A fully classical LLL algorithm for modules*

Gabrielle De Micheli and Daniele Micciancio

University of California, San Diego

Abstract. The celebrated LLL algorithm for Euclidean lattices is central to cryptanalysis of well-known and deployed protocols as it provides approximate solutions to the Shortest Vector Problem (SVP). Recent interest in algebrically structured lattices (e.g., for the efficient implementation of lattice-based cryptography) has prompted adapations of LLL to such structured lattices, and, in particular, to module lattices, *i.e.,* lattices that are modules over algebraic ring extensions of the integers. One of these adaptations is a *quantum* algorithm proposed by Lee, Pellet-Mary, Stehlé and Wallet (Asiacrypt 2019). In this work, we dequantize the algorithm of Lee et al., and provide a fully *classical* LLL-type algorithm for arbitrary module lattices that achieves same SVP approximation factors, single exponential in the rank of the input module. Just like the algorithm of Lee et al., our algorithm runs in polynomial time given an oracle that solves the Closest Vector Problem (CVP) in a certain, fixed lattice L_K that depends only on the number field K.

1 Introduction

The security of (public key) cryptographic protocols has always relied on the hardness of a variety of different mathematical problems. These problems are not only chosen for their presumed complexity, but often also selected for their usability and efficiency to achieve cryptographic purposes. The more recent necessity for these problems to guarantee quantum hardness has pushed cryptographers to deviate from traditional mathematical problems such as factoring and computation of discrete logarithms and focus on new mathematical objects and related computational problems. Euclidean lattices (in their general and structured versions) have emerged as a very interesting candidate due to their average-case complexity properties [2, 6, 32, 40] and support for a wide range of applications, including fully homormorphic encryption (e.g., see [5, 7, 16]). While basing cryptography on general Euclidean lattices [2, 6, 32, 40] provides a high level of security, the resulting protocols are usually quite inefficient due to their large key sizes. So, in practice, cryptographic protocols make frequent use of lattices that exhibit some (typically algebraic) structure [26,31] that allows for more compact representation of key material, as well as substantially faster operations. Widely-used mathematical problems related to these objects are for example the NTRU problem [17], RingSIS [23, 24, 31, 38], RingLWE [25,42], and their generalizations ModuleSIS and ModuleLWE [5,20] which interpolate between the standard and ring versions of the SIS (Short Integer Solution) and LWE (Learning With Errors) problems. The winner of the NIST post-quantum competition (Crystals-Kyber) for public-key encryption [1] and two out of three of the winners for digital signatures (Crystals-Dilithium and Falcon) rely on problems related to structured-lattices, such as the ones mentioned above or variants of them.

In order to assess the security of cryptographic schemes based on ideal/module variants of lattice problems, an important question to consider is whether the added algebraic structure impacts the security of these schemes. Over the past several years, significant effort has been given to understanding whether adding extra algebraic structure to a lattice introduces fundamental weaknesses, or lowers the security level by improving the performance of not only the intended cryptographic applications, but also the algorithms used for cryptanalysis.

A main research direction towards understanding the effect of added algebraic structure on the security of lattice-based protocols is to evaluate the hardness of solving the Short Vector Problem (SVP) in structured lattices. This question is crucial in cryptography as the hardness of Ring/Module- LWE or Ring/Module-SIS

* This work was supported in part by the Swiss National Science Foundation Early Postdoc.Mobility fellowship and the NSF Award 1936703.

is linked to the difficulty of finding short vectors in the lattice. A natural approach is to look at basis reduction algorithms. These are algorithms that transform a lattice basis in order to progressively produce a new basis of the same lattice with relatively short vectors. The best-known basis reduction algorithms are the LLL (Lenstra, Lenstra, Lovasz) algorithm [22] and its block BKZ (Block Korkine-Zolotarev) variant [41] given for arbitrary integer lattices. These algorithms are crucial for cryptanalysis as they provide approximate solutions to SVP, and still run in polynomial time (when the BKZ block size is sufficiently small.)

As of today, significant work has been put into adapting the LLL algorithm to structured lattices. First generalizations were proposed by Napias [35] and Fieker and Pohst [12] in 1996. It was already observed at the time that reduction algorithms obtained in specific frameworks were more efficient. The search for a module-LLL algorithm continued with Lee, Pellet-Mary, Stehlé and Wallet [21] who introduced two algorithms: an *oracle-based* generalization of LLL to modules which reduces the general basis reduction problem (on rank-n modules over the ring of integers of a number field K) to the problem of finding short vectors in module lattices of rank 2, and a *quantum* algorithm[1] that efficiently realizes the required rank-2 SVP oracle given black-box access to a Closest Vector Problem (CVP) solver for a fixed lattice L_K that depends only on the number field K. Taken together, these two components give an LLL-type quantum algorithm to find short vectors in arbitrary module lattices, assuming CVP on L_K can be efficiently solved. Due to the use of quantum computation and the need to solve CVP on L_K, the resulting algorithm is not quite usable in practice, except for fields K of very small degree. However, the algorithm of [21] is still an important theoretical step forward in understanding the potential weaknesses introduced by module lattices. On a different front, in 2019, Kirchner, Espitau and Fouque [19] proposed a general framework for lattice reduction algorithms applied to module lattices along with a usable implementation that can be used to break cryptographic schemes, such as those based on graded encodings. (See [19] for details.) This gives a very efficient variant of LLL for module lattices defined over the ring of integers of cyclotomic fields that can actually be run in practice. However, their algorithms are only heuristic and have worse approximation factors than the classical LLL. The focus of our work is to improve our understanding of algorithms (and the role of quantum computation) for module lattices that admit a rigorous theoretical analysis, and improve on the approximation factor achieved by LLL. An adaption of BKZ for module lattices based on slide-reduction was also suggested in [34]. However, their work rely on similar oracle calls when considering SVP in rank-2 modules.

One last, somehow related, line of research is that exemplified by Cramer, Ducas and Wesolowski [10] who showed that lattice problems restricted to principal ideals of some cyclotomic number fields can be quantumly solved more efficiently than for arbitrary lattices. This was generalized to all number fields with not too large discriminant (using preprocessing) by Pellet-Mary, Hanrot and Stehlé [39]. Differently from these works, which focus on the special class of principal ideal lattices, our paper deals with arbitrary ideals and (possibly non-free) modules. In fact, the primary obstacle we need to overcome to avoid quantum computation is directly related to the distinction between principal ideals, and arbitrary ideals, which in a Dedekind domain can be expressed using not one, but two generators.

Contribution and Techniques In this work, we build on the work of Lee, Pellet-Mary, Stehlé and Wallet [21] and show how essentially the same results can be obtained without using any quantum computation. The need for quantum computation in the algorithm of [21] comes from a step known as divide-and-swap which is part of their solution of SVP for rank-2 module lattices. This divide-and-swap step requires, in particular, the knowledge of the decomposition of an ideal as a product of a subset of fixed ideals. As noted in [21], if the input module is free, trivial decompositions of the input ideals are known and thus the algorithm runs in classical polynomial time. However, the same is not true for arbitrary input module lattices.

In this work, we modify their algorithm, which we call Module-LLL, in order for it to run classically, with no quantum operations, regardless of the input module. Note that their algorithm could already be dequantized using the results of [4] but at the cost of more heuristics and a classical running time of $2^{\tilde{O}(\sqrt{d})}$ (superpolynomial in the degree d of the number field) to find short vectors in rank-2 modules, even in the

[1] This is described as a heuristic in [21], but the heuristic assumptions are very mild, essentially just (plausible) mathematical conjectures on the geometric properties of a certain lattice L_K that depends only on the number field.

presence of an efficient CVP solver for L_K. Our algorithm on the other hand only requires a polynomial number of calls to CVP on the fixed lattice L_K.

In order to remove the quantum computations involved in Module-LLL, we transform the input rank-2 module to the divide-and-swap algorithm into a free rank-4 module using properties of Dedekind domains, thus allowing to run the divide-and-swap step classically. We then show how to generalize the divide-and-swap algorithm to rank-4 modules by recursively calling Module-LLL on a *free* module of fixed, constant rank 4, which, as already shown in [21], does not require any quantum computation.

We remark that our algorithm, fully-classical Module-LLL, does not make any new heuristic assumption, and only requires some very mild assumptions (inherited from [21] and [39]) on the geometric properties of L_K. It maintains a polynomial running-time and the same approximation factor as the one obtained in Module-LLL. Our main result is presented in the informal theorem below.

Theorem 1. *Let $\gamma \geq 1$ and K be a number field of degree d with discriminant Δ_K and ring of integers $\mathcal{O}_K$. There is an algorithm (Fully classical Module-LLL) that solves γ'-SVP for rank-n modules in K^n (with respect to the Euclidean norm) for approximation factor $\gamma' = \left(2\gamma\Delta_K^{1/d}\right)^{2n-1}$ running in classical polynomial time (in $\log \Delta_K$ and the bit-length of the input pseudo-basis) and with polynomially many calls to CVP in the fixed lattice L_K.*

Just as in [21], in comparing this approximation factor to LLL, it should be kept in mind that the dimension of the lattice is nd. So, LLL (or even BKZ with small enough block size to run in polynomial time) would achieve an approximation factor exponential in $\tilde{O}(nd)$. By contrast, for $\Delta_K = d^{O(d)}$, Module LLL achieves an asymptotically smaller factor, exponential in $O(n \log d)$.

At this point, the only obstacle to running Module LLL in practice remains the implementation of the CVP solver for L_K. We remark that this lattice has fairly high dimension $O(d^2)$, where d is the degree of K. So, using a generic CVP algorithm (for abitrary lattices) would still require time exponential in d^2. This should not be too much of a surprise if one notices (as explained in Section 3) that module LLL is essentially a structured version of Rankin reduction [14], and that the L_K CVP oracle is used to solve a densest sublattice problem in dimension d [11], a problem which in general seems to require even more than 2^{d^2} time. (See Section 3 for a more detailed explaination.) It is also known that there are sequences of fixed lattices (one per dimension) for which CVP is NP-hard [18,30]. So, just the fact that L_K is fixed does not imply the existence of an efficient CVP algorithm. On the other hand, there are many important and non-trivial families of lattices [27–29] for which CVP can be efficiently solved. So, the existence of an efficient CVP algorithm for L_K is an interesting possibility, and finding such algorithm is an important open problem.

2 Preliminaries

We recall in this section some necessary background on algebraic number theory and modules over Dedekind domains useful to the comprehension of this paper. We refer the reader to [8, 36], and [37, Chapter VIII] for more details on algebraic number theory, computational aspects of algebraic number theory and modules over Dedekind domains.

2.1 Algebraic background

A number field K is a finite extension of the field of rational numbers usually defined as $K = \mathbb{Q}[x]/(f(x))$ for an irreducible polynomial $f \in \mathbb{Q}[x]$ of degree d, the dimension of K as a vector space over $\mathbb{Q}$.

Let (r_1, r_2) be the signature of K, corresponding to the number r_1 of real roots, and number $2r_2$ of complex roots of f. (Complex roots always come in conjugate pairs.) All roots are distinct, so we have $r_1 + 2r_2 = d$. Each root $\alpha_i \in \mathbb{C}$ defines an embedding σ_i from K to $\mathbb{C}$ corresponding to the evaluation of polynomials at α_i. The canonical embedding allows to embed a number field K of degree d into $\mathbb{C}^d$ and is defined for any $x \in K$ as

$$\sigma(x) = (\sigma_1(x), \cdots, \sigma_d(x)) \in \mathbb{C}^d.$$

Since r_1 of the embeddings map K to $\mathbb{R}$, and the remaining ones come in conjugate pairs, the image of $\sigma(K)$ is actually contained in a real subspace $\mathbb{H} \subset \mathbb{C}^d$ of dimension d over $\mathbb{R}$. Moreover, the standard Hermitian product of $\mathbb{C}^d$ induces a real inner product on $\mathbb{H}$. So, the embedding of K into $\mathbb{H} \subset \mathbb{C}^d$ endows K with a geometry, defined by the inner product of $\mathbb{C}^d$.

Let $K_{\mathbb{R}} = K \otimes \mathbb{R}$ be the ring defined as the tensor product between K and the real numbers. The canonical embedding also gives a ring isomorphism between $K_{\mathbb{R}}$ and $\mathbb{H}$.

Let $\mathcal{O}_K$ denote the ring of integers of K. It is a free $\mathbb{Z}$-module of rank d. Through the canonical embedding, this free $\mathbb{Z}$-module can also be seen as a lattice of rank d. The absolute value of the discriminant of K is given by $\Delta_K = |\det((\sigma_i(x_j))_{i,j})|^2$ where the $(x_i)_{i \leq d}$ form a basis of $\mathcal{O}_K$.

The algebraic trace and norm of an element $\alpha \in K$ can be defined respectively as $\mathrm{Tr}(\alpha) = \sum_{i \leq d} \sigma_i(\alpha)$ and $\mathcal{N}(\alpha) = \prod_{i \leq d} \sigma_i(\alpha)$. They correspond to the trace and determinant of the endomorphism $x \mapsto \alpha x$ of K. The algebraic trace induces a Hermitian inner product over $K_{\mathbb{R}}$ where for any $x \in K_{\mathbb{R}}$, the associated Euclidean norm is defined as $||x||^2 = \sum_{1 \leq i \leq d} |\sigma_i(x)|^2$. The algebraic norm is multiplicative, *i.e.,* $\mathcal{N}(xy) = \mathcal{N}(x)\mathcal{N}(y)$ for $x, y \in K$. Note that the algebraic norm also extends to elements of $K_{\mathbb{R}}$ with the same definition.

A fractional ideal I of K is an additive subgroup of K which is stable under multiplication by any element of $\mathcal{O}_K$ and such that $xI \subseteq \mathcal{O}_K$ for some $x \in \mathcal{O}_K \setminus \{0\}$. (It may be assume without loss of generality that $x \in \mathbb{Z}$.) A non-zero ideal is a free $\mathbb{Z}$-module of rank d and can thus be seen as a lattice using the canonical embedding. These lattices are known as ideal lattices. The algebraic norm of a non-zero fractional ideal I is the determinant of the ideal I seen as a lattice in $\mathbb{R}^d$ divided by the determinant of $\mathcal{O}_K$. (*i.e.,* the determinant is normalized so that the algebraic norm of $\mathcal{O}_K$ is 1.) We have $\mathcal{N}(I) = \mathcal{N}(xI)/|\mathcal{N}(x)|$ for any $x \in \mathcal{O}_K \setminus \{0\}$ such that $xI \subseteq \mathcal{O}_K$. Moreover, for any $x \in I$, we have $|\mathcal{N}(x)| = \prod_{i \leq d} |\sigma_i(x)|$.

The ring $\mathcal{O}_K$ of algebraic integers of K is a Dedekind domain. A useful result on Dedekind domains which we will exploit in this work is the following theorem.

Theorem 2. *Let R be a Dedekind domain and let a be a non-zero element in an ideal I of R. Then there is an element $b \in R$ such that $I = (a, b)$. In particular, every ideal in R can be generated by two elements. Moreover, given a fractional ideal $I \in R$ and a non-zero element $a \in I$, there exists a probabilistic algorithm that computes $b \in I$ such that $I = (a, b)$ in expected polynomial time.*

Proof. It is a known fact that every non-zero ideal I in a Dedekind domain can be expressed uniquely as a product of prime ideals, *i.e.,* there exists prime ideals $\mathfrak{p}_i$ and $n_i \in \mathbb{Z}$ such that $I = \prod_i \mathfrak{p}_i^{e_i}$.

Consider $a \in I \setminus \{0\}$. Then, one can write $(a) = \prod_i \mathfrak{p}_i^{f_i} \prod_j \mathfrak{q}_j^{g_j}$ and since $(a) \subseteq I$, we have $e_i \leq f_i$ for all i. Note that the prime ideals $\mathfrak{q}_j$ are disjoint from the prime ideals $\mathfrak{p}_i$. Now, consider $b \in R$ such that $(b) = \prod_i \mathfrak{p}_i^{e_i} \prod_k \mathfrak{l}_k^{h_k}$ where the ideals $\prod_j \mathfrak{q}_j^{g_k}$ and $\prod_k \mathfrak{l}_k^{h_k}$ are coprime. The existence of this element comes from the Chinese Remainder Theorem (CRT). Indeed, consider an element $b_i \in \mathfrak{p}^{e_i} \setminus \mathfrak{p}^{e_i+1}$. Then CRT allows us to find the element $b \in R$ such that for each i, we have $b \equiv b_i \pmod{\mathfrak{p}_i^{e_i+1}}$ and $b \equiv 1 \pmod{\prod_j \mathfrak{q}_j^{g_j}}$. We finally have

$$(a, b) = (a) + (b) = \gcd((a), (b)) = \prod_i \mathfrak{p}_i^{\min(e_i, f_i)} = I.$$

Finally, we refer to [3, Algorithm 6.15] as an example of a probabilistic algorithm that finds two-element representations of ideals in expected polynomial time. The general idea is to arbitrarly take $a \in I$ and then to uniformly choose $b \in I \mod (a)$.

Concerning the size of the generators. Whereas the algorithm proposed in [3, Algorithm 6.15] succeeds with a high probability, it comes with the downside that the representation of the ideal $I = (a, b)$ might not be necessarily small. In [13, Theorem 3], the authors propose a modification of [3, Algorithm 6.15] where the norms of the elements a, b remain small. More particularly, they give the following result.

Theorem 3 (From [13, Theorem 3]). *Let $(s_i)_i$ be a $\mathbb{Z}$-basis of the ring of integers $\mathcal{O}_K$ of a number field K. There exists a probabilistic polynomial time algorithm that takes as inputs a $\mathbb{Z}$-basis of a non-zero fractional ideal $I \subset \mathcal{O}_K$ and a success parameter t and returns $a, b \in I$ such that $I = (a, b)$ holds with probability $1 - 2^{-t}$ and*

$$||a||, ||b|| \leq 4d^2\gamma^8 \Delta_K^{4/d} \max ||s_i||^4 \cdot \mathcal{N}(I)^{4/d},$$

where γ is the approximation factor of the lattice reduction algorithm considered (say LLL).

As a result, the bit-size representation of the ideal I is $O(\log \mathcal{N}(I) + \log \Delta_K + d(d + \log k + \log \max ||s_i||))$, where k is the smallest integer such that kI is integral and the s_i's are assumed LLL-reduced.

Remark 1. One can bound the quantity $\max ||s_i||^2 \leq d\gamma^d \Delta_K$, where $(s_i)_i$ is a reduced integral basis of K for any lattice-based reduction algorithm with approximation factor γ, see [13, Lemma 1]. Hence, the bound on $a, b \in I$ becomes

$$||a||, ||b|| \leq 4d^4 \gamma^{8+2d} \Delta_K^{(4/d)+2} \cdot \mathcal{N}(I)^{4/d}.$$

2.2 Module lattices

In [21], the authors consider basis vectors with coefficients in $K_\mathbb{R}$ and not just in K. This choice is motivated by the use of QR-factorisation and the fact that when computing the QR decomposition of a matrix with coefficients in K, the coefficients of the new matrix do not necessarily remain in K. Hence, they consider the basis vectors $\mathbf{b}_j \in K_\mathbb{R}$ and the $\mathcal{O}_K$ modules are thus defined to be in $K_\mathbb{R}$ as well.

In order to define such modules, we will need a notion of linear independence for the basis vectors. Note that because $K_\mathbb{R}$ is not an integral domain, the definition of linear independance does not directly derive from the usual equivalences between the following three statements: the determinant of the $\mathbf{b}_j$ matrix is invertible, there exist no a_i such that $\sum a_i \mathbf{b}_i = 0$ and for all j, the vector $\mathbf{b}_j$ is not in the span of the other $\mathbf{b}_i$. Hence, the strongest of these statements is used as definition, as in [21].

Definition 1 ($K_\mathbb{R}$-linear independence). *The vectors $\boldsymbol{b}_j$ for $j = 1, \cdots, n$ are said to be $K_\mathbb{R}$-linearly independent if and only if there is no non-trivial way to write the zero vector as a $K_\mathbb{R}$-linear combination of the $\boldsymbol{b}_j$.*

Now, consider a set of $K_\mathbb{R}$-linearly independent vectors $\mathbf{b}_1, \cdots, \mathbf{b}_n \in K_\mathbb{R}^m$ for $m > 0$ and $I_1, \cdots I_n$, fractional ideals of $\mathcal{O}_K$. An $\mathcal{O}_K$-module is a finitely generated set of elements closed under addition and multiplication by elements in $\mathcal{O}_K$. It can be defined as the set $M = \sum_{i \leq n} I_i \mathbf{b}_i$ and the tuple of pairs $((I_i, \mathbf{b}_i))_{i \leq n}$ is refered to as the pseudo-basis of M of rank n. The existence of such bases results from the fact that $\mathcal{O}_K$ is a Dedekind domain, see for example [37, Theorem 81:3]. In the particular case of free modules, the ideals in the pseudo-basis are all the trivial ideal $\mathcal{O}_K$. Let B denote the matrix formed by concatenating the basis vectors $\mathbf{b}_i$ for $i < n$. As defined in [21], let $\det_{K_\mathbb{R}} M = \det(\overline{B}^T B)^{1/2} \cdot \prod_i I_i$ which is an $\mathcal{O}_K$-module of rank 1 in $K_\mathbb{R}$ (as the vectors $\mathbf{b}_j \in K_\mathbb{R}$, the term $\det(\overline{B}^T B)^{1/2}$ is also an element of $K_\mathbb{R}$ and $\prod_i I_i$ is a non-zero fractional ideal).

The canonical embedding map from K to $\mathbb{C}^d$ also provides some geometrical structure to modules and thus any module in K^n can be seen as a lattice in $\mathbb{C}^{nd}$. Seen as a lattice L of dimension $n \times d$, one can also define the notion of volume of a module M as $\det M = \det L = \Delta_K^{n/2} \cdot \mathcal{N}(\det_{K_\mathbb{R}} M)$. Following the notations from [21], we define the following inner product for $\mathbf{a}, \mathbf{b} \in K_\mathbb{R}^m$

$$\langle \mathbf{a}, \mathbf{b} \rangle_{K_\mathbb{R}} = \sum_{i \in [m]} a_i \bar{b}_i \in K_\mathbb{R}.$$

This inner product is then used to extend the notion of norm to vectors $\mathbf{v} \in K_\mathbb{R}^m$. For any vector $\mathbf{v} \in K_\mathbb{R}^m$, the following norm can thus be defined: $||\mathbf{v}||_{K_\mathbb{R}} = \sqrt{\langle \mathbf{v}, \mathbf{v} \rangle_{K_\mathbb{R}}}$. Similarly, the notion of algebraic norm defined above can be extended to vectors $\mathbf{v} \in K_\mathbb{R}^m$ as $\mathcal{N}(\mathbf{v}) = \mathcal{N}(||\mathbf{v}||_{K_\mathbb{R}})$. The algebraic norm $\mathcal{N}(\mathbf{v})$ can be seen as the volume of the module of rank-1 in $K_\mathbb{R}^m$ generated by $\mathbf{v}$.

Whereas the notion of minimum distance in a Euclidean lattice usually refers to the shortest non-zero vector with respect to the Eucidean norm, in the case of module lattices, we will be interested in both the Euclidean norm and the algebraic norm. Hence, one can define the module minimum as $\lambda_1(M) = \min\{||\mathbf{v}|| : \mathbf{v} \in M \setminus \{0\}\}$ with respect to the Euclidean norm or $\lambda_1^{\mathcal{N}}(M) = \inf\{\mathcal{N}(\mathbf{v}) : \mathbf{v} \in M \setminus \{0\}\}$ with respect to the algebraic norm. The relation between both quantities is given by the following lemma.

Lemma 1 (From [21, Lemma 2.2]). *For any rank-n module M, we have:*

$$d^{-d/2}\lambda_1(M)^d \Delta_K^{-1/2} \leq \lambda_1^{\mathcal{N}}(M) \leq d^{-d/2}\lambda_1(M) \leq n^{d/2}\Delta_K^{1/2}\mathcal{N}(\det_{K_{\mathbb{R}}} M)^{1/n}.$$

We also recall the arithmetic-geometric inequality. For $\mathbf{s} \in M \setminus \{0\}$, we have $\mathcal{N}(\mathbf{s}) \leq d^{-d/2}||\mathbf{s}||^d$.

In [21], the authors also extend the Gram-Schmidt Orthogonalization and the QR-factorization to matrices over $K_{\mathbb{R}}$. Using the same notations, let $\mathbf{b}_1, \cdots, \mathbf{b}_n \in K_{\mathbb{R}}^{m\times n}$ be $K_{\mathbb{R}}$-linearly independent vectors and let $\mathbf{b}_i^*$ be the result of the extended Gram-Schmidt Orthogonalization presented in [21] for $1 \leq i \leq n$. Moreover, let $r_{ii} = ||\mathbf{b}_i^*||_{K_{\mathbb{R}}}$ for $i \leq n$, $r_{ij} = \mu_{ij} r_{ii}$ when $i < j$ and $r_{ii} = 0$ when $i > j$. The coefficients μ_{ij} are the Gram-Schmidt coefficients (we refer to [21] for details). Then $B = Q \cdot R$ where $Q \in K_{\mathbb{R}}^{m\times n}$ is the matrix defined by the column vectors $\mathbf{b}_i^*/||\mathbf{b}_i^*||_{K_{\mathbb{R}}}$ and $R = (r_{ij})_{ij}$. These notations also allow to re-write $\det_{K_{\mathbb{R}}} M = \prod_i r_{ii} I_i$, see [21, Lemma 2.6]. The aforementioned lemma also provides a lower bound on the shortest non-zero module element, *i.e.,* $\lambda_1^{\mathcal{N}}(M) \geq \min_i \mathcal{N}(r_{ii} I_i)$.

3 A classical LLL algorithm for modules

In this section we define the notion of reduced basis for module lattices, review the module LLL algorithm of [21], and describe our modification. We begin with an informal description and justification for the notion of reduced basis. Let $[\mathbf{b}_1, \ldots, \mathbf{b}_n]$ be a basis and $[\mathbf{b}_1^*, \ldots, \mathbf{b}_n^*]$ its Gram-Schmidt orthogonalization. The goal of the LLL basis reduction algorithm is, given an arbitrary basis for a lattice, find an equivalent basis such that the length of the orthogonalized vectors $\|\mathbf{b}_i^*\|$ does not decrease too fast, say $\|\mathbf{b}_i^*\| \leq \alpha \cdot \|\mathbf{b}_{i+1}^*\|$ for some factor $\alpha \geq 1$. This technical property is key to prove that the vectors in an LLL reduced basis are not too long, relative to the successive minima of the lattice. For example, since $\min_i \|\mathbf{b}_i^*\|$ is a lower bound on the first minimum λ_1 (the length of the shortest nonzero lattice vector), it follows that $\|\mathbf{b}\| \leq \alpha^{n-1} \cdot \lambda_1$, *i.e.,* the first vector in an LLL reduced basis is an approximately shortest lattice vector, with approximation factor $\gamma = \alpha^{n-1}$.

There are various generalizations of the notion of LLL reduced basis, which consider blocks of consecutive basis vectors, rather than single vectors. The one that is most relevant to the reduction of module lattices is Rankin reduction [14]. The idea is to split the lattice basis into n/d blocks $[\mathbf{B}_1, \ldots, \mathbf{B}_{n/d}]$, each consisting of d vectors,[2] and require that[3] $\det(\mathbf{B}_i^*) \leq \alpha_d \cdot \det(\mathbf{B}_{i+1}^*)$, where $\mathbf{B}_i^*$ be the projection of the ith block $\mathbf{B}_i$ orthogonal to the previous vectors $[\mathbf{B}_1, \ldots, \mathbf{B}_{i-1}]$. In other words, the determinants of the orthogonalized blocks should not decrease too quickly. Similarly to LLL reduction, this allows to show, for example, that the first block $\mathbf{B}_1 = \mathbf{B}_1^*$ is a relatively dense d-dimensional sublattice, and (by Minkowski's theorem) will contain a relatively short nonzero lattice vector. Notice how in the case of 1-dimensional blocks, the determinant of the lattice generated by $\mathbf{b}_i^*$ equals precisely the length of the vector. So, Rankin reduction generalizes the standard notion of LLL reduction to blocks of size larger than 1.

Now, moving to module lattices over a number field K degree d, we recall that each ring element $c \in \mathcal{O}_K$ (or more generally, any fractional ideal of $\mathcal{O}_K$) can be regarded (via the canonical embedding) as a d-dimensional lattice. Similarly, a module lattice of rank n over K can be seen as a lattice in dimension nd, which each coordinate (corresponding to a ring element) describing a d-dimensional sublattice. Moreover, the determinant of this sublattice is precisely the algebraic norm of the ring element (or associated fractional ideal in the cases of a pseudo-basis). So, the reduction condition requires the (orthogonalized) basis vectors (over the ring) to have algebraic norms that do not decrease too quickly. This is made more precise in the definitions given in the following subsection.

[2] More generally, one may consider blocks of different size, often alternating between two values $d_0, d_1, d_0, d_1, \ldots$, e.g., as in the case of slide reduction [15].

[3] The name "Rankin reduction" is motivated by this condition, as the worst case (over all lattices) of the normalized ratio between $\det(\mathbf{B}_i^*)$ and $\det(\mathbf{B}_{i+1}^*)$ is the so-called Rankin constant, a generalization of Hermite constant to larger block sizes.

We remark that the problem of finding the d-dimensional sublattice with smallest determinant (the so-called "densest sublattice problem") is a very hard computational problem in general. In fact, the fastest known algorithm to solve it in general has running time $d^{O(dn)}$ [11]. When $n = 2d$ (as in the case of Rankin reduction with block size d, or module LLL on rings of rank d), this is exponential in d^2, much slower than the single exponential time $2^{O(d)}$ required to solve other more standard lattice problems, like SVP and CVP. Interestingly, in the case of module lattices, the (quantum) lattice reduction algorithm of [21] only requires to make a polynomial number of calls to an oracle that solves CVP in a *fixed* lattice in dimension d^2. This is interesting on two fronts: first, this allows to solve the problem in time $2^{O(d^2)}$ (e.g., using the single exponential time algorithm of [33]), which is already slightly faster than general solutions to the densest sublattice problem requiring time $d^{O(d^2)}$. More importantly, the fact that the CVP lattice is fixed (and depends only on the number field K) leaves open the possibility that CVP in this lattice may be solvable in time much faster than $2^{O(d^2)}$, perhaps even in polynomial time.

The goal of the algorithm presented in this section is to show that an efficient solution to CVP in this fixed lattice leads not just to a quantum algorithm to compute reduced bases of module lattices (as given in [21]), but a fully classical algorithm.

3.1 A quick overview of the LLL algorithm for modules in [21]

In [21], the authors present an algorithm that efficiently finds short vectors in modules of rank n when given access to an oracle which finds short vectors in modules of rank 2. Their algorithm, which we refer to as Module-LLL in this paper, closely follows the structure of the well known LLL algorithm for classical lattices adapting the latter to the more complex algebraic structure of module lattices. A major obstacle addressed in their work is adapting Gauss' algorithm to modules of rank 2. Indeed, the classical LLL algorithm operates on 2-dimensional lattices by iteratively using a divide-and-swap algorithm to shorten basis vectors. A similar strategy is used in Module-LLL which we recall in Algorithm 1.

Algorithm 1 Module-LLL from [21]

Input: a scaled size-reduced pseudo-basis $((I_i, \mathbf{b}_i))_{i\leq n}$ of a module $M \subseteq K^m$.
Output: an LLL-reduced pseudo-basis of M.

1: **while** there exists $i < n$ such that $\alpha_K \mathcal{N}(r_{i+1,i+1} I_{i+1}) < \mathcal{N}(r_{i,i} I_i)$ **do**
2: Define M_i a rank-2 module spanned by the pseudo-basis $((I_i, \mathbf{a}_i), (I_{i+1}, \mathbf{a}_{i+1}))$ where $\mathbf{a}_i = (r_{i,i}, 0)^T$ and $\mathbf{a}_{i+1} = (r_{i,i+1}, r_{i+1,i+1})^T$.
3: Find $\mathbf{s}_i \in M_i \setminus \{0\}$ such that $\mathcal{N}(\mathbf{s}_i) \leq \gamma_{\mathcal{N}}^d \cdot \lambda_1^{\mathcal{N}}(M_i)$. ▷ Oracle-based Divide-and-swap
4: Set $\mathbf{s}_{i+1} = \mathbf{a}_i$ if it is linearly independent with $\mathbf{s}_i$ and $\mathbf{s}_{i+1} = \mathbf{a}_{i+1}$ othewise.
5: Call the algorithm of [21, Lemma 2.8] with inputs $((I_i, \mathbf{a}_i), (I_{i+1}, \mathbf{a}_{i+1}))$ and $(\mathbf{s}_i, \mathbf{s}_{i+1})$. Let $((I'_i, \mathbf{a}'_i), (I'_{i+1}, \mathbf{a}'_{i+1}))$ denote the output.
6: Update the basis ideals $I_i \leftarrow I'_i$, $I_{i+1} \leftarrow I'_{i+1}$ and the basis vectors $[\mathbf{b}_i | \mathbf{b}_{i+1}] \leftarrow [\mathbf{b}_i | \mathbf{b}_{i+1}] \mathbf{A}^{-1} \mathbf{A}'$ where $\mathbf{A} = [\mathbf{a}_i | \mathbf{a}_{i+1}]$ and $\mathbf{A}' = [\mathbf{a}'_i | \mathbf{a}'_{i+1}]$.
7: Update the current pseudo-basis by scaling the ideals (algorithm 3.2 in [21]) and size-reducing the pseudo-basis (algorithm 3.3 in [21]).
8: **end while**
9: **return** $((I_i, \mathbf{b}_i))_{i\leq n}$.

We recall the notion of LLL-reduced basis for module lattices given in [21].

Definition 2 (LLL-reducedeness of a pseudo-basis from [21, Definition 3.1]). *A module pseudo-basis $((I_i, \boldsymbol{b}_i))_{i\leq n}$ is called LLL-reduced with respect to a parameter $\alpha_K \geq 1$ if, for all $i < n$, we have*

$$\mathcal{N}(r_{i+1,i+1} I_{i+1}) \geq \frac{1}{\alpha_K} \cdot \mathcal{N}(r_{i,i} I_i),$$

where $R = (r_{i,j})_{i,j}$ refers to the R-factor of the matrix basis B defined previously.

Similar results than for the classical LLL algorithm on the length of the shortest vector in an LLL-reduced basis can be given.

Lemma 2 (From [21, Lemma 3.2]). *Assume that $((I_i, \boldsymbol{b}_i))_{i\leq n}$ is an LLL-reduced pseudo-basis of a module M. Then*

$$\mathcal{N}(I_1)\mathcal{N}(\boldsymbol{b}_1) \leq \alpha_K^{(n-1)/2} \cdot (\mathcal{N}(\det_{K_{\mathbb{R}}} M))^{1/n}$$

and

$$\mathcal{N}(I_1)\mathcal{N}(\boldsymbol{b}_1) \leq \alpha_K^{n-1} \cdot \lambda_1^{\mathcal{N}}(M).$$

Step 7 of Algorithm 1 scales and size-reduces the intermediate pseudo-bases during the execution of the algorithm. We recall these definitions.

Definition 3 (From [21, Definition 3.5]). *A pseudo-basis $((I_i, \boldsymbol{b}_i))_{i\leq n}$ with $I_i \subset K$ and $\boldsymbol{b}_i \in K_{\mathbb{R}}^m$ for all $i \neq n$ is said scaled if, for all $i \leq n$,*

$$\mathcal{O}_K \subset I_i, \quad \mathcal{N}(I_i) \geq 2^{-d^2} \Delta_K^{-1/2} \text{ and } ||r_{ii}|| \leq 2^d \Delta_K^{1/2d} \mathcal{N}(r_{ii} I_i)^{1/d}.$$

It is said size-reduced if $||r_{ij}/r_{ii}|| \leq (4d)^d \Delta_K^{1/2}$ for all $i < j < n$.

We refer to Algorithm 3.2 and Algorithm 3.3 in [21] for details.

Under the assumption there is an oracle that solves CVP in lattices L_K during the call to the divide-and-swap algorithm, Module-LLL runs in polynomial time in the input bit-length.

Theorem 4 (Adapted from [21, Theorem 3.8]). *Assume the existence of an oracle $\mathcal{O}$ used in the implementation of the divide-and-swap algorithm for some parameter $\gamma_{\mathcal{N}}$. Assume that $\alpha_K > \gamma_{\mathcal{N}}^{2d} 2^d \Delta_K$. Given as input a scaled and size-reduced pseudo-basis of a module $M \subseteq K^m$, Module-LLL presented in Algorithm 3.4 in [21] outputs an LLL-reduced pseudo-basis of M in time polynomial in $\log \Delta_K$, $1/\log(\alpha_K/(\gamma_{\mathcal{N}}^{2d} 2^d \Delta_K))$ and the input bit-length*

Module-LLL relies on quantum operations, more precisely in the divide-and-swap algorithm. The algorithm can be dequantized if a decomposition of every input ideal is known, see [21, Section 5]. In particular, if the input module is free, then all the ideals of the pseudo-basis are the trivial ideal $\mathcal{O}_K$ and thus a decomposition of them is trivially known. Hence, in this particular case, Module-LLL runs in classical polynomial time.

Because the only potentially quantum part of Module-LLL is the divide-and-swap algorithm it is actually sufficient that the input to the divide-and-swap algorithm is a free module in order to dequantize the entire algorithm. In the following section, we show how to transform a rank-2 module, input to step 3 of Algorithm 1, into a free rank-4 module and adapt Module-LLL to obtain a fully classical algorithm regardless of the initial input module.

3.2 Obtaining a free module of rank-4

Whereas $\mathcal{O}_K$ is not necessarily a Euclidean domain, it is however a Dedekind domain and thus given a pseudo-basis $((I_i, \mathbf{b}_i))_{i\leq n}$, every ideal $I_i \subset \mathcal{O}_K$ can be generated by two elements, *i.e.*, $I_i = (\alpha_i, \beta_i)$ by Theorem 2 for $\alpha_i, \beta_i \in \mathcal{O}_K$. Assume we are given a pseudo-basis $((I_1, \mathbf{b}_1), (I_2, \mathbf{b}_2))$ of a rank-2 module M_2. Moreover, we know that there exist $\alpha_1, \alpha_2, \beta_1, \beta_2 \in \mathcal{O}_K$ such that $I_1 = (\alpha_1, \beta_1)$ and $I_2 = (\alpha_2, \beta_2)$. We want to use the representation of these ideals to construct a rank-4 free module M_4.

What is a pseudo-basis of M_4? A basis of a free module is given by a pseudo-basis $((I_i, \mathbf{b}_i))$ where all the ideals I_i are the trivial ideal $\mathcal{O}_K$. We construct a rank-4 free module with the following pseudo-basis for an integer $\epsilon > 0$ which we discuss later. The pseudo-basis is given by the tuple of pairs

$$\{(\mathcal{O}_K, (\alpha_1\mathbf{b}_1, 0, 0)), (\mathcal{O}_K, (\alpha_2\mathbf{b}_2, 0, 0)), (\mathcal{O}_K, (\beta_1\mathbf{b}_1, \epsilon, 0)), (\mathcal{O}_K, (\beta_2\mathbf{b}_2, 0, \epsilon))\}.$$

The basis can be represented by the following 4×4 matrix (row-vectors):

$$B_4 = \left(\begin{array}{c|cc} \alpha_1\mathbf{b}_1 & 0 & 0 \\ \alpha_2\mathbf{b}_2 & 0 & 0 \\ \beta_1\mathbf{b}_1 & \epsilon & 0 \\ \beta_2\mathbf{b}_2 & 0 & \epsilon \end{array}\right)$$

It remains to prove that the vectors are indeed $K_\mathbb{R}$-linearly independant in $K_\mathbb{R}^m$ for $m > 0$ and for $\epsilon > 0$. Recall that because $K_\mathbb{R}$ is a ring and not a field, the vectors are said to be $K_\mathbb{R}$-linearly independant if and only if there is no non-trivial way to write the zero vector as a $K_\mathbb{R}$-linear combination of the vectors. Consider $x_i \in K_\mathbb{R}$ and the sum

$$x_1(\alpha_1\mathbf{b}_1, 0, 0) + x_2(\alpha_2\mathbf{b}_2, 0, 0) + x_3(\beta_1\mathbf{b}_1, \epsilon, 0) + x_4(\beta_2\mathbf{b}_2, 0, \epsilon) = 0$$

The vectors $\mathbf{b}_1$ and $\mathbf{b}_2$ are non-zero by construction. Indeed, the vectors $\mathbf{b}_1, \mathbf{b}_2$ are exactly the vectors given as inputs to the divide-and-swap algorithm in Algorithm 1 and are equal to $\mathbf{b}_1 = (r_{ii}, 0)$ and $\mathbf{b}_2 = (r_{i,i+1}, r_{i+1,i+1})$. We thus directly have $x_3 = x_4 = 0$ as $\epsilon > 0$. We are left with the equation $x_1\alpha_1\mathbf{b}_1 + x_2\alpha_2\mathbf{b}_2 = 0$. We also know that $((I_1, \mathbf{b}_1), (I_2, \mathbf{b}_2))$ is a pseudo-basis for M_2 and thus the vectors $\mathbf{b}_1$ and $\mathbf{b}_2$ are $K_\mathbb{R}$-linearly independent. As the generators $\alpha_1, \alpha_2 \neq 0$, we then have $x_1 = x_2 = x_3 = x_4 = 0$.

3.3 A fully-classical LLL algorithm

Module-LLL runs in classical polynomial time only if the input module has known ideal decomposition (and in particular if the input module is free). We present in this section a classical polynomial-time algorithm that takes as input *any* module without the restriction that one must know the ideal decomposition of it to run classically. In order to remove the quantum computations performed in Module-LLL, we substitute the (potentially quantum) divide-and-swap algorithm implemented in step 3 of Algorithm 1 with a call to Module-LLL itself but with a free (rank-4) module as input.

Indeed, because the rank-4 module M_4 constructed in Section 3.2 is free, one can run the Module-LLL algorithm *classically* with input the pseudo-basis of M_4 in order to extract a short vector $\mathbf{s}_i \in M_2 \setminus \{0\}$, see Section 4.1. The output of this call to Module-LLL is an LLL-reduced pseudo-basis of M_4. We provide the pseudo-code for our fully-classical Module-LLL algorithm in Algorithm 2 which heavily relies on Algorithm 3.4 in [21]. The main differences are steps 3-4 where we expand the rank-2 module M_i to a free rank-4 module $\tilde{M}_i$ and call Module-LLL classically instead of using [21, Algorithm 4.3], *i.e.,* divide-and-swap (for which we do not know a generalization to rank-4 modules).

4 Analysis of fully-classical Module-LLL

In this section, we provide an analysis of our algorithm. In particular we discuss technicalities related to the ouptut basis of Module-LLL ran on our free rank-4 module and the relation between short vectors in rank-4 modules and rank-2 modules. We also discuss the approximation factor and the running time of our fully classical Module-LLL algorithm.

Algorithm 2 Fully-classical Module-LLL

Input: a scaled size-reduced pseudo-basis $((I_i, \mathbf{b}_i))_{i\leq n}$ of a module $M \subseteq K^m$.
Output: an LLL-reduced pseudo-basis of M.

1: **while** there exists $i < n$ such that $\alpha_K \mathcal{N}(r_{i+1,i+1} I_{i+1}) < \mathcal{N}(r_{i,i} I_i)$ **do**
2: Define M_i a rank-2 module spanned by the pseudo-basis $((I_i, \mathbf{a}_i), (I_{i+1}, \mathbf{a}_{i+1}))$ where $\mathbf{a}_i = (r_{i,i}, 0)^T$ and $\mathbf{a}_{i+1} = (r_{i,i+1}, r_{i+1,i+1})^T$.
3: Expand M_i into a rank-4 free module $\tilde{M}_i$ as explained in Section 3.2.
4: Call Module-LLL (classically) with input the pseudo-basis of $\tilde{M}_i$ to extract a short vector $\mathbf{s}_i \in M_2 \setminus \{0\}$.
5: Set $\mathbf{s}_{i+1} = \mathbf{a}_i$ if it is linearly independent with $\mathbf{s}_i$ and $\mathbf{s}_{i+1} = \mathbf{a}_{i+1}$ otherwise.
6: Obtain a better basis by using the algorithm presented in [13, Theorem 4] with inputs $((I_i, \mathbf{a}_i), (I_{i+1}, \mathbf{a}_{i+1}))$ and $(\mathbf{s}_i, \mathbf{s}_{i+1})$. Let $((I'_i, \mathbf{a}'_i), (I'_{i+1}, \mathbf{a}'_{i+1}))$ denote the output.
7: Update the basis ideals $I_i \leftarrow I'_i$, $I_{i+1} \leftarrow I'_{i+1}$ and the basis vectors $[\mathbf{b}_i | \mathbf{b}_{i+1}] \leftarrow [\mathbf{b}_i | \mathbf{b}_{i+1}] \mathbf{A}^{-1} \mathbf{A}'$ where $\mathbf{A} = [\mathbf{a}_i | \mathbf{a}_{i+1}]$ and $\mathbf{A}' = [\mathbf{a}'_i | \mathbf{a}'_{i+1}]$.
8: Update the current pseudo-basis by scaling the ideals (algorithm 3.2 in [21]) and size-reducing the pseudo-basis (algorithm 3.3 in [21]).
9: **end while**
10: **return** $((I_i, \mathbf{b}_i))_{i\leq n}$.

4.1 Module-LLL with input a pseudo-basis of a free rank-4 module

Let us start by discussing Step 4 of Algorithm 2 in which we call Module-LLL (Algorithm 1) with input a pseudo-basis of a free rank-4 module. For ease of comprehension, let us consider one iteration of Algorithm 2 by fixing i and let M_2 denote the rank-2 module defined in Step 2 and M_4 the rank-4 module defined in Step 3. The module M_2 defined in Step 2 is spanned by the pseudo-basis $\{(I_1, (r_{ii}, 0)), (I_2, (r_{i,i+1}, r_{i+1,i+1}))\}$. Because M_4 is free, we know there exists a pseudo-basis for which all the ideals are equal to $\mathcal{O}_K$ and we provided such a basis in Section 3.2. The vectors of this basis are given by the lower-triangular matrix

$$B_4 = \begin{pmatrix} \alpha_1 r_{ii} & 0 & 0 & 0 \\ \alpha_2 r_{i,i+1} & \alpha_2 r_{i+1,i+1} & 0 & 0 \\ \beta_1 r_{ii} & 0 & \epsilon & 0 \\ \beta_2 r_{i,i+1} & \beta_2 r_{i+1,i+1} & 0 & \epsilon \end{pmatrix}$$

with determinant $\det B_4 = \alpha_1 \alpha_2 r_{ii} r_{i+1,i+1} \epsilon^2$ for a small value of $\epsilon > 0$ which we will discuss in Section 4.2.

Let us now consider the output of Module-LLL. The algorithm outputs a reduced pseudo-basis of M_4 of the form $\{(J_i, \mathbf{b}_{\mathrm{red}_i})\}_{i\leq 4}$ where the ideals J_i are most likely not all equal to $\mathcal{O}_K$ anymore. Indeed, the operations occuring during the execution of Module-LLL on the ideals are likely to result in a pseudo-basis where some, if not most, of the ideals are not equal to $\mathcal{O}_K$ anymore. This is the case for example in an HNF computation (and note that Module-LLL performs a BCP-HNF computation (a generalization of the Hermite Normal Form, HNF, to $\mathcal{O}_K$-modules contained in K^m, see [9, Chapter 1.4]) in Step 6 by calling the algorithm of [13, Theorem 4].)

Moreover, as we will argue in Section 4.2, there exists an upper bound on the value of ϵ for which we can predict the nature of the two first vectors of the reduced pseudo-basis. Indeed, we will see that for $\epsilon > 0$ small enough, the expected reduced pseudo-basis will have reduced basis vectors $B_{\mathrm{red}} = (\mathbf{b}_{\mathrm{red}_i})_{i\leq 4}$ of the form

$$B_{\mathrm{red}} = \begin{pmatrix} 0 & 0 & \epsilon\alpha_1 & 0 \\ 0 & 0 & 0 & \epsilon\alpha_2 \\ c_0 & c_1 & \epsilon c_2 & \epsilon c_3 \\ c_4 & c_5 & \epsilon c_6 & \epsilon c_7 \end{pmatrix}$$

for coefficients $c_i \geq 0$. Seeing that the vectors $(0, 0, \epsilon\alpha_1, 0)$ and $(0, 0, 0, \epsilon\alpha_2)$ belong to the lattice spanned by B_4 is easy. Indeed, multiplying the first vector of B_4 by $-\beta_1$ and adding to it α_1 times the third one produces the vector $(0, 0, \epsilon\alpha_1, 0)$. Similarly, adding $-\beta_2$ times the second vector of B_4 to α_2 times the fourth

produces the vector $(0,0,0,\epsilon\alpha_2)$. Both these vectors will appear in the output pseudo basis of Module-LLL if ϵ is small enough. Again, the value of ϵ for which we are garanteed that these two vectors appear as the two shortest vectors in the reduced pseudo-basis will be discussed in the next section.

Extracting a short vector of M_2. In order for Module-LLL to proceed, one needs to extract from the reduced pseudo-basis of M_4 a short vector of M_2. Indeed, Module-LLL finds a vector $\mathbf{s}_i \in M_2 \setminus \{0\}$ such that $\mathcal{N}(\mathbf{s}_i) \leq \gamma_{\mathcal{N}}^d \cdot \lambda_1^{\mathcal{N}}(M_2)$ (in step 3 of Algorithm 1) and our algorithm requires a similar short vector. We will now argue that running Module-LLL on M_4 actually finds a similar vector $\mathbf{s}_i$. As we will see in the next section, for a small enough value of ϵ, the sublattice spanned by

$$B'_{\text{red}} = \begin{pmatrix} 0 & 0 & \epsilon\alpha_1 & 0 \\ 0 & 0 & 0 & \epsilon\alpha_2 \end{pmatrix}$$

appears in the reduced pseudo-basis (as shortest vectors) of M_4. It is easy to see that if one considers the shortest vector in M_4 outputted by Module-LLL, then its projection on the rank-2 module M_2 corresponds to the zero vector (the first two coordinates). One possibility would be to choose the third vector in the reduced pseudo-basis of M_4, say $\mathbf{v}_3 = (c_0, c_1, \epsilon c_2, \epsilon c_3)$ and consider its projection $\mathbf{s}_i = (c_0, c_1) \in M_2 \setminus \{0\}$. If this projected vector can be shown to be a short vector in M_2 (by setting an appropriate value of ϵ), then the algorithm can proceed with this choice of $\mathbf{s}_i$. This would result in a classical Module-LLL with a slightly worst approximation factor.

However, one does not necessarily need to consider the third vector in the output pseudo-basis of Module-LLL on input M_4. Indeed, directly from the construction of M_4, one can note that M_2 can be obtained by projection of M_4. Moreover, we know from the properties of LLL-reduced pseudo-bases that if a pseudo-basis is LLL-reduced, then so is a projection of it. Hence running Module-LLL (classically) on M_4 will then find a short vector in M_2 as expected.

4.2 Discussion on the choice of ϵ

In this section, we provide an upper bound on the value of ϵ introduced in the construction of the free module M_4 such that for ϵ smaller than this bound, the vectors $(0,0,\epsilon\alpha_1,0)$ and $(0,0,0,\epsilon\alpha_2)$ appear as the two shortest vectors in the Module-LLL reduced pseudo-basis of M_4.

First, recall that the pseudo-basis of M_4 is given by the basis vectors represented by the matrix B_4 along with the coefficient ideals $\mathcal{O}_K$:

$$B_4 = \left(\begin{array}{c|cc} \alpha_1\mathbf{b}_1 & 0 & 0 \\ \alpha_2\mathbf{b}_2 & 0 & 0 \\ \hline \beta_1\mathbf{b}_1 & \epsilon & 0 \\ \beta_2\mathbf{b}_2 & 0 & \epsilon \end{array}\right) \begin{array}{c} \mathcal{O}_K \\ \mathcal{O}_K \\ \mathcal{O}_K \\ \mathcal{O}_K \end{array}$$

When running Module-LLL classically on M_4, we expect to obtain the reduced pseudo-basis $\{(J_i, \mathbf{b}_{\text{red}_i})\}_{i\leq 4}$ explicited in Section 4.1 for a small enough value of ϵ, *i.e.,*

$$B_{\text{red}} = \left(\begin{array}{cc|cc} 0 & 0 & \epsilon\alpha_1 & 0 \\ 0 & 0 & 0 & \epsilon\alpha_2 \\ \hline * & * & * & * \\ * & * & * & * \end{array}\right) \begin{array}{c} J_1 \\ J_2 \\ J_3 \\ J_4 \end{array}$$

Let us assume instead that the reduced pseudo-basis of M_4 has a vector basis of the form

$$B_{\text{red}} = \left(\begin{array}{cc|cc} v_{11} & v_{12} & * & * \\ v_{21} & v_{22} & * & * \\ \hline * & * & * & * \\ * & * & * & * \end{array}\right) \begin{array}{c} J_1 \\ J_2 \\ J_3 \\ J_4 \end{array}$$

with vectors $\mathbf{w}_i = (v_{i1}, v_{i2}, *, *)$ for $i = 1, 2$ and $\mathbf{v}_i = (v_{i1}, v_{i2})$ is the projection of $\mathbf{w}_i$ on the first two coordinates. In this case, the upper-left quadrant is not equal to zero. We want to provide a bound on ϵ for which this situation is not possible and we necessarily obtain an upper-left quadrant with coefficients all equal to 0.

Consider the two sublattices P_v and P_w of M_4 spanned respectively by the pseudo-bases $\{(J_i, \mathbf{v}_i)\}_{i=1,2}$ and $\{(J_i, \mathbf{w}_i)\}_{i=1,2}$ explicited just above. We know that

$$\mathcal{N}(\det_{K_\mathbb{R}}(P_v)) \leq \mathcal{N}(\det_{K_\mathbb{R}}(P_w)),$$

where the algebraic norm of these determinants can be seen as the volume of the rank-1 modules $\det_{K_\mathbb{R}}(P_v)$ and $\det_{K_\mathbb{R}}(P_w)$. The goal is to show that $\mathcal{N}(\det_{K_\mathbb{R}} P_v) = 0$, and thus prove that the coefficients in the upper-left quadrant are all equal to 0. Indeed, this would show that our sublattice P_v has volume equal to zero, and thus could only be spanned by zero-vectors.

Let us know prove that using the properties of Module-LLL, if P_w corresponds to the sublattice spanned by the two shortest vectors in the Module-LLL reduced pseudo-basis, then we have

$$\mathcal{N}(\det_{K_\mathbb{R}}(P_w)) \leq \alpha_K^2 \mathcal{N}(\det_{K_\mathbb{R}}(M_4))^{1/2}.$$

Proof. Recall that considering the QR-decomposition, one can write $\det_{K_\mathbb{R}} M_4 = \prod_{i=1}^4 r_{ii} I_i$. We would like to bound the algebraic norm of the determinant of P_w, the sublattice spanned by the first two vectors. Let $\Delta = \mathcal{N}(\det_{K_\mathbb{R}}(M_4)) = \mathcal{N}\left(\prod_i r_{ii} I_i\right)$ and $\Delta_i = \mathcal{N}(r_{ii} I_i)$ such that we have $\Delta = \prod_i \Delta_i$. From the properties of Module-LLL, we know that

$$\Delta_1 = \mathcal{N}(\mathbf{w}_1)\mathcal{N}(J_1) \leq \alpha_K^{3/2} \Delta^{1/4}.$$

If we project and remove the first vector $\mathbf{w}_1$, the pseudo-basis remains Module-LLL reduced. The first vector of this new basis is $\mathbf{w}_2$ with coefficient ideal J_2. Then again by the properties of Module-LLL, we have that

$$\Delta_2 \leq \alpha_K \left(\frac{\Delta}{\Delta_1}\right)^{1/3}.$$

Hence, by multiplying powers of these two equations we obtain

$$\Delta_1^2 \Delta_2^3 \leq \alpha_K^3 \alpha_K^3 \Delta^{1/2} \left(\frac{\Delta}{\Delta_1}\right),$$

which results in the bound

$$\Delta_1 \Delta_2 = \mathcal{N}(\det_{K_\mathbb{R}} P_w) \leq \alpha_K^2 \mathcal{N}(\det_{K_\mathbb{R}} M_4)^{1/2}.$$

Recall that M_2 is the rank-2 module with pseudo-basis with vectors $(\mathbf{b}_1, \mathbf{b}_2)$. Let us consider different cases depending on the rank of P_v.

– **If the rank is equal to 2:** note that $P_v \subseteq M_2$ is also a sublattice of M_2 of same rank in this case. Hence, we have that

$$\mathcal{N}(\det_{K_\mathbb{R}} P_v) \geq \mathcal{N}(\det_{K_\mathbb{R}} M_2).$$

Now, suppose we enforce the condition $\mathcal{N}(\det_{K_\mathbb{R}} M_2) > \alpha_K^2 \mathcal{N}(\det_{K_\mathbb{R}}(M_4))^{1/2}$. Then, since by the properties of Module-LLL, we know that $\mathcal{N}(\det_{K_\mathbb{R}}(P_w)) \leq \alpha_K^2 \mathcal{N}(\det_{K_\mathbb{R}}(M_4))^{1/2}$, this would imply that $\mathcal{N}(\det_{K_\mathbb{R}}(P_w)) < \mathcal{N}(\det_{K_\mathbb{R}} M_2)$. This last inequality leads to a contradiction since we already know that $\mathcal{N}(\det_{K_\mathbb{R}}(P_v)) \leq \mathcal{N}(\det_{K_\mathbb{R}}(P_w))$ and $\mathcal{N}(\det_{K_\mathbb{R}} P_v) \geq \mathcal{N}(\det_{K_\mathbb{R}} M_2)$. Thus, setting the condition

$\mathcal{N}(\det_{K_\mathbb{R}} M_2) > \alpha_K^2 \mathcal{N}(\det_{K_\mathbb{R}}(M_4))^{1/2}$ implies P_v cannot be of rank 2. This condition can be re-written (recalling that $\mathbf{b}_1 = (r_{ii}, 0)$ and $\mathbf{b}_2 = (r_{i,i+1}, r_{i+1,i+1})$) as

$$\begin{aligned}\mathcal{N}(\det_{K_\mathbb{R}}(M_2)) &= |\mathcal{N}(r_{ii})||\mathcal{N}(r_{i+1,i+1})|\mathcal{N}(I_1)\mathcal{N}(I_2) \\ &> \alpha_K^2 \mathcal{N}(\det_{K_\mathbb{R}} M_4)^{1/2} \\ &= \alpha_K^2 \sqrt{|\mathcal{N}(\alpha_1)| \cdot |\mathcal{N}(\alpha_2)| \cdot |\mathcal{N}(r_{ii})| \cdot |\mathcal{N}(r_{i+1,i+1})| \cdot \epsilon^{2d}}.\end{aligned}$$

which translates into the following upper bound for ϵ:

$$\epsilon < \left(\frac{1}{\alpha_K^2} \sqrt{\frac{|\mathcal{N}(r_{i,i} r_{i+1,i+1})|}{|\mathcal{N}(\alpha_1\alpha_2)|}} \mathcal{N}(I_1)\mathcal{N}(I_2) \right)^{1/d}$$

It remains to eliminate the case where P_v could be of rank 1.

– **If the rank is equal to 1:** In thise case, we know that $P_v \subseteq M_2$ is defined by a single non-zero vector $\mathbf{v} \neq 0$ along with a coefficient ideal J. We start by considering the first vector of the Module-LLL reduced pseudo-basis. We assume this first vector is of the form $\mathbf{v}_1 = (v_{11}, v_{12})$ and we want to give a bound on ϵ such that $v_{11} = v_{12} = 0$. We know that $\mathbf{v}_1 \in M_2$. We set the following bound on ϵ by imposing

$$\gamma_\mathcal{N}^d \cdot \mathcal{N}((0, 0, \epsilon\alpha_1, 0)) = \gamma_\mathcal{N}^d \cdot \epsilon^d |\mathcal{N}(\alpha_1)| \leq \lambda_1^\mathcal{N}(M_2) \leq \mathcal{N}(\mathbf{v}_1),$$

where $\gamma_\mathcal{N}^d$ is the approximation factor with respect to the algebraic norm. This would necessarily imply that $\mathbf{v}_1 = (0, 0)$. Recall that we have the lower bound $\lambda_1^\mathcal{N}(M_2) \geq \min_{i=1,2} \mathcal{N}(r_{ii} I_i)$. We can thus bound ϵ using the following inequality:

$$\gamma_\mathcal{N}^d \cdot \epsilon^d |\mathcal{N}(\alpha_1)| \leq \min\left(\mathcal{N}(r_{11} I_1), \mathcal{N}(r_{22} I_2)\right).$$

This results in the following bound for ϵ:

$$\epsilon \leq \left(\frac{\min\left(\mathcal{N}(r_{11} I_1), \mathcal{N}(r_{22} I_2)\right)}{\gamma_\mathcal{N}^d \cdot |\mathcal{N}(\alpha_1)|} \right)^{1/d}.$$

Let us know focus on the second vector $\mathbf{v}_2$. Similarly as before, we start by assuming that $\mathbf{v}_2 \neq (0, 0)$. We want to show that the sublattice P_ϵ with pseudo-basis $\{(J_1, (0, 0, \epsilon\alpha_1, 0)), (J_2, (0, 0, 0, \epsilon\alpha_2))\}$ constitute the upper part of the reduced pseudo-basis of M_4. Let us assume by contradiction that P_w with pseudo-basis $\{(J_1, (0, 0, \epsilon\alpha_1, 0)), (J_2, (v_{21}, v_{22}, *, *))\}$ is instead (assuming the previous bound on ϵ for the shortest vector). Both P_w and P_ϵ are sublattices of M_4 of rank-2. Moreover, we can lower-bound the volume of P_w by ignoring the $*$ coordinates and thus have

$$\mathcal{N}(\det_{K_\mathbb{R}} P_w) = \Delta_1 \Delta_2 \geq \epsilon^d |\mathcal{N}(\alpha_1)||\mathcal{N}(\mathbf{v}_2)||\mathcal{N}(J_1)||\mathcal{N}(J_2)|.$$

Additionally, we have

$$\mathcal{N}(\det_{K_\mathbb{R}} P_\epsilon) = \epsilon^{2d} |\mathcal{N}(\alpha_1)||\mathcal{N}(\alpha_2)||\mathcal{N}(J_1)||\mathcal{N}(J_2)|.$$

By contradiction we assumed that $\mathcal{N}(\det_{K_\mathbb{R}} P_w) < \mathcal{N}(\det_{K_\mathbb{R}} P_\epsilon)$. This implies the following inequalities

$$\epsilon^d |\mathcal{N}(\alpha_1)||\mathcal{N}(\mathbf{v}_2)||\mathcal{N}(J_1)||\mathcal{N}(J_2)| \leq \Delta_1 \Delta_2 \leq \epsilon^{2d} |\mathcal{N}(\alpha_1)||\mathcal{N}(\alpha_2)||\mathcal{N}(J_1)||\mathcal{N}(J_2)|.$$

This inequality is then equivalent to

$$\frac{|\mathcal{N}(\mathbf{v}_2)|}{|\mathcal{N}(\alpha_2)|} \leq \epsilon^d.$$

Recall that we have the bound $\mathcal{N}(\mathbf{v}_2) \geq \lambda_1^\mathcal{N}(M_2) \geq \min_i(\mathcal{N}(r_{ii} I_i))$. It then suffices to set

$$\epsilon \leq \left(\frac{\min(\mathcal{N}(r_{11} I_1), \mathcal{N}(r_{22} I_2))}{|\mathcal{N}(\alpha_2)|} \right)^{1/d}$$

to reach a contraction. We then necessarily have $\mathcal{N}(\det_{K_\mathbb{R}} P_\epsilon) \leq \mathcal{N}(\det_{K_\mathbb{R}} P_w)$ and thus $v_{21} = v_{22} = 0$.

We provided an upper bound on the ϵ value necessary for our algorithm to succeed. In order for Module-LLL to run in polynomial time not only do we need to make sure the ϵ value remains polynomially bounded, but it also has to be chosen not too small.

4.3 Approximation factor of fully-classical Module-LLL

Module-LLL from [21] gives a polynomial-time algorithm to find LLL-reduced basis of a (scaled and size-reduced) pseudo-basis for a module M assuming that step 3 (divide-and-swap) is implemented with some algorithm $\mathcal{O}$ and approximation factor γ. This algorithm $\mathcal{O}$ makes use of an oracle that solves γ-SVP in rank-2 modules. The approximation factor of Module-LLL is then given in the following result.

Theorem 5 (From [21, Theorem 3.9]). *Let $\gamma \geq 1$ and assume that an LLL-reduced $\mathbb{Z}$-basis of $\mathcal{O}_K$ is known. Then there exists a polynomial-time reduction from solving γ'-SVP in rank-n modules (w.r.t $||.||$) in K^n to solving γ-SVP in rank-2 modules in K^2 where*

$$\gamma' = \left(2\gamma \Delta_K^{1/d}\right)^{2n-1}.$$

This reduction runs in time polynomial in $\log \Delta_K$ and the bit-length of the input pseudo-basis.

Remark 2. The above theorem is given with respect to the Euclidean norm. If expressed using the algebraic norm, the oracle computes a $\gamma_{\mathcal{N}}^d = \left(\gamma \cdot \Delta_K^{1/2d}\right)^d$ approximation factor of $\lambda_1^{\mathcal{N}}(M)$ as can be seen using Lemma 1 and the arithmetic-geometric inequality. This results in a $\gamma_{\mathcal{N}}^{'d} = \gamma^{'d}\Delta_K^{1/2} = (2\gamma)^{d(2n-1)}\Delta_K^{2n-\frac{1}{2}}$ approximation factor for Module-LLL with respect to the algebraic norm.

In fully-classical Module-LLL presented in Algorithm 2, Step 4 is implemented using Module-LLL with a rank-4 module as input. However, this step of the algorithm still finds a short vector $\mathbf{s}_i \in M_2 \setminus \{0\}$, more precisely such that $\mathcal{N}(\mathbf{s}_i) \leq \gamma_{\mathcal{N}}^d \cdot \lambda_1^{\mathcal{N}}(M_2)$, as explained in Section 4.1. Hence the approximation factor of our fully-classical Module-LLL remains the same.

4.4 Running time of fully-classical Module-LLL

As stated in Theorem 4, the algorithm Module-LLL runs in polynomial time in $\log \Delta_K, 1/\log(\alpha_K/\gamma_{\mathcal{N}}^{2d}2^d\Delta_K)$ and the input bit-length given an oracle-based algorithm for the divide-and-swap algorithm and a sufficiently large α_K. In order for our algorithm to maintain a polynomial running time, one must make sure that the bitsize of the quantities introduced in fully-classical Module-LLL remain polynomially bounded. We have seen in Section 2 that the size of the generators a, b or equivalently the representation of the ideals $I = (a, b)$ have indeed bitsize polynomially bounded in terms of $\log \Delta_K$, $\log \mathcal{N}(I)$ and $\log k$, where k is the smallest positive integer such that $kI \subset \mathcal{O}_K$, see Theorem 3. Similarly, the bitsize of the value ϵ introduced in the construction of the basis of the free module is polynomially bounded in terms of $\log \mathcal{N}(I)$ and $\log \mathcal{N}(\alpha_i), \log \mathcal{N}(r_{i,j})$.

The rest of the analysis remains the same as in [21] and our algorithm runs in polynomial time in $\log \Delta_K, 1/\log(\alpha_K/\gamma_{\mathcal{N}}^{2d}2^d\Delta_K)$ and the input bit-length, just like Module-LLL.

5 Open questions and directions

In this work, we have replaced Step 3 of Module-LLL presented in [21, Algorithm 3.4] by a call to Module-LLL itself with a free module of rank 4 as input in order to run the entire algorithm classically. In Module-LLL, step 3 corresponds to the divide-and-swap algorithm which takes as input a pseudo-basis for a rank-2 module $M \subset K_{\mathbb{R}}^2$. Unfortunately, no generalization of this algorithm to higher rank modules is provided. Generalizing divide-and-swap as presented in Algorithm 3.4 in [21] to higher rank modules, for example modules of rank 4, would require among other modifications to adapt the construction of the lattice L_K.

Moreover, note that the divide-and-swap algorithm both in classical and module lattices rely on Euclidean division. In the case of modules however the ring $\mathcal{O}_K$ may not necessarily be a Euclidean domain and thus there are no trivial generalization of the Euclidean division of the integers. Algorithm 4.1 in [21] thus adapts the Euclidean division over $\mathbb{Z}$ to a Euclidean division over $\mathcal{O}_K$. The latter (heuristic) algorithm however relies on solving a CVP instance in a specific lattice L_K which only depends on the number field K, as already mentioned in the paper. Finding an efficient CVP solver for the specific lattices presented in [21] is an open question which would allow our classical Module-LLL to be run in practice.

Another interesting direction would be to obtain a fully proven LLL algorithm for modules. Indeed, both our algorithm and Module-LLL rely on heuristics (the same ones). Obtaining a fully proven algorithm would require in particular a much better understanding of particular lattices involved in the algorithm such as the log-unit lattice and the lattice of class group relations between ideals of small algebraic norms both used to construct the lattice L_K.

Finally, we know that cryptanalysis of lattice-based schemes usually rely on the block-variant of LLL, namely BKZ, for which we have trade-offs between the quality of the output and the running time. Having a module-version of BKZ would highly contribute to better analysing the security of schemes relying on structured lattices. It is however unclear whether a Module-BKZ would output a basis of same quality as classical BKZ for a fixed block-size given the constraint brought by the algebraic structure. Work in that direction was suggested in [34] where the authors propose a block-variant of Module-LLL based on slide-reduction. However, similarly as for Module-LLL, theoretical constraints still prevents the algorithm to be implemented and run for cryptographic relevant instances.

Acknowledgement

We are grateful to Alice Pellet-Mary for her insightful explanations of Module-LLL and related notions.

References

1. NIST, round 3 submissions. csrc.nist.gov/Projects/post-quantum-cryptography/round-3-submissions
2. Ajtai, M.: Generating hard instances of lattice problems (extended abstract). In: Miller, G.L. (ed.) Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996. pp. 99–108. ACM (1996). , https://doi.org/10.1145/237814.237838
3. Belabas, K.: Topics in computational algebraic number theory. Journal de théorie des nombres de Bordeaux **16**(1), 19–63 (2004)
4. Biasse, J.F., Espitau, T., Fouque, P.A., Gélin, A., Kirchner, P.: Computing generator in cyclotomic integer rings - A subfield algorithm for the principal ideal problem in $\mathrm{L}_{|\Delta_{\mathbb{K}}|}(\frac{1}{2})$ and application to the cryptanalysis of a FHE scheme. In: Coron, J.S., Nielsen, J.B. (eds.) EUROCRYPT 2017, Part I. LNCS, vol. 10210, pp. 60–88. Springer, Heidelberg (Apr / May 2017).
5. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (leveled) fully homomorphic encryption without bootstrapping. ACM Trans. Comput. Theory **6**(3), 13:1–13:36 (2014). , https://doi.org/10.1145/2633600
6. Brakerski, Z., Langlois, A., Peikert, C., Regev, O., Stehlé, D.: Classical hardness of learning with errors. In: Boneh, D., Roughgarden, T., Feigenbaum, J. (eds.) Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013. pp. 575–584. ACM (2013). , https://doi.org/10.1145/2488608.2488680
7. Brakerski, Z., Vaikuntanathan, V.: Efficient fully homomorphic encryption from (standard) LWE. SIAM J. Comput. **43**(2), 831–871 (2014). , https://doi.org/10.1137/120868669
8. Cohen, H.: A course in computational algebraic number theory, vol. 8. Springer-Verlag Berlin (1993)
9. Cohen, H.: Advanced topics in computational number theory, vol. 193. Springer Science & Business Media (2012)
10. Cramer, R., Ducas, L., Wesolowski, B.: Short stickelberger class relations and application to ideal-SVP. In: Coron, J.S., Nielsen, J.B. (eds.) EUROCRYPT 2017, Part I. LNCS, vol. 10210, pp. 324–348. Springer, Heidelberg (Apr / May 2017).
11. Dadush, D., Micciancio, D.: Algorithms for the densest sub-lattice problem. In: Khanna, S. (ed.) Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013, New Orleans, Louisiana, USA, January 6-8, 2013. pp. 1103–1122. SIAM (2013). , https://doi.org/10.1137/1.9781611973105.79

12. Fieker, C., Pohst, M.: On lattices over number fields. In: International Algorithmic Number Theory Symposium. pp. 133–139. Springer (1996)
13. Fieker, C., Stehlé, D.: Short bases of lattices over number fields. In: International Algorithmic Number Theory Symposium. pp. 157–173. Springer (2010)
14. Gama, N., Howgrave-Graham, N., Koy, H., Nguyen, P.Q.: Rankin's constant and blockwise lattice reduction. In: Dwork, C. (ed.) Advances in Cryptology - CRYPTO 2006, 26th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2006, Proceedings. Lecture Notes in Computer Science, vol. 4117, pp. 112–130. Springer (2006). , https://doi.org/10.1007/11818175_7
15. Gama, N., Nguyen, P.Q.: Finding short lattice vectors within mordell's inequality. In: Dwork, C. (ed.) Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008. pp. 207–216. ACM (2008). , https://doi.org/10.1145/1374376.1374408
16. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Mitzenmacher, M. (ed.) Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009. pp. 169–178. ACM (2009). , https://doi.org/10.1145/1536414.1536440
17. Hoffstein, J., Pipher, J., Silverman, J.H.: Ntru: A ring-based public key cryptosystem. In: International Algorithmic Number Theory Symposium. pp. 267–288. Springer (1998)
18. Khot, S., Popat, P., Vishnoi, N.K.: Almost polynomial factor hardness for closest vector problem with preprocessing. SIAM J. Comput. **43**(3), 1184–1205 (2014). , https://doi.org/10.1137/130919623
19. Kirchner, P., Espitau, T., Fouque, P.A.: Algebraic and euclidean lattices: Optimal lattice reduction and beyond. Cryptology ePrint Archive, Report 2019/1436 (2019), https://eprint.iacr.org/2019/1436
20. Langlois, A., Stehlé, D.: Worst-case to average-case reductions for module lattices. Designs, Codes and Cryptography **75**(3), 565–599 (2015)
21. Lee, C., Pellet-Mary, A., Stehlé, D., Wallet, A.: An LLL algorithm for module lattices. In: Galbraith, S.D., Moriai, S. (eds.) ASIACRYPT 2019, Part II. LNCS, vol. 11922, pp. 59–90. Springer, Heidelberg (Dec 2019).
22. Lenstra, A.K., Lenstra, H.W., Lovász, L.: Factoring polynomials with rational coefficients. Mathematische annalen **261**(ARTICLE), 515–534 (1982)
23. Lyubashevsky, V., Micciancio, D.: Generalized compact knapsacks are collision resistant. In: International Colloquium on Automata, Languages, and Programming. pp. 144–155. Springer (2006)
24. Lyubashevsky, V., Micciancio, D., Peikert, C., Rosen, A.: SWIFFT: A modest proposal for FFT hashing. In: Nyberg, K. (ed.) Fast Software Encryption, 15th International Workshop, FSE 2008, Lausanne, Switzerland, February 10-13, 2008, Revised Selected Papers. Lecture Notes in Computer Science, vol. 5086, pp. 54–72. Springer (2008). , https://doi.org/10.1007/978-3-540-71039-4_4
25. Lyubashevsky, V., Peikert, C., Regev, O.: On ideal lattices and learning with errors over rings. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 1–23. Springer, Heidelberg (May / Jun 2010).
26. Lyubashevsky, V., Peikert, C., Regev, O.: On ideal lattices and learning with errors over rings. J. ACM **60**(6), 43:1–43:35 (2013). , https://doi.org/10.1145/2535925
27. McCormick, S.T., Peis, B., Scheidweiler, R., Vallentin, F.: A polynomial time algorithm for solving the closest vector problem in zonotopal lattices. SIAM J. Discret. Math. **35**(4), 2345–2356 (2021). , https://doi.org/10.1137/20M1382258
28. McKilliam, R.G., Grant, A.J., Clarkson, I.V.L.: Finding a closest point in a lattice of voronoi's first kind. SIAM J. Discret. Math. **28**(3), 1405–1422 (2014). , https://doi.org/10.1137/140952806
29. McKilliam, R.G., Smith, W.D., Clarkson, I.V.L.: Linear-time nearest point algorithms for coxeter lattices. IEEE Trans. Inf. Theory **56**(3), 1015–1022 (2010). , https://doi.org/10.1109/TIT.2009.2039090
30. Micciancio, D.: The hardness of the closest vector problem with preprocessing. IEEE Trans. Inf. Theory **47**(3), 1212–1215 (2001). , https://doi.org/10.1109/18.915688
31. Micciancio, D.: Generalized compact knapsacks, cyclic lattices, and efficient one-way functions. Comput. Complex. **16**(4), 365–411 (2007). , https://doi.org/10.1007/s00037-007-0234-9, prelim. version in FOCS 2002.
32. Micciancio, D., Regev, O.: Worst-case to average-case reductions based on gaussian measures. SIAM J. Comput. **37**(1), 267–302 (2007). , https://doi.org/10.1137/S0097539705447360
33. Micciancio, D., Voulgaris, P.: A deterministic single exponential time algorithm for most lattice problems based on voronoi cell computations. SIAM J. Comput. **42**(3), 1364–1391 (2013). , https://doi.org/10.1137/100811970
34. Mukherjee, T., Stephens-Davidowitz, N.: Lattice reduction for modules, or how to reduce ModuleSVP to ModuleSVP. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020, Part II. LNCS, vol. 12171, pp. 213–242. Springer, Heidelberg (Aug 2020).
35. Napias, H.: A generalization of the lll-algorithm over euclidean rings or orders. Journal de théorie des nombres de Bordeaux **8**(2), 387–396 (1996)
36. Neukirch, J.: Algebraic number theory, vol. 322. Springer Science & Business Media (2013)

37. O'Meara, O.T.: Introduction to quadratic forms, vol. 117. Springer (2013)
38. Peikert, C., Rosen, A.: Efficient collision-resistant hashing from worst-case assumptions on cyclic lattices. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 145–166. Springer, Heidelberg (Mar 2006).
39. Pellet-Mary, A., Hanrot, G., Stehlé, D.: Approx-SVP in ideal lattices with pre-processing. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019, Part II. LNCS, vol. 11477, pp. 685–716. Springer, Heidelberg (May 2019).
40. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. J. ACM **56**(6), 34:1–34:40 (2009). , http://doi.acm.org/10.1145/1568318.1568324
41. Schnorr, C.P., Euchner, M.: Lattice basis reduction: Improved practical algorithms and solving subset sum problems. Mathematical programming **66**(1), 181–199 (1994)
42. Stehlé, D., Steinfeld, R., Tanaka, K., Xagawa, K.: Efficient public key encryption based on ideal lattices. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 617–635. Springer, Heidelberg (Dec 2009).