formal description is illustrated in Algorithm 3, and an upper bound of B' is shown in Lemma 7.

Algorithm 3. Eigenvalue reduction $\mathsf{EigenRed}(\mathbf{\Sigma}, B)$

Input: a symmetric matrix $\mathbf{\Sigma} \in \mathbb{Z}^{n \times n}$ with $\|\mathbf{\Sigma}\|_2 \leq B$.
Output: $(\mathbf{L}, \mathbf{\Pi})$ where $\mathbf{L} \in \mathbb{Z}^{n \times n}$, $\mathbf{\Pi} \in \mathbb{Z}^{n \times n}$ is symmetric and $B \cdot \mathbf{I}_n - \mathbf{\Sigma} = \mathbf{L}\mathbf{L}^t - \mathbf{\Pi}$.

1: $\mathbf{L} \leftarrow \left\lfloor \widetilde{\mathbf{L}} \right\rceil$ where $\widetilde{\mathbf{L}}$ is the Cholesky Gram root of $B \cdot \mathbf{I}_n - \mathbf{\Sigma}$
2: $\mathbf{\Pi} \leftarrow \mathbf{L}\mathbf{L}^t - (B \cdot \mathbf{I}_n - \mathbf{\Sigma})$
3: return $(\mathbf{L}, \mathbf{\Pi})$

For better description, we define a function $F_n : \mathbb{N} \to \mathbb{N}$ specified by n as

$$F_n(x) = \left\lceil \sqrt{n(n+1)x} + \frac{n(n+1)}{8} \right\rceil.$$

Lemma 7. *Let $\mathbf{\Sigma} \in \mathbb{Z}^{n \times n}$ be a symmetric matrix and $B \geq \|\mathbf{\Sigma}\|_2$. Let $(\mathbf{L}, \mathbf{\Pi}) = \mathsf{EigenRed}(\mathbf{\Sigma}, B)$, then $\|\mathbf{\Pi}\|_2 \leq F_n(B)$.*

Proof. Let $\mathbf{\Delta} = \mathbf{L} - \widetilde{\mathbf{L}}$, then $\|\mathbf{\Delta}\|_{\max} \leq \frac{1}{2}$ and $\mathbf{\Delta}$ is lower triangular. We have $\|\mathbf{\Delta}\|_2 \leq \|\mathbf{\Delta}\|_F \leq \frac{1}{2} \cdot \sqrt{\frac{n(n+1)}{2}}$ and $\|\widetilde{\mathbf{L}}\|_2 \leq \sqrt{2B}$, then

$$\|\mathbf{\Pi}\|_2 = \|\mathbf{\Delta}\widetilde{\mathbf{L}}^t + \widetilde{\mathbf{L}}\mathbf{\Delta}^t + \mathbf{\Delta}\mathbf{\Delta}^t\|_2 \leq 2\|\widetilde{\mathbf{L}}\|_2\|\mathbf{\Delta}\|_2 + \|\mathbf{\Delta}\|_2^2 \leq F_n(B).$$

We complete the proof. □

Corollary 2. *Let $n, B, d, m \in \mathbb{N}$. There is a deterministic reduction from $\mathsf{IGDP}_{n,B,d,m}$ to $\mathsf{IGDP}_{n,F_n(B),d-B,m-n}$ whose cost is dominated by one call to Cholesky decomposition on some positive semi-definite matrix $\mathbf{\Sigma} \in \mathbb{Z}^{n \times n}$ with $\|\mathbf{\Sigma}\|_2 \leq 2B$.*

Remark 5. One may fear the re-introduction of Cholesky within our algorithm, however we argue that it is in this context much less of an issue:

- costly FPA computation may still be needed, but they are now confined to a one-time pre-computation, rather than a many-time off-line phase,
- the weak determinism of this FPA computation can be mitigated by running pre-computation as part of the trapdoor generation algorithm, and providing the pre-computed integral Gram decomposition as part of the secret key,
- the eigenvalue reduction algorithm can tolerate a rather crude approximation of Cholesky without leading to a hard to detect statistical leak. At worse, insufficient precision will simply fail to solve the IGDP instance at hand. That is, only completeness is at stake, not security, and one may tolerate rare failures.
- one may also completely avoid FPA by resorting to potentially less efficient though more convenient square root approximation algorithm. In particular, we note that the Taylor series of $\sqrt{1-x}$ involves only power-of-2 denominators: one can design a "strongly deterministic" algorithm.

4.3 Putting Them Together

So far, we have introduced two algorithmic tools for $\mathsf{IGDP}_{n,B,d,m}$: the integral decomposition for diagonally dominant matrices $\mathsf{DiagDomIGD}(\mathbf{\Sigma}, d, b, k)$ and the eigenvalue reduction $\mathsf{EigenRed}(\mathbf{\Sigma}, B)$. They can be combined as follows: one first applies the eigenvalue reduction iteratively and then decomposes the final left-over matrix. We summarize this as $\mathsf{IGD}(\mathbf{\Sigma}, d, B, t, b, k)$ in Algorithm 4, and prove its correctness in Lemma 8.

We follow the notation F_n given in Sect. 4.2, and also define its iterated function $F_n^{(i)} : \mathbb{N} \to \mathbb{N}$ for $i \in \mathbb{N}$ by: $F_n^{(0)}(x) = x$ and $F_n^{(i+1)}(x) = F_n\left(F_n^{(i)}(x)\right)$.

Algorithm 4. Integral matrix decomposition $\mathsf{IGD}(\mathbf{\Sigma}, d, B, t, b, k)$

Input: a symmetric matrix $\mathbf{\Sigma} \in \mathbb{Z}^{n\times n}$ with $\|\mathbf{\Sigma}\|_2 \leq B$,
non-negative integers b, k, t such that $b^k \geq F_n^{(t)}(B) + k(n-1)b^2$,
an integer d such that $d \geq \frac{b^{2k}-1}{b^2-1} + b^k + \sum_{i=0}^{t-1} F_n^{(i)}(B)$.

Output: $\mathbf{A} = \begin{pmatrix}\mathbf{A}_1 & \mathbf{A}_2\end{pmatrix} \in \mathbb{Z}^{n\times(m_1+m_2)}$ such that $\mathbf{A}\mathbf{A}^t = d\mathbf{I}_n - \mathbf{\Sigma}$ where
$m_1 = nt$ and $m_2 = n(k+4)$,
$\mathbf{A}_1 \in \mathbb{Z}^{n\times m_1}$ consists of t lower-triangular matrices,
$\mathbf{A}_2 = \mathsf{DiagDomIGD}(\mathbf{\Pi}, d - \sum_{i=0}^{t-1} F_n^{(i)}(B), b, k) \in \mathbb{Z}^{n\times m_2}$ where $\|\mathbf{\Pi}\|_2 \leq F_n^{(t)}(B)$.

$\mathbf{A}_1 \leftarrow ()$ (an empty matrix $\in \mathbb{Z}^{n\times 0\cdot n}$), $\mathbf{\Pi} \leftarrow \mathbf{\Sigma}$
for $i = 1, \cdots, t$ **do**
  $(\mathbf{L}, \mathbf{\Pi}) \leftarrow \mathsf{EigenRed}(\mathbf{\Pi}, F_n^{(i-1)}(B))$
  $\mathbf{A}_1 \leftarrow \begin{pmatrix}\mathbf{A}_1 & \mathbf{L}\end{pmatrix}$
end for
$\mathbf{A}_2 \leftarrow \mathsf{DiagDomIGD}(\mathbf{\Pi}, d - \sum_{i=0}^{t-1} F_n^{(i)}(B), b, k) \in \mathbb{Z}^{n\times m_2}$
return $\begin{pmatrix}\mathbf{A}_1 & \mathbf{A}_2\end{pmatrix}$

Lemma 8. *Algorithm 4 is correct. More precisely, let $\mathbf{\Sigma} \in \mathbb{Z}^{n\times n}$ be symmetric and $d, B, t, b, k \in \mathbb{N}$ such that $\|\mathbf{\Sigma}\|_2 \leq B$, $b^k \geq F_n^{(t)}(B) + k(n-1)b^2$ and $d \geq \frac{b^{2k}-1}{b^2-1} + b^k + \sum_{i=0}^{t-1} F_n^{(i)}(B)$. Then $\mathsf{IGD}(\mathbf{\Sigma}, d, B, t, b, k)$ outputs $\mathbf{A} \in \mathbb{Z}^{n\times m}$ such that $\mathbf{A}\mathbf{A}^t = d\mathbf{I}_n - \mathbf{\Sigma}$ and $m = n(t+k+4)$.*

Remark 6. In practice, the calculation of the Gram root $\mathbf{A}$ can be accomplished during the key generation and needs to run only once. Therefore, we do not take into account the complexity of $\mathsf{IGD}(\mathbf{\Sigma}, d, B, t, b, k)$.

Proof. It can be verified that

$$\mathbf{A}_1\mathbf{A}_1^t = \left(\sum_{i=0}^{t-1} F_n^{(i)}(B)\right) \cdot \mathbf{I}_n - \mathbf{\Sigma} + \mathbf{\Pi}$$

and

$$\mathbf{A}_2\mathbf{A}_2^t = \left(d - \sum_{i=0}^{t-1} F_n^{(i)}(B)\right) \cdot \mathbf{I}_n - \mathbf{\Pi},$$

hence $\mathbf{A}\mathbf{A}^t = d\mathbf{I}_n - \mathbf{\Sigma}$. According to Lemmata 5 and 7, all conditions required by the calls of $\mathsf{EigenRed}(\mathbf{\Pi}, F_n^{(i-1)}(B))$ and $\mathsf{DiagDomIGD}(\mathbf{\Pi}, d - \sum_{i=0}^{t-1} F_n^{(i)}(B), b, k)$ are satisfied. Therefore, Algorithm 4 is correct. □

Now we give an upper bound of $\lambda_{m-n}(\Lambda^{\perp}(\mathbf{A}))$ in Lemma 9. Similar to Lemma 6, we also pad the Gram root $\mathbf{A}$ with $\mathbf{I}_n$.

Lemma 9. *Let* $\mathbf{A} = \mathsf{IGD}(\mathbf{\Sigma}, d-1, B, t, b, k) \in \mathbb{Z}^{n\times m}$ *where* $m = n(t+k+4)$. *Let* $\mathbf{A}' = \begin{pmatrix}\mathbf{I}_n & \mathbf{A}\end{pmatrix}$, *then* $\mathbf{A}' \cdot \mathbb{Z}^{n+m} = \mathbb{Z}^n$, *the dimension of* $\Lambda^{\perp}(\mathbf{A}')$ *is* m, *and*

$$\lambda_m(\Lambda^{\perp}(\mathbf{A}')) \le \max\left\{b^2\sqrt{n}, \sqrt{d + F_n^{(t)}(B) - \frac{b^{2k}-1}{b^2-1} - \sum_{i=0}^{t-1} F_n^{(i)}(B)}, \max_{0\le i<t}\sqrt{2F_n^{(i)}(B)} + n\right\}.$$

Proof. Let $\mathbf{A} = \begin{pmatrix}\mathbf{A}_1 & \mathbf{A}_2\end{pmatrix}$ where $\mathbf{A}_1 \in \mathbb{Z}^{n\times m_1}$ with $m_1 = nt$ and $\mathbf{A}_2 \in \mathbb{Z}^{n\times m_2}$ with $m_2 = n(k+4)$. Let $\mathbf{A}_1' = \begin{pmatrix}\mathbf{I}_n & \mathbf{A}_1\end{pmatrix}$ and $\mathbf{A}_2' = \begin{pmatrix}\mathbf{I}_n & \mathbf{A}_2\end{pmatrix}$, then

$$\lambda_m(\Lambda^{\perp}(\mathbf{A}')) \le \max\{\lambda_{m_1}(\Lambda^{\perp}(\mathbf{A}_1')), \lambda_{m_2}(\Lambda^{\perp}(\mathbf{A}_2'))\}.$$

The matrix $\mathbf{A}_1$ consists of t lower-triangular matrices, denoted by $\mathbf{L}_1, \cdots, \mathbf{L}_t$, such that $\mathbf{L}_i = \left\lfloor \widetilde{\mathbf{L}_i} \right\rceil$ and $\|\widetilde{\mathbf{L}_i}\|_{\mathrm{col}} \le \|\widetilde{\mathbf{L}_i}\|_2 \le \sqrt{2F_n^{(i-1)}(B)}$ by Lemma 7. It follows that $\|\mathbf{L}_i\|_{\mathrm{col}} \le \|\widetilde{\mathbf{L}_i}\|_{\mathrm{col}} + \frac{\sqrt{n}}{2} \le \sqrt{2F_n^{(i-1)}(B)} + \frac{\sqrt{n}}{2}$, and then we have

$$\lambda_{m_1}(\Lambda^{\perp}(\mathbf{A}_1')) \le \max_{1\le i\le t}\sqrt{\|\mathbf{L}_i\|_{\mathrm{col}}^2 + 1} \le \max_{0\le i<t}\sqrt{2F_n^{(i)}(B)} + n.$$

As for $\lambda_{m_2}(\Lambda^{\perp}(\mathbf{A}_2'))$, combining Lemmata 6 and 7 leads to that

$$\lambda_{m_2}(\Lambda^{\perp}(\mathbf{A}_2')) \le \max\left\{b^2\sqrt{n}, \sqrt{d + F_n^{(t)}(B) - \frac{b^{2k}-1}{b^2-1} - \sum_{i=0}^{t-1} F_n^{(i)}(B)}\right\}.$$

The proof is completed. □

As the parameters n and B have been determined before the key generation, one can first choose suitable (t, b, k) and then proceed to minimize d satisfying the requirements of Algorithm 4. We next discuss concrete parameter selections according to the size of B.

Case 1: $B = \omega(n^6)$. In this case, we insist on a common gadget setting: $b = 2$. One can first fix $(t, b) = (2, 2)$ and then choose $k = 1 + \left\lceil \frac{3}{2}\log(n+1) + \frac{1}{4}\log B \right\rceil$. The minimal d is bounded by $B + 2(n+1)^3\sqrt{B} = (1 + o(1))B$.

Under this setting, the final integral Gram root $\mathbf{A}' = \begin{pmatrix}\mathbf{I}_n & \mathbf{A}\end{pmatrix}$ is of size $n \times (n+m)$ with $m = n(k+6)$, and $\lambda_m(\Lambda^{\perp}(\mathbf{A}')) \le \sqrt{2B} + n$ for $d \le 3B$.

Case 2: $B = \omega(n^4)$. We now insist on minimizing the total size of the output $\mathbf{A}$. To this end, one first sets $(t, k) = (1, 3)$ and then selects $b = \left\lceil 3n + \sqrt{2}n^{\frac{1}{3}}B^{\frac{1}{6}} \right\rceil$. The minimal d is bounded by $B + 2b^4 = (1 + o(1))B$.

Under this setting, the final integral Gram root $\mathbf{A}' = \begin{pmatrix}\mathbf{I}_n & \mathbf{A}\end{pmatrix}$ is of size $n \times (n+m)$ with $m = 8n$, and $\lambda_m(\Lambda^{\perp}(\mathbf{A}')) \le \max\{\sqrt{2B} + n, b^2\sqrt{n} = O(n^{\frac{7}{6}}B^{\frac{1}{3}})\}$ for $d \le 3B$.

Case 3: $B = O(n^4)$. In some scenarios, B can be relatively small, say $\widetilde{O}(n)$ [4,25], so that the current algorithm does not work directly. But we can still compute an *almost integral* Gram root of $d\mathbf{I}_n - \mathbf{\Sigma}$. By *almost integral*, we mean that we resort to rationals of the form $i/2^\nu$, with only a few rational bits $\nu = O(\log n)$.

The trick is rather simple: by scaling both d and B by a factor of $2^{2\nu}$, one can reduce the case of small B to the case of a large one. This technique indeed applies for any B and $d > B + 1$, when the scaling factor is sufficiently large.

As an example, we choose arbitrary $\nu \in \mathbb{Z}$ such that $2^{2\nu}B = \omega(n^4)$. As shown in Case 2, selecting $(t, k, b) = \left(1, 3, \left\lceil 3n + \sqrt{2}n^{\frac{1}{3}}(2^{2\nu}B)^{\frac{1}{6}} \right\rceil\right)$ allows an almost integral decomposition, and the minimal d is bounded by $B + 2b^4 \cdot 2^{-2\nu} = (1 + o(1))B$.

Under this setting, the final integral Gram root $\mathbf{A}' = 2^{-\nu} \cdot \left(\mathbf{I}_n \ \mathbf{A}\right)$ is of size $n \times (n + m)$ with $m = 8n$. A minor modification to apply Theorem 3 is that one should consider $\lambda_m(\Lambda^\perp(2^\nu \cdot \mathbf{A}'))$ to fulfil Lemma 3. This can be done similarly: $\lambda_m(\Lambda^\perp(2^\nu \cdot \mathbf{A}')) \le \max\{2^\nu\sqrt{2B} + n, b^2\sqrt{n} = O(2^{\frac{2}{3}\nu}n^{\frac{7}{6}}B^{\frac{1}{3}})\}$ for $d \le 3B$.

We compare above parameter selections according to some values including: (1) $\overline{m} = m + n$, determining the size of the Gram root; (2) L, an upper bound of $\lambda_m(\Lambda^\perp(\mathbf{A}'))$ dominating L' in Theorem 3; and (3) d_{min}, the minimum of d proportional to the minimal final width. We summarize three cases in Table 1.

Table 1. Parameter selections of the integral Gram decomposition. In the first two cases, the final Gram root $\mathbf{A}'$ is integral of size $n \times \overline{m}$. In the third one, the final Gram root is $2^{-\nu}\mathbf{A}'$ where $\mathbf{A}' \in \mathbb{Z}^{n \times \overline{m}}$ and $\nu = 2\log n - \frac{1}{2}\log B + \omega(1)$ is an integer.

	$\overline{m} = m + n$	L	d_{min}
Gadget base $b = 2$: $B = \omega(n^6)$	$O(n \log B)$	$O\left(\sqrt{B}\right)$	$(1 + o(1))B$
Large gadget base: $B = \omega(n^4)$	$9n$	$O\left(\sqrt{B} + n^{\frac{7}{6}}B^{\frac{1}{3}}\right)$	$(1 + o(1))B$
Almost integral: $B = O(n^4)$	$9n$	$O\left(2^\nu\sqrt{B} + 2^{\frac{2}{3}\nu}n^{\frac{7}{6}}B^{\frac{1}{3}}\right)$	$(1 + o(1))B$

5 Comparisons with Peikert's Perturbation Sampler

Throughout this section, $\mathbf{\Sigma} \in \mathbb{Z}^{n \times n}$ is a positive semi-definite matrix, and $s = rs'$ is the final Gaussian width where r is the base sampling parameter and $s'^2 \in \mathbb{Z}$ such that $s'^2 \ge e_1(\mathbf{\Sigma}) + 1$. The covariance of perturbation vectors is $r^2(s'^2\mathbf{I}_n - \mathbf{\Sigma})$. The later discussions specialize to $e_1(\mathbf{\Sigma}) = \omega(n^6)$, which can occur in advanced cryptosystems, e.g. hierarchical IBE [1,8] and ABE [20].[5]

[5] When $e_1(\mathbf{\Sigma})$ is small, one can resort to the almost integral Gram root in Sect. 4.3.

Applying the integral matrix decomposition from Sect. 4 along with Theorem 3, we devise a variant of Peikert's perturbation sampling algorithm. This variant requires no floating-point arithmetic, and the intermediate matrix is integral. The centers of the base Gaussian samplings are integers scaled by a common factor L', which is easier to deal with. Moreover, our approach only enlarges the final width by a factor of $1 + o(1)$.

We next compare our sampler with Peikert's one [32] from the following aspects: the storage of the Gram root (Sect. 5.1), required base samplings (Sect. 5.2) and the quality of final Gaussians (Sect. 5.3). Additionally, we discuss the applications within the Micciancio-Peikert trapdoor framework [25] in Sect. 5.4, and show that exploiting the trapdoor, one can significantly reduce the size of the matrix to be decomposed.

5.1 Required Storage

For Peikert's sampler, we follow the suggested setting where the precomputation is a standard (real) Cholesky Gram root of $\sqrt{(s'^2-1)\mathbf{I}_n - \mathbf{\Sigma}}$. It requires $\frac{n(n+1)}{2}(\log s' + \lambda)$ bits of storage, where λ is a security parameter that is usually set to $O(n)$.

In our sampler, the intermediate matrix is $\mathbf{A} = \begin{pmatrix}\mathbf{I}_n & \mathbf{A}_1 & \mathbf{A}_2\end{pmatrix} \in \mathbb{Z}^{n+m}$ where $\begin{pmatrix}\mathbf{A}_1 & \mathbf{A}_2\end{pmatrix} = \mathsf{IGD}(\mathbf{\Sigma}, s'^2 - 2, B, t, b, k) \in \mathbb{Z}^{n\times m}$ and $m = n(t + k + 4)$. The sub-matrix $\mathbf{A}_1$ consists of t lower-triangular matrices, and its storage is about $\frac{n(n+1)}{2}\left(\sum_{i=0}^{t-1}\log\sqrt{F_n^{(i)}(B)}\right)$ bits. The parameter t can be very small, say $t = 1, 2$ (see Sect. 4.3), and $\{F_n^{(i)}(B)\}_i$ decreases very fast at the beginning for large B. Therefore, the actual storage of $\mathbf{A}_1$ is well bounded. As for $\mathbf{A}_2$, while it is even wider, namely $n \times n(k+4)$, its regular structure allows an efficient storage. More precisely, treating (b, k) as global variables, it suffices to store off-diagonal entries that are in $(-b, b)$ in the first k blocks and diagonal ones in the rest blocks (see Algorithm 2). Thus the storage of $\mathbf{A}_2$ is bounded by $\frac{n(n-1)}{2}k\log b + 2n\log(s')$, which is about $\frac{n(n-1)}{2}\log(F_n^{(t)}(B)) + 2n\log(s')$ when $b^k = O(F_n^{(t)}(B))$.

We summarize in Table 2 the storage comparison. Despite the integral Gram root $\mathbf{A}$ being wider, it can achieve asymptotically better storage efficiency than the FPA solution. In fact, the storage is still an advantage even we apply the almost integral decomposition in Sect. 4.3 to deal with small $e_1(\mathbf{\Sigma})$.

5.2 Base Samplings

To generate an integral perturbation, one first samples from a Gaussian of covariance $r^2((s'^2-1)\mathbf{I}_n - \mathbf{\Sigma})$. In Peikert's sampler, this is accomplished by continuous Gaussian sampling with high precision, which is expensive. In our sampler, this is accomplished by sampling from $D_{\mathbb{Z}^{n+m}, L'r}$ and then multiplying a scaled integral Gram root, i.e. $\frac{1}{L'}\mathbf{A}$.

Table 2. The storage comparison. The concrete parameter selections of $t = 1, 2$ are discussed in Sect. 4.3.

	Storage of Gram root
Peikert's sampler	$\approx \frac{n^2}{2}(\log s' + \lambda)$ where λ is the security parameter
Ours	$\approx \frac{n^2}{4}\left(\sum_{i=0}^{t} \log(F_n^{(i)}(s'^2)) + \log(F_n^{(t)}(s'^2))\right)$
Ours with $t = 1$	$\approx n^2(\log s' + \frac{1}{2}\log n)$
Ours with $t = 2$	$\approx n^2(\log s' + \log n)$

There are also some non-centered samplings. Peikert's algorithm requires to sample from $D_{\mathbb{Z},r,c}$ with a floating-point center. In our sampler, all Gaussian centers are in $\frac{1}{L'} \cdot \mathbb{Z}$ with some $L' \geq \lambda_m(\Lambda^\perp(\mathbf{A}))$.

We exhibit the comparison on the base samplings in Table 3. As shown in Sect. 4.3, we may choose some $L' = O(s' + n^{\frac{7}{6}} s'^{\frac{2}{3}})$. When the padding trick is used (see Sect. 3.1), L' can be even smaller, namely $O\left(\sqrt{n \log s'}\right)$. In concrete implementations, we would suggest to set L' to be a power-of-2 so that with a minor modification, all samplings can be done by only two base samplers $D_{\mathbb{Z},r}$ and $D_{\mathbb{Z},r,1/2}$ as in [28]. Therefore, the base samplings required by us are easier to implement than that by Peikert. While our sampler requires more centered samples, $(n + m)$ is $O(n \log s')$ even $O(n)$, which does not increase the base sample number too much.

Table 3. The base sampling comparison. Here $D_{\mathbb{R},r}$ denotes the continuous Gaussian over $\mathbb{R}$ of width r. In our sampler, the Gram root $\mathbf{A} \in \mathbb{Z}^{n \times (n+m)}$. If no padding, $m = O(n)$ (say $8n$) and $L' = O(s' + n^{\frac{7}{6}} s'^{\frac{2}{3}}) \geq \lambda_m(\Lambda^\perp(\mathbf{A}))$. If padding is used, $m = O(n \log s')$ and $L' = O(\sqrt{n \log s'})$.

	Centered samplings	Non-centered samplings
Peikert's sampler	$D_{\mathbb{R},r}$	$D_{\mathbb{Z},r,c}$ with $c \in \mathbb{R}$
	n times	n times
Ours	$D_{\mathbb{Z},L'r}$	$D_{\mathbb{Z},r,c}$ with $c \in \frac{1}{L'} \cdot \mathbb{Z}$
	$O(n)$ times	n times
Ours with padding (Sect. 3.1)	$D_{\mathbb{Z},L'r}$	$D_{\mathbb{Z},r,c}$ with $c \in \frac{1}{L'} \cdot \mathbb{Z}$
	$O(n \log s')$ times	n times

5.3 The Quality of Final Gaussians

In Peikert's sampler, $r \geq \eta_\epsilon(\mathbb{Z}^n)$ and $s'^2 \geq e_1(\mathbf{\Sigma}) + 1$ so that the minimal s is $\eta_\epsilon(\mathbb{Z}^n)\sqrt{e_1(\mathbf{\Sigma}) + 1}$. Our sampler also applies to any $r \geq \eta_\epsilon(\mathbb{Z}^n)$ according to Theorem 3. As for s', its minimum is $(1 + o(1))\sqrt{e_1(\mathbf{\Sigma})}$ (see Sect. 4.3). Thus the minimal s achieved by us is $(1 + o(1)) \cdot \eta_\epsilon(\mathbb{Z}^n)\sqrt{e_1(\mathbf{\Sigma})}$. As a conclusion, our sampler only leads to a very small loss in the quality of final Gaussians.

5.4 The Case of the Micciancio-Peikert Trapdoor

In [25], Micciancio and Peikert propose a celebrated trapdoor framework which has been the basis of various primitives [4,12,20]. In this framework, the matrix $\mathbf{\Sigma} = \begin{pmatrix} \mathbf{T} \\ \mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{T}^t & \mathbf{I} \end{pmatrix}$ where $\mathbf{T} \in \mathbb{Z}^{n_1 \times n_2}$ is the trapdoor with $n_1 \ll n_1 + n_2$. The Gaussian sampling is performed by an entity that knows the trapdoor $\mathbf{T}$.

We now explain how to use the trapdoor to reduce the input size at the beginning of the matrix decomposition of $\mathbf{\Sigma}' = d\mathbf{I}_n - \mathbf{\Sigma}$ (a similar idea is used in [17]). More precisely, notice that

$$\mathbf{\Sigma}' = d\mathbf{I}_n - \mathbf{\Sigma} = \begin{pmatrix} d\mathbf{I}_{n_1} - 2\mathbf{T}\mathbf{T}^t & \\ & (d-2)\mathbf{I}_{n_2} \end{pmatrix} + \begin{pmatrix} \mathbf{T} \\ -\mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{T}^t & -\mathbf{I} \end{pmatrix}.$$

Hence, it suffices to decompose $\mathbf{\Sigma}'_{new} = d\mathbf{I}_{n_1} - 2\mathbf{T}\mathbf{T}^t$ whose dimension is n_1, which is much less than $n = n_1 + n_2$.[6] This trick needs neither extra computation nor storage, and only enlarges the maximal elements in $\mathbf{\Sigma}_{new} (= 2\mathbf{T}\mathbf{T}^t)$ by a factor of 2. Therefore we suggest to use this trick as the preprocessing of the integral decomposition in the Micciancio-Peikert trapdoor framework.[7]

6 The Ring Setting

Many lattice cryptosystems use polynomial rings. In this setting, vectors and matrices consist of ring elements, which improves storage and running time. Some previous works [15,17,25,32] provide ring-efficient Gaussian sampling in which intermediate matrices preserve the ring structure, but require high-precision FPA. One the other hand, the generic techniques in Sect. 4 avoid high-precision FPA but require to take $\mathbb{Z}$ as a base ring, therefore not benefiting from efficiency gains that are typically expected in the ring setting.

The goal of this section is to get *the best of both worlds*: being ring-efficient and avoiding high-precision FPA. We realize that by proposing an integral decomposition algorithm for the ring setting. By Theorem 3, this will imply a Gaussian sampler that is both ring-efficient and FPA-free. To this end, we first formally define the *Integral Gram Decomposition Problem over the ring* $\mathcal{R}$.

Definition 4 ($\mathsf{IGDP}_{\mathcal{R},n,B,d,m}$). *Let* $\mathcal{R} = \mathbb{Z}[x]/\Phi(x)$ *where* $\Phi(x) \in \mathbb{Z}[x]$ *and* $n, B, d, m \in \mathbb{N}$. *The Integral Gram Decomposition Problem over* $\mathcal{R}$, *denoted by* $\mathsf{IGDP}_{\mathcal{R},n,B,d,m}$, *is defined as follows: given an integral symmetric matrix* $\mathbf{\Sigma} \in \mathcal{R}^{n \times n}$ *with* $\|\mathbf{\Sigma}\|_2 \leq B$, *find an integral matrix* $\mathbf{A} \in \mathcal{R}^{n \times m}$ *such that* $\mathbf{A}\mathbf{A}^t = d\mathbf{I}_n - \mathbf{\Sigma}$.

Clearly, $\mathsf{IGDP}_{\mathcal{R},n,B,d,m}$ is a natural generalization of the IGDP problem which introduces a new parameter: the ring $\mathcal{R}$. The initial definition of $\mathsf{IGDP}_{n,B,d,m}$ (Definition 3) corresponds to the case $\mathcal{R} = \mathbb{Z}$. For simplicity we only discuss

[6] The other block can be addressed by 4-square decomposition directly.

[7] Note that this requires some new analysis of smoothness conditions.

power-of-2 cyclotomic rings, i.e. $\mathcal{R}_{2w} = \mathbb{Z}[x]/(x^w + 1)$ with $w = 2^\ell$. Similarly to [15], the results can be extended to more general cyclotomic rings and convolution rings with smooth conductors. Why do we care about solving $\mathsf{IGDP}_{\mathcal{R},n,B,d,m}$ for trapdoor sampling? Indeed, a naive and functional approach is to embed $\mathbf{\Sigma}$ in $\mathbb{Z}^{wn\times wn}$ via the coefficient embedding, then solve $\mathsf{IGDP}_{\mathbb{Z},wn,B,d',m'}$ as in Sect. 4. However, that would break the ring structure and cancel the main advantage of rings: efficiency. Therefore our goal is to directly solve $\mathsf{IGDP}_{\mathcal{R},n,B,d,m}$; if $m' = m \cdot \tilde{\omega}(w)$, this improves the storage and running time (via the number theoretic transform) of lattice Gaussian sampling by a factor $\tilde{O}(w)$ compared to the naive approach.

Technical roadmap. We first recall some preliminaries on cyclotomic rings (Sect. 6.1). Next, we propose a solution for the special case $\mathsf{IGDP}_{\mathcal{R}_{2w},1,B,d,m}$; this particular case is also a generalization of the 4-square decomposition, for self-adjoint ring elements instead of natural numbers. Our solution relies on a technique called *ring gadgets* to reduce $\mathsf{IGDP}_{\mathcal{R}_{2w},1,B,d,m}$ to $\mathsf{IGDP}_{\mathcal{R}_w,1,B',d',m'}$. Naturally, a repeated application of ring gadgets allows to project the initial problem onto $\mathbb{Z}$ eventually, which is then solved by 4-square decomposition.

Once we know how to decompose a single polynomial, a general solution to $\mathsf{IGDP}_{\mathcal{R}_{2w},n,B,d,m}$ is easily derived by adapting Algorithm 2 to the ring setting (Sect. 6.3). Compared with the generic solution (Sect. 4), the ring-based integral decomposition reduces storage by a factor $O(w)$ and running time by a factor $\tilde{O}(w)$, at the cost of increasing the number of columns of the integral Gram root by $O(\log w)$. This leads to a simple ring-based sampler achieving the same efficiency as the state of the art [17] (Sect. 6.4).

Comparison with the generic technique. Algorithms in Sect. 4 and in this section operate on the same central idea: to recursively project the initial problem into smaller dimensions. The only conceptual difference is how this projection is done; as we now impose an additional constraint, i.e. preserving the ring structure, projection requires to use ring gadgets in addition to matrix decomposition. In addition, these algorithms treat the eigenvalue reduction and the bottom case differently. We summarize used techniques as Table 4.

Table 4. Different techniques in integral decompositions.

	Section 4 $\mathsf{IGDP}_{n,B,d,m}$	Section 6.2 $\mathsf{IGDP}_{\mathcal{R}_{2w},1,B,d,m}$	Section 6.3 $\mathsf{IGDP}_{\mathcal{R}_{2w},n,B,d,m}$
Eigenvalue reduction	Cholesky	–	Structured Cholesky
Projection	Integer gadget	Ring gadget	Integer gadget
Bottom case	4-square	4-square	$\mathsf{IGDP}_{\mathcal{R}_{2w},1,B,d,m}$

6.1 Preliminaries on Cyclotomic Rings

Let $w \in \mathbb{N}$ and $\Phi_w(x) \in \mathbb{Z}[x]$ be the w-th cyclotomic polynomial. The w-th *cyclotomic ring* is $\mathcal{R}_w = \mathbb{Z}[x]/(\Phi_w(x))$ and the w-th *cyclotomic field* is $\mathbb{K}_w = \mathbb{Q}[x]/(\Phi_w(x))$. In this paper, we only discuss the case of power-of-2 cyclotomic rings where $w = 2^\ell$ and $\Phi_{2w}(x) = x^w + 1$. For such kind of rings, we have the following *tower of rings*:

$$\mathcal{R}_{2w} \supseteq \mathcal{R}_w \supseteq \cdots \supseteq \mathcal{R}_2 = \mathbb{Z}. \tag{1}$$

Adjoints. Let $\Phi \in \mathbb{R}[x]$ be monic with distinct roots over $\mathbb{C}$, and $f, g \in \mathbb{R}[x]/(\Phi(x))$. We denote by $f^\star$ the (Hermitian) adjoint of f, that is, the unique element of $\mathbb{R}[x]/(\Phi(x))$ such that $f^\star(\xi) = \overline{f(\xi)}$ for any root ξ of Φ. This generalizes the complex conjugation of real numbers. We say that f is *self-adjoint* if $f = f^\star$. It is easy to verify that $ff^\star$ is self-adjoint and all self-adjoint elements form a ring. When Φ is a cyclotomic polynomial, it holds that $f^\star(x) = f(x^{-1})$.

Norms and gadgets. For $f = \sum_{i=0}^{w-1} f_i x^i \in \mathbb{K}_{2w}$, let $\|f\| = \sqrt{\sum_{i=0}^{w-1} |f_i|^2}$ be its ℓ_2*-norm*, and $\|f\|_\infty = \max_i |f_i|$ be its ℓ_∞*-norm*. For $\mathbf{f} = (f_0, \cdots, f_{n-1})^t$ and $\mathbf{g} = (g_0, \cdots, g_{n-1})^t$ in $\mathbb{K}_{2w}^n$, let $\|\mathbf{f}\| = \sqrt{\sum_{i=0}^{n-1} \|f_i\|^2}$, $\|\mathbf{f}\|_\infty = \max_i \|f_i\|_\infty$ and $\langle \mathbf{f}, \mathbf{g} \rangle = \sum_i f_i g_i^\star \in \mathbb{K}_{2w}$. For $\mathbf{\Sigma} \in \mathbb{K}_{2w}^{n \times n}$, let $\|\mathbf{\Sigma}\|_2 = \max_{\mathbf{x} \neq \mathbf{0}} \frac{\|\mathbf{\Sigma x}\|}{\|\mathbf{x}\|}$ and $\|\mathbf{\Sigma}\|_{\max} = \max_{i,j} \|\mathbf{\Sigma}_{i,j}\|_\infty$. Moreover, the gadget decomposition generalizes naturally: given the gadget vector $\mathbf{g} = (1, b, \cdots, b^{k-1})^t \in \mathcal{R}_{2w}^k$, for any $f \in \mathcal{R}_{2w}$ with $\|f\|_\infty < b^k$, there exists $\mathbf{c} \in \mathcal{R}_{2w}^k$ such that $\langle \mathbf{c}, \mathbf{g} \rangle = f$ and $\|\mathbf{c}\|_\infty < b$, and it can be efficiently computed.

Even and odd polynomials. Each $f \in \mathcal{R}_{2w}$ can be uniquely written as:

$$f(x) = f_e(x^2) + x f_o(x^2),$$

where f_e, f_o are elements of the subring $\mathcal{R}_w \subset \mathcal{R}_{2w}$. We say that f_e (resp. f_o) is the even (resp. odd) part of f, and indeed it consists of the even-index (resp. odd-index) coefficients of f, respectively. We also say that a polynomial is even (resp. odd) if its odd (resp. even) part is zero. Any even polynomial of $\mathbb{Z}[x]/(x^w + 1)$ can be seen as an element of $\mathbb{Z}[y]/(y^{w/2} + 1)$ by the ring morphism $y \to x^2$.

Ring gadgets. A key technical component of our algorithms in the ring setting consists of projecting a self-adjoint polynomial $f \in \mathcal{R}_{2w}$ onto the subring $\mathcal{R}_w$. More precisely, we exhibit a polynomial $a \in \mathcal{R}_{2w}$ such that $aa^\star + f$ is even (hence is in the subring $\mathcal{R}_w$) and self-adjoint. Let us write $f = f_e(x^2) + x f_o(x^2)$; since f is self-adjoint, it holds that $f_i = -f_{w-i}$ for each coefficient f_i of f, and we can

therefore write $xf_o(x^2) = x\bar{f}_o(x^2) + (x\bar{f}_o(x^2))^\star$ for some polynomial $\bar{f}_o \in \mathcal{R}_w$ with only its lower-half coefficients nonzero. Taking $a = 1 - x\bar{f}_o(x^2)$, we have:

$$\begin{aligned} f + aa^\star &= f_e(x^2) + x\bar{f}_o(x^2) + (x\bar{f}_o(x^2))^\star + (1 - x\bar{f}_o(x^2))(1 - x\bar{f}_o(x^2))^\star \\ &= f_e(x^2) + \bar{f}_o(x^2)(\bar{f}_o(x^2))^\star + 1 \end{aligned}$$

All the odd terms have been eliminated, and $f + aa^\star$ is isomorphic to an element of $\mathcal{R}_w$. As an example, let us consider the following self-adjoint element of $\mathcal{R}_{16}$:

$$\begin{aligned} f &= 32 - 8x + 2x^2 - 9x^3 + 9x^5 - 2x^6 + 8x^7, \\ f &= (32 + 2x^2 - 2x^6) + x(-8 - 9x^2) + (x(-8 - 9x^2))^\star. \end{aligned}$$

Then we will take: $a = 1 - x\bar{f}_o(x^2) = 1 + 8x + 9x^3$. One can check that:

$$f + aa^\star = -74x^6 + 74x^2 + 178,$$

which is indeed in the subring $\mathcal{R}_8$. This projection is compatible with the use of gadget matrices; more precisely, we can first decompose a polynomial using gadget decomposition, and then apply the projection to each element of the decomposition. Finally we have $f + \sum_{i=1}^{k} a_i a_i^\star$ is even, where $a_i = b^{i-1} + xc_i(x^2)$ and $\sum_{i=1}^{k} b^{i-1}c_i = -\bar{f}_o$. We will refer to these combined decomposition and projection as *ring gadgets*.

6.2 Decomposition for Ring Elements

We proceed to generalize 4-square decomposition to the ring setting. Precisely, our goal is to represent one integral self-adjoint ring element $f \in \mathcal{R}_{2w}$ as $\langle \mathbf{a}, \mathbf{a} \rangle$ where $\mathbf{a} \in \mathcal{R}_{2w}^m$ is an integral polynomial vector. Equivalently, we seek to solve the special case $\mathsf{IGDP}_{\mathcal{R}_{2w},1,B,d,m}$.

Our solution is build upon the use of *ring gadgets*, defined at the end of Sect. 6.1. As previously illustrated, a single application of a ring gadget can be viewed as the reduction $\mathsf{IGDP}_{\mathcal{R}_{2w},1,B,d,m} \to \mathsf{IGDP}_{\mathcal{R}_w,1,B',d',m'}$. Hence, we have projected our problem onto a subring.

We can go further. As recalled in (1), the $\mathcal{R}_i$'s are arranged in a tower of rings structure, thus we can repeatedly apply ring gadgets to project $\mathsf{IGDP}_{\mathcal{R},1,B,d,m}$ onto a subring, until it is projected in $\mathcal{R}_4$. We note that the set of all self-adjoint elements of $\mathcal{R}_4$ is exactly $\mathbb{Z}$ and thus $\mathsf{IGDP}_{\mathcal{R}_4,1,B,B,4}$ is easily solved via the Rabin-Shallit algorithm. We have this chain of reductions:

$$\mathsf{IGDP}_{\mathcal{R}_{2w},1,B,d,m} \to \mathsf{IGDP}_{\mathcal{R}_w,1,B',d',m'} \to \cdots \to \mathsf{IGDP}_{\mathcal{R}_4,1,B'',d'',m''}$$

We formally describe the procedure in Algorithm 5.

Lemma 10 shows the correctness and the complexity analysis of Algorithm 5.

Algorithm 5. Decomposition of a single ring element $\mathsf{REIGD}(d, f, b, k)$

Input: a self-adjoint $f \in \mathcal{R}_{2w}$ with $w = 2^\ell \geq 2$,
two integers $b, k \geq 2$ such that $b^k \geq \|f\|_\infty + kwb^2$,
an integer d such that $d \geq \frac{b^{2k}-1}{b^2-1}(\ell - 1) + b^k$.

Output: $\mathbf{a} = \begin{pmatrix} a_1 & \cdots & a_{k(\ell-1)} & x_1 & x_2 & x_3 & x_4 \end{pmatrix} \in \mathcal{R}_{2w}^{k(\ell-1)+4}$ such that $\langle \mathbf{a}, \mathbf{a} \rangle = d - f$, where $x_1, x_2, x_3, x_4 \in \mathbb{Z}$ and $a_{1+jk+i} = b^i + a'_{i,j}\left(x^{2^{\ell-2-j}}\right)$ with $a'_{i,j} \in \mathcal{R}_{2^{j+2}}$ for any $0 \leq i < k$ and $0 \leq j < \ell - 1$.

$\mathbf{g} \leftarrow (1, b, \cdots, b^{k-1})^t$
if $w = 2$ **then**
  calculate $\mathbf{x} = (x_1, x_2, x_3, x_4)^t \in \mathbb{Z}^4$ such that $\|\mathbf{x}\|^2 = d - f$ using the Rabin-Shallit algorithm (Theorem 2)
  return $\mathbf{x}$
end if
calculate $\begin{pmatrix} a_1 & \cdots & a_k \end{pmatrix} \in \mathcal{R}_{2w}^k$ by using ring gadgets such that $f + \sum_i a_i a_i^\star$ is even
let $f' \in \mathcal{R}_w$ such that $f'(x^2) = f - \frac{b^{2k}-1}{b^2-1} + \sum_i a_i a_i^\star$
$\mathbf{a}' = \begin{pmatrix} a'_1 & \cdots & a'_{k(\ell-2)} & x_1 & x_2 & x_3 & x_4 \end{pmatrix} \leftarrow \mathsf{REIGD}\left(d - \frac{b^{2k}-1}{b^2-1}, f', b, k\right)$
return $\mathbf{a} = \begin{pmatrix} a'_1(x^2) & \cdots & a'_{k(\ell-2)}(x^2) \parallel a_1 & \cdots & a_k \parallel x_1 & x_2 & x_3 & x_4 \end{pmatrix} \in \mathcal{R}_{2w}^{k(\ell-1)+4}$

Lemma 10. *Algorithm 5 is correct. More precisely, let $w = 2^\ell \geq 2$ and $f \in \mathcal{R}_{2w}$ be a self-adjoint polynomial, let $d, b, k \in \mathbb{Z}$ such that $b^k \geq \|f\|_\infty + kwb^2$ and $d \geq \frac{b^{2k}-1}{b^2-1}(\ell - 1) + b^k$. Then $\mathsf{REIGD}(d, f, b, k)$ outputs $\mathbf{a} \in \mathcal{R}_{2w}^{k(\ell-1)+4}$ such that $\langle \mathbf{a}, \mathbf{a} \rangle = d - f$.*

Moreover, $\mathsf{REIGD}(d, f, b, k)$ performs $O(kw \log w + \log^2 d' \log\log d')$ arithmetic operations on integers of bitsize $O(\log d')$ where $d' = d - \frac{b^{2k}-1}{b^2-1}(\ell - 1)$.

Proof. Let $f_e \in \mathcal{R}_w$ be the even part of f. Let $a_i = b^{i-1} + xc_i(x^2)$ where $c_i \in \mathcal{R}_w$ with $\|c_i\|_\infty < b$ and only its lower-half coefficients nonzero. Since $f(x) + \sum_i a_i a_i^\star = f'(x^2) + \frac{b^{2k}-1}{b^2-1}$, we inductively conclude that $\langle \mathbf{a}, \mathbf{a} \rangle = d - f$. A routine computation shows that $f' = f_e + \sum_i c_i c_i^\star$ and that $\|c_i c_i^\star\|_\infty \leq \frac{w}{2} b^2$. By the same argument as in the proof of Lemma 5, the correctness follows.

Algorithm 5 proceeds recursively. At the highest level, there is one gadget decomposition of $(-\bar{f}_o)$, k polynomial multiplications over $\mathcal{R}_w$ and one recursive call. At the bottom level, there is one 4-square decomposition. Thus the total complexity is $O(kw \log w + \log^2 d' \log\log d')$ if one uses NTT techniques during multiplication, and all involved integers are of bitsize at most $O(\log d')$. □

Lemma 10 implies a solution to $\mathsf{IGDP}_{\mathcal{R}_{2w},1,B,d,m}$ as the following corollary.

Corollary 3. *Let $\ell, B, d, b, k \in \mathbb{N}$ and $w = 2^\ell \geq 2$ such that $m = k(\ell-1)+4$ and $d - \frac{b^{2k}-1}{b^2-1}(\ell-1) \geq b^k \geq B + kwb^2$. Then there exists a solution to $\mathsf{IGDP}_{\mathcal{R}_{2w},1,B,d,m}$ and it can be calculated by Algorithm 5.*

The output of Algorithm 5 consists of a series of vectors built upon the tower of rings followed by 4 integers, hence the storage can be essentially the same as

that of f due to the polynomials in the tower of rings being gradually sparser. Detailed argument on the storage is given in Lemma 11.

Lemma 11. *Let* $\mathbf{a}$ *be the output of Algorithm 5, then* $\mathbf{a}$ *can be stored using* $\left(\frac{kw}{2}\log b + 2\log(2d')\right)$ *bits where* $d' = d - \frac{b^{2k}-1}{b^2-1}(\ell-1)$. *In particular, when* $d' = O(\|f\|_\infty)$, *the required storage is* $\left(\frac{w}{2}+2\right)(\log\|f\|_\infty + O(1))$ *bits.*

Proof. The storage of $(x_1, x_2, x_3, x_4) \in \mathbb{Z}^4$ is bounded by $2\log(2d')$. We notice that $a_{1+jk+i} = b^i + a'_{i,j}\left(x^{2^{\ell-2-j}}\right)$ for some $a'_{i,j} \in \mathcal{R}_{2^{j+2}}$ with even coefficients being 0, odd coefficients in $(-b, b)$, hence the storage of a_{1+jk+i} is $2^j \log b$ and then the storage of $\mathbf{a}$ is $\frac{kw}{2}\log b + 2\log(2d')$. □

6.3 Decomposition for Positive Definite $\boldsymbol{\Sigma' \in \mathcal{R}_{2w}^{n\times n}}$

We now show how to solve the generalized problem $\mathsf{IGDP}_{\mathcal{R}_{2w},n,B,d,m}$. Our ring-setting matrix decomposition is illustrated in Algorithm 6. The high level idea is the same in spirit to Algorithm 4, except that we replace the Rabin-Shallit algorithm with a decomposition based on ring gadgets (Algorithm 5). For $\boldsymbol{\Sigma}' = d\mathbf{I}_n - \boldsymbol{\Sigma} \in \mathcal{R}_{2w}^{n\times n}$, one first calculates some $\mathbf{T} \in \mathcal{R}_{2w}^{n\times k}$ such that $\mathbf{T}\mathbf{T}^t$ has the same first row and column as $\boldsymbol{\Sigma}'$, except the diagonal element, and then proceeds iteratively over $(\boldsymbol{\Sigma}' - \mathbf{T}\mathbf{T}^t)_{2:n,2:n} \in \mathcal{R}_{2w}^{(n-1)\times(n-1)}$. During construction of $\mathbf{T}$ we deal with off-diagonal elements by gadget decomposition, and decompose the remaining diagonal element with Algorithm 5. Detailed analysis is shown in Lemma 12.

Lemma 12. *Algorithm 6 is correct. More precisely, let* $w = 2^\ell \geq 2$ *and* $\boldsymbol{\Sigma} \in \mathcal{R}_{2w}^{n\times n}$ *be a symmetric matrix, let* $d, b, k \in \mathbb{Z}$ *such that* $b^k \geq \|\boldsymbol{\Sigma}\|_{\max} + knwb^2$ *and* $d \geq \frac{b^{2k}-1}{b^2-1}\ell + b^k$. *Then* $\mathsf{RMIGD}(d, \boldsymbol{\Sigma}, b, k)$ *outputs* $\mathbf{A} \in \mathcal{R}_{2w}^{n\times n(k\ell+4)}$ *such that* $\mathbf{A}\mathbf{A}^t = d\mathbf{I}_n - \boldsymbol{\Sigma}$.

Moreover, $\mathsf{RMIGD}(d, \boldsymbol{\Sigma}, b, k)$ *performs* $O(n^3kw\log w + n\log^2 d'\log\log d')$ *arithmetic operations on integers of bitsize at most* $O(\log d')$, *and* $\mathbf{A}$ *can be stored using* $\left(\frac{n^2}{2}kw\log b + 2n\log(2d')\right)$ *bits where* $d' = d - \frac{b^{2k}-1}{b^2-1}\ell$.

Proof (sketch). A routine computation shows that $\|\mathbf{C}\mathbf{C}^t\|_{\max} \leq kwb^2$. Following the same argument as the proof of Lemma 5, we confirm the correctness.

According to Lemma 10, all involved integers are of bitsize at most $O(\log d')$, and the complexity is mainly contributed by (1) the gadget decompositions, (2) calls to Algorithm 5 and (3) matrix multiplications. More specifically, there are $O(n^2)$ times gadget decompositions, hence the total complexity of this part is $O(kwn^2)$. There are n calls to Algorithm 5 that entirely costs $O(nkw\log w + n\log^2 d'\log\log d')$ according to Lemma 10. Furthermore, the cost of all matrix multiplications is bounded by $O(kn^3w\log w)$. To sum up, the running time of $\mathsf{RMIGD}(d, \boldsymbol{\Sigma}, b, k)$ is dominated by $O(n^3kw\log w + n\log^2 d'\log\log d')$.

From Lemma 11, the storage of $\mathbf{D}$ is $n\left(\frac{kw}{2}\log b + 2\log(2d')\right)$ and that of each $\mathbf{L}_i$ is $\frac{n(n-1)}{2}w\log b$. The overall storage thus is $\left(\frac{n^2}{2}kw\log b + 2n\log(2d')\right)$. □

Algorithm 6. Integral matrix decomposition in ring setting $\mathsf{RMIGD}(d, \mathbf{\Sigma}, b, k)$

Input: a symmetric matrix $\mathbf{\Sigma} \in \mathcal{R}_{2w}^{n \times n}$ with $w = 2^\ell \geq 2$,
integers $d, b, k \geq 2$ such that $b^k \geq \|\mathbf{\Sigma}\|_{\max} + knwb^2$ and $d \geq \frac{b^{2k}-1}{b^2-1}\ell + b^k$.

Output: $\mathbf{A} = \begin{pmatrix} \mathbf{L}_1 \cdots \mathbf{L}_k \ \mathbf{D} \end{pmatrix} \in \mathcal{R}_{2w}^{n \times n(k\ell+4)}$ such that $\mathbf{A}\mathbf{A}^t = d\mathbf{I}_n - \mathbf{\Sigma}$
where $\mathbf{L}_i \in \mathcal{R}_{2w}^{n \times n}$ is a lower-triangular matrix with diagonal elements being b^{i-1} and off-diagonal elements of ℓ_∞-norm less than b,
$\mathbf{D} \in \mathcal{R}_w^{n \times n(k(\ell-1)+4)}$ is a block diagonal matrix with each block being the output of $\mathsf{REIGD}(d, f, b, k)$ for some $f \in \mathcal{R}_{2w}$.

$\mathbf{g} \leftarrow (1, b, \cdots, b^{k-1})^t \in \mathcal{R}_{2w}^k$
$\mathbf{x} \leftarrow \mathsf{REIGD}(d - \|\mathbf{g}\|^2, \mathbf{\Sigma}_{1,1}, b, k) \in \mathcal{R}_{2w}^{k(\ell-1)+4}$ {Call to Algorithm 5}
if $n = 1$ **then**
  return $(\mathbf{g}^t, \mathbf{x}^t)$
end if
for $j = 2, \cdots, n$ **do**
  calculate $\mathbf{c}_j \in \mathcal{R}_{2w}^k$ such that $\langle \mathbf{c}_j, \mathbf{g} \rangle = -\mathbf{\Sigma}_{1,j}$ by gadget decomposition
end for
$\mathbf{C} \leftarrow \begin{pmatrix} \mathbf{c}_2 \cdots \mathbf{c}_n \end{pmatrix}^t \in \mathcal{R}_{2w}^{(n-1)\times k}$, $\mathbf{T} \leftarrow \begin{pmatrix} \mathbf{g}^t & \mathbf{x}^t \\ \mathbf{C} & \end{pmatrix} \in \mathcal{R}_{2w}^{n \times (k\ell+4)}$
$\mathbf{\Pi} \leftarrow \left(\mathbf{\Sigma} + \mathbf{T}\mathbf{T}^t\right)_{2:n,2:n}$
$\begin{pmatrix} \mathbf{L}'_1 \cdots \mathbf{L}'_k \ \mathbf{D}' \end{pmatrix} \leftarrow \mathsf{RMIGD}(d, \mathbf{\Pi}, b, k)$ {Recursive call}
$\begin{pmatrix} \mathbf{v}'_1 \cdots \mathbf{v}'_k \end{pmatrix} \leftarrow \mathbf{C}$
$\mathbf{L}_i \leftarrow \begin{pmatrix} b^{i-1} & \\ \mathbf{v}'_i & \mathbf{L}'_i \end{pmatrix} \in \mathcal{R}_{2w}^{n \times n}$ for $i = 1, \cdots, k$
$\mathbf{D} \leftarrow \begin{pmatrix} \mathbf{x}^t & \\ & \mathbf{D}' \end{pmatrix} \in \mathcal{R}_{2w}^{n \times n(k(\ell-1)+4)}$
return $\mathbf{A} = \begin{pmatrix} \mathbf{L}_1 \cdots \mathbf{L}_k \ \mathbf{D} \end{pmatrix}$

Corollary 4. *Let $\ell, B, d, b, k \in \mathbb{N}$ and $w = 2^\ell \geq 2$ such that $m = n(k\ell + 4)$ and $d - \frac{b^{2k}-1}{b^2-1}\ell \geq b^k \geq B + knwb^2$. Then there exists a solution to $\mathsf{IGDP}_{\mathcal{R}_{2w},n,B,d,m}$ and it can be calculated by Algorithm 6.*

Lemma 13 shows a result related to the smoothness condition. Arguments in the proof of Lemma 9 still apply to the ring setting due to the similar structure of the output Gram root. The minor difference is that we should use the ℓ_2-norm to measure the "size" of each entry that is a ring element instead of an integer. Therefore we omit the proof.

Lemma 13. *Let $\mathbf{A}' = \mathsf{RMIGD}(d, \mathbf{\Sigma}, b, k) \in \mathcal{R}_{2w}^{n \times m}$ with $w = 2^\ell \geq 2$ and $m = n(k\ell + 4)$ and $\mathbf{A} = \left(\mathbf{I}_{nw} \ \mathcal{M}_w(\mathbf{A}')\right) \in \mathbb{Z}^{nw \times (n+m)w}$ where $\mathcal{M}_w$ maps each entry of $\mathbf{A}'$ to its coefficient matrix of size $w \times w$. Then*

$$\lambda_{mw}(\Lambda^\perp(\mathbf{A})) \leq \max\left\{ b^2\sqrt{nw}, \sqrt{d - \frac{b^{2k}-1}{b^2-1}\ell + b^k + 1} \right\}.$$

The idea of eigenvalue reduction (Sect. 4.2) is compatible with the ring setting as well, if one uses structure-preserving Cholesky decomposition as in [15].

Additionally, for Algorithm 5, one may also subtract some $gg^\star$ approximation from f at the beginning, and then work on a small polynomial.

6.4 Comparative Results of the Ring-Based Sampler

Combining the eigenvalue reduction and Algorithm 6, a ring-based integral decomposition is available. Based on it, one can devise a perturbation sampler for the ring case. Here we skip detailed arguments and just present some comparisons. Let us first recall the following notations:

- $\ell \in \mathbb{N}$, $w = 2^\ell$, $n \in \mathbb{N}$ and $N = nw$.
- $\mathbf{\Sigma} \in \mathcal{R}_{2w}^{n\times n}$ is a symmetric matrix over $\mathcal{R}_{2w}$ that is identified with a symmetric matrix over $\mathbb{Z}^{N\times N}$. We focus on the case of $e_1(\mathbf{\Sigma}) = \omega(N^7)$.
- $s'^2 \in \mathbb{N}$ and $s'^2 > e_1(\mathbf{\Sigma}) + 1$.
- $M \in \mathbb{N}$ such that the integral Gram root $\mathbf{A} = \begin{pmatrix}\mathbf{I}_N & \mathbf{A}'\end{pmatrix} \in \mathbb{Z}^{N\times(N+M)}$.
- $L \in \mathbb{N}$ is an upper bound of $\lambda_M(\Lambda^\perp(\mathbf{A}))$. The base samplings include $D_{\mathbb{Z},L'r}$ and $D_{\mathbb{Z},r,c}$ with $c \in \frac{1}{L'}\cdot\mathbb{Z}$, where $L' \approx L$.

Comparison with the generic sampler. Table 5 shows the comparison between the ring-based sampler and the generic one. Note that in both the generic and ring cases, the parameter $L = O(s')$ and the minimal Gaussian width $s_{min} = (1 + o(1))\sqrt{e_1(\mathbf{\Sigma})}$. Thus we do not include them in Table 5.

Table 5. Comparisons between the ring-based sampler and the generic ones.

	Storage	M
Ring, large gadget base	$\approx N(\frac{2n+1}{2}\log s' + \frac{n}{2}\log N)$	$O(Nl)$
Generic, large gadget base	$\approx N^2(\log s' + \frac{1}{2}\log N)$	$O(N)$
Ring, gadget base $b = 2$	$\approx N(\frac{4n+3}{4}\log s' + n\log N)$	$O(Nl\log s')$
Generic, gadget base $b = 2$	$\approx N^2(\log s' + \log N)$	$O(N\log s')$

As a conclusion, our ring-based integral decomposition reduces the required memory by a factor of $O(w)$ but increases the number of centered base samplings (i.e. M) by $O(\log w)$. The smoothness condition and the quality of the output Gaussian are asymptotically the same in two kinds of samplers.

Comparison with the sampler of [17]. Genise and Micciancio proposed a ring-based perturbation sampler in [17]. To generate a perturbation vector in $\mathbb{Z}^{w(2+\log q)}$, they first sample $w\log q$ integer Gaussians and then sample a Gaussian of covariance $d\mathbf{I}_2 - \mathbf{\Sigma} \in \mathcal{R}_{2w}^{2\times 2}$. To minimize the storage, the sampler only stores the matrix $\mathbf{\Sigma}$ and performs all algebraic computation on the fly.[8]

As shown in Sect. 5.4, our ring-based sampler can also reduce the procedure to the sampling of $D_{\mathbb{Z}^{2w},\sqrt{d\mathbf{I}_2-\mathbf{\Sigma}}}$ in which $n = 2$. The storage comes from the integral Gram root of $\sqrt{d\mathbf{I}_2 - \mathbf{\Sigma}}$. We summarize the comparison in Table 6.

[8] It suffices to store 3 polynomials due to the symmetry.

Table 6. Comparisons between the Genise-Micciancio sampler and ours. We use large gadget base in our sampler. We do not take into account the storage of the trapdoor itself that is $O(w \log q \log s')$.

	Storage	Time
Genise-Micciancio sampler	$\approx 6w \log s'$	$\Theta(w \log w \log q)$
Our ring-based sampler	$\approx w(5 \log s' + 2 \log w)$	$\Theta(w \log w \log q)$

The Genise-Micciancio sampler and ours require asymptotically the same memory. Particularly, if one regards the integral Gram root as a part of the trapdoor, the increase is *negligible* compared with the storage of trapdoor itself. As for running time, the costs of two samplers are dominated by the matrix multiplication of the trapdoor $\mathbf{T} \in \mathcal{R}_{2w}^{2\times \log q}$. Applying FFT or NTT techniques yields the same complexity of $\Theta(w \log w \log q)$. Nevertheless, our sampler (Theorem 3) just requires base samplings and integral polynomial multiplications. This not only gets rid of FPA, but also makes the whole algorithm much simpler and highly parallelizable.

As a conclusion, our ring-based sampler achieves the same storage and time efficiency asymptotically as the state of the art [17] but in a simpler manner.

Acknowledgements. Léo Ducas is supported by a Veni Innovational Research Grant from NWO under project number 639.021.645 and by the European Union Horizon 2020 Research and Innovation Program Grant 780701 (PROMETHEUS). Steven Galbraith is funded by the Royal Society of New Zealand, Marsden Fund project 16-UOA-144. Thomas Prest is supported by the Innovate UK Research Grant 104423 (PQ Cybersecurity). Yang Yu is funded by a French government support managed by the National Research Agency in the "Investing for the Future" program, under the national project RISQ P141580-2660001/DOS0044216, and under the project TYREX granted by the CominLabs excellence laboratory with reference ANR-10-LABX-07-01.

References

1. Agrawal, S., Boneh, D., Boyen, X.: Efficient lattice (H)IBE in the standard model. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 553–572. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13190-5_28
2. Agrawal, S., Freeman, D.M., Vaikuntanathan, V.: Functional encryption for inner product predicates from learning with errors. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 21–40. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-25385-0_2
3. Bai, S., Langlois, A., Lepoint, T., Stehlé, D., Steinfeld, R.: Improved security proofs in lattice-based cryptography: using the Rényi divergence rather than the statistical distance. In: Iwata, T., Cheon, J.H. (eds.) ASIACRYPT 2015. LNCS, vol. 9452, pp. 3–24. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48797-6_1
4. Bert, P., Fouque, P.-A., Roux-Langlois, A., Sabt, M.: Practical implementation of ring-SIS/LWE based signature and IBE. In: Lange, T., Steinwandt, R. (eds.)

PQCrypto 2018. LNCS, vol. 10786, pp. 271–291. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-79063-3_13
5. Boneh, D., et al.: Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 533–556. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-55220-5_30
6. Bourse, F., Del Pino, R., Minelli, M., Wee, H.: FHE circuit privacy almost for free. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016. LNCS, vol. 9815, pp. 62–89. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53008-5_3
7. Brakerski, Z., Vaikuntanathan, V.: Efficient fully homomorphic encryption from (standard) LWE. In: FOCS 2011, pp. 97–106 (2011)
8. Cash, D., Hofheinz, D., Kiltz, E., Peikert, C.: Bonsai trees, or how to delegate a lattice basis. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 523–552. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13190-5_27
9. Cassels, J.W.S.: Rational quadratic forms. In: North-Holland Mathematics Studies, vol. 74 (1982)
10. Chen, Y., Genise, N., Mukherjee, P.: Approximate trapdoors for lattices and smaller hash-and-sign signatures. In: Galbraith, S.D., Moriai, S. (eds.) ASIACRYPT 2019. LNCS, vol. 11923, pp. 3–32. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-34618-8_1
11. Ducas, L., Lyubashevsky, V., Prest, T.: Efficient identity-based encryption over NTRU lattices. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014. LNCS, vol. 8874, pp. 22–41. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-45608-8_2
12. Ducas, L., Micciancio, D.: Improved short lattice signatures in the standard model. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014. LNCS, vol. 8616, pp. 335–352. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44371-2_19
13. Ducas, L., Nguyen, P.Q.: Faster Gaussian lattice sampling using lazy floating-point arithmetic. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 415–432. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-34961-4_26
14. Ducas, L., Nguyen, P.Q.: Learning a zonotope and more: cryptanalysis of NTRUSign countermeasures. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 433–450. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-34961-4_27
15. Ducas, L., Prest, T.: Fast fourier orthogonalization. In: ISSAC 2016, pp. 191–198 (2016)
16. Ducas, L., Galbraith, S., Prest, T., Yu, Y.: Integral matrix gram root and lattice Gaussian sampling without floats. IACR Cryptology ePrint Archive, report 2019/320 (2019)
17. Genise, N., Micciancio, D.: Faster Gaussian sampling for trapdoor lattices with arbitrary modulus. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018. LNCS, vol. 10820, pp. 174–203. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-78381-9_7
18. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: STOC 2009, pp. 169–178 (2009)
19. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: STOC 2008, pp. 197–206 (2008)
20. Gorbunov, S., Vaikuntanathan, V., Wee, H.: Attribute-based encryption for circuits. In: STOC 2013, pp. 545–554 (2013)

21. Klein, P.N.: Finding the closest lattice vector when it's unusually close. In: SODA 2000, pp. 937–941 (2000). http://dl.acm.org/citation.cfm?id=338219.338661
22. Laguillaumie, F., Langlois, A., Libert, B., Stehlé, D.: Lattice-based group signatures with logarithmic signature size. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013. LNCS, vol. 8270, pp. 41–61. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-42045-0_3
23. Langlois, A., Ling, S., Nguyen, K., Wang, H.: Lattice-based group signature scheme with verifier-local revocation. In: Krawczyk, H. (ed.) PKC 2014. LNCS, vol. 8383, pp. 345–361. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-54631-0_20
24. Lenstra, H.W.: Lattices (2008). http://www.math.leidenuniv.nl/~psh/ANTproc/06hwl.pdf
25. Micciancio, D., Peikert, C.: Trapdoors for lattices: simpler, tighter, faster, smaller. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 700–718. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-29011-4_41
26. Micciancio, D., Peikert, C.: Hardness of SIS and LWE with small parameters. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8042, pp. 21–39. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40041-4_2
27. Micciancio, D., Regev, O.: Worst-case to average-case reductions based on Gaussian measures. SIAM J. Comput. **37**(1), 267–302 (2007)
28. Micciancio, D., Walter, M.: Gaussian sampling over the integers: efficient, generic, constant-time. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017. LNCS, vol. 10402, pp. 455–485. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63715-0_16
29. mirtich: Bug 323 - optimized code gives strange floating point results. GCC Bugzilla. https://gcc.gnu.org/bugzilla/show_bug.cgi?id=323
30. Nguyen, P.Q., Regev, O.: Learning a parallelepiped: cryptanalysis of GGH and NTRU signatures. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 271–288. Springer, Heidelberg (2006). https://doi.org/10.1007/11761679_17
31. Nguyen, P.Q., Zhang, J., Zhang, Z.: Simpler efficient group signatures from lattices. In: Katz, J. (ed.) PKC 2015. LNCS, vol. 9020, pp. 401–426. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46447-2_18
32. Peikert, C.: An efficient and parallel Gaussian sampler for lattices. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 80–97. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14623-7_5
33. Pöppelmann, T., Ducas, L., Güneysu, T.: Enhanced lattice-based signatures on reconfigurable hardware. In: Batina, L., Robshaw, M. (eds.) CHES 2014. LNCS, vol. 8731, pp. 353–370. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44709-3_20
34. Prest, T.: Sharper bounds in lattice-based cryptography using the Rényi divergence. In: Takagi, T., Peyrin, T. (eds.) ASIACRYPT 2017. LNCS, vol. 10624, pp. 347–374. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70694-8_13
35. Rabin, M.O., Shallit, J.O.: Randomized algorithms in number theory. Commun. Pure Appl. Math. **39**(S1), S239–S256 (1986)
36. Wilson, J.: Floating point trouble with x86's extended precision. The gcc@gcc.gnu.org mailing list for the GCC project. https://gcc.gnu.org/ml/gcc/2003-08/msg01195.html
37. Yu, Y., Ducas, L.: Learning strikes again: the case of the DRS signature scheme. In: Peyrin, T., Galbraith, S. (eds.) ASIACRYPT 2018. LNCS, vol. 11273, pp. 525–543. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-03329-3_18