Cryptanalysis of rank-2 module-LIP in Totally Real Number Fields

Guilhem Mureau[1], Alice Pellet-Mary[1], Heorhii Pliatsok[2,3], and Alexandre Wallet[3]

[1] Univ Bordeaux, CNRS, Inria, Bordeaux INP, IMB, UMR 5251, Talence, France
guilhem.mureau@math.u-bordeaux.fr
alice.pellet-mary@math.u-bordeaux.fr
[2] Insitute of Mathematics, NAS of Ukraine
georgiipliatsok@icloud.com
[3] Univ Rennes, Inria, CNRS, Irisa, UMR 6074, France
alexandre.wallet@inria.fr

Abstract. At Asiacrypt 2022, Ducas, Postlethwaite, Pulles, and van Woerden introduced the Lattice Isomorphism Problem for module lattices in a number field K (module-LIP). In this article, we describe an algorithm solving module-LIP for modules of rank 2 in K^2, when K is a totally real number field. Our algorithm exploits the connection between this problem, relative norm equations and the decomposition of algebraic integers as sums of two squares. For a large class of modules (including $\mathcal{O}_K^2$), and a large class of totally real number fields (including the maximal real subfield of cyclotomic fields) it runs in classical polynomial time in the degree of the field and the residue at 1 of the Dedekind zeta function of the field (under reasonable number theoretic assumptions). We provide a proof-of-concept code running over the maximal real subfield of some cyclotomic fields. As a side contribution, we also provide some algorithmic and theoretical tools for the future study of the module-LIP problem.

Keywords: Module Lattices · Lattice Isomorphism Problem · Cryptanalysis.

1 Introduction

The Lattice Isomorphism Problem (LIP) is an algorithmic problem recently introduced in cryptography [15,2]. In its search version, LIP asks to find a linear *isometry*, that is, a distance-preserving linear transformation, mapping a lattice L_1 onto an *isomorphic* lattice L_2.[4] The decisional variant asks, given two lattices L_1 and L_2, to determine whether they are isomorphic. Algorithms for solving either variant of the problems have been studied independently of cryptography first [31,21]. In cryptography, the LIP problem was initially studied

[4] Lattices are geometric objects, so an isomorphism between lattices should respect the group structure *and* the geometry.

as a way to cryptanalyse the GGH and NTRU signatures [34], but the new cryptographic constructions based on LIP have motivated a lot of recent cryptanalytic works [7,27,13,12,14]. A classical approach for solving the search variant is to enumerate (possibly many) short vectors in both lattices and then to reconstruct linear isometries matching these short vectors by using a backtracking algorithm [31,16]. An alternative approach, which gives the best asymptotic complexity, is due to Haviv and Regev [21]: it finds all linear isometries between two isomorphic lattices in time $n^{O(n)}$, where n is the dimension of the lattice. To our current understanding, it seems that LIP is at least as hard as the Shortest Vector Problem (SVP), since all known algorithms solving LIP require to solve SVP first.

An algebraic variant of LIP (named module-LIP) was introduced in 2023 in [14], and used to construct the scheme Hawk, currently in evaluation in NIST's additional call for digital post-quantum signatures.[5] The security of the scheme is related to the hardness to find a given isomorphism to $\mathcal{O}_K^2$, where $\mathcal{O}_K$ is the ring of integers of a power-of-two cyclotomic field. More precisely, recovering such an isomorphism would provide a key-recovery attack on Hawk (up to an automorphism). Note that the converse is not known: there is for the moment no security proof reducing the module-LIP problem to the security of Hawk. The authors of [14] provided some analysis on the security of this new variant of LIP, based on the algorithms solving LIP in the unstructured case. In other words, so far, the only algorithms we know for solving the module-LIP variant from [14] consist in forgetting about the algebraic structure, treating the instance as a non-structured one. Generally, the cryptographic novelty of the problem means that there are few tools for cryptanalysts to understand it. The main goal of this work is to initiate the development of these tools, motivated by the following question:

Is it possible to exploit the algebraic structure of module-LIP to solve it more efficiently than LIP?

Contributions. Our main contribution is an algorithm solving module-LIP when the module $M \subset K^2$ has rank-2 and when the number field K is a totally real number field.[6] An important family of totally real fields is $K = \mathbb{Q}(\zeta + \zeta^{-1})$, with ζ a primitive root of unity, which is a degree 2 subfield of the cyclotomic field $\mathbb{Q}(\zeta)$, called its maximal totally real subfield. The precise complexity of our algorithm is related to arithmetic properties of the module. Informally, when the "gap" between the ideal generated by the coordinates of vectors in the module and the ideal generated by the norms of these vector is small, our algorithm runs in polynomial time (in the size of the input and in some field related quantities, including the residue ρ_K of the zeta function). This is the case for the important module $M = \mathcal{O}_K^2$ for which the result is stated in Corollary 4.7.

[5] https://csrc.nist.gov/Projects/pqc-dig-sig/round-1-additional-signatures

[6] Our algorithm relies on arithmetic properties of the module, and does *not* extend to modules $M \subset K_{\mathbb{R}}^2$, where $K_{\mathbb{R}} := K \otimes_{\mathbb{Q}} \mathbb{R}$.

The general complexity statement of the algorithm (for any totally real field K and any module $M \subset K^2$) is provided in Corollary 4.8. We want to stress that this result *does not* break the Hawk signature scheme. Indeed, our algorithm only works for totally real number fields, whereas the Hawk scheme is defined over a power-of-two cyclotomic field. However, our algorithm rules out a large class of possibly interesting fields, showing that using them for instantiations could result in an insecure scheme.

We provide a proof-of-concept implementation of our algorithm using a mix of `Sagemath` [36] and `PARI/GP` [35]. The code is available at https://gitlab.inria.fr/capsule/code-for-module-lip. We were able to run our algorithm successfully on module lattices isomorphic to $\mathcal{O}_K^2$, when K is the maximal totally real subfield of a cyclotomic field up to conductor 256 (which means that the lattices involved have dimension 128) — see also Table 1.

As a side contribution, we also develop some new tools and security foundations for the future study of the module-LIP problem. We extend the definition of module-LIP from [14] to cover all modules, using the notion of pseudo-bases [10] (the definition from [14] was restricted to free modules, represented by a basis). While our formulation is slightly different from the one in [14], we still have the property that solving module-LIP for our definition leads to a key recovery attack on Hawk (up to an isomorphism). We also extend standard tools for the unstructured case to the module setting, such as a worst-case to average-case reduction[7] and algorithms such as the Cholesky decomposition.

Technical details. A central tool for our definition of module-LIP is the notion of pseudo-Gram matrices. Modules usually do not admit bases but only pseudo-bases, as defined by Cohen in [10]. A pseudo-basis $\boldsymbol{B}$ of a module M consists of a matrix $B \in K^{\ell \times \ell}$ and a list of ℓ fractional ideals $I_1, \dots, I_\ell$, such that $M = \{\sum_i x_i \boldsymbol{b}_i \,|\, x_i \in I_i\}$, where $\boldsymbol{b}_i$ are the columns of B. In a natural way, we define the pseudo-Gram matrix associated to the pseudo-basis $\boldsymbol{B}$ to be $\boldsymbol{G} = (B^* B, (I_i)_{1 \leq i \leq \ell})$.

In the same way that a Gram matrix-based formulation is the better fit in the unstructured case, this basic ingredient leads us to a well-defined version of the module-LIP problem. Let $\boldsymbol{B}, \boldsymbol{C}$ be two pseudo-bases of a module M, and let $\boldsymbol{G}$ be the pseudo-Gram matrix of $\boldsymbol{C}$ (the pseudo-basis $\boldsymbol{B}$ is fixed, and will be a parameter of the problem). The module-LIP problem with parameter $\boldsymbol{B}$ (wc-smodLIP$_K^{\boldsymbol{B}}$) is, given as input $\boldsymbol{G}$, to recover a pseudo-basis $\boldsymbol{C}'$ of M such that its Gram matrix is $\boldsymbol{G}$ ($\boldsymbol{C}$ is a solution to this problem, but it may not be unique).[8] Let us explain briefly why this definition of the module-LIP problem includes the problem underlying the security of Hawk. In Hawk, the secret key is a basis C of $\mathcal{O}_K^2$, and the public key contains the Gram matrix $G = C^*C$. Since $\mathcal{O}_K^2$ is free, no ideals are involved. The secret key C is then a solution of the

[7] A structured variant of this reduction was already present in [14, Lemma 5] for cyclotomic number fields and free modules. We extend it to all number fields and modules.

[8] This definition is not formulated exactly in the same way as Definition 3.11 below, but both are equivalent.

module-LIP problem with parameter $\boldsymbol{I}_2$ (the trivial basis of $\mathcal{O}_K^2$), on input G. It can be checked that any other solution C' to this problem can be used to forge valid signatures.

The tools that we provide in Section 3 for manipulating module-LIP are generalization of the results known for non-structured lattices. This extension is easy once the formal framework is set up. For example, the worst-case to average-case reduction (Appendix A) is the adaptation to the module context of the reduction provided in [15]. The latter used a standard algorithm in lattice theory, which computes a short basis of a lattice L given as input short linearly independent vectors and a (potentially large) basis of L. A module variant of this algorithm was provided in [18], and we used it to complete our reduction.

Algorithm for module-LIP. We now explain how our algorithm for module-LIP works, when K is totally real. To simplify the description in this introduction, we will restrict ourselves to the case where the problem is parameterized by the module $\mathcal{O}_K^2$ (with basis $B = \boldsymbol{I}_2$), and where the input of the problem is a Gram matrix G (instead of a pseudo-Gram matrix $\boldsymbol{G}$). In other words, we are given as input $G \in M_2(\mathcal{O}_K)$ are we are asked to compute $C \in \mathrm{GL}_2(\mathcal{O}_K)$ a basis of $\mathcal{O}_K^2$ such that $C^*C = C^TC = G$. We are thus looking for $u, v, r, t \in \mathcal{O}_K^2$ such that

$$\begin{pmatrix} u\ v \\ r\ \ t \end{pmatrix} \cdot \begin{pmatrix} u\ r \\ v\ \ t \end{pmatrix} = \begin{pmatrix} u^2 + v^2 & ur + vt \\ ur + vt & r^2 + v^2 \end{pmatrix} = G.$$

The diagonal coefficients of G must hence be the sum of two squares in $\mathcal{O}_K$. Sums of two squares $x^2 + y^2 = a$ over $\mathbb{Z}$ are the topic of Fermat's two-squares theorem. Algorithms such as Cornacchias's allow to find all solutions of this equation efficiently, and the equation usually does not have too many solutions (this depends on the number of prime factors of a). A similar situation happens when $\mathbb{Z}$ is changed to the ring of integers $\mathcal{O}_K$ of a totally real field K.

Let us explain the principles behind the algorithm solving equations of the form $x^2+y^2 = a$ (with unknowns $x, y \in \mathcal{O}_K$). As K is totally real, it contains no square roots of -1. Thus there is a quadratic extension $L = K(i)$ where i is such that $i^2 = -1$. To any solution (x, y) of our equation corresponds the element $z := x + iy \in \mathcal{O}_L$. By definition of L, the relative norm $\mathcal{N}_{L|K}(z)$ of this element is precisely $z\bar{z} = x^2 + y^2 = a$. Hence, two-squares decompositions reduces to the computation of all solutions $z \in \mathcal{O}_L$ of $z\bar{z} = a$.[9]

To solve this norm equation $\mathcal{N}_{L|K}(z) = a$, we use the Gentry-Szydlo algorithm, as described by Lenstra and Silverberg in [24]. Given $a = z\bar{z} \in \mathcal{O}_K$ and $I = z\mathcal{O}_L$, the Gentry-Szydlo algorithms recovers z in polynomial time. To apply it in our setting, we then need to compute the ideal $I = z\mathcal{O}_L$ for all possible solutions z. To do so, we observe that $I \cdot \overline{I} = z\bar{z}\mathcal{O}_L = a\mathcal{O}_L$ is known. Hence, the ideals I we are looking for are ***principal*** divisors of the ideal $a\mathcal{O}_L$. It then only remains to factor the ideal $a\mathcal{O}_L$ and test all possible divisors I of it. The

[9] The set of $z \in \mathcal{O}_L$ with $z\bar{z} = a$ might be strictly larger than the set of solutions, since we may have $\mathcal{O}_K + i\mathcal{O}_K \subsetneq \mathcal{O}_L$, but we can easily check, given a $z = x + iy$ if (x, y) is a solution to our equation.

number of such divisors is exponential in the number of prime factors of $a\mathcal{O}_L$, so our algorithm is, also, exponential in the number of prime factors of $a\mathcal{O}_L$. These two steps are handled by our provided code.

Combining everything, we obtain an algorithm which finds all the matrices $C \in M_2(\mathcal{O}_K)$ such that $C^T C = G$ (so it solves module-LIP), but whose complexity is, for the moment, exponential in the number of prime factors of the diagonal coefficients q_1 and q_2 of the Gram matrix G. To make this algorithm polynomial time, we re-randomize the matrix G (by computing $G' = U^{-1}GU$ with $U \in \mathrm{GL}_2(\mathcal{O}_K)$, which transforms C into $C' = CU$), until its two diagonal coefficients q_1 and q_2 are prime. Under some heuristic (formalised in Assumption 1), we expect to find a good re-randomized matrix G' in polynomial time polynomial in the input size and in ρ_K, the residue of the Dedekind zeta function of K at 1.

2 Preliminaries

Notations. Vectors are by default column vectors, and are denoted by bold lower case letters (e.g., $\boldsymbol{v}$). Matrices are denoted by non-bold upper case letters (e.g., B), and pseudo-matrices by bold upper case letters (e.g., $\boldsymbol{B}$). For a ring R, we write $\mathrm{GL}_n(R)$ the set of $n \times n$ matrices whose determinant is an invertible element of R. We let $\mathcal{O}_n(\mathbb{R})$ be the set of orthogonal real matrices, $\mathcal{U}_n(\mathbb{C})$ be the set of unitary complex matrices, $\mathcal{S}_n^{>0}(\mathbb{R})$ be the set of real symmetric positive definite matrices, and $\mathcal{H}_n^{>0}(\mathbb{C})$ be the set of complex Hermitian positive definite matrices. For vectors $\boldsymbol{v}$ in $\mathbb{R}^n$ or in $\mathbb{C}^n$, we let $\|\boldsymbol{v}\|$ (resp. $\|\boldsymbol{v}\|_\infty$, $\|\boldsymbol{v}\|_1$) denote their ℓ_2 (resp. ℓ_∞, ℓ_1) norm. For a matrix $B \in \mathbb{C}^{n \times m}$ with column vectors $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_m$, we use the notation $\|B\| := \max_i \|\boldsymbol{b}_i\|$. For a field K and a subset S of a K-vector space H, we write $\mathrm{Span}_K(S)$ for the K-vector subspace of H spanned by S. We use log to refer to the natural logarithm (in base e).

2.1 Lattices

We call lattice any set of the form $L = \sum_{1 \leq i \leq r} \boldsymbol{b}_i \cdot \mathbb{Z}$, where $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n \in \mathbb{R}^m$ are $\mathbb{R}$-linearly independent vectors. It is a discrete additive subgroup of $\mathbb{R}^m$. The integer n is called the rank of L, when $n = m$ we say that the lattice has full rank. The matrix $B = (\boldsymbol{b}_1 \,|\, \ldots \,|\, \boldsymbol{b}_n)$ is called a basis of L and all bases of L are obtained by multiplication on the right by a unimodular matrix $U \in \mathrm{GL}_n(\mathbb{Z})$. For a given matrix $B \in \mathrm{GL}_n(\mathbb{R})$, we write $L(B)$ the lattice spanned by B. The determinant (or volume) of a lattice L is $\det(L) := \sqrt{\det(B^T \cdot B)}$ for any basis B of L. For $1 \leq i \leq n$, the i-th successive minimum $\lambda_i(L)$ of the lattice L is the smallest real number $r > 0$ such that the set $\{\boldsymbol{v} \in L \,|\, \|\boldsymbol{v}\| \leq r\}$ spans a real vector space of dimension at least i. We define analogously the successive minima of L in the ℓ_∞-norm. Minkowski's first theorem states that $\lambda_1^{(\infty)}(L) \leq \det(L)^{1/n}$ and $\lambda_1(L) \leq \sqrt{n} \cdot \det(L)^{1/n}$. The dual of a lattice L is the lattice $L^* = \{\boldsymbol{x} \in \mathrm{Span}_{\mathbb{R}}(L) \,|\, \langle \boldsymbol{x}, \boldsymbol{v} \rangle \in \mathbb{Z}, \forall\, \boldsymbol{v} \in L\}$.

The lattice isomorphism problem. Two lattices L, L' (with respective bases B, B') are said *isomorphic* if there exists an orthogonal transformation O such that $L' = O \cdot L$. In terms of matrices, $L(B)$ and $L(B')$ are isomorphic if and only if there exists a unimodular transformation $U \in \mathrm{GL}_n(\mathbb{Z})$ such that $B' = OBU$. The search version of the problem is precisely to find either such O or U.

Definition 2.1 (wc-sLIP^B)**.** *For a matrix $B \in \mathrm{GL}_n(\mathbb{R})$, the worst-case search Lattice Isomorphism Problem with parameter B (*wc-sLIP^B*) is, given a basis $B' \in \mathrm{GL}_n(\mathbb{R})$ of a lattice $L' \subset \mathbb{R}^n$ isomorphic to $L(B)$, to find $O \in \mathcal{O}_n(\mathbb{R})$ or $U \in \mathrm{GL}_n(\mathbb{Z})$ such that $B' = OBU$.*

For convenience, we restate LIP in terms of quadratic forms. The Gram matrix associated to a basis $B \in \mathrm{GL}_n(\mathbb{R})$ is the definite positive quadratic form $G = B^T B \in \mathcal{S}_n^{>0}(\mathbb{R})$. Two quadratic forms $Q, Q' \in \mathcal{S}_n^{>0}(\mathbb{R})$ are said *congruent* if there exists $U \in \mathrm{GL}_n(\mathbb{Z})$ such that $Q' = U^T Q U$.

Definition 2.2 (wc-sLIP^Q)**.** *For a quadratic form $Q \in \mathcal{S}_n^{>0}(\mathbb{R})$, the worst-case search LIP problem with parameter Q (*wc-sLIP^Q*) is, given any quadratic form $Q' \in \mathcal{S}_n^{>0}(\mathbb{R})$ congruent to Q, to find a unimodular transformation $U \in \mathrm{GL}_n(\mathbb{Z})$ such that $Q' = U^T Q U$.*

The two problems are polynomial-time equivalent, thanks to the Cholesky decomposition for quadratic forms.

Discrete Gaussian distributions. The Gaussian function over $\mathbb{R}^m$ with parameter $s \in \mathbb{R}_{>0}$ is $\rho_s(\boldsymbol{v}) = \exp(-\pi \cdot \|\boldsymbol{v}\|^2 / s^2)$ for all $\boldsymbol{v} \in \mathbb{R}^m$. For a rank-$n$ lattice $L \subset \mathbb{R}^m$, the discrete Gaussian distribution $\mathcal{D}_{L,s}$ over L with parameter s (and center 0) is the probability distribution defined over L by

$$\Pr_{X \sim \mathcal{D}_{L,s}}(X = \boldsymbol{v}) = \frac{\rho_s(\boldsymbol{v})}{\rho_s(L)},$$

for all $\boldsymbol{v} \in L$, where $\rho_s(L) = \sum_{\boldsymbol{v} \in L} \rho_s(\boldsymbol{v})$.

For $\varepsilon > 0$ and L a rank-n lattice, the smoothing parameter $\eta_\varepsilon(L)$ is the smallest real number $s > 0$ such that $\rho_{1/s}(L^* \setminus \{0\}) \leq \varepsilon$. Instantiating [28, Lemma 3.3] with $\varepsilon = 1/2$, we obtain the following upper bound, for any rank-n lattice L

$$\eta_{1/2}(L) \leq \sqrt{\frac{\log(6n)}{\pi}} \cdot \lambda_n(L). \tag{1}$$

Lemma 2.3 (Proof of [28, Lemma 4.4]). *Let L be a rank-n lattice, $\varepsilon > 0$ and $s \geq \eta_\varepsilon(L)$, then*

$$\rho_s(L) \in [1 - \varepsilon, 1 + \varepsilon] \cdot \frac{s^n}{\det(L)}.$$

One can efficiently sample discrete Gaussian distributions over lattices, provided the parameter s is large enough.

Lemma 2.4 (Weakening of [6, Lemma 2.3]). *There is a probabilistic polynomial time algorithm* `DiscreteGaussian` (B, s) *that takes as input a basis* B *of an* n*-dimensional lattice and a parameter* $s \geq \sqrt{\log(2n+4)/\pi} \cdot \max_i ||\boldsymbol{b}_i||$ *(where* $\boldsymbol{b}_i$ *are the columns of* B*) and returns a sample from* $\mathcal{D}_{L,s}$*.*

Note that the lower bound on s, in our statement above, is slightly larger than the one provided in [6, Lemma 2.3] (it involves the euclidean norm of the column vectors of B, instead of the Gram-Schmidt orthogonalization of B), but this will be sufficient for our purposes in this article.

Lemma 2.5 ([17, Lemma A.5]). *For any rank-*n *lattice* L *with basis* $B = (\boldsymbol{b}_i)_{1 \leq i \leq n}$*, if* $s \geq \sqrt{n} \cdot \max_i \|\boldsymbol{b}_i\|$*, then for any* $\varepsilon \in (0, 1]$ *it holds that*

$$\Pr_{\boldsymbol{v} \sim \mathcal{D}_{L,s}} \left(||\boldsymbol{v}|| > s\sqrt{4n + \log(1/\varepsilon)} \right) \leq \varepsilon.$$

2.2 Number fields

A number field K is a finite extension of the field of rational numbers $\mathbb{Q}$. Any such K is isomorphic to $\mathbb{Q}[X]/(P)$ for an irreducible monic polynomial $P \in \mathbb{Q}[X]$. The degree of P is exactly the degree of the extension. Furthermore, any extension K of degree d comes naturally with d embeddings $K \to \mathbb{C}$, sending the class of X to a complex root of P. An embedding $\sigma : K \to \mathbb{C}$ such that $\sigma(K) \subset \mathbb{R}$ is called a real embedding. An embedding σ which is not real is called complex. In this case, it can be composed with complex conjugation and gives another distinct complex embedding $\overline{\sigma}$. We denote by d_1 the number of real embeddings and d_2 the number of complex embeddings up to complex conjugation, so that $d = d_1 + 2d_2$. We order the embeddings $\sigma_1, \cdots, \sigma_d$ such that σ_i is a real embedding for $1 \leq i \leq d_1$ and $\sigma_{i+d_2} = \overline{\sigma_i}$ for $d_1 < i \leq d_1 + d_2$. When $d_1 = d$ (resp. $2d_2 = d$), we say that the extension $K|\mathbb{Q}$ is totally real (resp. totally imaginary).

The space $K_{\mathbb{R}}$*.* The $\mathbb{R}$-algebra $K_{\mathbb{R}} := K \otimes_{\mathbb{Q}} \mathbb{R}$ is a real vector space of dimension d. If $K \sim \mathbb{Q}[X]/(P)$, then we have an identification $K_{\mathbb{R}} \sim \mathbb{R}[X]/(P)$. The embeddings of K can be uniquely extended to $K_{\mathbb{R}}$, which leads to the so-called canonical embedding $\sigma(x) = (\sigma_1(x), \ldots, \sigma_d(x)) \in \mathbb{C}^d$. This map defines a ring homomorphism from $K_{\mathbb{R}}$ to $\sigma(K_{\mathbb{R}})$, which is the d-dimensional real vector space

$$\sigma(K_{\mathbb{R}}) := \left\{ z = (z_i)_i \in \mathbb{C}^d \,|\, z_1, \ldots, z_{d_1} \in \mathbb{R},\ \overline{z_{d_1+i}} = z_{d_1+d_2+i},\ 1 \leq i \leq d_2 \right\}.$$

To any $z \in K_{\mathbb{R}}$ we associate its complex conjugate $\bar{z} := \sigma^{-1}(\overline{\sigma(z)}) \in K_{\mathbb{R}}$, where $\overline{\sigma(z)}$ consists in taking the complex conjugation of $\sigma(z) \in \mathbb{C}^d$ coordinate-wise. We let $K_{\mathbb{R}}^{+}$ (resp. $K_{\mathbb{R}}^{++}$) be the subset of $K_{\mathbb{R}}$ corresponding to elements with all non-negative (resp. positive) embeddings (in particular, the complex embeddings are all real numbers). The norm map defined over $K_{\mathbb{R}}$ is $\mathcal{N}_K(z) = \prod_i \sigma_i(z)$, and extends the well-known notion of algebraic norm for elements of K. Similarly, the trace map is $\mathrm{Tr}_K(z) = \sum_i \sigma_i(z)$. When there is no ambiguity, we drop the subscript. If $z \in K$ (resp. $\mathcal{O}_K$), then $\mathcal{N}(z), \mathrm{Tr}(z) \in \mathbb{Q}$ (resp. $\mathbb{Z}$).

Another important notion of size for $K_\mathbb{R}$ is the corresponding "T_2-norm" $\|z\|^2 := \|\sigma(z)\|^2 = \mathrm{Tr}(z\overline{z}) \in \mathbb{R}^+$.

We extend the canonical embedding to vectors of $K_\mathbb{R}^k$ and matrices in $K_\mathbb{R}^{k \times k'}$ by applying it coordinate-wise. We say that r vectors $\boldsymbol{b}_1, \cdots, \boldsymbol{b}_r$ in $K_\mathbb{R}^k$ are ($K_\mathbb{R}$-)linearly independent if there exists no non-zero r-tuple $\boldsymbol{x} = (x_1, \cdots, x_r) \in K_\mathbb{R}^r \setminus \{\boldsymbol{0}\}$ such that $\sum_i x_i \boldsymbol{b}_i = 0$. Note that, since $K_\mathbb{R}$ is not a field, this notion can be interesting even for $r = 1$ vector: a vector $\boldsymbol{b} \in K_\mathbb{R}^k$ is linearly independent if and only if $\sigma_i(\boldsymbol{b}) \neq \boldsymbol{0}$ for all embeddings σ_i. For a matrix $B \in K_\mathbb{R}^{k \times k'}$, we let $B^* = \overline{B}^T$; this covers vectors (k or k' is 1). We let $\mathcal{U}_k(K_\mathbb{R})$ denote the group of unitary matrices of $K_\mathbb{R}$, that is the matrices $B \in \mathrm{GL}_k(K_\mathbb{R})$ such that $B^* \cdot B = I_k$. Thanks to the injectivity of the canonical embedding, it holds that B is in $\mathcal{U}_k(K_\mathbb{R})$ if and only if $\sigma_i(B) \in \mathcal{U}_k(\mathbb{C})$ for all embeddings σ_i. For $\boldsymbol{v} = (v_i)_i$ and $\boldsymbol{w} = (w_i)_i \in K_\mathbb{R}^k$ (for some $k \geq 1$), we define

$$\langle \boldsymbol{v}, \boldsymbol{w} \rangle_{K_\mathbb{R}} := \boldsymbol{v}^* \boldsymbol{w} = \sum_i \overline{v_i} \cdot w_i \in K_\mathbb{R}.$$

This extends the T_2-norm to $K_\mathbb{R}^k$ as $\|\boldsymbol{v}\|^2 := \mathrm{Tr}\, \langle \boldsymbol{v}, \boldsymbol{v} \rangle_{K_\mathbb{R}} = \sum_i \mathrm{Tr}(v_i \overline{v_i}) = \|\sigma(\boldsymbol{v})\|^2$.

Ring of integers. We denote by $\mathcal{O}_K$ the ring of integers of a number field K. The ring $\mathcal{O}_K$ is a free $\mathbb{Z}$-module of rank d. The discriminant of K is defined by $(\det(\sigma_i(e_j))_{i,j})^2 \in \mathbb{Z}$, where $(e_i)_{1 \leq i \leq d}$ is any $\mathbb{Z}$-basis of $\mathcal{O}_K$ (this does not depend on the choice of the basis). Its absolute value is denoted Δ_K. There exists some absolute constant c such that $\Delta_K \geq c^d$ for all number fields K. In particular, we always have $d = \mathrm{poly}(\log \Delta_K)$.

A special case that will be of interest for us in Section 4.3 is when K is totally real and $L = K[X]/(X^2+1)$. In this case, we have $\Delta_L \mid 4^d \Delta_K^2$. This can be obtained using the formula $\Delta_L = \mathcal{N}_K(\Delta_{L|K}) \cdot \Delta_K^{[L:K]}$ and the fact that $\Delta_{L|K}$ divides $\mathrm{disc}(X^2+1) \cdot \mathcal{O}_K = 4 \cdot \mathcal{O}_K$ (see [33, III, §4, Proposition 8] and [33, III, §2, Corollary to Proposition 5] with lattices $X = \mathcal{O}_L$ and $X' = \mathcal{O}_K[i]$). In particular for these extensions we always have $\log \Delta_L = \mathrm{poly}(\log \Delta_K)$. Moreover, we also know how to compute $\mathcal{O}_L$ efficiently from $\mathcal{O}_K$, as stated in the following lemma.

Lemma 2.6 (Specialization of [9, Theorem 1.2]). *Let K be a totally real number field and $L := K[X]/(X^2+1)$. There exists a polynomial time algorithm $\mathcal{A}$ that, given as input a $\mathbb{Z}$-basis B_K of $\mathcal{O}_K$, computes a $\mathbb{Z}$-basis B_L of $\mathcal{O}_L$.*

Proof. We apply Buchmann-Lenstra algorithm to the order $\mathcal{O} = \mathcal{O}_K[X]/(X^2+1)$ of L, with basis $B = B_0 \cup X \cdot B_0$. One can prove that in our situation, the index $[\mathcal{O}_L : \mathcal{O}]$ is a power of two so it is enough to make it 2-maximal, i.e., we can take $m = 2$ in input of the algorithm. □

Residue. We write ρ_K the residue of the zeta function of K at $s = 1$. This is a positive number, which is always $\leq O(\log \Delta_K)^d$ [25, Theorem 1]. In some number fields, such as when K is a cyclotomic fields, we have better bounds, and we know that $\rho_K = \mathrm{poly}(d)$ [5, Theorem A.5]. Unfortunately, we are not aware of a proven polynomial bound for the maximal totally real subfield of a cyclotomic field (for these fields, the best bound we found was $\rho_K = O(1.31^d)$ [37, Lemma 11.5]). However, in practice, in all our experiments with maximal totally real subfield of a cyclotomic field, we observed that the residue ρ_K was quite small.

Ideals. An integral ideal $\mathfrak{a}$ is an additive subgroup of $\mathcal{O}_K$ such that $x \cdot \mathfrak{a} \subset \mathfrak{a}$ for all $x \in \mathcal{O}_K$. Equivalently, it is an $\mathcal{O}_K$-module contained in $\mathcal{O}_K$.[10] An ideal generated by a single element a is called principal, and is denoted by $a\mathcal{O}_K$. A fractional ideal I of K is any additive subgroup of K of the form $x \cdot \mathfrak{a}$ where $x \in K \setminus \{0\}$ and $\mathfrak{a} \subset \mathcal{O}_K$ is an integral ideal. We use fraktur lower-case letters for integral ideals (e.g., $\mathfrak{a}$, $\mathfrak{b}$) and upper-case letters for fractional ideals (e.g., I, J). The product of two fractional ideals I and J is the smallest ideal containing all products xy for $x \in I$ and $y \in J$, denoted by IJ. This operation turns the set of non-zero fractional ideals into a multiplicative group. Given two fractional ideals I and J, we have that $I \subseteq J$ if and only if there exists an integral ideal $\mathfrak{a}$ such that $I = J \cdot \mathfrak{a}$. When this is the case, we say that J divides I, and we may write $J|I$. An integral ideal $\mathfrak{p}$ is prime when $\mathfrak{p} \neq \{0\}$ and $ab \in \mathfrak{p}$ implies $a \in \mathfrak{p}$ or $b \in \mathfrak{p}$ — such ideals are maximal in $\mathcal{O}_K$. We have unique factorization of integral ideals into prime ideals (up to permutation of the factors).

The algebraic norm of an integral ideal is $\mathcal{N}(\mathfrak{a}) = |\mathcal{O}_K/\mathfrak{a}|$. It is multiplicative, extends to fractional ideals as $\mathcal{N}(I) = \mathcal{N}(dI)/\mathcal{N}(d)$ for any d such that dI is integral. If $I = a\mathcal{O}_K$ is principal, then $\mathcal{N}(I) = |\mathcal{N}(a)|$.

Modules. In this article, an ($\mathcal{O}_K$-)module M will refer to a subset of $K_{\mathbb{R}}^{\ell}$ of the form $\boldsymbol{b}_1 I_1 + \cdots + \boldsymbol{b}_r I_r$, where the I_i are non-zero fractional ideals of K and $(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_r)$ are $K_{\mathbb{R}}$-linearly independent vectors of $K_{\mathbb{R}}^{\ell}$, for some $\ell > 0$. The integer r is called the rank of the module. When $r = \ell$, we say that the module has full rank. When $M \subset (\mathcal{O}_K)^{\ell}$ (resp. $M \subset K^{\ell}$), we say M is an integer module (resp. a rational module). We say that $\boldsymbol{B} = (B, \{I_i\}_{1 \leq i \leq r})$ is a pseudo-basis for M, where $B \in K_{\mathbb{R}}^{\ell \times r}$ is the matrix whose columns are the $\boldsymbol{b}_i' s$. Two pseudo-bases $\boldsymbol{B} = (B, \{I_i\}_{1 \leq i \leq r})$ and $\boldsymbol{B}' = (B', \{J_i\}_{1 \leq i \leq r})$ represent the same module if and only if there exists $U = (u_{i,j})_{1 \leq i,j \leq r} \in \mathrm{GL}_r(K)$ such that $B' = BU$ and $u_{i,j} \in I_i J_j^{-1}$, $v_{i,j} \in J_i I_j^{-1}$, where $V = (v_{i,j})_{1 \leq i,j \leq l} = U^{-1}$.[11] Given a module M and a non-zero fractional ideal J, we define $J \cdot M$ to be the smallest module containing all $\alpha \cdot \boldsymbol{v}$ for $\alpha \in J$ and $\boldsymbol{v} \in M$. If $\boldsymbol{B} = (B, \{I_i\}_{1 \leq i \leq r})$ is a pseudo-basis of M, then $\boldsymbol{B}' = (B, \{J \cdot I_i\}_{1 \leq i \leq r})$ is a pseudo-basis of $J \cdot M$.

If $M \subset K_{\mathbb{R}}^{\ell}$ is a module of rank r, then the set $\sigma(M) := \{\sigma(\boldsymbol{v}) \,|\, \boldsymbol{v} \in M\}$ is a *module* lattice, that is, a lattice of rank dr in the real space $\sigma(K_{\mathbb{R}})^{\ell}$ equipped with the (extended) T_2-norm. Since we always use the canonical embedding to view modules as lattices in this work, we will simplify notations and write $\lambda_i(M)$ and $\det(M)$ instead of $\lambda_i(\sigma(M))$ and $\det(\sigma(M))$.

[10] Note that $\mathcal{O}_K$ is a Dedekind ring. In particular, it is Noetherian, so all ideals are finitely generated as $\mathcal{O}_K$-module.

[11] When $M \subset K^l$ is a rational module, these are the conditions for U to be a pseudo-base change matrix, see [10].

A special case of module lattices is when $M = I$ is a non-zero fractional ideal of K. In this case, we know that $\det(I) = \sqrt{\Delta_K} \cdot \mathcal{N}(I)$ (see e.g., [30, Chapter I. Proposition 5.2]). We also know that the successive minima of $\sigma(I)$ cannot be very unbalanced. More precisely, we have

$$\lambda_d(I) \leq \Delta_K^{1/d} \cdot \lambda_1(I) \leq \sqrt{d} \cdot \Delta_K^{3/(2d)} \cdot \mathcal{N}(I)^{1/d}. \tag{2}$$

The second inequality follows from Minkowski's first theorem, and the equality $\det(I) = \Delta_K^{1/2} \cdot \mathcal{N}(I)$. This first inequality can be obtained by choosing $x \in I$ reaching $\lambda_1(I)$ (i.e., such that $\|\sigma(x)\| = \lambda_1(I)$) and $\{\alpha_1, \dots, \alpha_d\}$ linearly independent elements of $\mathcal{O}_K$ reaching the successive minima in infinity norm (i.e., such that $||\sigma(\alpha_i)||_\infty = \lambda_i^{(\infty)}(\mathcal{O}_K)$). Then $\{\alpha_1 x, \dots, \alpha_d x\}$ are $\mathbb{Z}$-linearly independent elements of I thus $\lambda_d(I) \leq \max_i ||\sigma(\alpha_i x)|| \leq ||\sigma(x)|| \cdot \lambda_d^\infty(\mathcal{O}_K)$. The desired upper bound is then obtained by the fact that $\lambda_d^{(\infty)}(\mathcal{O}_K) \leq \Delta_K^{1/d}$ (see [3] for the asymptotic result and [5, Theorem A.4] for the upper bound with an explicit constant).

In the general case, the volume of a module M of rank r with pseudo-basis $(B, (I_i)_i)$ is $\det(M)^2 = \Delta_K^r \cdot \det(B^*B) \cdot \prod_{i=1}^r \mathcal{N}(I_i)^2$.

CM extensions. A CM (complex multiplication) extension is any quadratic extension $L|K$ of number fields such that K is totally real and L is totally imaginary. In the rest of this article, we will always use the convention that the degree of K is d, and the degree of L is $2d$. A typical example to keep in mind is when L is a cyclotomic field and K is its maximal totally real subfield.

Any CM extension $L|K$ is Galois, and so L has two field automorphisms fixing K point-wise, namely the identity and another automorphism which we will call τ. The automorphism τ somehow plays the role of a complex conjugation on L, as can be seen in the following lemma.

Lemma 2.7. *Let $L|K$ be a CM extension of number fields and σ_i be an embedding of L in $\mathbb{C}$. Then $\sigma_i(\tau(x)) = \overline{\sigma_i(x)}$ for all $x \in L$.*

Proof. First, observe that the composition map $\sigma_i \circ \tau$ is an embedding of L. Moreover, since τ is the identity on K, then it must be that $\sigma_i \circ \tau = \sigma_i$ on K, which restricts it to be either σ_i or $\overline{\sigma_i}$. Finally, τ is not the identity on L, and σ_i is injective, hence we cannot have $\sigma_i \circ \tau = \sigma_i$ and we conclude that $\sigma_i \circ \tau = \overline{\sigma_i}$ as desired. □

For an element $x \in L$, the relative norm of x is defined as $\mathcal{N}_{L|K}(x) := x \cdot \tau(x)$, which is an element of K (since it is fixed by τ). We also define similarly the relative norm of a fractional ideal I of L: $\mathcal{N}_{L|K}(I) := I \cdot \tau(I) \cap K$, where $\tau(I) := \{\tau(x) \,|\, x \in I\}$ is also a fractional ideal of L. Finally, if $I = x\mathcal{O}_L$ is a principal ideal, then it holds that $\mathcal{N}_{L|K}(I) = \mathcal{N}_{L|K}(x) \cdot \mathcal{O}_K$.

Splitting of prime ideals in CM extensions. Let $\mathfrak{p}$ be a prime ideal in $\mathcal{O}_K$. Then, $\mathfrak{p}\mathcal{O}_L$ is an integral ideal of $\mathcal{O}_L$ and one of the three situations holds for its factorization into prime ideals (see *e.g.*, [30, Chap 2. Proposition 8.2])

$$\mathfrak{p}\mathcal{O}_L = \begin{cases} \mathfrak{q} & \text{with } \mathfrak{q} \text{ prime and } \mathcal{N}_{L|K}(\mathfrak{q}) = \mathfrak{p}^2 \text{ (inert case)}, \\ \mathfrak{q}_1\mathfrak{q}_2 & \text{with } \mathfrak{q}_1 \neq \mathfrak{q}_2 \text{ prime and } \mathcal{N}_{L|K}(\mathfrak{q}_i) = \mathfrak{p} \text{ (split case)}, \\ \mathfrak{q}^2 & \text{with } \mathfrak{q} \text{ prime and } \mathcal{N}_{L|K}(\mathfrak{q}) = \mathfrak{p} \text{ (ramified case)}. \end{cases} \tag{3}$$

This implies in particular the following result.

Lemma 2.8. *Let $\mathfrak{q}$ be a prime ideal of $\mathcal{O}_L$ and $\mathfrak{p}$ be a prime ideal of $\mathcal{O}_K$. If $\mathfrak{p}$ divides $\mathcal{N}_{L|K}(\mathfrak{q})$, then $\mathfrak{q}$ divides $\mathfrak{p}\mathcal{O}_L$.*

Proof. Since $\mathfrak{q}$ is prime, then $\mathfrak{a} = \mathfrak{q} \cap \mathcal{O}_K$ is a prime ideal of $\mathcal{O}_K$. Moreover, $\mathfrak{a}\mathcal{O}_L \subseteq \mathfrak{q}$, i.e., $\mathfrak{q}$ divides $\mathfrak{a}\mathcal{O}_L$. Equation (3) gives us all the possibilities for the factorization of $\mathfrak{a}\mathcal{O}_L$ into prime ideals, from which we conclude that $\mathcal{N}_{L|K}(\mathfrak{q}) = \mathfrak{a}$ or $\mathcal{N}_{L|K}(\mathfrak{q}) = \mathfrak{a}^2$. This implies that $\mathfrak{p} = \mathfrak{a}$, and concludes the proof. □

2.3 Algorithmic considerations

This section collects various computational results. In this article, we assume a model of computation where computers have access to infinite precision real numbers. Most of the time, we will work with elements of K which we can represent with rational numbers, and for which we define a notion of size. When we need to work with complex numbers, we assume that we have enough precision so that all computations are correct.

Representation of ideals and modules. We assume in this article that we are always given an LLL reduced basis $\alpha_1, \ldots, \alpha_d$ of $\mathcal{O}_K$, i.e., such that $(\sigma(\alpha_i))_i$ is an LLL-reduced basis of $\mathcal{O}_K$. Elements in K (resp. $K_\mathbb{R}$) are represented by their coordinates in the basis $(\alpha_1, \ldots, \alpha_d)$, which is a vector in $\mathbb{Q}^d$ (resp $\mathbb{R}^d$). For $x \in K$ represented by the vector $(x_1, \ldots, x_d) \in \mathbb{Q}^d$, we define $\text{size}(x) := \sum_i \text{size}(x_i)$, where the size of a rational number a/b with a and b coprime is $\lceil \log_2 |a| \rceil + \lceil \log_2 |b| \rceil$. Since $(\alpha_i)_i$ is LLL-reduced for the canonical embedding, [18, Lemma 2] states that if $x = \sum_i x_i \alpha_i \in K$, then $\max_i |x_i| \leq 2^{3d/2} \cdot \|\sigma(x)\|$. This implies in particular that for any integral $x \in \mathcal{O}_K$, $\text{size}(x) = \text{poly}(d, \|\sigma(x)\|)$. Inversely, we have $\|\sigma(x)\| \leq \sum_i |x_i| \cdot \|\sigma(\alpha_i)\| \leq d^{3/2} \cdot 2^d \cdot \Delta_K^{1/d} \cdot \max_i |x_i|$ (where we used the fact that $\lambda_d(\mathcal{O}_K) \leq \sqrt{d} \cdot \Delta_K^{1/d}$). This implies that for any rational $x \in K$, we have $\|\sigma(x)\| \leq \text{poly}(\log \Delta_K, \text{size}(x))$.

A fractional ideal I is represented by a $\mathbb{Z}$-basis $(y_1, \ldots, y_d)$ of the ideal, such that $(\sigma(y_i))_i$ is an LLL-reduced basis of $\sigma(I)$. In particular, we have

$$\|\sigma(y_i)\| \leq 2^d \cdot \lambda_d(I) \leq \sqrt{d} \cdot 2^d \cdot \Delta_K^{3/(2d)} \cdot \mathcal{N}(I)^{1/d}, \tag{4}$$

where we used Inequality (2). We define $\text{size}(I) := \sum_i \text{size}(y_i)$. If $\mathfrak{a}$ is an integral ideal, then (4) shows that $\text{size}(\mathfrak{a}) = \text{poly}(\log \Delta_K, \log \mathcal{N}(\mathfrak{a}))$. By default, in the rest of this article, whenever an algorithm takes as input, manipulates, or returns an ideal, we implicitly assume that the ideal is represented as described above.

For vectors and matrices with coefficients in K, we define their size as the sum of the size of their coordinates. Modules in K^ℓ are represented by a pseudo-basis $\boldsymbol{B} = (B, (I_i))$, where B is a matrix with coefficients in K and the I_i's are fractional ideals, represented by LLL-reduced bases as discussed above. We define $\mathrm{size}(\boldsymbol{B}) := \mathrm{size}(B) + \sum_i \mathrm{size}(I_i)$.

Basic algorithms for number fields. If $x, y \in K$, then $\mathrm{size}(x \cdot y) \leq \mathrm{poly}(\mathrm{size}(x), \mathrm{size}(y), \log(\Delta_K))$, where we used the relations between $\mathrm{size}(x)$ and $\|\sigma(x)\|$ mentioned above, and the fact that $\|\sigma(x \cdot y)\| \leq \|\sigma(x)\| \cdot \|\sigma(y)\|$. Such a product can be computed in time $\mathrm{poly}(\mathrm{size}(x), \mathrm{size}(y), \log(\Delta_K))$.

With the representation of ideals as described above, one can multiply two ideals I and J in time $\mathrm{poly}(\mathrm{size}(I), \mathrm{size}(J), \log \Delta_K)$. Indeed, if $(x_i)_i$ is an LLL reduced basis of I and $(y_i)_i$ is an LLL reduced basis of J, then $(x_i \cdot y_i)_{i,j}$ is a generating set of $I \cdot J$, and one can run the LLL algorithm on this generating set to extract an LLL-reduced basis from it.

Lemma 2.9. *Let $\boldsymbol{B} = (B, \{I_i\}_i)$ be a pseudo-basis of a rank r module M in K^ℓ. Then one can compute in polynomial time a basis $C \in \mathbb{C}^{d\ell \times dr}$ of $\sigma(M)$ such that the column vectors $\boldsymbol{c}_i$ of C satisfy*

$$\|\boldsymbol{c}_i\| \leq \sqrt{d} \cdot 2^d \cdot \Delta_K^{3/(2d)} \cdot \max_{1 \leq j \leq r} \left(\|\sigma(\boldsymbol{b}_j)\| \cdot \mathcal{N}(I_j)^{1/d} \right),$$

where $\boldsymbol{b}_j$ is the j-th column of B.

Proof. Let $(\alpha_i^{(j)})_i$ be the LLL reduced basis representing the ideal I_j for all $1 \leq j \leq r$ and $\boldsymbol{b}_i$ be the columns of the matrix B. The basis C of $\sigma(M)$ is composed of the vectors $\boldsymbol{c}_{d(j-1)+i} := \sigma(\alpha_i^j \cdot \boldsymbol{b}_j)$, for $1 \leq j \leq r$ and $1 \leq i \leq d$. These vectors can be computed in polynomial time from B and the $\alpha_i^{(j)}$'s. The upper bound on their size follows from the upper bound $\|\sigma(\alpha_i^{(j)})\| \leq \sqrt{d} \cdot 2^d \cdot \Delta_K^{3/(2d)} \cdot \mathcal{N}(I_j)^{1/d}$ (see Equation (4)). □

The computation of roots of unity is handled by the following lemma.

Lemma 2.10 (Factoring polynomials over a number field [1]). *There is a polynomial time algorithm that given a number field K and a polynomial $P \in K[X]$, factorizes P in K.*

Corollary 2.11 (Computing roots of unity in a number field). *Let K be a degree d number field. Then, K has at most $2d^2$ roots of unity and there is a polynomial time algorithm that given a basis of $\mathcal{O}_K$, computes the roots of unity in K.*

Proof. Let K be a number field of degree K. By Dirichlet's unit theorem, we know that the group $\mathbb{U}_K$ of roots of unity in K is finite and cyclic. Let ζ be a generator of this group, and m be its order. Since K contains $\mathbb{Q}(\zeta)$, and $\mathbb{Q}(\zeta)$ has degree $\varphi(m)$, it should be that $\varphi(m) \leq d$. Also, we remark that for all $n \geq 2$, it holds that $\varphi(n) \geq \sqrt{n/2}$ (this is obtained by observing that $p - 1 \geq \sqrt{p}$

for all prime $p \geq 3$ and $p - 1 \geq \sqrt{p}/\sqrt{2}$ for $p = 2$, and using the fact that $\varphi(\prod_i p_i^{e_i}) = \prod_i (p_i - 1)p_i^{e_i - 1})$. Hence, the number m of roots of unity in K satisfies $m \leq 2\varphi(m)^2 \leq 2d^2$. Then to compute $\mathbb{U}_K$, it is enough to factor the polynomials $X^k - 1$ over K, for $1 \leq k \leq 2d^2$. This can be done in polynomial time using the algorithm from Lemma 2.10.

We note that the knowledge of a basis of $\mathcal{O}_K$ is used here because we said that elements of K are represented as rational linear combination of a given basis of $\mathcal{O}_K$. If one chooses a different representation for elements of K (for example, polynomials in $\mathbb{Q}[X]/P(X)$ with P a defining polynomial of K), the knowledge of this basis of $\mathcal{O}_K$ might not be needed. □

Factoring ideals. Given a prime integer $p \in \mathbb{Z}$, we can compute all prime ideals of $\mathcal{O}_K$ above p. This can be done using Buchmann-Lenstra's algorithm [8], whose detailed analysis can be found in Cohen's book [11, Section 6.2.5].

Lemma 2.12 ([11, Section 6.2.5]). *There exists a polynomial time algorithm that takes as input any prime integer $p \in \mathbb{Z}$ and a basis of the ring of integers $\mathcal{O}_K$ of a number field K, and computes all the prime ideals of $\mathcal{O}_K$ dividing $p \cdot \mathcal{O}_K$.*

As a corollary of this lemma, we have the following two algorithms, to test primality of ideals, and factor them.

Corollary 2.13. *There is a polynomial time algorithm `PrimalityTest`$(\mathfrak{a})$ that takes as input any integral ideal $\mathfrak{a}$ and decides whether $\mathfrak{a}$ is a prime ideal or not.*

There is a polynomial time algorithm `FactorIdeal`$(\mathfrak{a})$ that takes as input an integral ideal $\mathfrak{a}$ and the factorization of $\mathcal{N}(\mathfrak{a}) \in \mathbb{Z}$, and returns the factorisation of $\mathfrak{a}$ into prime ideals.

Proof. To verify whether $\mathfrak{a}$ is a prime ideal or not, we first compute its algebraic norm $\mathcal{N}(\mathfrak{a})$ and check if it a prime power. This can be done by computing roots $\mathcal{N}(\mathfrak{a})^{1/e}$ (for $1 \leq e \leq \log|\mathcal{N}(\mathfrak{a})|$) with good enough precision and see if the result is a prime integer. If we do have $\mathcal{N}(\mathfrak{a}) = p^e$ for some prime p and $e \geq 1$, then using the previous lemma we compute the prime ideals $p \cdot \mathcal{O}_K$. For each of them, we test for equality with $\mathfrak{a}$.

To factorize an integral ideal $\mathfrak{a}$, the first step is to factor its algebraic norm $\mathcal{N}(\mathfrak{a}) = \prod_{1 \leq i \leq r} p_i^{e_i}$, which is done in time $T_{\text{factor}}(\mathcal{N}(\mathfrak{a}))$. For each prime factor p_i we compute the splitting of $p_i \cdot \mathcal{O}_K$ with the preceding lemma. There are at most $d = [K : \mathbb{Q}]$ prime ideals above each p_i and for each of them, say $\mathfrak{p}_{i,j}$ (a prime ideal of $\mathcal{O}_K$ above p_i) we test if $\mathfrak{p}_{i,j}$ divides $\mathfrak{a}$. If it does, we check if powers $\mathfrak{p}_{i,j}^e$ of this prime factor still divide $\mathfrak{a}$ (see that the largest e such $\mathfrak{p}_{i,j}^e | \mathfrak{a}$ is smaller than e_i). All together, the number of divisions is at most $r \cdot d \cdot \max_i\{e_i\} = \text{poly}(\log|\mathcal{N}(\mathfrak{a})|)$. □

Let us write $T_{\text{factor}}(N)$ is the time needed to factor an integer $N \in \mathbb{Z}_{>0}$. Then Algorithm `FactorIdeal` can be used to factor an integral ideal $\mathfrak{a}$ in time $\text{poly}(\text{size}(\mathfrak{a})) + T_{\text{factor}}(\mathcal{N}(\mathfrak{a}))$.

Norm equations in CM fields. Let us first consider a simple case of norm equation, which will be useful later on.

Lemma 2.14. *Let $L|K$ be a CM extension of number fields with K of degree d. The units $u \in \mathcal{O}_L^\times$ that satisfy $\mathcal{N}_{L|K}(u) = 1$ are exactly the roots of unity in L.*

Proof. Let $u \in \mathcal{O}_L^\times$ be a unit such that $\mathcal{N}_{L|K}(u) = u\tau(u) = 1$. Because u is an algebraic integer, its minimal polynomial μ_u over $\mathbb{Q}$ has coefficients in $\mathbb{Z}$. By Lemma 2.7, we have $|\sigma_i(u)|^2 = \sigma_i(u \cdot \tau(u)) = 1$ for all $2d$ complex embeddings σ_i of L. Since the $\sigma_i(u)$'s are exactly the complex roots of μ_u (appearing possibly multiply times), this implies that all the roots of μ_u over $\mathbb{C}$ have modulus 1. A theorem of Kronecker[12] implies u and its conjugates are roots of unity. □

The Gentry-Szydlo algorithm [20] was originally presented in a cyclotomic field K and allows one to recover in polynomial-time an element $x \in K$, given as input a basis of the ideal $x\mathcal{O}_K$ and the element $x\bar{x}$.[13] A generalization of this algorithm to CM-fields (and more generally CM-orders) was proposed by Lenstra and Silverberg in [24].

Theorem 2.15 (Specialization of [24, Theorem 1.3]). *There exists a deterministic polynomial time algorithm* `GentrySzydlo` *such that the following holds. Let $L|K$ be a CM extension of number fields, $w \in L$ and I be a fractional ideal of L. Given as input a basis of $\mathcal{O}_L$, the ideal I and the element w, algorithm* `GentrySzydlo` *decides whether there exists $v \in \mathcal{O}_L$ such that $I = v \cdot \mathcal{O}_L$ and $v\bar{v} = w$, and if so computes such an element v.*

We note that Theorem 1.3 from [24] is stated for an arbitrary CM order A. We specialized the statement to $A = \mathcal{O}_L$, which is a CM order (see Example 3.7 (i) from [24]). With this specialization, the $\mathbb{Q}$-algebra $A_{\mathbb{Q}}$ (using the notations from [24]) is the field L, and the set $(A_{\mathbb{Q}}^+)_{\gg 0}$ is the set $L \cap L_{\mathbb{R}}^{++}$. In the original statement from [24], the element w is taken in this set $(A_{\mathbb{Q}}^+)_{\gg 0}$. We note that in our case, if $w \in L$ does not have all its embeddings real and positive, then we know that there is no solution to the equation $v\bar{v} = w$, and so this case is easily discarded. For this reason, we did not restrict the statement to specific elements w. Similarly, we did not impose in the statement that the ideal I satisfies $I\bar{I} = w\mathcal{O}_L$, because this condition can be tested in polynomial time, and the two equations $I = v \cdot \mathcal{O}_L$ and $v\bar{v} = w$ can be satisfied only if $I\bar{I} = (v\bar{v})\mathcal{O}_L = w\mathcal{O}_L$.

With this theorem, one obtains the following algorithm (Algorithm 2.1), which solves norm equations in CM extensions. This is a generalization to any CM field extensions of an algorithm by Howgrave-Graham and Szydlo [22] which solves relative norm equations in some cyclotomic number fields (over their totally real subfields). Note that the algorithm by Howgrave-Graham and Szydlo

[12] Let $P \in \mathbb{Z}[X]$ be a monic polynomial such that $P(0) \neq 0$. If all the complex roots of P have modulus less or equal to 1, then these are roots of unity.

[13] The element x is recovered up to a root of unity.

was published in 2004, before the generalization of the Gentry-Szydlo algorithm to all CM orders by Lenstra and Silverberg. This explains why Howgrave-Graham and Szydlo only mention cyclotomic fields in [22], but the generalization to all CM fields is immediate once we have the generalization of the Gentry-Szydlo algorithm for such fields. A result similar to Theorem 2.16 below and its proof are provided in [23, Theorem 14]. Since the latter article does not seem to be published, we provide a proof for completeness.

Algorithm 2.1 Solve relative norm equations (`NormEquation`)

```
Input: A basis of $\mathcal{O}_K$ and $\mathcal{O}_L$, an element $q \in \mathcal{O}_K$, the prime factorization of $|\mathcal{N}_K(q)|$.
Output: All elements $z \in \mathcal{O}_L$ such that $\mathcal{N}_{L|K}(z) = q$.
 1: Compute the roots of unity $\mathbb{U}_L$ in $L$, using Corollary 2.11.
 2: Factor $q \cdot \mathcal{O}_K = \prod_{i=1}^r \mathfrak{p}_i^{\alpha_i}$, using Corollary 2.13.
 3: $\mathfrak{I}_0 \leftarrow \{\mathfrak{q} \subset \mathcal{O}_L \text{ prime ideal} \mid \exists\, i \in \{1, \dots, r\},\ \mathfrak{q} \mid \mathfrak{p}_i \mathcal{O}_L\}$
 4: $\mathfrak{I} \leftarrow \{\mathfrak{a} = \mathfrak{q}_1 \dots \mathfrak{q}_s \mid s > 0, \mathfrak{q}_i \in \mathfrak{I}_0, \mathcal{N}_{L|K}(\mathfrak{a}) = q \cdot \mathcal{O}_K\}$
 5: $S \leftarrow \{\}$
 6: for $\mathfrak{a} \in \mathfrak{I}$ do
 7:     $z_\mathfrak{a} \leftarrow$ GentrySzydlo($\mathcal{O}_L, \mathfrak{a}, q$) (see Theorem 2.15)
 8:     if GentrySzydlo did not fail then
 9:         $S \leftarrow S \cup \{\zeta \cdot z_\mathfrak{a} \mid \zeta \in \mathbb{U}_L\}$
10:     end if
11: end for
12: return $S$
```

Theorem 2.16. *Let $L|K$ be a CM extension of number fields and $q \in \mathcal{O}_K$. Given as input a basis of $\mathcal{O}_K$, a basis of $\mathcal{O}_L$, the element $q \in \mathcal{O}_K$, and the factorization of $|\mathcal{N}_K(q)|$ over $\mathbb{Z}$, Algorithm* 2.1 *(`NormEquation`) computes the list of all elements $z \in \mathcal{O}_L$ such that $\mathcal{N}_{L|K}(z) = q$. Moreover, the algorithm runs in time* $\mathrm{poly}(\log \Delta_L, \log|\mathcal{N}_K(q)|) \cdot (1 + \log|\mathcal{N}_K(q)|)^r$*, where r is the number of distinct prime factors of the ideal $q \cdot \mathcal{O}_K$.*

Note that the size of the output of the algorithm, and so in particular the number of solutions to the equation $\mathcal{N}_{L|K}(z) = q$, is bounded by the running time of the algorithm.

Proof. Correctness. Let us first show that the set $\mathfrak{I}$ from Step 4 contains all integral ideals of $\mathcal{O}_L$ with relative norm $q \cdot \mathcal{O}_K$. To do so, consider an integral ideal $\mathfrak{b} \subset \mathcal{O}_L$ with relative norm $q \cdot \mathcal{O}_K$ and write down its factorization $\mathfrak{b} = \prod_{j=1}^s \mathfrak{q}_j^{\beta_j}$ in $\mathcal{O}_L$. Since the relative norm is multiplicative, we have $\prod_{j=1}^s \mathcal{N}_{L|K}(\mathfrak{q}_j)^{\beta_j} = q \cdot \mathcal{O}_K = \prod_{i=1}^r \mathfrak{p}_i^{\alpha_i}$ (using the notations from Step 1). By uniqueness of the factorization into prime ideals, we deduce that for any $j \in \{1, \dots, s\}$, there exists $i_j \in \{1, \dots, r\}$ such that $\mathfrak{p}_{i_j}$ divides $\mathcal{N}_{L|K}(\mathfrak{q}_j)$. By Lemma 2.8, this implies that $\mathfrak{q}_j \mid \mathfrak{p}_{i_j}$. This means that $\mathfrak{b}$ has all its prime factors in $\mathfrak{I}_0$, and so $\mathfrak{b}$ is in $\mathfrak{I}$ (by definition of $\mathfrak{I}$), as desired. In particular, for any $z \in \mathcal{O}_L$ such that $z\bar{z} = q$, the corresponding principal ideal $z\mathcal{O}_L$ is in $\mathfrak{I}$.

In Step 7, the Gentry-Szydlo algorithm outputs, when it exists, a generator of $\mathfrak{a}$ with relative norm q for an ideal $\mathfrak{a} \in \mathfrak{I}$ (it always succeeds when a solution exists). Let us fix a principal ideal $\mathfrak{a}$ that has a generator $z_{\mathfrak{a}}$ of relative norm q. Let $z'_{\mathfrak{a}}$ be another generator with the same relative norm, so there is a unit u such that $z_{\mathfrak{a}} = uz'_{\mathfrak{a}}$. But then we have $\mathcal{N}_{L|K}(u) = q/q = 1$. By Lemma 2.14, we conclude that $u \in \mathbb{U}_L$ is a root of unity. Conversely, for any root of unity $\zeta \in \mathbb{U}_L$, one can check that $z_{\mathfrak{a}} \cdot \zeta$ is a generator of $\mathfrak{a}$ with relative norm q. We then conclude that the generators of $\mathfrak{a}$ whose relative norm is q are exactly the elements $\{\zeta \cdot z_{\mathfrak{a}} \,|\, \zeta \in \mathbb{U}_L\}$. This concludes the correctness of the algorithm.

Complexity. The roots of unity $\mathbb{U}_L$ can be computed in polynomial time thanks to Corollary 2.11. Recall that the factorization of $\mathcal{N}_K(q)$ is known. For Step 2 of the Algorithm, one can use `FactorIdeal` from Corollary 2.13 to factor the ideal $q\mathcal{O}_K$ in polynomial time, thanks to the knowledge of the factorization of $\mathcal{N}_K(q)$. In Step 3, computing all the prime ideals dividing $\mathfrak{p}_i\mathcal{O}_L$ for a fixed prime ideal $\mathfrak{p}_i$ can be done in time polynomial in $\log \mathcal{N}(\mathfrak{p}_i)$ and $\log \Delta_L$ thanks to Lemma 2.12. Note that each prime factor $\mathfrak{p}_i$ of $q\mathcal{O}_K$ has its algebraic norm bounded by $|\mathcal{N}_K(q)|$. Moreover, there are at most $\log_2 |\mathcal{N}_K(q)|$ distinct prime ideals $\mathfrak{p}_i$ dividing $q\mathcal{O}_K$. Hence, in Step 3, the set $\mathfrak{I}_0$ can be computed in time $\mathrm{poly}(\log \Delta_L, \log |\mathcal{N}_K(q)|)$.

Recall that, above each prime ideal $\mathfrak{p}$ of $\mathcal{O}_K$ are at most two prime ideals of $\mathcal{O}_L$. More precisely, for each $\mathfrak{p}_i$ appearing in the prime decomposition of $q \cdot \mathcal{O}_K$, either $\mathfrak{p}_i\mathcal{O}_L$ is inert (and thus stays prime), splits into $\mathfrak{q}_i$ and $\tau(\mathfrak{q}_i)$ or ramifies as $\mathfrak{q}_i^2$. The respective relative norm ideals are then $\mathcal{N}_{L|K}(\mathfrak{p}_i\mathcal{O}_L) = \mathfrak{p}_{\mathrm{i}}{}^2$, $\mathcal{N}_{L|K}(\mathfrak{q}_i) = \mathcal{N}_{L|K}(\tau(\mathfrak{q}_i)) = \mathfrak{p}_i$ or $\mathcal{N}_{L|K}(\mathfrak{q}_i) = \mathfrak{p}_i$. To build the set $\mathfrak{I}$ from the set of prime ideals $\mathfrak{q}_{i,j}$, we proceed as follows. Let $\mathfrak{p}_i$ be a prime factor of $q \cdot \mathcal{O}_K$. Any ideal $\mathfrak{a} \in \mathfrak{I}$ is divisible by a prime ideal above $\mathfrak{p}_i$ (recall the correctness part of the proof). From the condition on the norm of $\mathfrak{a}$ and the computation of the $\mathcal{N}_{L|K}(\mathfrak{q}_{i,j})$, we can enumerate all the possibilities for the prime factors of $\mathfrak{a}$.

Keeping the notation of the algorithm, if $\mathfrak{p}_i\mathcal{O}_L$ is inert, then α_i is an even number and $\mathfrak{a}$ must be divisible by $\mathfrak{q}_{i,0}^{\alpha_i/2}$. If $\mathfrak{p}_i\mathcal{O}_L$ ramifies as $\mathfrak{q}_i^2$, then $\mathfrak{a}$ must be divisible by $\mathfrak{q}_i^{\alpha_i}$. Lastly, if $\mathfrak{p}_i\mathcal{O}_L$ splits as $\mathfrak{q}_i$ and $\tau(\mathfrak{q}_i)$, then there exists $0 \leq a \leq \alpha_i$ such that $\mathfrak{q}_i^a\tau(\mathfrak{q}_i)^{\alpha_i - a}$ divides $\mathfrak{a}$. This number is bounded by $(\log_2 |\mathcal{N}_K(q)| + 1)$ so we get at most $(\log_2 |\mathcal{N}_K(q)| + 1)^r$ different ways to choose the prime factors of $\mathfrak{a}$. This is a bound for the cardinal of $\mathfrak{I}$, and one can construct the set $\mathfrak{I}$ (by following the procedure described above) in time $\mathrm{poly}(\log \Delta_L, \log |\mathcal{N}_K(q)|) \cdot |\mathfrak{I}|$.

The last part of the algorithm consists in repeating $|\mathfrak{I}|$ times Steps 7 to 10. These steps can be performed in time $\mathrm{poly}(\log \Delta_L, \log |\mathcal{N}_K(q)|)$ (using Theorem 2.15). The upper bound on the size of $\mathfrak{I}$ gives us the desired upper bound on the running time, which concludes the proof. □

3 Module-LIP for all modules

In this section, we extend the definition of the module-LIP problem from [14] to any (not necessarily free) module. We also provide some tools which are the analogue in the module settings of standard results in the unstructured case (such as Cholesky decomposition). We believe that these basic tools may be useful for further study of the module-LIP problem. Finally, at the end of the section, we compare our new definition of module-LIP with the one from [14], and we explain why breaking the module-LIP problem that we define in Definition 3.11 allows to break the Hawk scheme.

3.1 Pseudo-Gram matrices

By applying the canonical embedding coordinate-wise to vectors and matrices, many well-known matrix decompositions and operations also extends *mutatis mutandis* to matrices over $K_\mathbb{R}$ — another expression of the algebraic structure. This happens since field embeddings preserves linear (and even polynomial) combinations, and we will follow this intuition in this section.

Definition 3.1. *A matrix $G \in M_\ell(K_\mathbb{R})$ is said to be Hermitian definite positive*[14] *if it satisfies $G^* = G$ and $\boldsymbol{x}^* G \boldsymbol{x} \in K_\mathbb{R}^{++}$ for any linearly independent vector $\boldsymbol{x} \in K_\mathbb{R}^\ell$ (i.e., such that $\sigma_i(\boldsymbol{x}) \neq \mathbf{0}$ for all embeddings σ_i, see preliminaries). The set of Hermitian definite positive forms is denoted $\mathcal{H}_\ell^{>0}(K_\mathbb{R})$, and $\mathcal{H}_\ell^{>0}(K)$ when restricted to matrices with entry in K.*

Basic properties. Below we give some standard results on Hermitian definite positive matrices in $K_\mathbb{R}$. The key ingredient is to observe that a matrix G is in $\mathcal{H}_\ell^{>0}(K_\mathbb{R})$ if and only if for all embeddings σ_i, the complex matrix $\sigma_i(G)$ is in $\mathcal{H}_\ell^{>0}(\mathbb{C}) = \{A \in M_\ell(\mathbb{C}) \mid A^* = A,\ x^* A x > 0,\ \forall x \in \mathbb{C}^\ell \setminus \{\mathbf{0}\}\}$. One can then apply the known results over complex Hermitian definite positive matrices and lift them to $\mathcal{H}_\ell^{>0}(K_\mathbb{R})$.

Lemma 3.2. *A matrix $G \in M_\ell(K_\mathbb{R})$ is in $\mathcal{H}_\ell^{>0}(K_\mathbb{R})$ if and only if $\sigma_i(G) \in \mathcal{H}_\ell^{>0}(\mathbb{C})$ for all embeddings σ_i of K.*

Proof. Let $G \in M_\ell(K_\mathbb{R})$. First of all, observe that from the definition of the complex conjugation over $K_\mathbb{R}$, we have that $G^* = G$ if and only if $\sigma_i(G)^* = \sigma_i(G)$ for all embeddings, i.e., $\sigma_i(G)$ is Hermitian (or symmetric if σ_i is real).

Recall also that σ induces a bijection from $K_\mathbb{R}$ to the set $\{z = (z_i)_i \in \mathbb{C}^d \mid z_1, \ldots, z_{d_1} \in \mathbb{R},\ \overline{z_{d_1+i}} = z_{d_1+d_2+i},\ 1 \le i \le d_2\}$. Hence, it holds that $\boldsymbol{x}^* G \boldsymbol{x} \in K_\mathbb{R}^{++}$ for all linearly independent vector $\boldsymbol{x} \in K_\mathbb{R}^\ell$ if and only if $\boldsymbol{y}_i^* \sigma_i(G) \boldsymbol{y}_i > 0$ for all $\boldsymbol{y}_i \in \mathbb{R}^\ell \setminus \{\mathbf{0}\}$ $(1 \le i \le d_1)$ and $\boldsymbol{y}_i \in \mathbb{C}^\ell \setminus \{\mathbf{0}\}$ $(d_1 < i \le d_1 + d_2)$. In other words, $G \in \mathcal{H}_\ell^{>0}(K_\mathbb{R})$ if and only if $\sigma_i(G) \in \mathcal{S}_\ell^{>0}(\mathbb{R})$ for $1 \le i \le d_1$ and $\sigma_i(G) \in \mathcal{H}_\ell^{>0}(\mathbb{C})$ for all $d_1 < i \le d_1 + d_2$. We conclude using the fact that $\mathcal{S}_\ell^{>0}(\mathbb{R}) = \mathcal{H}_\ell^{>0}(\mathbb{C}) \cap M_\ell(\mathbb{R})$. □

Proposition 3.3. *Let $B \in \mathrm{GL}_\ell(K_\mathbb{R})$, then $B^* B \in \mathcal{H}_\ell^{>0}(K_\mathbb{R})$.*

Proof. Note that since $B \in \mathrm{GL}_\ell(K_\mathbb{R})$, then $\sigma_i(B) \in \mathrm{GL}_\ell(\mathbb{C})$ for all complex embedding σ_i. Then, $\sigma_i(B^* \cdot B) = \sigma_i(B)^* \cdot \sigma_i(B) \in \mathcal{H}_\ell^{>0}(\mathbb{C})$ and we conclude using Lemma 3.2. □

[14] In prior literature, this is sometimes called *Humbert forms.*

Proposition 3.4 (Cholesky factorization). *Let $G \in \mathcal{H}_\ell^{>0}(K_\mathbb{R})$, then there exists a unique lower triangular matrix $L \in \mathrm{GL}_\ell(K_\mathbb{R})$ with diagonal coefficients in $K_\mathbb{R}^{++}$ such that $G = LL^*$. Moreover, this matrix L can be computed from G in polynomial time.*

Proof. By Cholesky factorization over $\mathcal{H}_\ell^{>0}(\mathbb{C})$, we know that for every $i \in \{1, \ldots, d_1 + d_2\}$ there exists a unique lower triangular matrices L_i in $M_\ell(\mathbb{C})$ with real positive diagonal coefficients such that $\sigma_i(G) = L_i \cdot L_i^*$ (we even have $L_i \in M_\ell(\mathbb{R})$ for $1 \leq i \leq d_1$). By bijectivity of the canonical embedding, there exists then a unique matrix $L \in M_\ell(K_\mathbb{R})$ such that $\sigma_i(L) = L_i$ for all $1 \leq i \leq d_1 + d_2$, and this matrix has its diagonal coefficients in $K_\mathbb{R}^{++}$ since all the L_i have real positive diagonal coefficients. This proves the existence of L. Unicity is obtained by observing that if L is as in the theorem statement, then $\sigma_i(L) \cdot \sigma_i(L)^*$ is a Cholesky decomposition of $\sigma_i(G)$, and so $\sigma_i(L) = L_i$ by unicity of Cholesky decomposition over $\mathcal{H}_\ell^{>0}(\mathbb{C})$.

To compute the matrix L, a possibility is to compute $\sigma_i(G)$ for all $1 \leq i \leq d_1 + d_2$, then compute the Cholesky decomposition of $\sigma_i(G) \in \mathcal{H}_\ell^{>0}(\mathbb{C})$ to obtain L_i, and finally reconstruct $L \in M_\ell(K_\mathbb{R})$ such that $\sigma_i(L) = L_i$. Computing σ can be done by evaluating polynomials, and computing σ^{-1} (to recover L from the L_i's) can be done by polynomial interpolation. Both steps can be done in polynomial time. Cholesky decomposition over $\mathcal{H}_\ell^{>0}(\mathbb{C})$ can also be performed in polynomial time, hence the full procedure is polynomial. □

Proposition 3.5. *Let B and $C \in \mathrm{GL}_\ell(K_\mathbb{R})$ be such that $B^*B = C^*C$. Then there exists $O \in \mathcal{U}_\ell(K_\mathbb{R})$ such that $B = O \cdot C$.*

Proof. The equality $B^*B = C^*C$ can be rearranged as $(BC^{-1})^* \cdot (BC^{-1}) = I_\ell$ (note that C is invertible), which exactly means that $BC^{-1} \in \mathcal{U}_\ell(K_\mathbb{R})$. □

Pseudo-Gram matrices. We define an analog of quadratic forms and Gram matrices for module lattices, namely the pseudo-Gram matrices.

Definition 3.6. *Let $\boldsymbol{B} = (B, (I_i)_i)$ be a pseudo-basis of a rank-ℓ module M in $K_\mathbb{R}^k$. The pseudo-Gram matrix associated to $\boldsymbol{B}$ is $\boldsymbol{G} := (G, (I_i)_i)$, where $G = B^*B \in \mathcal{H}_\ell^{>0}(K_\mathbb{R})$.*

Using the Cholesky decomposition that we reviewed above, one can see that for any Hermitian positive definite matrix $G \in \mathcal{H}_\ell^{>0}(K_\mathbb{R})$ and any non-zero fractional ideals $I_1, \ldots, I_\ell$, there always exists a pseudo-basis $\boldsymbol{B} = (B, (I_i)_i)$ whose pseudo-Gram matrix is $\boldsymbol{G} = (G, (I_i)_i)$. Note however that Cholesky only guarantees the existence of such pseudo-basis in $K_\mathbb{R}$ (the matrix B is in $M_\ell(K_\mathbb{R})$). Even when $G \in M_\ell(K)$, there may not exist a pseudo-basis in K (with $B \in M_\ell(K)$) whose pseudo-Gram matrix is $\boldsymbol{G}$.

Definition 3.7. *Let $\boldsymbol{G} = (G, (I_i)_{1 \leq i \leq \ell})$ and $\boldsymbol{G'} = (G', (J_i)_{1 \leq i \leq \ell})$ be two pseudo-Gram matrices (with G and G' in $\mathcal{H}_\ell^{>0}(K_\mathbb{R})$). They are said congruent if there exists $U = (u_{i,j})_{1 \leq i,j \leq \ell} \in \mathrm{GL}_\ell(K)$ such that $G' = U^*GU$ and $u_{i,j} \in I_i J_j^{-1}$, $v_{i,j} \in$*

$J_i I_j^{-1}$, where $V = (v_{i,j})_{1\leq i,j\leq \ell} := U^{-1}$. Such U is called a congruence matrix between $\boldsymbol{G}$ and $\boldsymbol{G'}$. This defines an equivalence relation $\sim$ on the set of pseudo-Gram matrices.[15] *The class of $\boldsymbol{G}$ is denoted by $[\boldsymbol{G}]$.*

Sampling (implicit) Gaussian vectors. As in the case of standard (non-structured) lattices, it is possible to sample vectors from a discrete Gaussian distribution in an implicit module M given as input a pseudo-Gram matrix G of this module.

Lemma 3.8. *There is a probabilistic polynomial time algorithm* `GaussianGram` *such that the following holds. Let $\boldsymbol{B} = (B, (I_i)_{1\leq i\leq \ell})$ be a pseudo-basis of a rank-ℓ module M in $K_{\mathbb{R}}^{\ell}$, and $\boldsymbol{G} = (G, (I_i)_{1\leq i\leq \ell})$ be the pseudo-Gram matrix of $\boldsymbol{B}$. Let $s > 0$ be a real number satisfying*

$$s \geq \sqrt{\frac{d\cdot \log(2d\ell+4)}{\pi}} \cdot 2^d \cdot \mathit{\Delta}_K^{3/(2d)} \cdot \max_{1\leq j\leq r} \left(\|\sigma(g_{j,j})^{1/2}\| \cdot \mathcal{N}(I_j)^{1/d} \right),$$

where $G = (g_{i,j})_{1\leq i,j\leq \ell}$ and the square-root is applied coordinate-wise to $\sigma(g_{j,j}) \in \mathbb{C}^d$. On input $\boldsymbol{G}$ and s, `GaussianGram`$(\boldsymbol{G}, s)$ outputs $\boldsymbol{z} \in I_1 \times \cdots \times I_\ell$ such that $\boldsymbol{v} := B \cdot \boldsymbol{z}$ follows a discrete Gaussian distribution of parameter s in M (i.e., $\sigma(\boldsymbol{v}) \sim D_{\sigma(M),s}$).

Proof. The algorithm `GaussianGram` first computes a pseudo-basis $\boldsymbol{C} = (C, (I_i)_i)$ such that $C^*C = G$, using Cholesky decomposition. According to Proposition 3.4, this can be done in polynomial time. The pseudo-basis $\boldsymbol{C}$ generates a rank-ℓ module $M' \subset K_{\mathbb{R}}^{\ell}$, and from Proposition 3.5, since $\boldsymbol{C}$ and $\boldsymbol{B}$ have the same pseudo-Gram matrix $\boldsymbol{G}$, we know that there exists $O \in \mathcal{U}_\ell(K_{\mathbb{R}})$ such that $C = O \cdot B$.

From $\boldsymbol{C}$, the algorithm then computes a basis $D \in \mathbb{C}^{d\ell\times d\ell}$ of the lattice $\sigma(M')$. This can be done in polynomial time from Lemma 2.9, and the same lemma also tells us that the column vectors $\boldsymbol{d}_i$ of D have Euclidean norm upper bounded by

$$\|\boldsymbol{d}_i\| \leq \sqrt{d} \cdot 2^d \cdot \mathit{\Delta}_K^{3/(2d)} \cdot \max_{1\leq j\leq r} \left(\|\sigma(\boldsymbol{c}_j)\| \cdot \mathcal{N}(I_j)^{1/d} \right),$$

where $\boldsymbol{c}_j$ are the column vectors of C. Observe that, since $G = C^*C$, then $g_{i,j} = \langle \boldsymbol{c}_i, \boldsymbol{c}_j \rangle_{K_{\mathbb{R}}}$ (where $G = (g_{i,j})_{1\leq i,j\leq \ell}$). Hence, the quantity $\max_j \|\sigma(\boldsymbol{b}_j)\|$ in the bound above can be replaced by $\max_j \|\sigma(g_{j,j})^{1/2}\|$, where the square-root is applied coordinate-wise to $\sigma(g_{j,j}) \in \mathbb{C}^d$.

Finally, the algorithm uses the basis D to sample $\boldsymbol{x} \in \sigma(M')$, following the discrete Gaussian distribution $D_{\sigma(M'),s}$. This is doable in polynomial time, using Lemma 2.4 and the fact that $s \geq \sqrt{\log(2d\ell+4)/\pi} \cdot \max_i ||\boldsymbol{d}_i||$.

[15] For the transitivity ; let $\boldsymbol{G} = (G, (I_i)_{1\leq i\leq \ell})$, $\boldsymbol{G'} = (G', (J_i)_{1\leq i\leq \ell})$, $\boldsymbol{G''} = (G'', (L_i)_{1\leq i\leq \ell})$ and U (resp. U') a congruence matrix between $\boldsymbol{G}$ and $\boldsymbol{G'}$ (resp. between $\boldsymbol{G'}$ and $\boldsymbol{G''}$), then $U'' := U \cdot U'$ satisfies $G'' = U''^* \cdot G \cdot U''$ and has coefficients $U''_{i,j} = \sum_{k=1}^{\ell} u_{i,k} \cdot u'_{k,j}$. All terms of the sum are in $I_i J_j^{-1}$ by definition. The same observation for $(U'')^{-1}$ finally gives $\boldsymbol{G} \sim \boldsymbol{G''}$.

The algorithm then reconstructs $\boldsymbol{w} \in M'$ such that $\sigma(\boldsymbol{w}) = \boldsymbol{x}$. This can again be done in polynomial time. The vector $\boldsymbol{w} \in M'$ is distributed according to a discrete Gaussian distribution over M' of parameter s. Let $\boldsymbol{z}$ be the coordinates of $\boldsymbol{w}$ in the pseudo-basis $\boldsymbol{C}$. Since $\boldsymbol{w}$ is in M', then $\boldsymbol{z}$ must be in $I_1 \times \cdots \times I_\ell$. This vector $\boldsymbol{z}$ is then output by the algorithm.

We have seen so far that the algorithm runs in polynomial time, and outputs a vector $\boldsymbol{z} \in I_1 \times \cdots \times I_\ell$ such that $\boldsymbol{w} = C \cdot \boldsymbol{z}$ follows a discrete Gaussian distribution in M' of parameter s. To conclude, remember that $B = O \cdot C$ and that $O \in \mathcal{U}_\ell(K_\mathbb{R})$ preserves the euclidean norm (i.e., for any $\boldsymbol{y} \in K_\mathbb{R}^\ell$, we have $\|\sigma(y)\| = \|\sigma(O \cdot y)\|$). Hence the vector $\boldsymbol{v} = B \cdot \boldsymbol{z} = O \cdot \boldsymbol{w}$ follows a Gaussian distribution of parameter s in M. □

3.2 Module-LIP

We are now ready to define the module-LIP problem for any number field K and any module M. Using the formalism of pseudo-bases and pseudo-Gram matrices, the situations for module lattices and non-structured lattices become quite similar. Consequently, the definitions in this subsection will be very reminiscent of the ones from preliminaries. We start by introducing the notion of isomorphism for module lattices.

Definition 3.9. *Let $M, M' \subset K_\mathbb{R}^\ell$ be two modules of rank ℓ. We say that M, M' are isomorphic as module lattices if there exists a unitary transformation $O \in \mathcal{U}_\ell(K_\mathbb{R})$ such that $M' = O \cdot M$.*

Note that if M and M' are isomorphic as module lattices, then the lattices $\sigma(M)$ and $\sigma(M')$ are isomorphic for the standard (non-structured) definition of lattice isomorphism. In the case of module lattice isomorphism, we restrict ourselves to specific lattices (the ones of the form $\sigma(M)$ for M a module) *and* specific orthogonal transformations (they should be K-linear on the modules).

Isomorphism of module lattices can be restated in terms of pseudo-bases or pseudo-Gram matrices.

Lemma 3.10. *Let $M, M' \subset K_\mathbb{R}^\ell$ be two modules of rank ℓ with respective pseudo-bases $\boldsymbol{B} = (B, (I_i)_{1 \le i \le \ell})$ and $\boldsymbol{B}' = (B', (J_i)_{1 \le i \le \ell})$. Let $\boldsymbol{G}$ (resp. $\boldsymbol{G}'$) be the pseudo-Gram matrix associated to $\boldsymbol{B}$ (resp $\boldsymbol{B}'$). Then, the three following assertions are equivalent*

(1) M and M' are isomorphic as module lattices;
(2) there exists $O \in \mathcal{U}_\ell(K_\mathbb{R})$ and $U \in \mathrm{GL}_\ell(K)$ with $u_{i,j} \in I_i J_j^{-1}$ and $v_{i,j} \in J_i I_j^{-1}$ (where $U = (u_{i,j})_{1 \le i,j \le \ell}$ and $U^{-1} = (v_{i,j})_{1 \le i,j \le \ell}$) such that $B' = OBU$;
(3) $\boldsymbol{G}$ and $\boldsymbol{G}'$ are congruent (see Definition 3.7).

Proof. We prove that (1) ⇒ (2) ⇒ (3) ⇒ (1). Assume that M and M' are isomorphic as module lattices. By definition, there exists $O \in \mathcal{U}_\ell(K_\mathbb{R})$ such that $M' = O \cdot M$. A pseudo-basis of OM is given by $(OB, (I_i)_i)$. Since $\boldsymbol{B}'$ is also a pseudo-basis of $O \cdot M$, there must exist a matrix $U \in \mathrm{GL}_\ell(K)$ such that $u_{i,j} \in I_i J_j^{-1}$ and $v_{i,j} \in J_i I_j^{-1}$ and $B' = (OB)U$, as desired.

Let us now assume that $B' = OBU$. Then $G' = U^*GU$ and this means that $\boldsymbol{G}$ and $\boldsymbol{G}'$ are congruent.

Finally, let us assume that $\boldsymbol{G}$ and $\boldsymbol{G}'$ are congruent. By definition, there exists $U \in \mathrm{GL}_\ell(K)$ (with $u_{i,j} \in I_i J_j^{-1}$ and $v_{i,j} \in J_i I_j^{-1}$) such that $G' = U^*GU$. Let us consider $\boldsymbol{C} = (BU, (J_i)_i)$. Thanks to the condition on U, we know that $\boldsymbol{C}$ and $\boldsymbol{B}$ are both pseudo-bases of the same module M. Moreover, the pseudo-Gram matrix of $\boldsymbol{C}$ is exactly $\boldsymbol{G}'$. Using Proposition 3.5, we conclude that there exists $O \in \mathcal{U}_\ell(K_\mathbb{R})$ such that $B' = OBU$, i.e., $M' = O \cdot M$. □

Let us now define the module Lattice Isomorphism Problem (module-LIP), in its worst-case variant. As in the unstructured case, we could define two variants, one using the formalism of pseudo-bases and another one using the formalism of pseudo-Gram matrices. Instead, we decided to introduce only one variant, which uses both pseudo-bases and pseudo-Gram matrices. More precisely, we define a collection of problems parametrized by the field K and a pseudo-basis $\boldsymbol{B}$ of a module M, and whose input is a pseudo-Gram matrix $\boldsymbol{G}'$ (of a different pseudo-basis $\boldsymbol{B}'$ of M). This variant is tailored to fit our attack in the next section, but we also believe that it makes sense for itself. Indeed, so far, the instantiations of LIP and module-LIP in cryptography have been parameterized by very specific (module) lattices, for which a good (pseudo-)basis was known (e.g., in Hawk, the module lattice is $\mathcal{O}_K^2$).[16] On the other hand, the algorithm then usually manipulates only (pseudo-)Gram matrices, and so an attacker would get as input such a (pseudo-)Gram matrix and not a (pseudo-)basis of an isomorphic (module) lattice.

Definition 3.11 (wc-smodLIP$_K^{\boldsymbol{B}}$)**.** *For $\boldsymbol{B}$ a pseudo-basis of a module-lattice $M \subset K_\mathbb{R}^\ell$ with associated pseudo-Gram matrix $\boldsymbol{G}$, the worst-case search module-Lattice Isomorphism Problem with parameter K and $\boldsymbol{B}$ denoted by* wc-smodLIP$_K^{\boldsymbol{B}}$ *is, given as input any pseudo-Gram matrix $\boldsymbol{G'} \sim \boldsymbol{G}$ (see Definition 3.7), to find a congruence matrix between $\boldsymbol{G}$ and $\boldsymbol{G'}$.*

When we say that the problem is parameterized by K and $\boldsymbol{B}$, we mean that this is a collection of algorithmic problems, one for each choice of $(K, \boldsymbol{B})$.[17] In particular, the number field K and pseudo-basis $\boldsymbol{B}$ are known to an adversary.

A definition of module-LIP was also provided in [14, Definition 7], which differs slightly from ours. There are three main differences between the two definitions: [14, Definition 7] defines module-LIP for free modules M (with $\boldsymbol{B} = (B, (\mathcal{O}_K)_{1 \le i \le 2})$ a *basis* of the module), over a CM number field, and restricts $\boldsymbol{G}'$ to be the Gram-matrix of a *basis* $\boldsymbol{B}' = (B', (\mathcal{O}_K)_{1 \le i \le 2})$ of M, and such that the

[16] By good, we mean here a (pseudo)-basis with rational coefficients. This will be needed for our attack, and we do not know how to recover it efficiently from the (pseudo-)Gram matrix since Cholesky decomposition only provides a basis with coefficients in $\mathbb{R}$ (or $K_\mathbb{R}$).

[17] The same terminology is used for, e.g., the LWE problem. We usually say that n, m, q are parameters of the problem, which means that for each choice of (n, m, q), we have a different algorithmic problem.

determinant of B' is the same as the one of B. On the other hand, Definition 3.11 above holds for any module M (and any pseudo-basis $\boldsymbol{B}$), any number field, and the input $\boldsymbol{G'}$ of the problem is the Gram-matrix of any pseudo-basis $\boldsymbol{B'}$ of M. Note that even if one takes K a CM field, and $\boldsymbol{B}$ a basis of a free module M, then our problem wc-smodLIP$_K^{\boldsymbol{B}}$ from Definition 3.11 is still not exactly the same as the problem defined in [14, Definition 7]. Indeed, in our definition, the input of the problem can be any Gram-matrix $\boldsymbol{G'}$ congruent to $\boldsymbol{G}$, whereas in [14, Definition 7] the input is restricted to specific congruent Gram-matrices $\boldsymbol{G'}$.

Hawk and module-LIP. In the case of Hawk [14], the authors consider a power-of-two cyclotomic field K and the free rank-2 module-lattice $M = \mathcal{O}_K^2$. The secret key consists in a short basis U of M and the public key is the Gram matrix $G' = U^*U \in \mathcal{H}_2^{>0}(K)$.[18] With the formalism we introduced in the previous paragraph, U is a congruence matrix between $\boldsymbol{I_2} = (I_2^* I_2, (\mathcal{O}_K)_{1 \le i \le 2})$ and $\boldsymbol{G'} = (G', (\mathcal{O}_K)_{1 \le i \le 2})$. We now recall how (uncompressed) Hawk proceeds (Section 3.1, [14]).

Given a message $\boldsymbol{m}$ to sign, first hash $\boldsymbol{m}$ together with a salt $\boldsymbol{r}$ to a point $\boldsymbol{h} \in \{0,1\}^{2d}$, which is interpreted as an element of $\mathcal{O}_K^2$ thanks to the following bijective map, called coefficient embedding $K \to \mathbb{Q}^d$; $(\sum_i a_i X^i) \mapsto (a_0, \dots, a_{d-1})^T$. Then, a vector $\boldsymbol{x} \in \mathcal{O}_K^2 + \frac{1}{2} U \cdot \boldsymbol{h}$ close to $\boldsymbol{0}$ is sampled using Gaussian samples in $\mathbb{Z}$ and $\frac{1}{2}\mathbb{Z}$ (the identification between $\mathcal{O}_K^2$ and $\mathbb{Z}^{2d}$ being possible via coefficient embedding). Finally, the signer computes $\boldsymbol{s} := \frac{1}{2}\boldsymbol{h} - U^{-1} \cdot \boldsymbol{x}$. The (uncompressed) signature consists of the pair $\boldsymbol{sig} = (\boldsymbol{r}, \boldsymbol{s})$.

A verifier receiving $\boldsymbol{sig}$ computes the hash $\boldsymbol{h}$ and checks if $||\frac{1}{2}\boldsymbol{h} - \boldsymbol{s}||_{G'}$ is smaller than a fixed security parameter (here, $\|\boldsymbol{z}\|_{G'} := \|\boldsymbol{z}^* G' \boldsymbol{z}\|$ for any $\boldsymbol{z} \in K^2$). It also checks that $\boldsymbol{s} \in \mathcal{O}_K^2$. If both tests succeed, it accepts the signature. Using the fact that $||\frac{1}{2}\boldsymbol{h} - \boldsymbol{s}||_{G'} = \|U \cdot (\frac{1}{2}\boldsymbol{h} - \boldsymbol{s})\| = \|\boldsymbol{x}\|$, one can check that the first verification test must be satisfied when the signature is honestly generated. For the second one, writing $\boldsymbol{x} = \boldsymbol{y} + \frac{1}{2}U \cdot \boldsymbol{h}$ with $\boldsymbol{y} \in \mathcal{O}_K^2$, one can see that $\boldsymbol{s} = U^{-1}\boldsymbol{y}$. The fact that $\boldsymbol{s} \in \mathcal{O}_K^2$ follows from the fact that $U \in \mathrm{GL}_2(\mathcal{O}_K)$ (since U is a basis of $\mathcal{O}_K^2$).

Lemma 3.12. *With the notations of the previous paragraph, finding any congruence matrix V between $\boldsymbol{I_2}$ and the public key $\boldsymbol{G'}$ allows to forge signatures in Hawk. In other words, if one can solve the* wc-smodLIP$_K^{\boldsymbol{I_2}}$ *problem, then one can forge signatures in Hawk, given only the public key.*

Proof. By definition, a congruence matrix V between $\boldsymbol{G}$ and $\boldsymbol{G'}$ satisfies $V \in \mathrm{GL}_2(\mathcal{O}_K)$ and $G' = V^*V$. We show that any such matrix can be used to produce valid signatures : let $(\boldsymbol{r}, \boldsymbol{s})$ be a signature produced by Hawk with input basis V. Then $\boldsymbol{x} = \boldsymbol{y} + \frac{1}{2}V \cdot h$ is small, with $\boldsymbol{y} \in \mathcal{O}_K^2$, and $\boldsymbol{s} = \frac{1}{2}h - V^{-1}\boldsymbol{x} = V^{-1}\boldsymbol{y}$. Since $V \in \mathrm{GL}_2(\mathcal{O}_K)$, the vector $\boldsymbol{s}$ is in $\mathcal{O}_K^2$ as desired. Moreover, $\|\frac{1}{2}\boldsymbol{h} - \boldsymbol{s}\|_{G'} = \|V \cdot (\frac{1}{2}\boldsymbol{h} - \boldsymbol{s})\| = \|\boldsymbol{x}\|$, since $G' = V^*V$, which means that the latter norm is small the pair $(\boldsymbol{r}, \boldsymbol{s})$ is then accepted by the verifier. □

[18] Here M is a free module so it has a basis (equivalently, the coefficient ideals are equal to $\mathcal{O}_K$) so the authors use bases (resp. Gram-matrices) instead of pseudo-bases (resp. pseudo-Gram matrices).

The previous lemma underlines that, to forge signatures in Hawk it is enough to recover *any* congruence matrix V and not specifically the secret key U. Therefore, the problem of signature forgery in Hawk reduces to solving wc-smodLIP$_K^{\boldsymbol{I_2}}$, as defined in Definition 3.11.

4 An algorithm for module-LIP in rank 2 over totally real fields

We now present our main algorithm, which solves wc-smodLIP$_K^{\boldsymbol{B}}$ for totally real fields K and when the module generated by the pseudo-basis $\boldsymbol{B}$ lives in K^2 (see Algorithm 4.2 when the module is in $\mathcal{O}_K^2$ and Algorithm 4.3 for the general case). At a high level, it is based on the observation that, when the module is in $\mathcal{O}_K^2$, a diagonal element q in the pseudo-Gram matrix $\boldsymbol{G} = \boldsymbol{B}^*\boldsymbol{B} = \boldsymbol{B}^T\boldsymbol{B}$ can be written as $q = x^2 + y^2$ with $x, y \in \mathcal{O}_K$, that is, a *sum of two squares.* Already when $K = \mathbb{Q}$, a common way to find such sums is to go through a quadratic imaginary extension $L := K(i)$, where i is a root of $X^2 + 1$ over K. Indeed, all solutions to a 2-square decompositions are also solutions of a *relative norm equation* $q = \mathcal{N}_{L|K}(a) = a^*a$, which we can solve algorithmically thanks to Lenstra-Silverberg's algorithm. Our algorithm will thus use the procedure `NormEquation` (from preliminaries) to solve the latter, then restrict to solutions of the former in another algorithm named `TwoSquares`. From these solutions, we will be able to build the set of all congruence matrices to solve wc-smodLIP$_K^{\boldsymbol{B}}$, when the module generated by $\boldsymbol{B}$ is in $\mathcal{O}_K^2$. For rational modules in K^2, we then simply multiply by a common denominator, to reduce to the situation where the module is in $\mathcal{O}_K^2$.

An apparently unavoidable component of `NormEquation` is the factorization of the ideal $q \cdot \mathcal{O}_K$, so there is no hope to obtain a classical polynomial-time algorithm this way. Therefore, we use a re-randomization procedure to generate random vectors $\boldsymbol{z} = (z_1, z_2)$ until $\boldsymbol{z}^*G\boldsymbol{z}$ generates a prime ideal (for the rest of this paragraph, we say that $\boldsymbol{z}$ is a "good vector" when this is the case). Primality testing can be done in polynomial time for ideals as well, and if we let $B \cdot (z_1, z_2)^T = (x, y)^T$, then $\boldsymbol{z}^*G\boldsymbol{z} = x^2 + y^2$ is again a sum of two squares. This allows us to avoid the potentially costly factorization. With two linearly independent good vectors, we can then use a two-square decomposition algorithm together with some linear algebra to solve the module-LIP instance. Estimating the probability of finding a good vector relies on a heuristic assumption, supported by standard (but non-trivial) number-theoretic arguments: the discussion and justifications are the topic of Section 4.2.

4.1 Gram ideal

In order to describe our results in this section for all modules of rank 2 in K^2, we introduce the Gram ideal and the relative Gram ideal of a module $M \subset K^\ell$ of rank ℓ. From now on, we will consider the arithmetic properties of the coefficients of vectors in M, and hence restrict ourselves to modules in K^ℓ (as opposed to $K_\mathbb{R}^\ell$ which we considered so far).

Definition 4.1. *Let K be a totally real number field and $M \subset K^\ell$ be a module of rank ℓ. Let $\boldsymbol{B} = ((B_{i,j})_{i,j}, (J_i)_i)$ be a pseudo-basis of M and $\boldsymbol{G} = ((G_{i,j})_{i,j}, (J_i)_i)$ be the associated pseudo-Gram matrix. We define the following fractional ideals*

$$\mathcal{G}(M) := \sum_{1 \le i \le l} G_{i,i} \cdot J_i^2 + \sum_{1 \le i < j \le l} 2 \cdot G_{i,j} \cdot J_i \cdot J_j$$

$$\mathcal{C}(M) := \sum_{1 \le i,j \le l} B_{i,j} \cdot J_j \quad ; \quad \mathcal{RG}(M) := \mathcal{G}(M)/\mathcal{C}(M)^2.$$

We call $\mathcal{G}(M)$ the Gram ideal and $\mathcal{RG}(M)$ the relative Gram ideal of M.

The first two ideals correspond respectively to the ideal generated by the (squared) norm of vectors of M, and the ideal generated by the coordinates of the vectors of M. We show below that these ideals indeed depend only on the module lattice M and not on the pseudo-basis $\boldsymbol{B}$.

Lemma 4.2. *Let K be totally real and $M \subset K^\ell$ be a module of rank ℓ. Let $\boldsymbol{B}$ be a pseudo-basis of M with associated pseudo-Gram matrix $\boldsymbol{G}$. Then $\mathcal{G}(M)$ is the smallest ideal (for inclusion) containing the set $\{\langle \boldsymbol{v}, \boldsymbol{v} \rangle_{K_\mathbb{R}} \,|\, \boldsymbol{v} \in M\}$, and $\mathcal{C}(M)$ is the smallest ideal containing the set $\{v_j \,|\, \boldsymbol{v} = (v_i)_i \in M, 1 \le j \le \ell\}$. In particular, they do not depend on the choice of the pseudo-basis $\boldsymbol{B}$ of M.*

Proof. Any module vector can be uniquely written as $\boldsymbol{v} = \sum_{i=1}^{l} x_i \cdot \boldsymbol{b}_i$, where $\boldsymbol{b}_i$ is the i-th vector column of B, and $x_i \in J_i$ (with $\boldsymbol{B} = (B, (J_i)_i)$). Then, $\langle \boldsymbol{v}, \boldsymbol{v} \rangle_{K_\mathbb{R}} = \sum_{1 \le i \le \ell} x_i^2 G_{i,i} + 2 \sum_{1 \le i < j \le \ell} x_i x_j G_{i,j} \in \mathcal{G}(M)$, where $G = (G_{i,j})_{i,j}$. Note that here, we used the fact that K is totally real, and so $\overline{v_i} = v_i$ for all i's. Conversely, any (fractional) ideal I containing the square euclidean norm of the vectors of M must contain $\langle x\boldsymbol{b}_i, x\boldsymbol{b}_i \rangle_{K_\mathbb{R}} = x^2 \cdot G_{i,i}$ for all $x \in J_i$ and all $1 \le i \le \ell$. Since the ideal generated by $\{x^2 \,|\, x \in J_i\}$ is precisely J_i^2, we conclude that $G_{i,i} \cdot J_i^2$ is included in I.[19] Using the polarization formula and since I is stable by addition, we see that it contains $||x_i\boldsymbol{b}_i + x_j\boldsymbol{b}_j||^2 - ||x_i\boldsymbol{b}_i||^2 - ||x_j\boldsymbol{b}_j||^2 = 2G_{i,j} \cdot x_i \cdot x_j$, for all $1 \le i < j \le l$ and all $x_i \in J_i$ and $x_2 \in J_2$. Hence, I contains $2G_{i,j}J_i \cdot J_j$ for all i and j's. The assertion on $\mathcal{C}(M)$ follows from its definition. □

Corollary 4.3. *The relative Gram ideal $\mathcal{RG}(M)$ is integral, and do not depend on the choice of the pseudo-basis $\boldsymbol{B}$ of M.*

Proof. The fact that $\mathcal{RG}(M)$ does not depend on the choice of $\boldsymbol{B}$ follows from its definition, and the fact that neither $\mathcal{G}(M)$ nor $\mathcal{C}(M)$ depend on $\boldsymbol{B}$. To see that the ideal is integral, we show that $\mathcal{G}(M) \subseteq \mathcal{C}(M)^2$ (which implies that $\mathcal{G}(M)/\mathcal{C}(M)^2 \subseteq \mathcal{O}_K$). Let $\boldsymbol{v} = (v_i)_i \in M$. From Lemma 4.2, we know that $\langle \boldsymbol{v}, \boldsymbol{v} \rangle_{K_\mathbb{R}} = \sum_i v_i^2 \in \mathcal{C}(M)^2$. Since $\mathcal{G}(M)$ is generated by the $\langle \boldsymbol{v}, \boldsymbol{v} \rangle_{K_\mathbb{R}}$ for $\boldsymbol{v} \in M$ (see again Lemma 4.2), we conclude that $\mathcal{G}(M) \subseteq \mathcal{C}(M)^2$ as desired. □

[19] To prove that the ideal generated by $\{x^2 \,|\, x \in J_i\}$ is equal to J_i^2, let $a, b \in J_i$ be such that $J_i = a\mathcal{O}_K + b\mathcal{O}_K$. Then $a\mathcal{O}_K = J_i \cdot \mathfrak{a}$ and $b\mathcal{O}_K = J_i \cdot \mathfrak{b}$ with $\mathfrak{a}$ and $\mathfrak{b}$ integral coprime ideals. The ideal containing all squares x^2 for $x \in J_i$ must in particular contain $(a^2 + b^2) \cdot \mathcal{O}_K = J_i^2 \cdot (\mathfrak{a}^2 + \mathfrak{b}^2) = J_i^2$. The other inclusion is immediate.

The Gram ideal and relative Gram ideal of a module M can be computed in polynomial time from any pseudo-basis $\boldsymbol{B}$ of M (directly from Definition 4.1). A simple but important case is when $M = \mathcal{O}_K^\ell$, e.g, in Hawk. In this case, we have $\mathcal{G}(M) = \mathcal{C}(M) = \mathcal{RG}(M) = \mathcal{O}_K$. In the following we prove that the relative Gram ideal is invariant when scaling the module (by a scalar, or even a fractional ideal).

Lemma 4.4. *Let K be totally real, $M \subset K^\ell$ be a module of rank ℓ, and J be a fractional ideal. Then,*

(1) $\mathcal{G}(J \cdot M) = J^2 \cdot \mathcal{G}(M)$.
(2) $\mathcal{C}(J \cdot M) = J \cdot \mathcal{C}(M)$.
(3) $\mathcal{RG}(J \cdot M) = \mathcal{RG}(M)$.
(4) $\mathcal{C}(M)^{-1} \cdot M \subset \mathcal{O}_K^\ell$ is an integer module lattice not contained in $\mathfrak{p}^\ell$ for any prime ideal $\mathfrak{p}$.

We will later use Lemma 4.4 to scale our input module M, in order to make it integer, but as small as possible (with respect to its determinant) among all the scaled variants of M included in $\mathcal{O}_K^\ell$.

Proof. (1) − (3) Let $\boldsymbol{B} = (B, (I_i)_{1 \leq i \leq \ell})$ be a pseudo-basis of M with pseudo-Gram matrix $\boldsymbol{G}$. Recall from preliminaries that $J \cdot M$ has pseudo-basis $(B, (J \cdot I_i)_{1 \leq i \leq \ell})$ so the Gram ideal associated is by definition

$$\begin{aligned}\mathcal{G}(J \cdot M) &= \sum_{1 \leq i \leq \ell} G_{i,i} \cdot (J \cdot I_i)^2 + 2 \sum_{1 \leq i < j \leq \ell} G_{i,j} \cdot (J \cdot I_i) \cdot (J \cdot I_j) \\ &= J^2 \cdot \mathcal{G}(M).\end{aligned}$$

Similarly, $\mathcal{C}(J \cdot M) = J \cdot \mathcal{C}(M)$ thus $\mathcal{RG}(J \cdot M) = \mathcal{RG}(M)$ holds.
(4) $M' = \mathcal{C}(M)^{-1} \cdot M$ is such that $\mathcal{C}(M') = \mathcal{O}_K$ (using (2)) so any lattice vector $\boldsymbol{v} \in M'$ has its coordinates in $\mathcal{O}_K$ thus $M' \subset (\mathcal{O}_K)^\ell$. If M' were contained in some $\mathfrak{p}^\ell$ then the ideal spanned by the coordinates of M' would be contained in $\mathfrak{p}$, i.e, we would have $\mathcal{O}_K = \mathcal{C}(M') \subseteq \mathfrak{p}$ (using Lemma 4.2), which is impossible. □

4.2 The assumption

In this section, we formalize the assumption that will be used by our algorithm in Section 4.3. This assumption essentially states that, when sampling a random vector $\boldsymbol{u}$ in an integer rank-2 module $M \subseteq \mathcal{O}_K^2$, the probability that the ideal $\langle \boldsymbol{u}, \boldsymbol{u} \rangle_{K_\mathbb{R}} \cdot \mathcal{O}_K$ is prime is not too small. Of course, we need to exclude the cases where this assumption is obviously false, for instance if $M = 2\mathcal{O}_K^2$, then $\langle \boldsymbol{u}, \boldsymbol{u} \rangle_{K_\mathbb{R}}$ will always be divisible by 4. More generally, we need to exclude the cases where the Gram ideal $\mathcal{G}(M)$ is not $\mathcal{O}_K$. We provide in Appendix B both theoretical and experimental justifications in favor of our assumption.

Assumption 1. *There exists some absolute polynomial P (with non-negative coefficients) such that the following holds. Let K be a totally real number field* *of degree d, $M \subseteq \mathcal{O}_K^2$ be a module of rank 2, and $s > 0$ be a real number such that $s \geq \eta_{1/2}(M)$. Let $I = \mathcal{G}(M)$ be the Gram ideal of the module M. Let $(z_1, z_2)^T \leftarrow D_{M,s}$ and $q = z_1^2 + z_2^2$. Then*

$$\Pr(q \cdot I^{-1} \textit{ is prime }) \geq \frac{1}{\rho_K \cdot \log(s/\mathcal{N}(I)^{1/d}) \cdot P(d)},$$

where ρ_K is the residue of the Dedekind zeta function of K at 1.

Note that the ideal $q \cdot I^{-1}$ in our assumption is always an integral ideal. Indeed, thanks to Lemma 4.2, we know that the ideal I contains all square norms of vectors of M, hence it contains in particular $\langle \boldsymbol{z}, \boldsymbol{z} \rangle_{K_\mathbb{R}} = q$ (where $\boldsymbol{z} = (z_1, z_2)^T$), which implies $qI^{-1} \subseteq II^{-1} = \mathcal{O}_K$.

4.3 The algorithm

Before describing the algorithm for module-LIP, let us start with an algorithm, called `TwoSquares`, which solves sum of two squares equations in $\mathcal{O}_K$ for a totally real number field K (i.e., equations of the form $x^2+y^2 = q$ where $q \in \mathcal{O}_K$ is given and $x, y \in \mathcal{O}_K$ are the unknown). Algorithm `TwoSquares` works by reformulating this as a relative norm equation in some well chosen CM extension L of K, and then applying Algorithm `NormEquation` from preliminaries (Algorithm 2.1).

Algorithm 4.1 Finding sums of two squares in totally real fields (`TwoSquares`)

```
Input: A basis $B_K$ of $\mathcal{O}_K$, an element $q \in \mathcal{O}_K$, the factorization of $|\mathcal{N}_K(q)|$.
Output: All elements $(x, y) \in \mathcal{O}_K^2$ such that $q = x^2 + y^2$.
Define $L \leftarrow K[X]/(X^2 + 1)$.
Compute $B_L \leftarrow$ a basis of $\mathcal{O}_L$, using Lemma 2.6.
$S \leftarrow$ NormEquation($B_K, B_L, q,$ the factorization of $|\mathcal{N}_K(q)|$).
Cast elements of $S$ from $L$ into $K^2$ (via the map $a + bX \in L \mapsto (a, b) \in K^2$)
return $S \cap \mathcal{O}_K^2$
```

Lemma 4.5. *Let K be a totally real number field and $q \in \mathcal{O}_K$. Given as input a basis of $\mathcal{O}_K$, the element $q \in \mathcal{O}_K$ and the factorization of $|\mathcal{N}_K(q)|$, Algorithm* 4.1 *(`TwoSquares`) computes all $(x, y) \in \mathcal{O}_K^2$ such that $q = x^2 + y^2$. Moreover, the algorithm runs in time* $\mathrm{poly}\big(\log \Delta_K, (\log |\mathcal{N}_K(q)|)^r\big)$*, where r is the number of distinct prime factors of the ideal $q \cdot \mathcal{O}_K$.*

Proof. Correctness. The field $L = K[X]/(X^2+1)$ is a totally imaginary quadratic extension of K, and so $L|K$ is a CM extension as desired. Moreover, the non-trivial automorphism τ of L fixing K is the map sending X to $-X$. Hence, for any element $z = a + bX \in L$, we have that $\mathcal{N}_{L|K}(z) = z \cdot \tau(z) = a^2 + b^2$. To any solution (x, y) of the sum of two squares equation $x^2 + y^2 = q$, we can then associate a solution $z = x + y \cdot X \in \mathcal{O}_L$ to the relative norm equation

$\mathcal{N}_{L|K}(z) = q$. By correctness of the `NormEquation` algorithm (see Theorem 2.16), the set S contains all solutions to this relative norm equation. Note that not all of them provide a solution to the sum of two squares equations, since some $z = x + y \cdot X \in S$ may have x or y not belonging to $\mathcal{O}_K$ (the set $\{a + bX \mid a, b \in \mathcal{O}_K\}$ is included in $\mathcal{O}_L$, but the inclusion may be strict). This is why we intersect the set S with the set $\{a + bX \mid a, b \in \mathcal{O}_K\}$.

Complexity. A basis of $\mathcal{O}_L$ can be computed in polynomial time from a basis of $\mathcal{O}_K$, using Lemma 2.6, which proves that Step 2 can be run in polynomial time. For Step 3, we know from Theorem 2.16 that it can be run in time poly($\log \Delta_L$, $(\log|\mathcal{N}_K(q)|)) \cdot (1 + \log|\mathcal{N}_K(q)|)^r$ = poly $\big(\log \Delta_L, (\log|\mathcal{N}_K(q)|)^r\big)$. Recall from the preliminaries that $\log \Delta_L$ = poly($\log \Delta_K$), hence the previous quantity is also poly $\big(\log \Delta_K, (\log|\mathcal{N}_K(q)|)^r\big)$. Finally, testing whether an element $x + y \cdot X$ in S satisfies $(x, y) \in \mathcal{O}_K^2$ can be done in polynomial time using the basis B_K. □

We now describe our main algorithm, which solves wc-smodLIP$_K^{\boldsymbol{B}}$ when K is a totally real number field and $\boldsymbol{B}$ is a pseudo-basis of a rank-2 module in $\mathcal{O}_K^2$. The general case follows as a by-product in Algorithm 4.3, thanks to the coordinate ideal $\mathcal{C}(M)$. Once a preliminary factorization of $\mathcal{G}(M)$ is done, the algorithm tries to find two "nice" vectors for the input instance $\boldsymbol{G'} = (G', (I_1', I_2'))$. By nice, we mean that we sample random vectors until we obtain somewhat short and linearly independent $\boldsymbol{v}, \boldsymbol{v}'$, such that both $q = \boldsymbol{v}^* G' \boldsymbol{v}$ and $q' = \boldsymbol{v}'^* G' \boldsymbol{v}$ generate a prime multiple of $\mathcal{G}(M)$ (recall that $q, q' \in \mathcal{G}(M)$, so the ideals they generate have to be a multiple of $\mathcal{G}(M)$). As mentioned in the beginning of this section, we will call the `TwoSquares` algorithm for q, q', which requires to know the factorization of $\mathcal{N}_K(q)$ and $\mathcal{N}_K(q')$. Computing these factorizations can be done efficiently once we know the factorization of $\mathcal{N}(\mathcal{G}(M))$, since we ensured that $q/\mathcal{G}(M)$ and $q'/\mathcal{G}(M)$ are prime ideals. Pairing these two-square decompositions gives as many linear equations as there are entries in B: some of them must lead to congruence matrices (others may be non-integral). We actually prove that we find them all this way.

Theorem 4.6 (Assumption 1). *Let K be a totally real number field and $M \subset \mathcal{O}_K^2$ an integer module lattice of rank 2 with pseudo-basis $\boldsymbol{B} = (B, I_1, I_2)$ and associated pseudo-Gram matrix $\boldsymbol{G}$. Algorithm 4.2 takes as input a basis of $\mathcal{O}_K$, the pseudo-matrix $\boldsymbol{B}$ and $\boldsymbol{G'} = (G', I_1', I_2')$ an instance of wc-smodLIP$_K^{\boldsymbol{B}}$ and finds all congruence matrices between $\boldsymbol{G}$ and $\boldsymbol{G'}$. Under Assumption 1, the algorithm runs in expected polynomial time in the size of its input and in*

$$\Big(\mathrm{poly}(\rho_K, \log \Delta_K, \mathrm{size}(\boldsymbol{G'}))\Big)^r + T_{factor}(\mathcal{N}(\mathcal{G}(M))),$$

where ρ_K is the residue of the Dedekind zeta function of K at 1, and r is the number of distinct prime ideals dividing the Gram ideal $\mathcal{G}(M)$.

At Step 13, we actually do not need to factorize again, since standard primality tests will give us both the prime and its valuation. Note that Algorithm 4.2 solves wc-smodLIP$_K^{\boldsymbol{B}}$ on input $\boldsymbol{G'}$, but it does even more than this, since it finds

Algorithm 4.2 Finding all congruence matrices for integer rank-2 modules.

```
Input: A basis $B_K$ of $\mathcal{O}_K$, a pseudo-basis $\boldsymbol{B} = (B, (I_1, I_2))$ of $M \subseteq \mathcal{O}_K^2$ with pseudo-Gram matrix $\boldsymbol{G}$, and $\boldsymbol{G'} = (G', (I'_1, I'_2)) \sim \boldsymbol{G}$ an instance of wc-smodLIP$_K^{\boldsymbol{B}}$.
Output: All congruence matrices between $\boldsymbol{G}$ and $\boldsymbol{G'}$.

$I \leftarrow \mathcal{G}(M)$; $\alpha = \rho_K \cdot P(d)$   (with $P(d)$ from Assumption 1)
Factor $\mathcal{N}(I) = \prod_j q_j^{f_j}$.
Generating two "nice" instances of TwoSquares
$q \leftarrow 0$; $(u, v) \leftarrow (0, 0)$; $s \leftarrow 4^d \cdot \Delta_K^{3/(2d)} \cdot \max_{1 \le j \le 2}(\|\sigma(g'_{j,j})^{1/2}\| \cdot \mathcal{N}(I'_j)^{1/d})$
    (where $G' = (g'_{i,j})_{1 \le i,j \le 2}$)
while $\sqrt{\|\sigma(q)\|_1} > s \cdot \sqrt{8d + \log(4\alpha \log(s))}$ or $q \cdot I^{-1}$ is not a prime ideal do
    $I'_1 \times I'_2 \ni (u, v)^T \leftarrow$ GaussianGram($\boldsymbol{G'}, s$).
    $q \leftarrow (u, v) \cdot G' \cdot (u, v)^T$.
end while
$q' \leftarrow 0$; $(u', v') \leftarrow (0, 0)$; $s' = 144 \cdot d \cdot \Delta_K^{1/d} \cdot \max(\alpha^2, 1) \cdot s^4$.
while $(u', v') \in \mathrm{Span}_K((u, v))$ or $\sqrt{\|\sigma(q)\|_1} > s' \cdot \sqrt{8d + \log(4\alpha \log(s'))}$ or $q' I^{-1}$ is not a prime ideal do
    $I'_1 \times I'_2 \ni (u', v')^T \leftarrow$ GaussianGram($\boldsymbol{G'}, s'$).
    $q' \leftarrow (u', v') \cdot G' \cdot (u', v')^T$.
end while
Solving the two instances of TwoSquares
Factor $\mathcal{N}(qI^{-1}) = p^e$ and $\mathcal{N}(q' I^{-1}) = (p')^f$
$S_1 \leftarrow$ TwoSquares($B_K, q, p^e \cdot \prod_j q_j^{f_j}$); $S_2 \leftarrow$ TwoSquares($B_K, q', (p')^f \cdot \prod_j q_j^{f_j}$)
Recovering the congruence matrices from the solutions to TwoSquares
$S \leftarrow \emptyset$.
for $((t_1, t_2), (t'_1, t'_2)) \in S_1 \times S_2$ do
    $D \leftarrow \begin{pmatrix} t_1 & t'_1 \\ t_2 & t'_2 \end{pmatrix} \cdot \begin{pmatrix} u & u' \\ v & v' \end{pmatrix}^{-1}$
    $V \leftarrow B^{-1} \cdot D$
    if $V$ is a congruence matrix between $\boldsymbol{G}$ and $\boldsymbol{G'}$ then
        $S \leftarrow S \cup \{V\}$.
    end if
end for
return $S$.
```

all congruence matrices between $\boldsymbol{G}$ and $\boldsymbol{G}'$, when wc-smodLIP$_K^{\boldsymbol{B}}$ only asks to find one such congruence matrix.

Proof. Correctness. First, note that thanks to the `if` condition in Steps 19 of the algorithm, all matrices V that are output by our algorithm are indeed congruence matrices between $\boldsymbol{G}$ and $\boldsymbol{G}'$.

Conversely, let us fix $U \in \mathrm{GL}_2(K_{\mathbb{R}})$ a congruence matrix between $\boldsymbol{G}$ and $\boldsymbol{G}'$ and show that $U \in S$ at the end of the algorithm. Let $C = B \cdot U$. By definition of congruence matrices, we know that $\boldsymbol{C} = (C, I_1', I_2')$ is another pseudo-basis of the module M. Moreover, we also know that $U^*GU = G' = C^*C$. This means that $\boldsymbol{G}'$ is the pseudo-Gram matrix of the pseudo-basis $\boldsymbol{C}$ of M.

Let us define $\boldsymbol{z} = (z_1, z_2)^T := C \cdot (u, v)^T$ (where u and v are as in the algorithm, after exiting the while loop in Step 7). Since s satisfies the constraints from Lemma 3.8 (see the discussion below for the choice of s), the algorithm `GaussianGram` is correct, and so we have $u \in I_1'$ and $v \in I_2'$. This implies that $\boldsymbol{z} \in M$. Moreover, we have that $q := (u, v) \cdot G' \cdot (u, v)^T = \boldsymbol{z}^*\boldsymbol{z} = z_1^2 + z_2^2$. Since $M \subset \mathcal{O}_K^2$, the pair (z_1, z_2) thus gives a sum of two squares for q. By Lemma 4.5, `TwoSquares` finds them all so the pair (z_1, z_2) must belong to the set S_1 computed in Step 14. A similar argument works for $\boldsymbol{z}' = (z_1', z_2')^T = C \cdot (u', v')^T$ and S_2.

From this, we know that during the `for` loop of Step 16, there must be one iteration where $(t_1, t_2) = (z_1, z_2)$ and $(t_1', t_2') = (z_1', z_2')$. When this is the case, then the matrix D computed in Step 17 must be equal to C. Note that this computation makes sense since (u, v) and (u', v') are linearly independent. This finally implies that the corresponding matrix V from Step 18 is equal to U, and so $U \in S$ at the end of the algorithm. Overall, this proves that, if the algorithm terminates, then S contains all the congruence matrices between $\boldsymbol{G}$ and $\boldsymbol{G}'$.

Complexity. We have seen that the Gram ideal $\mathcal{G}(M)$ can be computed in polynomial time from the knowledge of $\boldsymbol{G}'$. At Step 2, the norm of the Gram ideal is factored, which takes time $T_{factor}(\mathcal{N}(\mathcal{G}(M)))$ (note that since $M \subseteq \mathcal{O}_K^2$, its Gram ideal $\mathcal{G}(M)$ is integral, and so $\mathcal{N}(\mathcal{G}(M)) \in \mathbb{Z}$).

We discuss the choice of the parameters s and s' for the sampling algorithm. To work under our Assumption 1, they must be chosen above the smoothing parameter $\eta_{1/2}(\sigma(M))$ of the module lattice $\sigma(M)$. Equation (1) tells us that it is enough to take s larger than $\sqrt{\log(12d)/\pi} \cdot \lambda_{2d}(\sigma(M))$. Now, we know from Lemma 2.9 that there exists a basis of $\sigma(M)$ whose vectors are all smaller than $\sqrt{d} \cdot 2^d \cdot \Delta_K^{3/(2d)} \cdot \max_{1 \le j \le 2} \left(\|\sigma(g_{j,j}')^{1/2}\| \cdot \mathcal{N}(I_j')^{1/d} \right)$, so this must be an upper bound on $\lambda_{2d}(\sigma(M))$. Since $2^d \geq \sqrt{d \log(12d)/\pi}$ for all $d \geq 1$, we conclude that $s' \geq s \geq \eta_{1/2}(\sigma(M))$ as needed to apply Assumption 1 and Lemma 3.8.

As a consequence, Lemma 3.8 ensures that steps 5 and 10 run in polynomial time. Moreover, by correctness of the `GaussianGram` algorithm, we know that $\sigma(\boldsymbol{z}) \sim D_{\sigma(M),s}$ and $\sigma(\boldsymbol{z}') \sim D_{\sigma(M),s'}$, where $\boldsymbol{z} = C \cdot (u, v)^T$ and $\boldsymbol{z}' = C \cdot (u', v')$, as before.

Now we estimate the number of trials before satisfying the conditions on Step 4. By Lemma 2.5 applied to $\varepsilon = (4\alpha \log s)^{-1}$, we have $||\sigma(\boldsymbol{z})|| > s \cdot \sqrt{8d + \log(4\alpha \log(s))}$ with probability at most ε. Recall that $q = z_1^2 + z_2^2$, so we have that $\|\sigma(q)\|_1 = \|\sigma(\boldsymbol{z})\|^2$. This implies that the first condition in the while loop is satisfied with probability $\geq 1 - \varepsilon$. For the second condition in the while loop, we have seen that Assumption 1 applies, so qI^{-1} is prime with probability at least $(\alpha \log(s/\mathcal{N}(I)^{1/d}))^{-1} \geq (\alpha \log s)^{-1}$ (recall that $\alpha = \rho_K \cdot P(d)$ and that $M \subseteq \mathcal{O}_K^2$ so that $\mathcal{N}(I) := \mathcal{N}(\mathcal{G}(M)) \geq 1$).

Overall, the probability to exit the first `while` loop in Step 4 is larger than $(\alpha \log s)^{-1} - \varepsilon \geq (2\alpha \log s)^{-1}$, using the definition of ε.[20] The expected number of iterations of the `while` loop is then $\leq 2\alpha \log s = \text{poly}(\log \Delta_K, \rho_K, \text{size}(\boldsymbol{G'}))$.

To conclude on this first while loop, note that all the operations performed during one iteration of the while loop (including the two tests in Step 4) can be performed in time $\text{poly}(\log \Delta_K, \rho_K, \text{size}(\boldsymbol{G'}))$ (for the test that qI^{-1} is prime, we use Corollary 2.13, and we apply it only when the first test passes, which ensures that $\mathcal{N}(qI^{-1}) \leq \mathcal{N}_K(q) \leq \|\sigma(q)\|_1^d$ is not too big).

The reasoning for the second while loop is similar, except that we now also want that (u', v') be non-colinear with (u, v). This is equivalent to asking that $\boldsymbol{z'}$ is not colinear to $\boldsymbol{z}$. In other words, we want $\boldsymbol{z'}$ to avoid the rank-1 submodule $N := M \cap \text{Span}_K(\boldsymbol{z})$. Again, Lemma 3.8 tells us that $\boldsymbol{z'} \sim D_{M,s'}$, hence we want to upper bound

$$\Pr_{\boldsymbol{y} \sim \mathcal{D}_{M,s'}}(\boldsymbol{y} \in N) = \frac{\rho_{s'}(N)}{\rho_{s'}(M)}.$$

To estimate $\rho_{s'}(N)$ and $\rho_{s'}(M)$ we would like to apply Lemma 2.3 (with $\varepsilon = 1/2$). We have already seen that $s \geq \eta_{1/2}(M)$ and by definition $s' \geq s$, so this handles the denominator. Since N has rank 1, observe that $\lambda_d(N) \leq \lambda_d^{(\infty)}(\mathcal{O}_K) \cdot \lambda_1(N) \leq \Delta_K^{1/d} \cdot \|\boldsymbol{z}\|$. Using the upper bound on $\|\boldsymbol{z}\| = \sqrt{\|q\|_1}$ and Equation (1), one can check that $s' \geq \eta_{1/2}(N)$ as desired. We can then apply Lemma 2.3 and use the fact that $\det(N) \geq 1$ since $N \subset \mathcal{O}_K^2$ to obtain

$$\begin{aligned}\Pr_{\boldsymbol{y} \sim \mathcal{D}_{M,s'}}(\boldsymbol{y} \in N) &\leq 3/2 \cdot \frac{\det(M)}{(s')^{2d}} \cdot 2 \cdot \frac{(s')^d}{\det(N)} \leq \frac{3\det(M)}{(s')^d} \\ &\leq \frac{(s')^{d/2}}{4\alpha \cdot (s')^d} \leq \frac{1}{4\alpha \cdot (\sqrt{s'})^d} \leq \frac{1}{4\alpha \log(s')},\end{aligned}$$

where the last inequality follows from the fact that $s' \geq 1$ (so that $(\sqrt{s'})^d \geq \log(s')$) and on the second line we used the fact that $\sqrt{s'} \geq (12\alpha \det(M))^{1/d}$ by choice of s'. Indeed, replacing s' by its value, the latter inequality is equivalent to having $12 \cdot \sqrt{d} \cdot \Delta_K^{1/2d} \cdot \max(\alpha, 1) \cdot s^2 \geq (12\alpha \det(M))^{1/d}$ which is implied by $s^2 \geq (\det(M))^{1/d}$. We know that $\det(M) \leq \lambda_{2d}(M)^{2d}$ (this holds in any lattice), so it is sufficient to prove that $s \geq \lambda_{2d}(M)$, which we have already proved since this was required for the Gaussian sampling algorithm `GaussianGram` (see Lemma 3.8).

Similarly to the first `while` loop, we have that, under Assumption 1, the probability that $q'I^{-1}$ is prime is $\geq (\alpha \log(s'))^{-1}$. The probability that $\sqrt{\|q\|_1} =$

[20] Here we also use the general fact that for two events A and B, we can upper bound $\Pr(A \cap B) = \Pr(A)$ - $\Pr(A \cap \neg B) \geq \Pr(A) - \Pr(\neg B)$.

$\|\boldsymbol{z}'\|$ is larger than the bound in the condition is $\leq (4\alpha \log(s'))^{-1}$ (the argument is again similar to the first loop). Combining everything, the probability to exit the second while loop is at least

$$\frac{1}{\alpha \log(s')} - \frac{1}{4\alpha \log(s')} - \frac{1}{4\alpha \log(s')} = \frac{1}{2\alpha \log(s')}.$$

We conclude that the expected number of iterations of the second while loop is $\mathrm{poly}(\log \Delta_K, \rho_K, \mathrm{size}(\boldsymbol{G'}))$. Similarly to the first while loop, all the operations performed during one execution of the while loop can be done in time $\mathrm{poly}(\log \Delta_K, \rho_K, \mathrm{size}(\boldsymbol{G'}))$.

Step 13 can be done in polynomial time since $\mathcal{N}(qI^{-1})$ and $\mathcal{N}(q'I^{-1})$ are prime powers. The two calls to the `TwoSquares` algorithm in Step 14 can be performed in time $\mathrm{poly}\big(\log \Delta_K, (\log|\mathcal{N}_K(q)|)^{r+1}\big) = \mathrm{poly}(\log \Delta_K, \rho_K, \mathrm{size}(\boldsymbol{G'}))^r$. Here we used the fact that $q\mathcal{O}_K$ and $q'\mathcal{O}_K$ have at most $r+1$ distinct prime factors since $I = \mathcal{G}(M)$ has r distinct prime factors and qI^{-1} and $q'I^{-1}$ are prime ideals. We also used the the fact that $\log|\mathcal{N}_K(q)|$ and $\log|\mathcal{N}_K(q')|$ are $\mathrm{poly}(\log \Delta_K, \rho_K, \mathrm{size}(\boldsymbol{G'}))$, thanks to the conditions on $\|\sigma(q)\|_1$ and $\|\sigma(q')\|_1$ in the while loops.

Finally, let us consider the final `for` loop from Step 16. Each step in this loop can be done in polynomial time. For the number of iterations of the loop, the proof of Theorem 2.16 gives $|S_1| \leq d^2(\log(|\mathcal{N}_K(q)|) + 1)^{r+1}$ and the same holds for $|S_2|$. Hence, the number of iterations of the loop is upper bounded by $\mathrm{poly}(\log \Delta_K, \rho_K, \mathrm{size}(\boldsymbol{G'}))^r$. This concludes the analysis of the running time of the algorithm. □

Before extending the previous result to modules in K^2, we focus a little bit on the particular case when $M = \mathcal{O}_K^2$ and K is the maximal totally real subfield of a cyclotomic field $L = \mathbb{Q}(\zeta_m)$. In this case, the Gram ideal is simply $\mathcal{O}_K$, leading to a polynomial complexity in d (the degree of K), the residue ρ_K and the size of the input.

Corollary 4.7 (Assumption 1). *Let K be the maximal totally real subfield of a cyclotomic field, and define $\boldsymbol{I_2} = (I_2, (\mathcal{O}_K)_{1\leq i\leq 2})$ (this is a pseudo-basis of the rank-2 module $\mathcal{O}_K^2$). Under Assumption 1, there exists a probabilistic algorithm solving* wc-smodLIP$_K^{\boldsymbol{I_2}}$ *in expected polynomial time in the degree d of K, the residue ρ_K, and in the size of the input.*

Proof. Since $M = \mathcal{O}_K^2$, the Gram ideal is trivial thus with the notation of Theorem 4.6, $r = 1$, and Algorithm 4.2 solves wc-smodLIP$_K^{\boldsymbol{I_2}}$ in expected polynomial time in ρ_K, $\log \Delta_K$ and the size of the inputs. Also, notice that the discriminant of K can be computed ([38, Proposition 3.1]) and verifies $\log \Delta_K = \mathrm{poly}(d)$. □

Corollary 4.8 (Assumption 1). *Let K be a totally real number field and $M \subset K^2$ a module lattice of rank 2 with pseudo-basis $\boldsymbol{B} = (B, I_1, I_2)$ and associated pseudo-Gram matrix $\boldsymbol{G}$. There exists a probabilistic algorithm (Algorithm 4.3) that takes as input a basis of $\mathcal{O}_K$, the pseudo-basis $\boldsymbol{B}$, and $\boldsymbol{G'} = (G', I_1', I_2')$ an*

Algorithm 4.3 Finding all congruence matrices for rank-2 modules.

```
Input: A basis of $\mathcal{O}_K^2$, a pseudo-basis $\boldsymbol{B} = (B, (I_1, I_2))$ of $M \subset K^2$, with pseudo-Gram
    matrix $\boldsymbol{G}$, and $\boldsymbol{G'} = (G, (I'_1, I'_2)) \sim \boldsymbol{G}$ an instance of wc-smodLIP$_K^{\boldsymbol{B}}$.
Output: All congruence matrices between $\boldsymbol{G}$ and $\boldsymbol{G'}$.
1: $J \leftarrow \mathcal{C}(M)^{-1}$
2: $\boldsymbol{B_J} \leftarrow (B, \{J \cdot I_i\}_{i=1,2})$ ; $\boldsymbol{G_J} \leftarrow (G, \{J \cdot I_i\}_{i=1,2})$ ; $\boldsymbol{G'_J} \leftarrow (G', \{J \cdot I'_i\}_{i=1,2})$
3: $S \leftarrow$ Run Algorithm 4.2 with $\boldsymbol{B_J}, \boldsymbol{G_J}$ and $\boldsymbol{G'_J}$
4: return $S$.
```

instance of wc-smodLIP$_K^{\boldsymbol{B}}$ and finds all congruence matrices between $\boldsymbol{G}$ and $\boldsymbol{G'}$. Under Assumption 1, the algorithm runs in expected time polynomial in its input size and in

$$\Big(\mathrm{poly}(\rho_K, \log \Delta_K, \mathrm{size}(\boldsymbol{G'}))\Big)^r + T_{factor}(\mathcal{N}(\mathcal{RG}(M))),$$

where r is the number of distinct prime ideals dividing the relative Gram ideal $\mathcal{RG}(M)$.

Proof. Correctness. By Lemma 4.4 (4), $M' = \mathcal{C}(M)^{-1} \cdot M$ is an integer module lattice not contained in $\mathfrak{p}^2$ for any prime ideal $\mathfrak{p} \subset \mathcal{O}_K$. Using (3) we get $\mathcal{RG}(M) = \mathcal{RG}(M') = \mathcal{G}(M')$ (since $\mathcal{C}(M') = \mathcal{O}_K$). The last thing to observe is that scaling the module doesn't change the set of solutions to modLIP. This is because a congruence matrix U between $\boldsymbol{G_J} := (G, J \cdot I_1, J \cdot I_2)$ and $\boldsymbol{G'_J} := (G', J \cdot I'_1, J \cdot I'_2)$ satisfies $G' = U^*GU$ and $u_{i,j} \in (J \cdot I_i) \cdot (J \cdot I'_j)^{-1} = I_i(I'_j)^{-1}$, so it is also a congruence matrix between $\boldsymbol{G}$ and $\boldsymbol{G'}$.

Complexity. The relative Gram ideal can be computed in polynomial time in the size of $\boldsymbol{B}$ and $\boldsymbol{G'_J}$ has size $\mathrm{poly}(\mathrm{size}(\boldsymbol{G'}), \mathrm{size}(J)) = \mathrm{poly}(\mathrm{size}(\boldsymbol{G'}))$, thus the complexity follows from Theorem 4.6. □

5 Implementation of the algorithm

We have implemented a proof-of-concept of our algorithm mixing `Sagemath` and `PARI/GP`, for the totally real (maximal) subfield K of cyclotomic fields $L = \mathbb{Q}(\zeta_m)$. The current version of the code only works for fields with conductor $m = 4k$: this ensures that L contains a primitive 4-th root of unity (equivalently $L = K[X]/(X^2 + 1)$), which simplifies the code and covers the cryptographic relevant case where $m = 2^t$. This also allows us to use our implementation of the Gentry-Szydlo algorithm as in this case, $K[X]/(X^2+1) = L$ is a cyclotomic field. For other conductors or more general CM-extensions, a previous version of the code worked on toy-sized examples as a proof-of-concept, but we did not push more in this direction.

Our approach follows the structure of Algorithm 4.2. We give more details in Appendix C, and refer to our code in our public repository. Nonetheless, we briefly report some experimental results and observations. In our experiments, we selected instances (Q, B), with B a basis of $\mathcal{O}_K^2$ and $Q = B^T \cdot B$, where B had a quite short column (to emulate the situation in Hawk) but usually the second column would be much bigger. Our values of choice for m included the power-of-two case but also various other of the form $4k$ (see Table 1) below). In the table, we also display the quantity $2d$, which is the dimension over $\mathbb{Z}$ of the module lattices involved in the problem (recall that they are rank-2 modules over K, and that $d = \deg(K)$). For a larger experiment, using `PARI/GP` 2.16 and its LLL relying on the "Flatter" algorithm [32], we could also solve the congruence problem for $m = 512$ (lattices of dimension $2d = 256$) in about 29 hours, essentially all spent in the two call to Gentry-Szydlo algorithm.

$(m, 2d)$	$(64, 32)$	$(128, 64)$	$(256, 128)$
Time	2	25	850

$(m, 2d)$	$(124, 60)$	$(204, 64)$	$(228, 72)$	$(276, 88)$	$(260, 96)$	$(232, 112)$	$(340, 128)$	$(296, 144)$
Time (s)	33	53	74	195	434	652	2980	4205

Table 1. Times in seconds for attacks over various maximal totally real subfields K of cyclotomic fields with conductors $m = 4k$, averaged over 5 instances. The degree d of K is $\varphi(m)/2$, and the lattices involved have dimension $2d$. The upper table are powers-of-two. Experiments performed on a MacBook Pro (Apple M2), with Sagemath 10.2 and Pari/GP 2.15.5.

Acknowledgments. We are grateful to Aurel Page and Bill Allombert for their help with the implementation of the Gentry-Szydlo algorithm, Sébastien Labbé, Xavier Caruso and Vincent Delecroix for organizing the Sage Days 125, where a significant part of the implementation took place. Thanks to Paul Kirchner and Thomas Espitau for their preliminary implementation of Gentry-Szydlo that helped for proof-of-concepts. Finally we thank Koen de Boer and Wessel van Woerden for enlightening discussions.
Guilhem Mureau and Alice Pellet-Mary were supported by the CHARM ANR-NSF grant (ANR-21-CE94-0003). All the authors were supported by the PEPR quantique France 2030 programme (ANR-22-PETQ-0008). Alice Pellet-Mary was supported by the TOTORO ANR (ANR-23-CE48-0002).

References

1. Belabas, K., van Hoeij, M., Klüners, J., Steel, A.: Factoring polynomials over global fields. Journal de théorie des nombres de Bordeaux **21**(1), 15–39 (2009)
2. Bennett, H., Ganju, A., Peetathawatchai, P., Stephens-Davidowitz, N.: Just how hard are rotations of $\mathbb{Z}^n$? algorithms and cryptography with the simplest lattice. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 252–281. Springer (2023)

3. Bhargava, M., Shankar, A., Taniguchi, T., Thorne, F., Tsimerman, J., Zhao, Y.: Bounds on 2-torsion in class groups of number fields and integral points on elliptic curves. Journal of the American Mathematical Society **33**(4), 1087–1099 (2020)
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 and application to the cryptanalysis of a fhe scheme. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 60–88. Springer (2017)
5. Boer, K.d.: Random walks on Arakelov class groups. Ph.D. thesis, Leiden University (2022)
6. Brakerski, Z., Langlois, A., Peikert, C., Regev, O., Stehlé, D.: Classical hardness of learning with errors. In: Proceedings of the forty-fifth annual ACM symposium on Theory of computing. pp. 575–584 (2013)
7. Bruin, P.J., Ducas, L., Gibbons, S.: Genus distribution of random q-ary lattices. Cryptology ePrint Archive (2022)
8. Buchmann, J.A., Lenstra, H.W.: Computing maximal orders and factoring over $\mathbb{Z}_p$. Preprint (1994)
9. Buchmann, J.A., Lenstra, H.W.: Approximatting rings of integers in number fields. Journal de théorie des nombres de Bordeaux **6**(2), 221–260 (1994)
10. Cohen, H.: Advanced topics in computational number theory, vol. 193. Springer Science & Business Media (2012)
11. Cohen, H.: A course in computational algebraic number theory, vol. 138. Springer Science & Business Media (2013)
12. Ducas, L.: Provable lattice reduction of $\mathbb{Z}^n$ with blocksize n/2. Cryptology ePrint Archive (2023)
13. Ducas, L., Gibbons, S.: Hull attacks on the lattice isomorphism problem. In: IACR International Conference on Public-Key Cryptography. pp. 177–204. Springer (2023)
14. Ducas, L., Postlethwaite, E.W., Pulles, L.N., Woerden, W.v.: Hawk: Module lip makes lattice signatures fast, compact and simple. In: Advances in Cryptology–ASIACRYPT 2022: 28th International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, December 5–9, 2022, Proceedings, Part IV. pp. 65–94. Springer (2023)
15. Ducas, L., van Woerden, W.: On the lattice isomorphism problem, quadratic forms, remarkable lattices, and cryptography. In: Advances in Cryptology–EUROCRYPT 2022: 41st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Trondheim, Norway, May 30–June 3, 2022, Proceedings, Part III. pp. 643–673. Springer (2022)
16. Dutour Sikirić, M., Haensch, A., Voight, J., van Woerden, W.P.: A canonical form for positive definite matrices. Open Book Series **4**(1), 179–195 (2020)
17. Felderhoff, J., Pellet-Mary, A., Stehlé, D., Wesolowski, B.: Ideal-svp is hard for small-norm uniform prime ideals. In: Theory of Cryptography Conference. pp. 63–92. Springer (2023)
18. Fieker, C., Stehlé, D.: Short bases of lattices over number fields. In: International Algorithmic Number Theory Symposium – ANTS. pp. 157–173. Springer (2010)
19. Garg, S., Gentry, C., Halevi, S.: Candidate multilinear maps from ideal lattices. In: Advances in Cryptology–EUROCRYPT 2013: 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings 32. pp. 1–17. Springer (2013)
20. Gentry, C., Szydlo, M.: Cryptanalysis of the revised ntru signature scheme. In: Advances in Cryptology—EUROCRYPT 2002. pp. 299–320. Springer (2002)

21. Haviv, I., Regev, O.: On the lattice isomorphism problem. In: Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms. pp. 391–404. SIAM (2014)
22. Howgrave-Graham, N., Szydlo, M.: A method to solve cyclotomic norm equations. In: International Algorithmic Number Theory Symposium. pp. 272–279. Springer (2004)
23. Kirchner, P.: Algorithms on ideal over complex multiplication order. arXiv preprint arXiv:1602.09037 (2016)
24. Lenstra Jr, H.W., Silverberg, A.: Testing isomorphism of lattices over cm-orders. SIAM Journal on Computing **48**(4), 1300–1334 (2019)
25. Louboutin, S.: Explicit bounds for residues of dedekind zeta functions, values of l-functions at s= 1, and relative class numbers. Journal of Number Theory **85**(2), 263–282 (2000)
26. Marcus, D.A., Sacco, E.: Number fields, vol. 1995. Springer (1977)
27. Mehta, S.K., Rajasree, M.S.: On the bases of z n lattice. In: 2022 24th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC). pp. 100–107. IEEE (2022)
28. Micciancio, D., Regev, O.: Worst-case to average-case reductions based on gaussian measures. SIAM Journal on Computing **37**(1), 267–302 (2007)
29. Narkiewicz, W.: Elementary and analytic theory of algebraic numbers, vol. 57. Springer (1974)
30. Neukirch, J.: Algebraic number theory, vol. 322. Springer Science & Business Media (2013)
31. Plesken, W., Souvignier, B.: Computing isometries of lattices. Journal of Symbolic Computation **24**(3-4), 327–334 (1997)
32. Ryan, K., Heninger, N.: Fast practical lattice reduction through iterated compression. In: Handschuh, H., Lysyanskaya, A. (eds.) Advances in Cryptology - CRYPTO 2023 - Proceedings, Part III. Lecture Notes in Computer Science, vol. 14083, pp. 3–36. Springer (2023)
33. Serre, J.P.: Local fields, vol. 67. Springer Science & Business Media (2013)
34. Szydlo, M.: Hypercubic lattice reduction and analysis of ggh and ntru signatures. In: Advances in Cryptology—EUROCRYPT 2003: International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4–8, 2003 Proceedings 22. pp. 433–448. Springer (2003)
35. The PARI Group, Univ. Bordeaux: PARI/GP version `2.16.2` (2024), available from http://pari.math.u-bordeaux.fr/
36. The Sage Developers: SageMath, the Sage Mathematics Software System (Version 10.3.0) (2024), `https://www.sagemath.org`
37. Washington, L.C.: Introduction to cyclotomic fields, vol. 83. Springer Science & Business Media (1997)
38. Yamagata, K., Yamagishi, M.: On the ring of integers of real cyclotomic fields (2016)

A Worst-case to average-case reduction for module-LIP

In [14] is given an average-case version of a module-LIP problem in which the instances of the problem are sampled following a certain distribution. This is used to generate secret keys (*i.e.* congruence matrices) and the security of the signature scheme Hawk thus lies on the average-case version of this problem (see [14, Algorithm 1 and Algorithm 6]). To ensure security, a worst-case to average-case reduction is proved ([14, Lemma 5]).

This appendix contains somehow a generalization of the result mentioned above.[21] To do so, we provide a polynomial time algorithm (Algorithm A.1) which, given as inputs a pseudo-matrix $\boldsymbol{B}$ (with associated pseudo-Gram matrix $\boldsymbol{G}$) and a parameter $s > 0$, samples congruent forms $\boldsymbol{G'} \sim \boldsymbol{G}$, together with a congruence matrix between them. An average-case version of module-LIP follows naturally and we prove a worst-case to average-case reduction. That means that solving module-LIP with an instance generated from this distribution is as hard as solving module-LIP for any instance, up to some polynomial-time operations.

Sampling module vectors. From $\boldsymbol{G}$ we can define a distribution $\mathcal{D}_{\boldsymbol{G},s}$ (with parameter $s > 0$) on $I_1 \times \cdots \times I_\ell$ as usual (Definition A.1). Fixing $\mathbb{Z}$-bases of the coefficient ideals, we can use Minkowski embedding to associate a positive definite quadratic form $\psi(\boldsymbol{G}) \in \mathcal{S}_{d\ell}^{>0}(\mathbb{R})$ (Definition A.2). In this paragraph we prove that $\mathcal{D}_{\boldsymbol{G},s}$ and the distribution on $I_1 \times \cdots \times I_\ell$ obtained by sampling the coordinates (in $\mathbb{Z}^{d\ell}$) with $\texttt{DiscreteGaussian}(\psi(\boldsymbol{G}), s)$ coincide (Lemma A.4). This is a more detailed proof of the fact that the outputs to `GaussianGram` introduced in Lemma 3.8 follow a Gaussian distribution.

Definition A.1. *The distribution $\mathcal{D}_{\boldsymbol{G},s}$ on $I_1 \times \cdots \times I_\ell$ with parameter $s > 0$ is defined by*

$$\underset{X \sim \mathcal{D}_{I_1 \times \cdots \times I_\ell, s}}{\mathbb{P}(X = x)} := \frac{\exp(-\pi ||x||_G^2 / s^2)}{\sum\limits_{y \in I_1 \times \cdots \times I_\ell} \exp(-\pi ||y||_G^2 / s^2)},$$

where $x \in I_1 \times \cdots \times I_\ell$ and $||y||_G^2 := Tr\Big(y^* G y\Big) \geq 0$, for any $y \in I_1 \times \cdots \times I_\ell$.

Definition A.2. *For any $g \in K$ and ordered set $\alpha = \{\alpha_1, \ldots, \alpha_d\} \subset K$ we put*

$$\psi_\alpha(g) := \Big(Tr\Big(\overline{\alpha_i} \cdot g \cdot \alpha_j\Big)\Big)_{1 \leq i,j \leq d} \in \mathcal{M}_d(\mathbb{R}).$$

[21] Module-LIP as defined in [14] asks for a congruence matrix with determinant one, this is not a condition we kept in our definition of module-LIP. Also, the distribution used in [14] for key generation is different from the one we will present so the average-case problems are not the same.

Now writing $G = (g_{i,j})_{1 \leq i,j \leq \ell}$ *and given* $\mathbb{Z}$*-bases* $\alpha^{(j)} = \{\alpha_1^{(j)}, \ldots, \alpha_d^{(j)}\}$ *of* I_j *for every* $j \in \{1, \ldots, \ell\}$*, we define*

$$\psi(\boldsymbol{G}) := \begin{pmatrix} \boxed{\psi_{\alpha^{(1)}}(g_{1,1})} & \cdots & \boxed{\psi_{\alpha^{(\ell)}}(g_{1,\ell})} \\ \vdots & \cdots & \vdots \\ \boxed{\psi_{\alpha^{(1)}}(g_{\ell,1})} & \cdots & \boxed{\psi_{\alpha^{(\ell)}}(g_{\ell,\ell})} \end{pmatrix} \in \mathcal{S}_{d\ell}(\mathbb{R}).$$

Note that this depends on the choice of a $\mathbb{Z}$-basis for each coefficient ideals. When the notation $\psi(\boldsymbol{G})$ is used, we mean that LLL-reduced $\mathbb{Z}$-bases of the coefficient ideals are fixed. Now we prove that this is indeed the matrix of a positive definite quadratic form. For a matrix $B = (\boldsymbol{b}_1 | \ldots | \boldsymbol{b}_\ell) \in \mathcal{M}_\ell(K_\mathbb{R})$ and ideals $I_1, \ldots, I_\ell$ with respective $\mathbb{Z}$-bases $\alpha^{(1)}, \ldots, \alpha^{(\ell)}$, we define $\widetilde{B} \in \mathcal{M}_{d\ell}(\mathbb{R})$ whose columns are given by $\tilde{\boldsymbol{b}}_{j(d-1)+i} = \sigma(\alpha_i^{(j)} \cdot \boldsymbol{b}_j) \in \mathbb{R}^{d\ell}$, for $1 \leq j \leq \ell$ and $1 \leq i \leq d$.

Lemma A.3. *Given* $B = (\boldsymbol{b}_1 | \ldots | \boldsymbol{b}_\ell) \in \mathrm{GL}_\ell(K_\mathbb{R})$ *such that* $G = B^* \cdot B$*, we have*

$$\psi(\boldsymbol{G}) = \widetilde{B}^T \cdot \widetilde{B},$$

In particular $\psi(\boldsymbol{G})$ *is symmetric, positive and definite.*

Proof. Let us write $G = (g_{i,j})_{1 \leq i,j \leq \ell}$ and let $1 \leq s, t \leq d\ell$. There are unique $1 \leq i_s, i_t \leq d$ and $1 \leq j_s, j_t \leq \ell$ such that $s = j_s(d-1) + i_s$ and $t = j_t(d-1) + i_t$. Then the coefficient (s,t) on the right hand side is

$$\begin{aligned} \langle \tilde{\boldsymbol{b}}_s, \tilde{\boldsymbol{b}}_t \rangle_{\mathbb{C}^{d\ell}} &= \langle \sigma(\alpha_{i_s}^{(j_s)} \cdot \boldsymbol{b}_{j_s}), \sigma(\alpha_{i_t}^{(j_t)} \cdot \boldsymbol{b}_{j_t}) \rangle_{\mathbb{C}^{d\ell}} \\ &= Tr(\overline{\alpha_{i_s}^{(j_s)}} \cdot \boldsymbol{b}_{j_s}^* \cdot \boldsymbol{b}_{j_t} \cdot \alpha_{i_t}^{(j_t)}) \\ &= Tr(\overline{\alpha_{i_s}^{(j_s)}} \cdot g_{j_s,j_t} \cdot \alpha_{i_t}^{(j_t)}), \end{aligned}$$

which is by definition the coefficient (s,t) of $\psi(\mathbf{G})$.

Lemma A.4. *Let* $\alpha^{(j)} = \{\alpha_1^{(j)}, \ldots, \alpha_d^{(j)}\}$ *be a* $\mathbb{Z}$*-basis of* I_j *and* $x := (x_1, \ldots, x_l) \in I_1 \times \cdots \times I_\ell$*. For every* $1 \leq j \leq \ell$ *we put* $(z_i^{(j)})_i$ *the coefficients of* x_j *in the* $\mathbb{Z}$*-basis of* I_j *i.e.,* $x_j = \sum_{i=1}^d z_i^{(j)} \alpha_i^{(j)}$ *and* $z = (z_i^{(j)})_{i,j} \in \mathbb{Z}^{dl}$*. Then for any* $s > 0$*,*

$$\underset{X \sim \mathcal{D}_{I_1 \times \cdots \times I_\ell, s}}{\mathbb{P}}(X = x) = \underset{Z \sim \mathcal{D}_{\psi(\boldsymbol{G}), s}}{\mathbb{P}}(Z = z).$$

Proof. Let $s > 0$. It is enough to prove that

$$\exp(-\pi ||x||_G^2 / s^2) = \rho_{\psi(\mathbf{G}),s}(z) = \exp(-\pi (z^T \psi(\mathbf{G}) z)/s^2).$$

Let $B = (\boldsymbol{b}_1|\dots|\boldsymbol{b}_\ell) \in \mathrm{GL}_\ell(K_\mathbb{R})$ such that $G = B^*B$, then by the previous lemma $z^T\psi(\mathbf{G})z = (\widetilde{B}z)^T \cdot (\widetilde{B}z)$ and by definition of $\widetilde{B}$,

$$\begin{aligned}\widetilde{B} \cdot z &= \sum_{j=1}^{d}\sum_{i=1}^{\ell} \tilde{\boldsymbol{b}}_{j(d-1)+i} \cdot z_i^{(j)} \\ &= \sigma\Big(\sum_{j=1}^{\ell} \boldsymbol{b}_j \sum_{i=1}^{d} \alpha_i^{(j)} \cdot z_i^{(j)}\Big) \\ &= \sigma\Big(\sum_{j=1}^{\ell} \boldsymbol{b}_j \cdot x_j\Big) \\ &= \sigma(B \cdot x).\end{aligned}$$

Therefore we obtain as expected,

$$\begin{aligned}z^T\psi(\mathbf{G})z &= \langle \widetilde{B}z, \widetilde{B}z\rangle_{\mathbb{C}^{d\ell}} \\ &= Tr((Bx)^*(Bx)) \\ &= Tr(x^*Gx) \\ &= ||x||_G^2.\end{aligned}$$

Sampling congruent forms. Let us fix a pseudo-Gram matrix $\boldsymbol{G}$ associated to the pseudo-basis $\boldsymbol{B}$ of a module lattice M. Using the algorithm `GaussianGram` of Lemma 3.8 with input $\boldsymbol{G}$ and parameter $s > 0$ we can sample in the product $I_1 \times \dots \times I_\ell$ (recall that the coefficient ideals are represented by fixed $\mathbb{Z}$-bases). Following the idea developed in [18, Fig. 2] we sample enough vectors (enough to get a rank ℓ matrix) and apply the CHNF algorithm to extract another pseudo-basis of M together with the pseudo-bases change U. Then from $\boldsymbol{G}$ and U we can build a congruent pseudo-Gram matrix $\boldsymbol{G'} \sim \boldsymbol{G}$.

Lemma A.5. *For any pseudo-matrix $\boldsymbol{B} = (B, (I_j)_{1\le j\le\ell})$ with $B \in \mathrm{GL}_\ell(K)$ and associated pseudo-Gram matrix $\boldsymbol{G} = ((g_{i,j})_{1\le i,j\le\ell}, (I_j)_{1\le j\le\ell}))$, and parameter*

$$s \ge \sqrt{\frac{d \cdot \log(2d\ell+4)}{\pi}} \cdot 2^d \cdot \Delta_K^{3/(2d)} \cdot \max_{1\le j\le\ell} \Big(\|\sigma(g_{j,j})^{1/2}\| \cdot \mathcal{N}(I_j)^{1/d}\Big),$$

Algorithm A.1 returns a pseudo-Gram matrix $\boldsymbol{G'} = (G', (J_j)_{1\le j\le\ell})$ congruent to $\boldsymbol{G} := (B^*B, (I_j)_{1\le j\le l})$ together with a congruence matrix U between $\boldsymbol{G}$ and $\boldsymbol{G'}$. Is it a probabilistic algorithm which runs in expected time poly$(d, \ell, \log s)$.

Moreover, the result depends only on the equivalence class of the input, in the sense that for any congruent pseudo-Gram matrix $\boldsymbol{H} = (W^*GW, (H_i)_{1\le i\le\ell})$, running steps 9-11 with $\boldsymbol{H}$ and $W^{-1}Y$ instead of $\boldsymbol{G}$ and Y gives the same output. This defines a Gaussian distribution on $[\boldsymbol{G}]$ with parameter s, denoted $\mathcal{D}_s([\boldsymbol{G}])$.

Algorithm A.1 Sample congruent pseudo-Gram matrix.

```
Input: A pseudo-Gram matrix $\boldsymbol{G} = (G, (I_i)_{1\leq i\leq \ell})$ coming from a known pseudo-basis
    of $M$, and a parameter $s \geq \sqrt{\frac{d\cdot\log(2d\ell+4)}{\pi}} \cdot 2^d \cdot \Delta_K^{3/(2d)} \cdot \max_{1\leq j\leq\ell}\left(\|\sigma(g_{j,j})^{1/2}\| \cdot \mathcal{N}(I_j)^{1/d}\right)$.
Output: A pseudo-Gram matrix $\boldsymbol{G'} = (G', (J_i)_{1\leq i\leq\ell})$ equivalent to $\boldsymbol{G}$ together with
    a congruence matrix $U$ between $\boldsymbol{G}$ and $\boldsymbol{G'}$.
 1: $C \leftarrow 1 - (1 + e^{-\pi})^{-1}$ and $m \leftarrow \lceil \frac{2\ell}{C} \rceil$
 2: for $1 \leq i \leq m$ do
 3:     $I_1 \times \cdots \times I_\ell \ni y_i \leftarrow$ GaussianGram($\boldsymbol{G}, s$)
 4: end for
 5: $Y \leftarrow (y_1 \,|\, \ldots \,|\, y_m) \in \mathcal{M}_{\ell\times m}(K)$
 6: if $Y$ has rank $< \ell$ then
 7:     Restart
 8: end if
 9: $(H, (J_i^{-1})_{1\leq i\leq\ell}, U_0) \leftarrow$ CHNF$(Y^T, (I_i^{-1})_{1\leq i\leq\ell})$, [10, Algorithm 1.4.7].
10: $U \leftarrow U_0^{-T}$
11: return $\boldsymbol{G'} = (U^*GU, (J_i)_{1\leq i\leq\ell}), U)$
```

Proof. Correctness. At step 9, properties of the CHNF give $Y^TU_0 = H$ and $u^0_{i,j} \in I_i^{-1}J_j$ (where $U_0 = (u^0_{i,j})_{1\leq i,j\leq\ell}$), $v^0_{i,j} \in J_i^{-1}I_j$ (where $U_0^{-1} = (v^0_{i,j})_{1\leq i,j\leq\ell}$). Thus at step 10, the matrix $U = U_0^{-T}$ has coefficient (i,j) in $I_iJ_j^{-1}$ and it is a congruence matrix between $\boldsymbol{G}$ and $\boldsymbol{G}'$ (with notations of step 11). Thus the algorithm ensures a pseudo-Gram matrix $\boldsymbol{G'} \sim \boldsymbol{G}$ together with the congruence matrix.

Complexity. Sampling from GaussianGram (thanks to Lemma 3.8) and the CHNF Algorithm [10, Algorithm 1.4.7] run in polynomial time in d, ℓ and $\log s$. We need to estimate the probability of failure at step 6. Let T be the random variable counting the number of iteration before founding a set of full rank vectors. By Lemma 5.1 of [21] (and because $s \geq \max_{1\leq j\leq\ell}\|\sigma(g_{j,j})^{1/2}\| \geq \lambda_{d\ell}(\sigma(M))$), for any $i \in \{1, \ldots, d\ell - 1\}$ and set of vectors $\{y_1, \ldots, y_i\}$ sampled from GaussianGram($\boldsymbol{G}, s$), the probability that $y \leftarrow$ GaussianGram($\boldsymbol{G}, s$) does not belong to the span of $y_1, \ldots, y_i$ is greater than $C := 1 - (1 + e^{-\pi})^{-1}$. Let $m = \lceil\frac{2\ell}{C}\rceil$ and $X_1, \ldots, X_m$ be Bernoulli variables with success parameter C, and $S_m = X_1 + \cdots + X_m$. Then, the probability p_{fail} of not finding ℓ linearly independent vectors within m sampled vectors is upper bounded by the probability $\mathbb{P}(S_m \leq \ell - 1)$, where S_m follows a binomial distribution with parameters m and C *i.e.*,

$$p_{fail} \leq \mathbb{P}(S_m \leq \ell - 1).$$

Using Hoeffding's inequality,

$$\begin{aligned} p_{fail} &\leq \mathbb{P}\Big(\frac{S_m}{m} - C \leq \frac{\ell-1}{m} - C\Big) \\ &\leq \exp\Big(-2m\Big(C - \frac{\ell-1}{m}\Big)^2\Big) \\ &\leq \exp(-mC) \\ &\leq \exp(-2\ell) \\ &\leq e^{-2}. \end{aligned}$$

Therefore,

$$\mathbb{E}[T] \leq \frac{1}{1 - p_{\text{fail}}} < 2.$$

Independance from $\boldsymbol{G}$*.* Finally we prove that the result depends only on the equivalence class of $\boldsymbol{G}$. Let $(W^*GW, (I'_i)_{1\leq i\leq \ell})$ be equivalent to $\boldsymbol{G}$, where $W = (w_{i,j})_{1\leq i,j\leq \ell} \in \mathrm{GL}_\ell(K)$. Denote by $(H', ({J'}_i^{-1})_{1\leq i\leq \ell}, U'_0)$ the CHNF Algorithm of [10] applied to the pseudo-matrix $(Y^TW^{-T}, ({I'}_i^{-1})_{1\leq i\leq \ell})$ and $U' = {U'}_0^{-T}$. By unicity of the CHNF (there is a unique pseudo-matrix in CHNF in the orbit of $\boldsymbol{G}$ for the multiplication equivalence relation), we obtain

$$(U')^{-1}W^{-1}Y = {U'}_0^TW^{-1}Y = (Y^TW^{-T}U'_0)^T = H' = H = (Y^TU_0)^T = U^{-1}Y,$$

so $(U')^{-1}W^{-1} = U^{-1}$ and $U' = W^{-1}U$. Then,

$$(U')^*W^*GWU' = U^*W^{-*}W^*GWW^{-1}U = U^*GU.$$

Also, the unicity of the CHNF implies $J'_i = J_i$ for all $1 \leq i \leq \ell$ so the pseudo-Gram matrix returned is the same and this concludes the proof. □

Average-case problem. Now that we have an efficient algorithm to generate instances of module-LIP, it leads us to an average-case version of the problem.

Definition A.6 (ac-smodLIP$_K^{\boldsymbol{B},s}$). *For* $\boldsymbol{B}$ *a pseudo-basis of a module lattice* $M \subset K^\ell$ *with associated pseudo-Gram matrix* $\boldsymbol{G}$*, the average-case search module-Lattice Isomorphism Problem with parameter* K*,* $\boldsymbol{B}$ *and* s *denoted by ac-smodLIP*$_K^{\boldsymbol{B},s}$ *is, given as input any pseudo-Gram matrix* $\boldsymbol{G'} \sim \boldsymbol{G}$ *sampled from Algorithm A.1 with parameters* $\boldsymbol{G}$ *and* $s > 0$*, to find a congruence matrix between* $\boldsymbol{G}$ *and* $\boldsymbol{G'}$*.*

The following results states that an oracle solving ac-smodLIP$_K^{\boldsymbol{B},s}$ also solves wc-smodLIP$_K^{\boldsymbol{B}}$, up to a call to Algorithm A.1.

Proposition A.7 (ac-smodLIP$_K^{\boldsymbol{B},s}$ $\geq$ wc-smodLIP$_K^{\boldsymbol{B}}$). *Given an oracle that solves ac-smodLIP*$_K^{\boldsymbol{B},s}$ *in time* τ *with probability* $p > 0$*, one can solve wc-smodLIP*$_K^{\boldsymbol{B}}$ *in time* $\tau + poly(d, \ell, \log s)$ *and probability* p*, where*

$$s \geq \sqrt{\frac{d \cdot \log(2d\ell + 4)}{\pi}} \cdot 2^d \cdot \varDelta_K^{3/(2d)} \cdot \max_{1\leq j\leq \ell} \Big(\|\sigma(g_{j,j})^{1/2}\| \cdot \mathcal{N}(I_j)^{1/d}\Big).$$

Proof. Let $\boldsymbol{G'} = (G', (J_i)_i)$ be any pseudo-Gram matrix equivalent to $\boldsymbol{G}$. First, we sample $\boldsymbol{G''} = (G', (H_i)_i)$ equivalent to $\boldsymbol{G}$ together with $U'' = (u''_{i,j})$ such that $G'' = U''^* G U''$ and $u''_{i,j} \in I_i H_j^{-1}$. This is done in time $poly(d, \ell, \log s)$. Now we can apply our oracle to solve an average-case LIP instance ; we find $U' = (u'_{i,j})$ with inverse $V' = (v'_{i,j})$ such that $G'' = U'^* G' U'$ and $u'_{i,j} \in J_i H_j^{-1}$, $v'_{i,j} \in H_i J_j^{-1}$. Let $U = U'' U'^{-1} =: (u_{i,j}) \in \mathrm{GL}_\ell(K)$, then $G' = U^* G U$ and

$$\forall\, (i,j) \in \{1, \ldots, \ell\}^2,\ u_{i,j} = \sum_{k=1}^{\ell} \underbrace{u''_{i,k} \cdot v'_{k,j}}_{\in (I_i H_k^{-1})(H_k J_j^{-1})} \in I_i J_j^{-1},$$

so U is a solution to the worst-case problem *i.e.* a congruence matrix between $\boldsymbol{G}$ and $\boldsymbol{G'}$. □

B Justifications for Assumption 1

In this appendix, we provide theoretical and experimental evidences in favor of Assumption 1.

Theoretical justification. First, recall that by Lemma 4.2, the ideal I is generated by the elements of the form $z_1^2 + z_2^2$ for $(z_1, z_2)^T \in M$. Hence, there exists no prime ideal $\mathfrak{p}$ such that $\mathfrak{p}$ divides $(z_1^2 + z_2^2)I^{-1}$ for all $(z_1, z_2)^T$ in M. This does not mean that all integral ideals $\mathfrak{a}$ are realisable as $(z_1^2 + z_2^2)I^{-1}$, but at least, these ideals should be co-prime when $(z_1, z_2)^T$ ranges over M. In terms of support, there is then some hope that some prime ideals can be realisable as $(z_1^2 + z_2^2)I^{-1}$. Note also that we chose $s \geq \eta_{1/2}(M)$ to ensure that the Gaussian distribution does not always fall in a sublattice of $\sigma(M)$ of smaller dimension.

With these precautions taken, we assume that the ideal qI^{-1} behaves like a "random" ideal of $\mathcal{O}_K$, with the condition that its class in the class-group should be the same as the one of I^{-1}. We now argue that if one picks a random integral ideal $\mathfrak{a}$ in a fixed class of the class group, then the probability that $\mathfrak{a}$ is prime is roughly $(\rho_K \cdot \log \mathcal{N}(\mathfrak{a}))^{-1}$. While founded on non-trivial results in number theory, the next arguments are somewhat standard.

Let C be a fixed class of the class group. We will write $i_C(X)$ the number of integral ideals in the class C of norm bounded by X and $p_C(X)$ the number of prime ideals in the class C of norm bounded by X. The next results give asymptotic bounds (when K is fixed and X tends to infinity) on the quantities $i_C(X)$ and $p_C(X)$, from which we will derive the (asymptotic) density of prime ideals in a given class C.

First, it is known that $p_C(X) \sim X/(h_K \cdot \log X)$ when X tends to infinity (and K is fixed). A reference for this claim can be found in [29, Se. 7.2, Cor. 4], when instantiated with $I = \mathcal{O}_K$.[22]

[22] In this case, the quantities $H_I^*(K)$ and $h_I^*(K)$ from the statement are simply the class group $Cl(K)$ and the class number h_K of K (the notations $H_I^*(K)$ and $h_I^*(K)$ are defined in Section 3.2 of [29]).

For the quantity $i_C(X)$, Theorems 39 and 40 from [26] and the class number formula provide the estimates $i_C(X) \sim \rho_K \cdot X/h_K$. Combining these two estimates, we obtain that the probability that a uniform ideal of norm $\leq X$ in a given class C is a prime ideal is asymptotically equivalent to $(\rho_K \cdot \log X)^{-1}$.

All these computations give us an expected behaviour, when the bound X tends to infinity and K is fixed. In our case, we would like to consider ideals of potentially small algebraic norms, and we are not even sure that our distribution produces ideals uniformly distributed in a given class below a bound X. Hence, our main assumption is to suppose that even though we do not satisfy all the requirements needed, the previous asymptotic bound still holds somehow in our case (up to a potential polynomial loss). In other words, we assume that the probability that qI^{-1} is prime is roughly $(\rho_K \cdot \log \mathcal{N}(qI^{-1}) \cdot \text{poly}(d))^{-1}$.

Finally, we upper bound the quantity $\mathcal{N}(q)$ by $\|\sigma(q)\|^d \leq (s\sqrt{d})^d$, which holds except with negligible probability using Gaussian tails bounds (see Lemma 2.5). Note also that the quantity appearing in the assumption is $\log(s/\mathcal{N}(I)^{1/d})$ and not $\log(s^d/\mathcal{N}(I))$, because we moved the factor d in the polynomial $P(d)$.

Experimental justification. We verified experimentally our assumption on number fields K that are the maximal real subfields of cyclotomic fields. The code for the experiments is available in our public repository (in the folder `experiments_for_assumption`). With these experiments, we tried to assess two claims:

- that the ideal qI^{-1} is prime with probability roughly $(\rho_K \cdot \log \mathcal{N}(qI^{-1}))^{-1}$,
- and that the quantity $\big(\rho_K \cdot \log(s/\mathcal{N}(I)^{1/d}) \cdot \Pr(q \cdot I^{-1} \text{ is prime })\big)^{-1}$ grows at most polynomially with d (it is supposed to be upper bounded by the polynomial $P(d)$ if our assumption holds).

In order to test these two claims, we performed multiple tests with different number fields, different modules for a given number field, and many random q for each module. More formally, we tested all K that are the maximal real subfield of a cyclotomic field of conductor ≤ 160 (those have degree d at most 78). We then tested some more maximal real subfields of cyclotomic fields (but not all of them), up to degrees $d = 125$.

In each of these number fields K, if the degree d of K was ≤ 45, we generated 10 random free modules M of rank 2 (even 20 modules if $d \leq 21$). We did not impose any restriction on the Gram ideal $\mathcal{G}(M)$ of the module. Most of them were equal to $\mathcal{O}_K$, but some of them were non-trivial. When d was larger than 45, we did not consider 10 random modules, but only 1 module per field, namely the module $\mathcal{O}_K^2$. We also added this specific module to the 10 and 20 modules already generated for the small dimensional number fields.

For each number field and each module M generated, we then sampled between 1000 and 5000 Gaussian vectors $(z_1, z_2)^T \in M$ and computed the empirical probability $P_{\text{empirical}}$ that qI^{-1} be a prime ideal (K and M are fixed here, and the empirical probability is computed only over the random choice of q). We also computed the expected probability P_{expected} to find a prime ideal. To do so, for each random q generated, we computed $1/(\rho_K \log \mathcal{N}(q/I))$, and we took the average of this quantity over all q's (for a fixed module).

In Figure 1, we plotted the ratios $P_{\text{empirical}}/P_{\text{expected}}$ for all number fields, and all modules, as a function of the degree d of K. The red crosses represent the modules $\mathcal{O}_K^2$ and the blue dots are the random modules (only for small degrees). One can check that this ratio is always between $[0.4, 2.2]$, hence, the quantity P_{expected} seems to approximate relatively well the empirical probability $P_{\text{empirical}}$. One can also check on the small dimensional fields that the module $\mathcal{O}_K^2$ does not seem to have a behaviour significantly different from the random modules. The slightly larger deviation observed in the blue dots for number fields of degree between 21 and 45 might be explained by the fact that those empirical probabilities were obtained using only 1000 random q, whereas the blue dots for $d \leq 21$ and the red crosses use at least 2000 random q's.

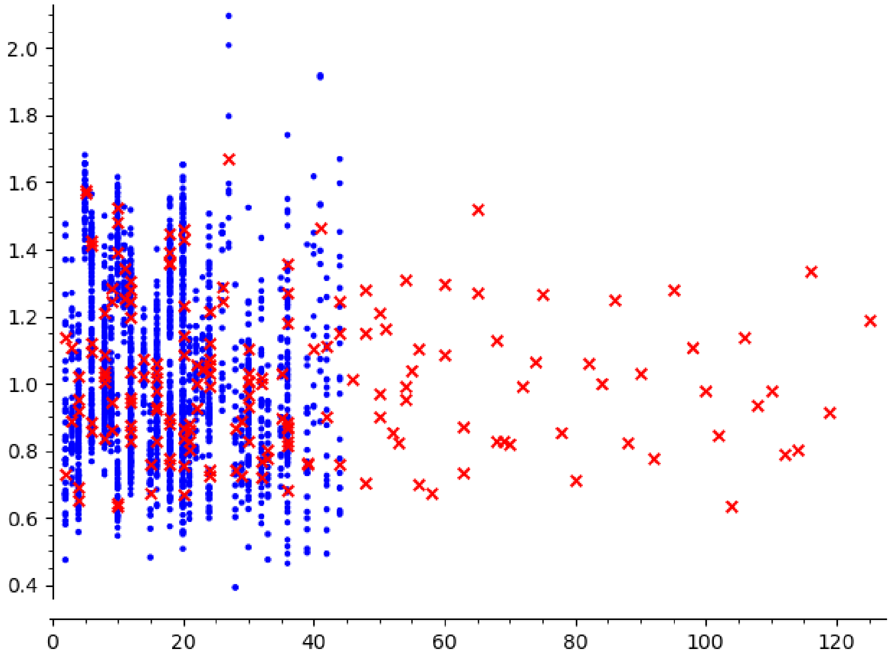

Fig. 1. Ratios $P_{\text{empirical}}/P_{\text{expected}}$ as a function of d for various fields K and various modules M. The red crosses correspond to $M = \mathcal{O}_K^2$ and the blue dots correspond to a random free module M

In Figure 2, we plotted the quantity $Q := (P_{\text{empirical}} \cdot \rho_K \cdot \log(\tilde{s}))^{-1}$ as a function of d. Here, $\tilde{s} = \max(\mathrm{e}, s/\mathcal{N}(I)^{1/d})$, where the max is here to ensure that $\log \tilde{s} \geq 1$ is not too small (this can happen in small dimensional examples and caused some weird results, which disappear when the dimension increases). Recall that this quantity is supposed to be upper bounded by $P(d)$ for some polynomial P (if our assumption holds). The result on the figure seem consistent with this assumption (the polynomial $P(d)$ even seems linear in d). Once again, there is one blue point per random module and one red cross per $\mathcal{O}_K^2$ module. Overall, our experimental results seem consistent with Assumption 1.

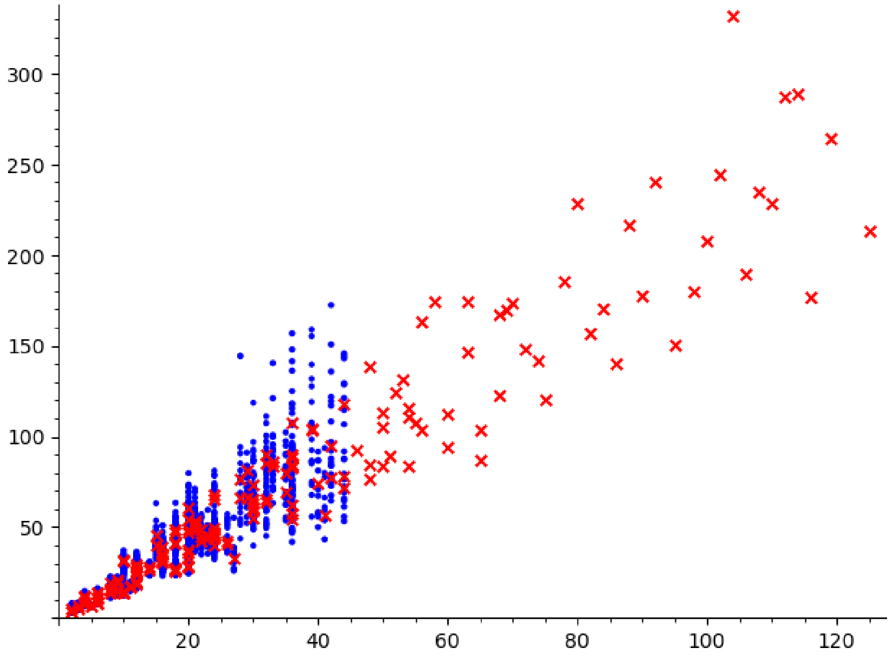

Fig. 2. The quantity Q as a function of d for various fields K and various modules M. The red crosses correspond to $M = \mathcal{O}_K^2$ and the blue dots correspond to a random free module M

C Details about the implementation

This section contains more technical details about our implementation of the attack, and some explanation for the choices we made there. Our code is available in our public repository (in the folder `attack`), and a high-level description of the implementation (with timings) is provided in Section 5.

On the Gentry-Szydlo's algorithm. The Gentry-Szydlo algorithm (in its initial form, over cyclotomic fields [20]) is a relatively complicated to implement algorithm. The main idea of the algorithm is to notice that if P is a prime number that splits completely in the cyclotomic field L, then $\mathcal{O}_L/(P\mathcal{O}_L) \cong (\mathbb{F}_P)^{\deg(L)}$, and so, by Fermat's theorem, any $x \in \mathcal{O}_L$ coprime to P satisfies

$x^{P-1} = 1 \bmod P$. Assume now that J is a principal ideal generated by some element $g \in \mathcal{O}_L$, then J^{P-1} is still principal, generated by g^{P-1}. If one is able to recover a small multiple $a \cdot g^{P-1}$ of this generator, then reducing it modulo P reveals $a \bmod P$. Now, if P is large enough compared to a, this allows to recover a exactly, and then recover g^{P-1} from $a \cdot g^{P-1}$. Finally, computing a root of g^{P-1} in L enables to recover g (up to roots of unity). This high-level description of the algorithm does not actually work, since it would require P to be exponentially large, causing the size of the elements to explode. A nice explanation of how to handle this issue can be found in [19, Section 7.3].

The Gentry-Szydlo algorithm is not natively implemented in `Sagemath`. We found a previous implementation of the algorithm in `Pari/GP`, done by de Vilmarest and Kirchner[23] and used in the article [4]. Unfortunately, we failed to use this implementation of the Gentry-Szydlo algorithm for our attack. More precisely, we managed to use the implementation of de Vilmarest and Kirchner on some instances, but the algorithm failed on some other instances without us understanding why (we guess that this was due to a bad choice of parameters at the beginning of the algorithm).

We decided to re-implement the Gentry-Szydlo algorithm in `Sagemath` for our attack. We did not target efficiency, but mainly stability of the algorithm. We followed the description of the algorithm provided in [19, Section 7.3], and used built-in Sage functions for manipulating ideals and number fields. The main bottleneck of the algorithm in large dimension are the (multiple) calls to an LLL algorithm in ideal lattices. We run the LLL algorithm on the Gram matrix of the ideals, and do so by calling Sage's function `LLL_Gram`, which itself calls the `qflllgram` function of `Pari/GP`. This function has recently been improved in `Pari/GP`, using the "Flatter" framework [32], but this optimization is not yet available when `Pari/GP` is called through Sage. In order to improve the running time of the Gentry-Szydlo algorithm, we enabled the user to specify a path to a recent installation of `Pari/GP` (with a fast `qflllgram` implementation), and use it instead of Sage's default `LLL_Gram` function. With this optimization, the cost of LLL is still dominating the running time of the algorithm, but it can be decreased by a factor 2 is large dimensions ($\gtrsim 200$).

On the `NormEquation` algorithm. We now explain `NormEquation` for our relevant fields, that is, $L = \mathbb{Q}(\zeta_m)$ is a cyclotomic field with $m = 4k$, and K is its maximal totally real subfield. If τ is the non-trivial automorphism of $L|K$, we write $\tau(a) = a^*$. Note that if a is a solution of $q = a^*a$, then $\rho \cdot a$ is also a solution for all roots of unity $\rho \in L$. On input an arbitrary principal ideal $q\mathcal{O}_K$, we factor it as a product of prime ideals in $\mathcal{O}_K$, which are then separated in inerts, ramified and splits parts in $\mathcal{O}_L$. The separation can be done e.g. by checking how the polynomial $X^2 + 1$ factors in finite fields. For simplicity, we

[23] available at https://alexgelin.fr/doc/GenRec.tar.gz.

will consider unramified primes in this section, so that

$$q\mathcal{O}_K = \prod_{\text{inert}} \mathfrak{p}^e \cdot \prod_{\text{split}} \mathfrak{q}^f.$$

If $q = a^*a$, we observe that primes above the inerts dividing $q\mathcal{O}_K$ must have an even valuation since they must appear both as divisors of $a\mathcal{O}_L$ and $a^*\mathcal{O}_L$. We can early-abort if this is not the case since there would be no solution. Now let us write

$$q\mathcal{O}_L = I^2 \cdot \prod_{\mathfrak{Q}\cap\mathcal{O}_K \text{ split}} (\mathfrak{Q}\mathfrak{Q}^*)^f,$$

where $I = \prod_{\mathfrak{P}\cap\mathcal{O}_K \text{ inert}} \mathfrak{P}^e$ is the product of all primes above the inerts of $p\mathcal{O}_K$. We then need to find principal ideals among all possible products $I \cdot \prod_{\mathfrak{Q}\cap\mathcal{O}_K \text{ split}} \mathfrak{Q}^\alpha \cdot (\mathfrak{Q}^*)^{f-\alpha}$ as their generators are candidates for solutions of the equation $q = a^*a$. Having computed all the products, we have in fact $\mathbb{Z}$-bases for all the ideals to be tested, as well as the relative norm q of its potential generators: this is where using Gentry-Szydlo algorithm makes this step polynomial time. As mentioned above, we provide an implementation of this algorithm on our repository that works well for small dimensions ($\phi(m) \leq 100$) and reasonably well up to $m \leq 300$.

On the `TwoSquares` algorithm. We now assume that `NormEquation` finds all solutions for the equation $q = a^*a$. The `TwoSquares` algorithm takes in input the set of all solutions to a relative norm equation $q = a^*a$ from $L = K(i)$ to K, and find those corresponding to $a = u + iv$ with $u, v \in \mathcal{O}_K$. The next lemma characterizes these particular solutions. Recall that $\mathrm{Tr}_{L|K}(x) = x + x^*$ is the relative trace map, and that it sends $\mathcal{O}_L$ into $\mathcal{O}_K$.

Lemma C.1. *We have* $\mathcal{O}_K + i\mathcal{O}_K = \{x \in \mathcal{O}_L \ : \ \mathrm{Tr}_{L|K}(x) \in 2\mathcal{O}_K\}$.

Proof. On the one hand, if $x = a + ib$ with $a, b \in K$, then $\mathrm{Tr}_{L|K}(x) = 2a$. On the other hand, we have the identity $2x = \mathrm{Tr}_{L|K}(x) + i\,\mathrm{Tr}_{L|K}(-ix)$. Then if $x \in \mathcal{O}_L$ is such that $\mathrm{Tr}_{L|K}(x) \in 2\mathcal{O}_K$, then $\mathrm{Tr}_{L|K}(-ix) \in 2\mathcal{O}_L \cap i\mathcal{O}_K$, which gives the result. □

Then, let b be any solution of the relative norm equation. Using the above lemma, when the relative trace is in $2\mathcal{O}_K$ we directly compute $u = (1/2) \cdot \mathrm{Tr}_{L|K}(b), v = (1/2) \cdot \mathrm{Tr}_{L|K}(-ib) \in \mathcal{O}_K$ such that $(u+iv)(u-iv) = u^2 + v^2 = q$. We implement this with a simple parity check of the coefficients for a given basis of $\mathcal{O}_K$, and this allows to filter out sums-of-two-squares. In e.g. `sagemath` where the ring $\mathcal{O}_K$ can be instantiated, this can also be done in a black-box by testing membership.

On the `CongruenceSolver` algorithm. To test our complete attack, we provide an algorithm to generate bases of $\mathcal{O}_K^2$ and their associated quadratic form Q of determinant 1. The idea behind the algorithm is a basis completion algorithm: take any vector $u \in \mathcal{O}_K^2$, find a $v \in \mathcal{O}_K^2$ such that $\det(u, v) = 1$. One can also check [10, Alg. 1.3.16] for the basis completion step, which we have borrowed in our implementation. We also do not aim for efficiency, and thus do not try to exploit any tower structure of the totally real subfield. However, if we stop at finding a v as described in the given reference, it will likely have large coefficients and so we also do a step of algebraic "size-reduction" using a basis of $\mathcal{O}_K$ to express our elements. For the NTRU enthousiasts out there, this is a similar procedure as in Hawk [14] and other NTRU key-generation algorithms, except that the ambient field is different so the operations are less efficient. In particular, there are less standard choices for the basis that is used to represent $\mathcal{O}_K$. Let $K = \mathbb{Q}(\theta)$, so that $1, \theta, \dots, \theta^{d-1}$ is a $\mathbb{Z}$-basis of $\mathcal{O}_K$. Experimentally, we observed that although easy to use, this tended to not be the best basis from a geometric point of view; in other words, the coefficients tended to be much larger with this basis than the other one we describe below.

Lemma C.2. *Let $\tau_0 = 1, \tau_1 = \theta$ and for $2 \leq i \leq d-1$, $\tau_i = \zeta_m^i + \zeta_m^{-i}$. Then $(\tau_i)_{0 \leq i \leq d-1}$ is a $\mathbb{Z}$-basis of $\mathcal{O}_K$.*

Proof. This is observed by expanding each θ^i, and noting that $\tau_i \in K$. Let us write $\zeta = \zeta_m$ to lighten the notation For an even exponent, we readily compute

$$\begin{aligned}\theta^{2i} &= \zeta^{2i} + \zeta^{-2i} + \sum_{j=1}^{2i-1} \binom{2i}{j} \zeta^{2i-j} \zeta^{-j} \\ &= \tau_{2i} + \sum_{j=1}^{i-1} \binom{2i}{j} \left(\zeta^{2(i-j)} + \zeta^{-2(i-j)}\right) + \binom{2i}{i} \\ &= \tau_{2i} + \sum_{j=1}^{i} \binom{2i}{j} \tau_{2(i-j)},\end{aligned}$$

and a similar expression is derived analogously for the odd case. This shows that the matrix to express the θ^i's from the τ_i's is triangular with 1 on the diagonal, and has integer entries. In other words, it is unimodular and thus $(\tau_i)_i$ is a basis of $\mathcal{O}_K$. □

To see that the geometry is better, one can look at the matrix $G = [\mathrm{Tr}_{K|\mathbb{Q}}(\tau_i \tau_j)]_{i,j}$, and notice that it is indeed quite sparser and has much smaller entries as when the degree of K grows. This comes from the vanishing of many sums of roots of unity. For the best case where m is a power of two, we can even show that G is diagonal with small entries. For completeness we gather this in the lemma below.

Lemma C.3. *If $m = 2^\ell$, we have $G = \mathrm{Diag}(d, 2d, \dots 2d) \in \mathbb{Z}^{d \times d}$.*

As a direct application, we find that $2\Delta_K = \sqrt{\Delta_L}$ for these conductors.

Proof. The upper left coefficient is seen as $\mathrm{Tr}_{K|\mathbb{Q}}(1) = d$. Let $\epsilon_{ij} = \tau_i \tau_j$ for $0 \leq i, j \leq d-1$. We have that $\epsilon_{ij} = \zeta^{i+j} + \zeta^{-(i+j)} + \zeta^{i-j} + \zeta^{-(i-j)}$. The chain rule for relative traces is $\mathrm{Tr}_{L|\mathbb{Q}} = \mathrm{Tr}_{K|\mathbb{Q}} \circ \mathrm{Tr}_{L|K}$, from which we see $\mathrm{Tr}_{K|\mathbb{Q}}(\epsilon_{ij}) =$

$\mathrm{Tr}_{L|\mathbb{Q}}(\zeta^{i+j}) + \mathrm{Tr}_{L|\mathbb{Q}}(\zeta^{i-j})$. By symmetry of G we can restrict to $j \leq i$. When $i = j$ we have $\zeta^{i-j} = 1$ and then $\mathrm{Tr}_{L|\mathbb{Q}}(1) = 2d$. Recall that the Galois group acts on ζ by exponentiation to an integer in $(\mathbb{Z}/m\mathbb{Z})^\times$, which in this case means exponentiation to an odd power. For the other term, for all k not multiple of d, we thus have

$$\mathrm{Tr}_{L|\mathbb{Q}}(\zeta^k) = \sum_{i=0}^{d-1} \zeta^{k(2i+1)} = \zeta^k \sum_{i=0}^{d-1} \zeta^{2i} = \zeta^k \cdot \frac{\zeta^m - 1}{\zeta - 1} = 0,$$

first by definition of the absolute trace, then using the geometric sum and the fact that ζ is a primitive m-th root of 1. This gives the result. □

For these conductors, the Gram matrix of the power basis is much worse. To compare, the bottom right coefficient in the Gram matrix of the power basis, for example, can be calculated from the lemma and properties of the trace to be $\binom{2(d-1)}{d-1} \cdot d$. The general case is more tedious as one needs to take into account the possible values of the trace at exponents that are not coprime to the conductor.

On the `FindPrimeNorm` algorithm. Moving onto the attack, we need to find sums-of-two squares that are also prime in K from the knowledge of Q. We implemented an implicit Gaussian sampler to do so, outputting vectors $z = (u, v) \in \mathcal{O}_K^2$ while keeping the coefficients of the found $q = x^2 + y^2 = z^t Q z$ as small as possible. Indeed, in previous versions our more naive approach resulted in substantially larger q's, which incurred a non-negligible cost on computations. To check for primality of the ideal $q\mathcal{O}_K$, our first approach was a to rely on a (probabilistic) prime-power testing implemented in sage, giving us (p, e) such that $N(q) = p^e$. Then we would factor the defining polynomial of K over $\mathbb{F}_p$ to find the possible two-element representation of $q\mathcal{O}_K$ among the factors. Experimentally, it was almost always the case that $e = 1$, or equivalently, that p totally splits in K. Therefore in our current version we directly aim at finding a element q of *prime* norm, which also makes the next step cheaper. This is of course a trade-off on the amount of repetitions, but experimentally, the benefit on the following computations was significant and we therefore kept this alternative strategy in our published code.

Finalizing the attack. Once a prime q is found in K, we need to find its factors in L, and we want to avoid to re-factor (or deduce the primality of) the absolute norm by using black-box functions. As $L|K$ is quadratic we know that either $q\mathcal{O}_K$ splits totally, is inert, or ramifies. By construction, we have $q = x^2 + y^2 = (x + iy)(x - iy)$ so in fact, q must always split.

This is similar to the situation over $\mathbb{Z}$, where primes that are sums of two squares must satisfy $p \equiv 1$ [4], which equivalently means that they split in the ring $\mathbb{Z}[i]$. Moreover, the two factors must be conjugate by the involution, so if we know the generator g for one, we have the other. This means that we only need to do one call to the Gentry-Szydlo algorithm by prime that we find at this step, making it a total of two to get to the last phase. We still need to feed input to Gentry-Szydlo. We can use the polynomial $X^2 - \theta X + 1$ defining the extension $L|K$; as we know it will have two factors modulo p, we just need to compute a square root of $\Delta_{L|K} = \theta^2 - 4$ modulo p to then deduce the linear factors. Note that p could be large if we did not control its size previously.

In the end we obtain two linearly independent vectors $z_1 = (u_1, v_1)$ and $z_2 = (u_2, v_2)$ which are also sums of two squares in $\mathcal{O}_K^2$, and for which we have all the possible sums. Following Step 17 to 20 in our algorithm, we then build all possible matrices corresponding to pairs of solutions, then check if they indeed give a congruence matrix (this amounts to check if the entries have a denominator). As we know already the determinant of the potential congruence matrices, we can filter out the matrices that would not fit and only do linear algebra with the ones we need. Additionally, we know that the group of $\mathcal{O}_K$-isometries acts on the set of solutions. This group has 4 elements (identity, negation, permutation of the two basis vectors and the composition of these two), which is also what we expect as a number of final solutions. Once one is found, we could obtain the other by using this action. In our published code, we did not implement this idea, and merely enumerated all (filtered) matrices until exhaustion. Overall, we could successfully recover the congruence class of Hawk-type Gram matrices[24] for different conductors in about $1100s$ for conductors as large as 256 (or a bit larger, when not a power of 2). Up to conductors about 50, the algorithm performs well and the attack takes mere seconds. For m from 50 to about 200, the attack is slower, with the two Gentry-Szydlo steps clearly dominating the running time. Notably, the attack never fails in the sense that we always find back the basis B such that $B^T B = Q$.

[24] Hermitian forms corresponding to a (secret) basis of the free module $\mathcal{O}_K^2$.