doi:10.3934/amc.2021055

A PROOF OF THE CONJECTURED RUN TIME OF THE HAFNER-MCCURLEY CLASS GROUP ALGORITHM

Jean-François Biasse* and Muhammed Rashad Erukulangara

University of South Florida
Center for Cryptographic Research
4202 E Fowler Ave, Tampa, FL 33620, USA

(Communicated by Sihem Mesnager)

Abstract. We present a proof under a generalization of the Riemann Hypothesis that the class group algorithm of Hafner and McCurley runs in expected time $e^{\left(3/\sqrt{8}+o(1)\right)\sqrt{\log d \log\log d}}$ where $-d$ is the discriminant of the input imaginary quadratic order. In the original paper, an expected run time of $e^{\left(\sqrt{2}+o(1)\right)\sqrt{\log d \log\log d}}$ was proven, and better bounds were conjectured. To achieve a proven result, we rely on a mild modification of the original algorithm, and on recent results on the properties of the Cayley graph of the ideal class group.

1. Introduction

Let K be an imaginary quadratic field , and $\mathcal{O}_{-d}$ be an order of K of discriminant $-d$, the computation of the ideal class group $\mathrm{Cl}(-d)$ of $\mathcal{O}_{-d}$ consists in finding positive integers $m_1, ...m_k$ such that $m_1 \mid m_2 \mid ... \mid m_k$ and

$$\mathrm{Cl}(-d) \cong \bigoplus_{i=1}^{k} \mathbb{Z}/m_i\mathbb{Z}.$$

The computation of the ideal class group is one of the four major tasks in computational number theory postulated by Zassenhaus [31, p. 2] (together with the computation of the unit group, the Galois group and the ring of integers). In particular, the efficient computation of the ideal class group has applications in the study of unproved heuristics in number theory and in the resolution of Diophantine equations. This problem is also equivalent to the computation of the group of $\mathrm{SL}(2,\mathbb{Z})$-equivalence classes of positive definite binary quadratic forms of discriminant $-d$.

1.1. Prior Work.

The first algorithms to compute the ideal class group of a quadratic number field had exponential complexity. In 1968, Shanks [35, 36] proposed an algorithm relying on the baby-step giant-step method to compute the class number and the regulator of a quadratic number field in time $O\left(|d|^{1/4+\epsilon}\right)$, or $O\left(|d|^{1/5+\epsilon}\right)$ under a generalization of the Riemann hypothesis (GRH) [28]. A

2020 *Mathematics Subject Classification:* Primary: 11Y16, 20C05, 11R32; Secondary: 11R29, 11R04, 11Y40, 11R18, 11R27.

Key words and phrases: Ideal class groups, imaginary quadratic fields, Index Calculus, Groups of unknown order.

The first author is supported by NSF grant 1846166.

* Corresponding author: Jean-François Biasse.

major breakthrough came in 1989 with the first subexponential algorithm for the computation of the ideal class group of an imaginary quadratic field by Hafner and McCurley [24] (under GRH). More specifically, they described a Las Vegas algorithm for the computation of $\mathrm{Cl}(-d)$ that runs in expected time $L(d)^{\sqrt{2}+o(1)}$ where the subexponential function $L(d)$ is defined as

$$L(d) = e^{\sqrt{\log d \log\log d}}$$

where log denotes the natural logarithm. After that, most of the improvements on the computation of the ideal class group of a quadratic number field focused on the practical implementation of the Hafner-McCurley algorithm rather than on the improvement of its asymptotic run time. For instance, practical improvements were presented in [22] by Cohen, Diaz Y Diaz and Olivier, and Jacobson successfully applied sieving techniques borrowed from factoring algorithms to reach significantly larger discriminants [26]. In [6, 13], Biasse and Jacobson incorporated more practical optimizations adapted from factoring algorithms to establish new records including the calculation of the class group of a 110-digit discriminant. Kleinjung further improved this with the computation of the class group of a 130-digit discriminant [27]. As of today, the largest d for which $\mathrm{Cl}(-d)$ was calculated is a 154-digit discriminant [5].

In parallel to these practical improvements to the original Hafner-McCurley algorithm, significant effort has been devoted to generalizing it to number fields of higher degree. This poses non-trivial challenges as lattice reduction (which is inefficient in large degree) is needed to emulate the subexponential approach of [24]. First, Buchmann [18] generalized this result to the case of infinite classes of number fields with fixed degree (which essentially circumvents issues related to lattice reduction). Then a new line of work started by Biasse [7] showed that despite the issues caused by lattice reduction, there were some infinite families of fields with degree growing to infinity where the computation of the ideal class group can be done in heuristic complexity better than that of Hafner and McCurley's algorithm. In [8, 10], Biasse and Fieker showed that there was a heuristic subexponential algorithm for the computation of the ideal class group in all classes of number fields. The methods of [10] can be specialized to the case of cyclotomic fields for a better asymptotic complexity [9]. Based on [2], Biasse and van Vredendaal [16] described an algorithm for computing the class group of $K = \mathbb{Q}(\sqrt{d_1}, \ldots, \sqrt{d_n})$ in heuristic asymptotic runtime in $\mathrm{Poly}(\log|d|)e^{\tilde{O}(\sqrt{\log|d_0|})}$ with $d_0 = d_1 \cdots d_n$. For some families of $d_1, \ldots, d_n$, this yields a polynomial time algorithm. The strategy behind this algorithm was later generalized by Biasse, Fieker, Hofmann and Page who described a systematic method to leverage computations in subfields [11]. This allowed the computation of the class group of $K = \mathbb{Q}(\zeta_{6552})$ of degree 1728 with a 5258-digit discriminant. These improvements in high degree number fields that reduced asymptotic complexity and achieved record-breaking computations do not impact the quadratic case, where until now the best known asymptotic complexity was the one stated by Hafner and McCurley [24]. Finally, note that due to an algorithm of Biasse and Song, ideal class group computation runs in polynomial time on quantum computers for all classes of number fields [15].

1.2. Relevance to cryptography.

Recently, ideal class group computation in large degree number fields received significant attention from the cryptographic research community because of the connection between this task and the search for (small) generators of principal ideals. However, this line of work does not include quadratic number fields. On the other hand, there are several areas in cryptography where class group computation in quadratic fields directly applies.

Schemes based on the DLP. The first and most obvious way where ideal class group computations occur in cryptography is through schemes based on the Discrete Logarithm Problem (DLP). Indeed, Quadratic fields were proposed as a setting for public key cryptosystems in the late 1980s by Buchmann and Williams [20, 19]. Two types of schemes were described: those that use imaginary fields, and the others that use real fields. In the imaginary case, cryptosystems are based on arithmetic in the ideal class group (which is a finite abelian group), and the discrete logarithm problem is the computational problem on which the security is based. In the real case, the so-called infrastructure is used instead, and the security is based on the analogue of the discrete logarithm problem in this structure, namely the principal ideal problem. The most up-to-date security estimates deriving from computational data on the resolution of the underlying hard problems (which derive from the hardness of computing the ideal class group) are due to Biasse, Jacobson and Silvester [14].

Computation of isogenies. In ordinary elliptic curves defined over finite fields, as well as in supersingular elliptic curves defined over prime fields, the ring of endomorphisms is isomorphic to an order in an imaginary quadratic field. Isomorphism classes of curves are in correspondence with classes of ideals in $\mathrm{Cl}(-d)$, and the class of a split prime ideal of norm ℓ acts as an isogeny of degree ℓ. As shown by Biasse, Fieker and Jacobson in [12, Prop. 6.2], the computation of a reduced basis of relations yields efficient ideal decompositions in $\mathrm{Cl}(-d)$, which in turns allows the fast evaluation of isogenies of large degree. This strategy was used in the signature scheme CSI-FiSh [5] where the precomputation of the class group of a 154-digit discriminant field enables good performances because signing involves the evaluation of large degree isogenies through the decomposition of the corresponding ideal class.

Groups of unknown order. One of the interesting features of the ideal class group is that it allows analogues of certain cryptographic schemes based on group arithmetic (in particular for schemes based on the DLP) despite the fact that the order of the group is unknown. Computing the cardinality of the class group seems to be similar to the computation of $\mathrm{Cl}(-d)$. Other groups have been suggested to run protocols requiring a group of unknown order, most notably the units modulo an RSA integer $N = pq$. However, in this case, since $|\mathbb{Z}_N^*| = (p-1)(q-1)$, the entity that sets up the public parameter N needs to be trusted not to reveal the factorization of N to certain users. On the other hand, computations in $\mathrm{Cl}(-d)$ do not need a trusted setup, which makes them very appealing for these families of schemes. A notable example of cryptographic schemes relying on groups of unknown order is the Verifiable Delay Functions (VDF) based on repeated exponentiation such as those of Pietrzak [30] and Wesolowski [40]. In such schemes, one of the participants needs to compute g^{2^T} for a large T and convince a verifier of the validity of the computation. Clearly, the use of $\mathrm{Cl}(-d)$ requires that the computation of the structure of this group be intractable. However, it is worth noting that other assumptions may be needed to allow a succinct verification of this computation, including the low-order assumption which states that it is hard to find elements of low order. Recent work from Belabas, Kleinjung, Sanso and Wesolowski [3] suggests that there are wrong choices of discriminant for which the low-order assumption does not hold true. Other applications of groups of unknown orders include secure Multi-Party Computation (see for example [23, Sec. 3.2]).

1.3. Our contribution.

In this paper, we propose a proof of the conjectured run time of our modified Hafner-McCurley class group algorithm.

Main Theorem. *Under a generalization of the Riemann hypothesis, there is a Las Vegas algorithm which computes* $\mathrm{Cl}(-d)$ *with probability* $1-\frac{1}{d^{1+o(1)}}$ *in time*

$$L(d)^{3/\sqrt{8}+o(1)} = e^{\left(3/\sqrt{8}+o(1)\right)\sqrt{\log d \log\log d}}.$$

The complexity proved in the main result is not surprising. Indeed, as early as 1988, McCurley [29] gave heuristic arguments showing that $|\operatorname{Cl}(-d)|$ should be computable in time $L(d)^{3/\sqrt{8}+o(1)}$ (note that this conjecture only concerns the computation of the cardinality of $\mathrm{Cl}(-d)$, not the group structure itself). Then, in [24, Sec. 5], Hafner and McCurley described conjectural improvements to their subexponential techniques for computing $\mathrm{Cl}(-d)$ that included the observation that the collection of the relations could be improved, and that such an improvement (as proposed in the present paper) would yield a run time of $L(d)^{2/\sqrt{3}+o(1)}$. Combined with better linear algebra methods than the ones used in [24], this yields the run time of $L(d)^{3/\sqrt{8}+o(1)}$ we prove in this paper. In 2000, Ulrich Vollmer even claimed a proof of this run time [38, Th. 2]. However, this claim was retracted in the Corrigendum of a 2002 follow-up paper [39]. In 2016, Biasse, Fieker, and Jacobson conjectured the main result of our paper [12, Conj. 1].

Our methods rely on a result from Jao, Miller and, Venkatesan [25] showing that under the GRH, the Cayley graph of $\mathrm{Cl}(-d)$ is an expander graph (for a good choice of edges). This allowed us to show that relations between generators of $\mathrm{Cl}(-d)$ could be obtained from short products of generators. With that knowledge, we prove that a minor modification of the original algorithm of Hafner and McCurley yields the conjectured expected run time. We do not rule out the existence of a more elementary proof involving techniques available at the time of Hafner and McCurley's paper [24] in addition to cubic complexity linear algebra methods (as observed in [24, Sec. 5], linear algebra with quartic complexity can only yield a total run time of $L(d)^{2/\sqrt{3}+o(1)}$).

On the other hand, we do not foresee any improvement of these methods that would yield a subexponential run time that is not conditional on a generalization of the Riemann Hypothesis being true. Indeed, all variations around the subexponential algorithm of [24] rely on the manipulation of vectors whose length is given by the first n primes generating the class group. So far, the only known unconditional bounds on the length of such vectors are all exponential in the input size. We state the relevant generalization of the Riemann hypothesis which we assume to be true to guarantee the validity of the main theorem. We abbreviate it by "GRH". Note that the same assumption is sometimes referred to as Extended Riemann Hypothesis (ERH), in particular in [24] and [1].

Conjecture 1 (GRH)**.** *Let K be an number field and χ be a Hecke character on K. Then the Hecke L-function given by $L(s,\chi)=\sum_{\mathfrak{a}}\frac{\chi(\mathfrak{a})}{\mathrm{N}(\mathfrak{a})}$ is zero-free in the half-plane $\mathfrak{Re}(s)>1/2$.*

2. Background

In this section, we briefly discuss quadratic fields. In particular, we define the ideal class group, we highlight the connection between positive definite quadratic forms and ideal classes, and we give a high level explanation of the class group computation of Hafner and McCurley [24].

2.1. Quadratic fields.

An integer Δ is a quadratic discriminant if it is not a perfect integral square and $\Delta \equiv 0, 1 \bmod 4$. In this paper, we assume that $d > 0$ and that $-d$ is a quadratic discriminant. The imaginary quadratic order of discriminant $-d$ is defined as the $\mathbb{Z}$-module $\mathcal{O}_{-d} = \mathbb{Z} + \frac{-d+\sqrt{-d}}{2}\mathbb{Z}$. A quadratic order $\mathcal{O}_{-d}$ is called a maximal order if it is not contained in a larger quadratic order. The discriminant of a maximal order is called fundamental discriminant. Let $-d$ be a fundamental discriminant. Then, the quadratic field of discriminant $-d$ is the $\mathbb{Q}$-vector space $\mathbb{Q}(\sqrt{-d}) = \mathbb{Q} + \sqrt{-d}\mathbb{Q}$.

2.2. The ideal class group.

An ideal of $\mathcal{O}_{-d}$ is a two dimensional $\mathbb{Z}$-module $\mathfrak{a} = m(a\mathbb{Z} + \frac{b+\sqrt{-d}}{2}\mathbb{Z})$, where $a, b, m \in \mathbb{Z}$ and $4a|b^2 + d$. The integers a and m are unique, and b is defined modulo $2a$.The ideal $\mathfrak{a}$ is said to be primitive if $m = 1$. The norm of an ideal is $N(\mathfrak{a}) = am^2$. The prime ideals of $\mathcal{O}_{-d}$ have the form $\mathfrak{p} = p\mathcal{O}_{-d}$ for p prime with Kronecker symbol $\left(\frac{-d}{p}\right) = -1$ and $\mathfrak{p} = p\mathbb{Z} + \frac{b_p+\sqrt{-d}}{2}\mathbb{Z}$ for primes p with Kronecker symbol $\left(\frac{-d}{p}\right) \neq -1$ and b_p the uniquely determined square root b of $-d$ modulo $4p$ in $[0, p]$. In both cases, we say that $\mathfrak{p}$ lies over p and we denote this by $\mathfrak{p} \mid p$. When $\left(\frac{-d}{p}\right) \leq 0$, only one prime ideal lies over p. When $\left(\frac{-d}{p}\right) = 1$, there are two prime ideals lying over p, which correspond to the two possible roots modulo p. If the prime ideal $\mathfrak{p}$ corresponds to the choice of root b_p, then its conjugate $\overline{\mathfrak{p}}$ corresponds to $-b_p \bmod 4p$. Since $\mathcal{O}_{-d}$ is a Dedekind domain, every ideal can be factored uniquely as a product of prime ideals. An ideal $\mathfrak{a}$ can be decomposed as a product of primes ideals by factoring $N(\mathfrak{a})$. For each prime p dividing the norm, the prime ideal $\mathfrak{p}$ or $\overline{\mathfrak{p}}$ necessarily divides $\mathfrak{a}$ for $\mathfrak{p} \mid p$. We can decide which one does depending on whether b is congruent to b_p or $-b_p$. A subset $\mathfrak{a}$ of $\mathbb{Q}(\sqrt{-d})$ is said to be a *fractional ideal* of $\mathcal{O}_d$ if there is an integer $d_0 > 0$ such that $d_0\mathfrak{a}$ is an ideal of $\mathcal{O}_{-d}$. The minimal such d_0 is called the denominator of $\mathfrak{a}$ and is denoted $d(\mathfrak{a})$. If $d(\mathfrak{a})\mathfrak{a} = m(a\mathbb{Z} + \frac{b+\sqrt{-d}}{2}\mathbb{Z})$, then

$$\mathfrak{a} = q\left(a\mathbb{Z} + \frac{b+\sqrt{-d}}{2}\mathbb{Z}\right)$$

for $q = m/d(\mathfrak{a})$, and this representation is unique. A fractional ideal $\mathfrak{a}$ is said to be invertible if there exists another fractional ideal $\mathfrak{a}^{-1}$ such that $\mathfrak{a}\mathfrak{a}^{-1} = \mathcal{O}_{-d}$. The inverse of $\mathfrak{a}$ is given by $\mathfrak{a}^{-1} = \frac{q}{\mathrm{N}(\mathfrak{a})}\left(a\mathbb{Z} + \frac{-b+\sqrt{-d}}{2}\mathbb{Z}\right)$, and the invertible fractional ideals form a multiplicative group with neutral element $\mathcal{O}_{-d}$. Two fractional ideals $\mathfrak{a}$ and $\mathfrak{b}$ are said to be equivalent, which is denoted by $\mathfrak{a} \sim \mathfrak{b}$, if there exist $u, v \in \mathcal{O}_{-d}$ such that $(u)\mathfrak{a} = (v)\mathfrak{b}$, where (u) denotes the principal ideal generated by u. This is an equivalence relation and the class group is the set of equivalence classes, that is

$$\mathrm{Cl}(-d) = \frac{\{\text{Invertible fractional ideals of } \mathcal{O}_{-d}\}}{\{\text{Principal invertible fractional ideals of } \mathcal{O}_{-d}\}}.$$

In particular, $\mathrm{Cl}(-d)$ is a finite abelian group. The order of the class group is called the class number. An ideal $\mathfrak{a}$ is reduced if it is primitive and $N(\mathfrak{a})$ is a minimum in $\mathfrak{a}$. Reduced ideals have the property that $a, b < \sqrt{d}$. So the reduced ideals gives reasonably small representative for each element of $\mathrm{Cl}(-d)$. The group operation is multiplication of two reduced ideals and computing a reduced ideal equivalent to the product. This operation is efficient and can be performed in $O(\log^2 d)$ bit operations.

2.3. Binary quadratic forms. A binary quadratic form is a bivariate polynomial

$$f = ax^2 + bxy + cy^2$$

where $a, b, c \in \mathbb{Z}$. The discriminant of the form f is $d = b^2 - 4ac$. Two forms f and g of discriminant d are said to be equivalent if there is a linear, unimodular transformation of variables such that when applied to f yields g. Under this notion of equivalence, the binary quadratic forms of discriminant $-d$ partition themselves into equivalence classes and this forms a group under the composition of forms. This group is isomorphic to the class group of the quadratic order $\mathcal{O}_{-d}$. Also, there is a correspondence between binary quadratic forms and the ideals, given by the following map:

$$ax^2 + bxy + cy^2 \rightarrow \left(a\mathbb{Z} + \frac{b + \sqrt{-d}}{2}\mathbb{Z}\right).$$

Since we have a natural bijection between the set of classes of quadratic forms and the class group, we can transport the group structure of class group to the set of equivalence classes of quadratic forms. The multiplication between ideals corresponds to the composition of quadratic forms, while a reduction procedure returns the unique reduced form in its equivalence class. The reduction algorithm is a variant of Euclid's algorithm as in Algorithm 5.4.2 of [21]. Also, the composition of quadratic forms can be done by using the Algorithm 5.4.7 of [21]. Both composition and reduction of quadratic forms can be done in $O(\log^2 d)$ bit operations. Given two reduced forms f, g of discriminant $-d$, we denote by $f \cdot g$ the reduced form obtained by composing f and g and then reducing the result. This allows us to efficiently implement arithmetic operations between elements of $\mathrm{Cl}(-d)$. We also denote the quadratic form $f = ax^2 + bxy + cy^2$ by (a, b, c) and its equivalence class by $[(a, b, c)]$. We use the notation $\mathrm{Cl}(-d)$ to denote the class group formed by the equivalence classes of binary quadratic forms of negative discriminant $-d$ and $h(-d)$ to denote its class number.

2.4. Subexponential class group computation. The set of generators for the class group $\mathrm{Cl}(-d)$ is given by the following result from [33, Th. 6.1].

Theorem 2.1. *Let p_i be the ith prime with $(-d/p_i) = 1$, and let*

$$b_i = \min\{b \in \mathbb{Z}_{\geq 0} : b^2 \equiv -d (mod\ 4p_i)\}.$$

Then, under GRH, there exists an absolute, effectively computable constant c such that the classes $[(p_i, b_i, .)]$, $1 \leq i \leq n_0$, generate $\mathrm{Cl}(-d)$, where n_0 is the largest integer such that $p_{n_o} \leq c \log^2 d$.

Let us use f_i to denote the reduced prime form $(p_i, b_i, .)$. Let $n = L(d)^z$ for a fixed positive number z. Then the classes $[(p_i, b_i, .)]$, $1 \leq i \leq n$ generate the class group $\mathrm{Cl}(-d)$ by Theorem 2.1. Also, as in [34] and [29], we define a homomorphism $\varphi : \mathbb{Z}^n \to \mathrm{Cl}(-d)$ by

$$\varphi(x_1, ..., x_n) = \prod_{i=1}^{n} f_i^{x_i}$$

An integer relation on $f_1, ..., f_n$ is a vector $(x_1, ..., x_n) \in \mathbb{Z}^n$ such that $\varphi(x_1, ..., x_n) = \prod_{i=1}^{n} f_i^{x_i} = 1_{\mathrm{Cl}(-d)}$ where $1_{\mathrm{Cl}(-d)}$ is the identity element of the class group $\mathrm{Cl}(-d)$. Relations in $f_1, ..., f_n$ form an additive subgroup of $\mathbb{Z}^n$ (i.e. a Euclidean lattice) which we denote as Λ. Since φ is surjective, we have

$$\mathbb{Z}^n/\Lambda \cong \mathrm{Cl}(-d).$$

Therefore, the computation of $\mathrm{Cl}(-d)$ reduces to the search for relations between the $f_1, \ldots, f_n$. Once enough relations are collected to generate Λ, a polynomial time linear algebra phase yields the quotient $\mathbb{Z}^n/\Lambda$, and therefore the ideal class group $\mathrm{Cl}(-d)$.

3. Overview of the algorithm

As we have seen in the previous section, the subexponential algorithm of Hafner and McCurley consists in the choice of a factor basis $f_1, \ldots, f_n$, and the resolution of the following two main tasks

- Finding a generating set of elements of Λ, the lattice of relations between factor basis elements.
- Computing the quotient $\mathbb{Z}^n/\Lambda$.

The quotient computation is well understood, and essentially corresponds to the computation of the Smith Normal Form (SNF) of the matrix representing a basis for Λ. Such a basis is typically obtained by computing the Hermite Normal Form (HNF) of a rectangular matrix representing a generating set for Λ. Polynomial time methods for the computation of HNF and SNF of integer matrices are known. Therefore, the main challenge of the algorithm is the calculation of a generating set for Λ.

The main issue with computing a basis of Λ is that we cannot easily sample random elements of Λ. In practice, products of the generators $f_1, \ldots, f_n$ with exponents drawn uniformly at random seem to be equivalent to random smooth form, and thus yield relations between the generators that appear to be distributed closely to the uniform distribution. This explains why in practical implementations, the randomization of the elements in Λ that are calculated is never an issue, and the number m of elements of Λ we need to draw is never significantly larger than n. At the end of the procedure, we can efficiently test whether enough relations were collected, thus certifying the computation. This test is based on a bound h^* that we can efficiently compute, which satisfies $h^* \leq \det(\Lambda) < 2h^*$. If $\det(\Lambda) \geq 2h^*$, then more relations are needed.

Making a formal case for the run time without the heuristic that elements sampled in Λ are distributed uniformly at random is more difficult. This was achieved by Hafner and McCurley in their original paper [24] at the price of a procedure that makes relation collection artificially more expensive than what is done in practical implementations in order to prove its expected run time under the ERH. In this paper, we show how to enhance this phase of the algorithm without having to rely on additional heuristics. The relation collection can be divided into phases. The original algorithm of Hafner and McCurley has two phases. The first one consists in the creation of n relations that are linearly independent. This means that at the end of it, we know $\Lambda_0 \subseteq \Lambda$ of full rank. However, since we typically have $\det(\Lambda_0) \gg \det(\Lambda)$, more relations are needed to find a generating set of Λ. Then the second phase consists in creating new relations with an expensive randomization strategy such that the the corresponding lattices $(\Lambda_i)_{i\leq m}$ they generate satisfy

$$\Lambda_0 \subseteq \Lambda_1 \subseteq ... \subseteq \Lambda_m \subseteq \Lambda$$

Our algorithm introduces an intermediate phase before the expensive randomization strategy of Hafner and McCurley in order to make sure that Λ_1 has a moderate determinant. We label our phases from 1 to 3.
Phase 1. For each $k = 1, \ldots, n$, we compute a relation whose k-th coefficient is significantly larger than the others. This method due to Seysen [34] ensures the fact that at the end of the collection of the first n relations, the lattice Λ_0 they generate has full rank.
Phase 2. We construct additional relations in order to ensure that at the end of this phase, the lattice Λ_1 they generate satisfies $\det(\Lambda_1) \in 2^{O(\log^4 d)}$. This is our main technical addition to the original method from [24]. We achieve this by observing that as long as $\det(\Lambda_1) > e^{\log^4 d}$, the lattice Λ_1 is very sparse within Λ, and therefore we can ensure the creation of new relations outside of Λ_1 with large enough probability.
Phase 3. Once we have $\det(\Lambda_1) \in 2^{O(\log^4 d)}$, we resume the creation of new relations with the expensive last phase of [24]. Since $|\Lambda/\Lambda_1|$ is small, the number of expensive steps is significantly less than in [24].

4. Phase 1

In this section, we show how to create n linearly independent relations between the $(f_i)_{i\leq n}$. We follow the approach of Seysen [34, Sec. 4] which consists in ensuring that the matrix $(a_{i,j})$ whose rows are the relation vectors satisfies $|a_{ii}| > \sum_{j\neq i} |a_{i,j}|$, which in turns guarantees that the matrix $(a_{i,j})$ has full rank. To improve on the run time of Seysen, we use the fact that the Cayley graph of $\mathrm{Cl}(-d)$ is an expander graph.

Proposition 1 (Deriving from Cor.1.3 of [25])**.** *Let $d > 0$ such that $-d$ is a fundamental discriminant, $\varepsilon > 0$ a constant, and let n_0 be such that*

$$f_1, \ldots, f_{n_0} = \left\{ \textit{Prime forms corresponding to } p \leq \log^{2+\varepsilon}(d) \textit{ and } \left(\frac{d}{p}\right) \neq 1 \right\}.$$

Then let f be an arbitrary form of $\mathrm{Cl}(-d)$ and $S \subseteq \mathrm{Cl}(-d)$. There is a constant $C > 0$ such that under GRH, the probability that a vector $\vec{x} \in \mathbb{Z}^{n_0}$ chosen uniformly at random among the vectors of ℓ_1-norm t for any $t \geq C\frac{\log|\mathrm{Cl}(-d)|}{\log\log d}$ satisfies

$$f \cdot \left(\prod_{i\leq n_0} f_i^{x_i} \right) \in S$$

is at least $\frac{1}{2}\frac{|S|}{|\mathrm{Cl}(-d)|}$.

The above means that a random walk of length at least $C\frac{\log|\mathrm{Cl}(-d)|}{\log\log d}$ starting from any vertex of the Cayley graph of $\mathrm{Cl}(-d)$ ends on a node whose distribution is close to the uniform one.

We can use this result to produce an analogue of the method of Seysen [34, Sec. 4] which requires only short products in order to produce a sparse relation with a dominant i-th coefficient. Here we assume that

$$\mathcal{B} = f_1, \dots, f_n = \left\{ \text{Prime forms corresponding to } p \leq L(d)^z \text{ and } \left(\frac{d}{p}\right) \neq 1 \right\},$$

which means that $n = L(d)^{z+o(1)}$. Choosing $t = \log(d) \gg C\frac{\log|\mathrm{Cl}(-d)|}{\log\log d}$, we draw random vectors $\vec{x}$ of ℓ_1-norm t until $f \cdot \left(\prod_{i \leq n_0} f_i^{x_i}\right)$ factors as a product of elements of $\mathcal{B}$ (i.e. is $\mathcal{B}$-smooth). The cost and the chances of success are given by Proposition 1 applied to the set S of $\mathcal{B}$-smooth reduced forms. We fix an amount of attempts such that the probability of obtaining the desired decomposition is high enough. To test for smoothness, we use Bernstein's batch smoothness test [4] rather than trial division, which allows us to reduce the asymptotic complexity. If at the end of the procedure, no smooth form was found, we declare a failure. If not, the result is guaranteed to be correct. This is compatible with a Las Vegas algorithm. All procedures presented throughout this paper satisfy this property.

```
Input  : Fundamental discriminant $-d < 0$, reduced form $f$, set of prime
         forms $\mathcal{B} = \{f_1, \dots, f_n\}$, $\varepsilon > 0$
Output: $\vec{x} \in \mathbb{Z}^n$ such that $\prod_{i \leq n} f_i^{x_i} = f$, or FAILURE

1  $B \leftarrow 4 \cdot e^{u(\log u + \log\log u + c(\varepsilon))}$ for $c(.)$ as in [34, Th. 5.2] and $u = \frac{\log(\sqrt{d}/2)}{\log(L(d)^z)}$;
2  $f'_1, \dots, f'_{n_0} \leftarrow \left\{ \text{Prime forms with } p \leq \log^{2+\varepsilon}(d) \text{ and } \left(\frac{d}{p}\right) \neq 1 \right\}$;
3  if $(f_i)_{i \leq n_0} \neq (f'_i)_{i \leq n_0}$ then
4      return: FAILURE
5  end
6  $t \leftarrow \log d$. Initiate an empty list $L_{\text{forms}}$;
7  for $k \leftarrow 1$ to $\lceil B \rceil$ do
8      Draw $\vec{y} \in \mathbb{Z}^{n_0}$ uniformly at random among the $\ell_1$-norm $t$ vectors;
9      Store $\vec{y}$ and $\left(\prod_{i \leq n_0} f_i^{y_i}\right) \cdot f$ in $L_{\text{forms}}$;
10 end
11 Test the $\mathcal{B}$-smoothness of all forms in $L_{\text{forms}}$ using [4];
12 if There is $\left(\prod_{i \leq n_0} f_i^{y_i}\right) \cdot f$ that is $\mathcal{B}$-smooth in $L_{forms}$ then
13     Let $\vec{x'} \in \mathbb{Z}^n$ with $\left(\prod_{i \leq n_0} f_i^{y_i}\right) \cdot f = \prod_{i \leq n} f_i^{x'_i}$ and $\vec{y'} = (\vec{y} \,||\, \vec{0}) \in \mathbb{Z}^n$;
14     return: $\vec{x} = \vec{x'} - \vec{y'}$;
15 else
16     return: FAILURE
17 end
```

Algorithm 1: Relation search algorithm

Proposition 2. *Under GRH, Algorithm 1 with input*

$$\mathcal{B} = \left\{ \textit{Prime forms corresponding to } p \leq L(d)^z \textit{ and } \left(\frac{d}{p}\right) \neq 1 \right\},$$

succeeds with probability at least $1-\frac{1}{d^{1+o(1)}}$ *in time* $L(d)^{1/4z+o(1)}+L(d)^{z+o(1)}$ *and returns a vector* $\vec{x}$ *whose* ℓ_1*-norm is in* $O(\log d)$*.*

Proof. We denote by S the set of $\mathcal{B}$-smooth reduced forms of $\mathrm{Cl}(-d)$. From the proof of [34, Prop. 4.4], we know that the probability $\frac{|S|}{2|\,\mathrm{Cl}(-d)|}$ that a single product of the form $\left(\prod_{i\le n_0} f_i^{y_i}\right)\cdot f$ be $\mathcal{B}$-smooth satisfies

$$\frac{|S|}{2|\,\mathrm{Cl}(-d)|}\ge\frac{\psi_{-d}(\sqrt{d}/2,L(d)^z)}{2\sqrt{d}\log d},$$

where the function $\psi_{-d}(x,y)$ satisfies

$$\psi_{-d}(x,y)\ge x\cdot e^{-u(\log u+\log\log u+c(\varepsilon))}\text{ for }u=\frac{\log x}{\log y},$$

provided that $x>10$ and $\log(x)^{1+\varepsilon},\log^{2+\varepsilon}(d)\le y\le e^{\log^{1-\varepsilon}d}$. The pair $x=\sqrt{d}/2$ and $y=L(d)^z$ satisfies these constraints, and we can thus deduce that

$$\frac{\psi_{-d}(\sqrt{d}/2,L(d)^z)}{2\sqrt{d}\log d}\ge\frac{1}{4\log d}e^{-u(\log u+\log\log u+c(\varepsilon))}\text{ for }u=\frac{\log\left(\sqrt{d}/2\right)}{\log\left(L(d)^z\right)}.$$

Let us denote by p_s the probability of smoothness of one of the forms in the list. We repeat the experiment $\lceil B\rceil$ times where $B\ge\frac{\log d}{p_s}$. Therefore the probability of failure is given by

$$(1-p_s)^{\lceil B\rceil}=e^{-\lceil B\rceil p_s(1+o(1))}\le e^{-\log d(1+o(1))}.$$

Moreover, as shown in the proof of [34, Prop. 4.4], the number of times Steps 8 and 9 are repeated satisfies $\lceil B\rceil=L(d)^{1/4z+o(1)}$.

To test the $\mathcal{B}$-smoothness of the elements of L_{forms}, we use Bernstein's batch smoothness test [4] on the set $\mathcal{S}$ of first coefficients of the corresponding reduced form, with the set of primes $\mathcal{P}$ that are bounded by $L(d)^z$. This algorithm takes time $b\log(b)^{2+o(1)}$ where b is the total number of bits of all integers in $\mathcal{S}$ and $\mathcal{P}$ combined, which is in $L(d)^{1/4z+o(1)}+L(d)^{z+o(1)}$. Therefore, the time taken for this step is in $L(d)^{1/4z+o(1)}+L(d)^{z+o(1)}$. □

Algorithm 1 will be used for different input forms f in the rest of this paper. To use it in the context of Seysen's relation collection method [34], we set $f=f_i^{2nd}$ to ensure the creation of a row vector of the relation matrix that satisfies $|a_{ii}|>\sum_{j\neq i}|a_{i,j}|$.

Proposition 3. *Assuming GRH, Algorithm 2 is valid, and succeeds with probability at least* $1-\frac{1}{d^{1+o(1)}}$ *in time*

$$L(d)^{z+o(1)}\left(L(d)^{z+o(1)}+L(d)^{1/4z+o(1)}\right).$$

Proof. The running time for Algorithm 1 is $L(d)^{1/4z+o(1)}+L(d)^{z+o(1)}$. So the running time for Algorithm 2 is $n.\left(L(d)^{z+o(1)}+L(d)^{1/4z+o(1)}\right)$. Since $n=L(d)^{z+o(1)}$, the running time for Algorithm 2 is $L(d)^{z+o(1)}\left(L(d)^{z+o(1)}+L(d)^{1/4z+o(1)}\right)$. The probability of success of Algorithm 2 is at least $(1-\frac{1}{d^{1+o(1)}})^n$. Since $n/d^{1+o(1)}\ll 1$ for large d, we can use the binomial approximation, $(1+x)^\alpha\approx 1+x\alpha$, which yields $(1-\frac{1}{d^{1+o(1)}})^n=1-\frac{n}{d^{1+o(1)}}(1+o(1))$. Then, since $n=d^{o(1)}$, the Algorithm 2 succeeds with probability at least $1-\frac{1}{d^{1+o(1)}}$. □

```
    Input  : Fundamental discriminant $-d < 0$, set of prime forms
             $\mathcal{B} = \{f_1, \ldots, f_n\}$, $\varepsilon > 0$
    Output: Matrix $(a_{i,j})$ with $\forall i$, $\prod_j f_j^{a_{i,j}} = 1_{\mathrm{Cl}(-d)}$, $|a_{ii}| > \sum_{j \neq i} |a_{i,j}|$, or
             FAILURE
1   $(a_{i,j}) \leftarrow 0^{n \times n}$;
2   for $i \leq n$ do
3       Use Algorithm 1 with input $f = f_i^{2nd}$, $\mathcal{B}$ and $\varepsilon$ and add the vector
        $\vec{x} - 2nd\vec{e_i}$ to the $i$-th row of $(a_{i,j})$;
4       if Algorithm 1 outputs FAILURE then return FAILURE;
5   end
6   return $(a_{i,j})_{i,j}$;
```

Algorithm 2: Phase 1

5. Phase 2

The second phase is where a modification of the strategy of Hafner and McCurley needs to be made in order to lower the cost of the search for relations with good guarantees of randomness. From a high level perspective, we quantify how sparse the relation lattice Λ' is at a given step to measure the chances of drawing a relation outside of it. When the determinant of $\Lambda' \subseteq \Lambda$ is large enough, new relations are outside of Λ' with overwhelming probability. When this happens, the addition of a new vector to Λ' means that its determinant gets divided by at least a factor 2. At some point, success in enriching Λ' with new relations results in a determinant that is no longer large enough to guarantee the probability that new relations are outside of it. At this point, we need to switch to Phase 3 to finish the calculation of Λ. We fix the target determinant for the switch to $e^{\log^4 d}$. The determinant is calculated only once after collecting $n^{1+o(1)}$ relations. If it does not fall below the required bound, then the procedure returns a failure.

The specificity of the method to search for relations in Phase 2 is that for each new attempt at finding a relation, we first draw a large vector of exponents $\vec{x} = (x_1, \ldots, x_n)$ and compute the form $f = \prod_i f_i^{x_i}$. This product is expensive to evaluate, and the odds of f being $\mathcal{B}$-smooth are low. Then, rather than drawing a new f, we multiply short random products to f in order to generate a relation involving f via Algorithm 1. Due to the properties of Algorithm 1 analyzed in the previous section, each short product, which is inexpensive to evaluate, has a reasonable chance to yield a relation. Moreover, once a relation $\vec{v} \in \Lambda$ is found, we have that $\|\vec{v} - \vec{x}\|$ is short, which means that if we were able to argue that $\vec{x}$ was far enough from Λ', then we know that the new relation satisfies $v \notin \Lambda'$.

To efficiently test the determinant at the end of Phase 2, we need to introduce a new building block to the algorithm. The original Hafner-McCurley algorithm [24] proceeds with computing the (row) Hermite Normal Form of the integer matrix whose rows are the vectors of exponents of the relations that are collected (i.e. the *relation matrix*). However, the Hermite normal form takes time $n^{4+o(1)}$ to compute for a $n^{1+o(1)} \times n$ matrix with polynomial sized entries such as the one we have at the end of Phase 2. Better techniques are now available to find the Smith Normal Form (SNF) of a basis of the lattice generated by the rows of the relation matrix without having to compute the Hermite Normal Form at all. The advantage of using such a method is two-fold:

- It allows the intermediate verification of the determinant of a full-rank $\Lambda' \subseteq \Lambda$ at a lower cost.
- Once a generating set for Λ is found, the SNF of a basis for Λ is $\mathrm{diag}(d_1, \dots, d_n)$ where $\mathrm{Cl}(-d) = \mathbb{Z}/d_1\mathbb{Z} \times \dots \times \mathbb{Z}/d_n\mathbb{Z}$, thus giving us the final result of the class group algorithm.

Lemma 5.1. *Given $n^{1+o(1)}$ vectors in $\mathbb{Z}^n$ with polynomial-sized entries that generate a full-rank sublattice $\Lambda_1 \subseteq \Lambda$, there is a Las Vegas algorithm to compute the SNF of a basis for Λ_1 in time $n^{3+o(1)}$.*

Proof. First, we need to reduce the problem to the computation of the SNF of a square matrix. This is a direct application of [37, Th. 58] which states that there is a Las Vegas algorithm to compute a matrix $B \in \mathbb{Z}^{(2m+5)\times(2m+5)}$ (where $m = n^{1+o(1)}$ is the number of input vectors) whose Hermite Normal Form has the shape $\begin{pmatrix} H & (0) \\ (0) & I \end{pmatrix}$ with entries of polynomial bit-size, in time $n^{\omega+o(1)}$, where $\omega \leq 3$ is the exponent for the complexity of matrix multiplication. This directly implies that the SNF of B is $\mathrm{diag}(d_1, \dots, d_n, 1 \dots, 1)$ where $\mathrm{diag}(d_1, \dots, d_n)$ is the SNF of H, which is a basis of Λ_1. The Las Vegas SNF algorithm recently introduced in [17] allows us to compute the SNF of B in time $n^{3+o(1)}$, which concludes this proof. □

We summarize Phase 2 in Algorithm 3. We analyze its run time and correctness separately from its probability of success, as the former is straightforward, while the latter requires a careful analysis of the odds of the random vector $\vec{z}$ being far enough from the lattice Λ_1 in Step 3.

Proposition 4. *Algorithm 3 is valid, and terminates in time*

$$L(d)^{3z+o(1)} + L(d)^{z+1/4z+o(1)}.$$

Proof. The validity immediately derives from the results previously stated. In particular, note that if $f = \prod_i f_i^{z_i} = \prod_i f_i^{x_i}$ where $\vec{x}$ is the output of Algorithm 1 on input f, then $\vec{x} - \vec{z} \in \Lambda$. The running time of the "for loop" of Steps 2 to 11 is $L(d)^z(L(d)^{z+o(1)} + L(d)^{1/4z+o(1)})$ as Algorithm 1 is repeated $n^{1+o(1)}$ times. Then Step 12 takes time $L(d)^{3z+o(1)}$, and therefore the overall time is

$$L(d)^z(L(d)^{z+o(1)} + L(d)^{1/4z+o(1)}) + L(d)^{3z+o(1)} = L(d)^{3z+o(1)} + L(d)^{z+1/4z+o(1)}.$$

□

We now turn to the analysis of the probability of success of Algorithm 3. Whenever we generate a new relation in Step 5, we need to find a bound on the probability that it does not belongs to Λ_1 already. Each time $\vec{x} - \vec{z} \notin \Lambda_1$, the update Step 9 divides the index Λ_1 within Λ by a factor at least 2, thus getting us one step closer to passing the test of Step 13 on the determinant of Λ_1. Let us denote the box $\{x : x \in \mathbb{Z}^n, \|x\|_\infty \leq d^2\}$ by $W_n(d^2)$ as in [24]. We also denote an n-dimensional sphere of radius r for the Euclidean distance, centered at x by $B(x, r)$. We define the subspace of $W_n(d^2)$,

$$V := \left\{ \bigcup B(x, (2+\varepsilon)\log d) : x \in \Lambda_1 \cap W_n(d^2) \right\} \cap W_n(d^2)$$

By construction, the relations added to the sublattice Λ_1 on Step 9 of Algorithm 3 are within distance $\log(d)$ for the ℓ_1-norm of a vector $\vec{x} \in W_n(d^2)$ drawn uniformly at random. This means that they are also at distance $\log(d)$ of $\vec{x}$ for the Euclidean

```
Input  : Fundamental discriminant $-d < 0$, $\mathcal{B} = (f_i)_{i\leq n}$ for $n = L(d)^z$,
         $\varepsilon > 0$, and $n$ generators of $\Lambda_0 \subseteq \Lambda$ of full rank
Output: $n + \log_2\left(n^{5n/2}d^n\right)$ generators of $\Lambda_1 \subseteq \Lambda$ with $\det(\Lambda_1) \leq e^{\log^4 d}$
         or FAILURE
1  $\Lambda_1 \leftarrow \Lambda_0$;
2  for $i \leq \log_2\left(n^{5n/2}d^n\right)$ do
3      Choose $\vec{z} \in [-d^2, d^2]^n$ uniformly at random ;
4      $\tilde{f} \leftarrow \prod_{k=1}^{n} f_k^{z_k}$;
5      Use Algorithm 1 with input $f = \tilde{f}$, $\mathcal{B}$ and $\varepsilon$ ;
6      if Algorithm 1 outputs FAILURE then
7          return FAILURE;
8      else
9          $\Lambda_1 \leftarrow \Lambda_1 + \mathbb{Z}(\vec{x} - \vec{z})$;
10     end
11 end
12 Compute the determinant $h_1$ of $\Lambda_1$ using Lemma 5.1;
13 if $h_1 > e^{\log^4 d}$ then
14     return: FAILURE
15 else
16     return: the $n + \log_2\left(n^{5n/2}d^n\right)$ generators of $\Lambda_1$
17 end
```

Algorithm 3: Phase 2

distance. By the triangular inequality, they cannot be in Λ_1 as long as the random vector $\vec{x}$ is outside of V.

Lemma 5.2. *Let Λ_1 be a full rank sublattice of Λ with discriminant greater than $e^{\log^4 d}$. Then the probability that a vector $\vec{x}$ drawn uniformly at random in $W_n(d^2)$ be outside of V is at least $1 - \frac{1}{e^{\log^4 d(1+o(1))}}$.*

Proof. We obtain a lower bound on the cardinality $W_n(d^2) \setminus V$ by subtracting an upper bound on the total number of integer points contained inside V from $|W_n(d^2)|$. According to [32, Corollary 1.4], the number of integer elements contained inside each individual n-dimensional sphere is bounded from above by $\frac{3.e^{\pi.k^2.r^2}}{2}$, where r is the radius of the sphere and $k = 10(\log n + 2)$. According to [24, Lemma 1], the number of elements of Λ_1 inside the box $W_n(d^2)$ is $\frac{(2d^2)^n}{\det(\Lambda_1)}.(1 + O(nD/d^2))$, where D is a bound on the diameter of the fundamental domain of Λ_1. In the proof of [24, Lem. 2], it is specified that the diameter of the fundamental domain of Λ_0 is in $O(n^2 d)$ by a triangular inequality argument on the vectors of the basis of Λ_0. Since the fundamental domain of Λ_1 is contained in that of Λ_0, it satisfies the same bound and

$$\left|\Lambda_1 \cap W_n(d^2)\right| \in \frac{(2d^2)^n}{\det(\Lambda_1)}.\left(1 + O\left(\frac{n^3}{d}\right)\right).$$

Therefore, the number of integer elements inside V is bounded from above by

$$\frac{(2d^2)^n}{\det(\Lambda_1)}.\left(1 + O\left(\frac{n^3}{d}\right)\right).\frac{3.e^{\pi.k^2.r^2}}{2}$$

In turn, this implies that the number of integer elements on the space $W_n(d^2) \setminus V$ is at least

$$(2d^2)^n - \frac{(2d^2)^n}{\det(\Lambda_1)} \cdot \left(1 + O\left(\frac{n^3}{d}\right)\right) \cdot \frac{3.e^{\pi.k^2.r^2}}{2}$$

So the probability to draw an $\vec{x}$ in $W_n(d^2) \setminus V$ is at least

$$1 - \frac{3.e^{\pi.k^2.r^2}}{2\det(\Lambda_1)}(1 + o(1)) = 1 - \frac{1}{e^{\log^4 d(1+o(1))}}$$

because $r = (2+\varepsilon)\log d$, $n = L(d)^{z+o(1)}$, $k = 10(\log n + 2)$, and $\det(\Lambda_1) \geq e^{\log^4 d}$. $\square$

Note that we did not discuss the probability of success of the Las Vegas SNF method used in Step 12 of Algorithm 3. The probability of success of the methods described in [37, 17] is at least 1/2. Therefore, we can achieve a probability of success of $1 - \frac{1}{e^{\log^c d}}$ for any constant $c > 0$ without (asymptotically) increasing the cost by repeating the procedure a polynomial amount of times. To assess the overall success probability of Algorithm 3, we need to find a lower bound on the probability of finding enough relations outside of Λ_1 so that at the end of the procedure $\det(\Lambda_1) \leq e^{\log^4 d}$.

Proposition 5. *Under GRH, the probability of success of Algorithm 3 is at least* $1 - \frac{1}{d^{1+o(1)}}$.

Proof. We derive a conservative lower bound on the success probability of Algorithm 3 by first noticing that according to the Hadamard inequality, $\det(\Lambda_0) < n^{5n/2}d^n$. This means that the number of times a vector $\vec{x} - \vec{z} \notin \Lambda_1$ needs to be added on Step 9 of Algorithm 3 is less than the number $\log_2\left(n^{5n/2}d^n\right) = n^{1+o(1)}$ of relation collected. The probability of drawing enough random vectors $\vec{z}$ outside of V in Step 4 is higher than $\left(1 - \frac{1}{e^{\log^4 d(1+o(1))}}\right)^{n^{1+o(1)}}$. Combining this with the probability $1 - \frac{1}{d^{1+o(1)}}$ of success of Algorithm 1, we get that the probability of finding enough relations to bring $\det(\Lambda_1)$ below $e^{\log^4 d}$ is higher than

$$\left(1 - \frac{1}{e^{\log^4 d(1+o(1))}}\right)^{n^{1+o(1)}} \cdot \left(1 - \frac{1}{d^{1+o(1)}}\right)^{n^{1+o(1)}}.$$

Since $n = L(d)^{z+o(1)}$, we have $\frac{n^{1+o(1)}}{e^{\log^4 d(1+o(1))}} \ll 1$ and $\frac{n^{1+o(1)}}{d^{1+o(1)}} \ll 1$ for very large d. Therefore, we can use the binomial approximation $(1+x)^\alpha \approx 1 + x\alpha$ as in the proof of Proposition 3. Moreover, we have $n = d^{o(1)}$ and $d = e^{o(1)\cdot\log^4 d}$, therefore,

$$\left(1 - \frac{1}{e^{\log^4 d(1+o(1))}}\right)^{n^{1+o(1)}} \cdot \left(1 - \frac{1}{d^{1+o(1)}}\right)^{n^{1+o(1)}} = 1 - \frac{1}{d^{1+o(1)}}.$$

$\square$

6. Phase 3

The last phase or the relation search uses the exact same method described in [24, Sec. 3]. In a nutshell, it consists in creating a tower of sublattices $(\Lambda_i)_{2\leq i\leq m}$ of the lattice of relations such that

$$\Lambda_0 \subseteq \Lambda_1 \subseteq \ldots, \subseteq \Lambda_m = \Lambda.$$

The key observation proved in [24, Lem. 2] is that when relations are obtained simply by testing the $\mathcal{B}$-smoothness of elements $f = \prod_i f_i^{x_i}$ for a vector $\vec{x}$ drawn uniformly at random in $W_n(d^2)$, the probability that they belong to a given coset in Λ/Λ_1 is essentially given by $\det(\Lambda)/\det(\Lambda_1)$. This means that new relations have somewhat comparable chances of landing in different cosets of Λ/Λ_1. Once every coset has been hit at least once, the relation collection is complete. What makes Phase 3 less expensive than the analogue procedure in [24, Sec. 3] is the fact that we start the procedure from Λ_1 where $|\Lambda/\Lambda_1| \leq e^{\log^4 d}$ instead of starting from Λ_0 which is only known to satisfy the more pessimistic bound $|\Lambda/\Lambda_0| < n^{5n/2}d^n$, thus requiring the drawing of exponentially more vectors to complete the lattice of relations.

```
Input  : Fundamental discriminant $-d < 0$, $\mathcal{B} = (f_i)_{i\leq n}$ for $n = L(d)^z$,
         $\varepsilon > 0$, and $m = n^{1+o(1)}$ generators of $\Lambda_1 \subseteq \Lambda$ of full rank with
         $\det(\Lambda_1) \leq e^{\log^4 d}$
Output: Cl$(-d)$, or FAILURE
1  $B \leftarrow 4 \cdot e^{u(\log u + \log\log u + c(\varepsilon))}$ for $c(.)$ as in [34, Th. 5.2] ;
2  for $k \leftarrow 1$ to $\frac{\log^4 d + \log d}{\log(1.5)}$ do
3      Initiate an empty list $L_{\text{forms}}$;
4      for $r \leftarrow 1$ to $\lceil B \rceil$ do
5          Choose $\vec{x}$ uniformly at random in $W_n(d^2)$. $f \leftarrow \prod_{i=1}^n f_i^{x_i}$ ;
6          Store $f, \vec{x}$ in $L_{\text{forms}}$;
7      end
8      Test the $\mathcal{B}$-smoothness of all forms in $L_{\text{forms}}$ using [4];
9      if there is $f$ that is $\mathcal{B}$-smooth in $L_{forms}$ then
10         Let $\vec{y} \in \mathbb{Z}^n$ with $f = \prod_{i\leq n} f_i^{y_i}$. $\Lambda_{k+1} \leftarrow \Lambda_k + \mathbb{Z}(\vec{x} - \vec{y})$;
11     else
12         return: FAILURE
13     end
14 end
15 Compute $h^*$ such that $h^* \leq \det(\Lambda) < 2h^*$;
16 Compute $d_1, \ldots, d_n$ such that $\Lambda/\Lambda_k = \prod_i \mathbb{Z}/d_i\mathbb{Z}$ using Lemma 5.1;
17 if $\prod_i d_i \geq 2h^*$ then
18     return: FAILURE
19 else
20     return: $\mathbb{Z}/d_1\mathbb{Z} \times \ldots \times \mathbb{Z}/d_n\mathbb{Z}$
21 end
```

Algorithm 4: Phase 3

Proposition 6. *Under GRH, Algorithm 4 returns* Cl$(-d)$ *with probability at least* $1 - \frac{1}{d^{1+o(1)}}$ *in time*

$$L(d)^{3z+o(1)} + L(d)^{z+1/4z+o(1)}$$

Proof. We rely on [24, Lem. 3] which states that if we generate at least $\frac{\log|\Lambda/\Lambda_1| + \log d}{\log(2/\alpha)}$ new relations stemming from a random choice of $\vec{x} \in W_n(d^2)$ as in Step 5 with $\alpha = 1 + O\left(\frac{n^3}{d}\right)$, then we have a probability at least $1 - \frac{1}{d}$ of generating Cl$(-d)$. For d large enough, $\log(2/\alpha) \geq \log(1.5)$, which ensures that if all attempts at finding relations in the "for loop" of Steps 3 to 14 succeed then we have a probability $1 - \frac{1}{d}$

of obtaining $\mathrm{Cl}(-d)$. The probability of succeeding in finding these relations is at least

$$\left(1-\frac{1}{d^{1+o(1)}}\right)^{\frac{\log^4 d+\log d}{\log(1.5)}} = 1-\frac{1}{d^{1+o(1)}},$$

which proves the result on the probability of success. The evaluation of a product in Step 5 takes time $n^{1+o(1)}$, The number $\lceil B\rceil$ of attempts to generate a relation with probability $1-\frac{1}{d^{1+o(1)}}$ is $L(d)^{1/4z+o(1)}$. This means that the loop between Step 2 and Step 14 costs $L(d)^{z+1/4z+o(1)}$. The computation of h^* runs in polynomial time under the ERH, and the computation of $d_1,\dots,d_n$ via the SNF costs $L(d)^{3z+o(1)}$ as seen before. □

Corollary 1. *Assuming GRH, there is a Las Vegas algorithm to compute* $\mathrm{Cl}(-d)$ *in time* $L(d)^{3/\sqrt{8}+o(1)}$ *with probability at least* $1-\frac{1}{d^{1+o(1)}}$.

Proof. The setup of the algorithm for computing $\mathrm{Cl}(-d)$ consists in computing $\mathcal{B}$ in time $L(d)^{z+o(1)}$. The time of the subsequent phases is bounded by $L(d)^{3z+o(1)} + L(d)^{z+1/4z+o(1)}$. Therefore, the total run time is optimal for $z = 1/\sqrt{8}$. □

Acknowledgments

The authors are thankful to David Jao and Stephen Miller for useful discussions on the role of GRH in [25]. We also thank Oded Regev and Noah Stephens-Davidowitz for their insight on [32], and Arne Storjohann for his help on the complexity of the best SNF and HNF algorithms.

References

[1] E. Bach, Explicit bounds for primality testing and related problems, *Math. Comp.*, **55** (1990), 355–380.

[2] J. Bauch, D. Bernstein, H. de Valence, T. Lange and C. van Vredendaal, Short generators without quantum computers: The case of multiquadratics, in *Proceedings of EUROCRYPT 2017*, **10210** (2017), 27–59.

[3] K. Belabas, T. Kleinjung, A. Sanso and B. Wesolowski, A note on the low order assumption in class group of an imaginary quadratic number fields, Cryptology ePrint Archive, Report 2020/1310, 2020, https://eprint.iacr.org/2020/1310.

[4] D. Bernstein, How to find smooth parts of integers.

[5] W. Beullens, T. Kleinjung and F. Vercauteren, Csi-fish: Efficient isogeny based signatures through class group computations, in *Advances in Cryptology–ASIACRYPT 2019–25th International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, December 8-12, 2019, Proceedings, Part I* (eds. S. D. Galbraith and S. Moriai), vol. 11921 of Lecture Notes in Computer Science, Springer, 2019, 227–247.

[6] J.-F. Biasse, Improvements in the computation of ideal class groups of imaginary quadratic number fields, *Adv. in Math. of Comm.*, **4** (2010), 141–154.

[7] J.-F. Biasse, An L(1/3) algorithm for ideal class group and regulator computation in certain number fields, *Math. Comp.*, **83** (2014), 2005–2031.

[8] J.-F. Biasse, Subexponential time relations in the class group of large degree number fields, *Advances in Mathematics of Communications*, **8** (2014), 407–425, http://aimsciences.org/journals/displayArticlesnew.jsp?paperID=10551.

[9] J. Biasse, T. Espitau, P. Fouque, A. Gélin and P. Kirchner, Computing generator in cyclotomic integer rings, in *Advances in Cryptology–EUROCRYPT 2017–36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part I* (eds. J. Coron and J. Nielsen), vol. 10210 of Lecture Notes in Computer Science, 2017, 60–88.

[10] J.-F. Biasse and C. Fieker, Subexponential class group and unit group computation in large degree number fields, *LMS Journal of Computation and Mathematics*, **17** (2014), 385–403.

[11] J.-F. Biasse, C. Fieker, T. Hofmann and A. Page, Norm relations and computational problems in number fields, 2020.

[12] J.-F. Biasse, C. Fieker and M. Jacobson, Fast heuristic algorithms for computing relations in the class group of a quadratic order, with applications to isogeny evaluation, *LMS Journal of Computation and Mathematics*, **19** (2016), 371–390.

[13] J.-F. Biasse and M. Jacobson, Practical improvements to class group and regulator computation of real quadratic fields, in *Algorithmic Number Theory, 9th International Symposium, ANTS-IX, Nancy, France, July 19-23, 2010. Proceedings* (eds. G. Hanrot, F. Morain and E. Thomé), vol. 6197 of Lecture Notes in Computer Science, Springer, 2010, 50–65.

[14] J.-F. Biasse, M. J. Jr. and A. Silvester, Security estimates for quadratic field based cryptosystems, in *Information Security and Privacy–15th Australasian Conference, ACISP 2010, Sydney, Australia, July 5-7, 2010. Proceedings* (eds. R. Steinfeld and P. Hawkes), vol. 6168 of Lecture Notes in Computer Science, Springer, 2010, 233–247.

[15] J.-F. Biasse and F. Song, Efficient quantum algorithms for computing class groups and solving the principal ideal problem in arbitrary degree number fields, in *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, 2016, 893–902.

[16] J.-F. Biasse and C. van Vredendaal, Fast multiquadratic S-unit computation and application to the calculation of class groups, in *Proceedings of ANTS XIII*, **2** (2019), 103–118.

[17] S. Birmpilis, G. Labahn and A. Storjohann, A las vegas algorithm for computing the smith form of a nonsingular integer matrix, in *ISSAC '20: International Symposium on Symbolic and Algebraic Computation, Kalamata, Greece, July 20-23, 2020* (eds. I. Z. Emiris and L. Zhi), ACM, 2020, 38–45.

[18] J. Buchmann, A subexponential algorithm for the determination of class groups and regulators of algebraic number fields, in *Séminaire de Théorie des Nombres, Paris 1988–1989* (ed. S. Goldstein), Birkhauser, Boston, **91** (1990), 27–41.

[19] J. Buchmann and H. C. Williams, A key exchange system based on real quadratic fields, in *Advances in Cryptology–CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings* (ed. G. Brassard), vol. 435 of Lecture Notes in Computer Science, Springer, 1989, 335–343.

[20] J. Buchmann and H. C. Williams, A key-exchange system based on imaginary quadratic fields, *J. Cryptol.*, **1** (1988), 107–118.

[21] H. Cohen, A course in computational algebraic number theory, *Graduate texts in Math.*, **138** (1993), 88, `https://ci.nii.ac.jp/naid/10006515766/en/`.

[22] H. Cohen, F. D. Y. Diaz and M. Olivier, Subexponential algorithms for class group and unit computations, *Journal of Symbolic Computation*, **24** (1997), 433–441.

[23] E. D. Cristofaro, J. Kim and G. Tsudik, Linear-complexity private set intersection protocols secure in malicious model, in *Advances in Cryptology–ASIACRYPT 2010–16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings* (ed. M. Abe), vol. 6477 of Lecture Notes in Computer Science, Springer, 2010, 213–231.

[24] J. Hafner and K. McCurley, A rigorous subexponential algorithm for computation of class groups, *Journal of the American Mathematical Society*, **2** (1989), 837–850.

[25] D. Jao, S. D. Miller and R. Venkatesan, Expander graphs based on GRH with an application to elliptic curve cryptography, *J. Number Theory*, **129** (2009), 1491–1504.

[26] M. J. J. Jr., Applying sieving to the computation of quadratic class groups, *Math. Comput.*, **68** (1999), 859–867.

[27] T. Kleinjung, Quadratic sieving, *Math. Comput.*, **85** (2016), 1861–1873.

[28] A. Lenstra, On the calculation of regulators and class numbers of quadratic fields, in *Journées Arithmétiques*, Cambridge Univ. Press, **56** (1982), 123–150.

[29] K. S. McCurley, Cryptographic key distribution and computation in class groups, in *Number Theory and Applications (Banff, AB, 1988)*, vol. 265 of NATO Adv. Sci. Inst. Ser. C Math. Phys. Sci., Kluwer Acad. Publ., Dordrecht, 1989, 459–479.

[30] K. Pietrzak, Simple verifiable delay functions, in *10th Innovations in Theoretical Computer Science Conference, ITCS 2019, January 10-12, 2019, San Diego, California, USA* (ed. A. Blum), vol. 124 of LIPIcs, Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2019, Art. No. 60, 15 pp.

[31] M. Pohst, *Algorithmic Methods in Algebra and Number Theory*, 1, Academic Press, 1987, `https://books.google.com/books?id=2SqKQgAACAAJ`.

[32] O. Regev and N. Stephens-Davidowitz, A reverse minkowski theorem, in *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2017, Association for Computing Machinery, New York, NY, USA, 2017, 941–953.

[33] R. Schoof, Quadratic fields and factorization, *Computational Methods in Number Theory, Part II*, Math. Centre Tracts, Math. Centrum, Amsterdam, **155** (1982), 235–286.

[34] M. Seysen, A probabilistic factorization algorithm with quadratic forms of negative discriminant, *Mathematics of Computation*, **48** (1987), 757–780.

[35] D. Shanks, Class number, a theory of factorization, and genera, in *Proceedings of Symposia in Pure Mathematics* (eds. W. J. LeVeque and E. G. Straus), vol. 20, American Mathematical Society, 1971, 415–440.

[36] D. Shanks, The infrastructure of a real quadratic field and its applications, in *Proceedings of the 1972 Number Theory Conference*, American Mathematical Society, 1972, 217–224.

[37] A. Storjohann, The shifted number system for fast linear algebra on integer matrices, *J. Complex.*, **21** (2005), 609–650.

[38] U. Vollmer, Asymptotically fast discrete logarithms in quadratic number fields, in *Algorithmic Number Theory, 4th International Symposium, ANTS-IV, Leiden, The Netherlands, July 2-7, 2000, Proceedings* (ed. W. Bosma), vol. 1838 of Lecture Notes in Computer Science, Springer, 2000, 581–594,

[39] U. Vollmer, An accelerated buchmann algorithm for regulator computation in real quadratic fields, in *Algorithmic Number Theory, 5th International Symposium, ANTS-V, Sydney, Australia, July 7-12, 2002, Proceedings* (eds. C. Fieker and D. R. Kohel), vol. 2369 of Lecture Notes in Computer Science, Springer, 2002, 148–162,

[40] B. Wesolowski, Efficient verifiable delay functions, *J. Cryptology*, **33** (2020), 2113–2147.

Received July 2021; revised September 2021; early access November 2021.

E-mail address: biasse@usf.edu
E-mail address: erukulangara@usf.edu