On the Complexity of Computing Units in a Number Field

V. Arvind and Piyush P Kurur
Institute of Mathematical Sciences
C.I.T Campus,Chennai, India 600 113
{arvind,ppk}@imsc.res.in

August 2, 2008

Abstract

Given an algebraic number field K, such that $[K : \mathbb{Q}]$ is constant, we show that the problem of computing the units group $\mathcal{O}_K^*$ is in the complexity class SPP. As a consequence, we show that *principal ideal testing* for an ideal in $\mathcal{O}_K$ is in SPP. Furthermore, assuming the GRH, the class number of K, and a presentation for the class group of K can also be computed in SPP. A corollary of our result is that solving PELL′S EQUATION, recently shown by Hallgren [12] to have a quantum polynomial-time algorithm, is also in SPP.

1 Introduction

The computation of units in a number field is a fundamental problem in computational number theory and is considered an important algorithmic task in the area. It has been the subject of considerable research in the last two decades and several algorithmic results as well some complexity-theoretic results have been pioneered. Much of this research (e.g. [17, 7, 6]) is based on ideas developed by Buchmann in [3, 4].

In the present paper we are interested in the following problems in computational number theory, from a structural complexity perspective. Let K be a number field given by its minimal polynomial.

1. Computing a fundamental system of units that generates the units group $\mathcal{O}_K^*$ in $\mathcal{O}_K$.
2. Computing a presentation (i.e. a set of generators and relators) for the class group $Cl(K)$ of K and the class number $h(K)$.
3. Testing if a given ideal A of $\mathcal{O}_K$ is a principal ideal.

From a purely complexity theory perspective, earlier research on these problems was by McCurley [16], and Buchmann and Williams [7]. This was followed by the Thiel's work [17] where it is shown that the problem of principal ideal testing is in NP. Furthermore, *assuming the Generalized Riemann Hypothesis,* it is shown in [17] that principal ideal testing and verifying the class number are in NP $\cap$ coNP.

Our interest to further investigate the computational complexity of these problems is motivated by the recent exciting work of Hallgren [12] where it is shown that computing a solution to PELL′S EQUATION is in BQP (the class of problems that have polynomial-time quantum algorithms). Hallgren's main result is that given a *quadratic* number field K, its regulator R_K can be computed by a polynomial-time quantum algorithm. The regulator is the solution to PELL′S EQUATION. For quadratic fields, Hallgren also indicates how principal ideal testing and computing the class group are problems in BQP. Hallgren's results, however, do not appear to generalize to number fields of larger degree. Thus, it remains an open problem if these problems are in BQP for number fields of degree more than two.

How does the class BQP relate to standard complexity classes defined using classical Turing machines? Fortnow and Rogers [11] show that BQP is contained in the counting complexity class AWPP (definitions in Section 1.1). Thus, in a sense, we can think of BQP as a *counting class.* Counting classes is an area of research in structural complexity theory motivated by Valiant's class #P (see e.g. [10]). Intuitively, counting complexity classes are defined by suitable restrictions on the number of accepting and rejecting paths in nondeterministic Turing machines. In the rest of this section we give formal definitions followed by a summary of our results.

1.1 *SPP and other Counting Complexity Classes*

Let $\Sigma = \{0, 1\}$ be the finite alphabet. Let lg denote logarithm to base 2. Let FP denote the class of polynomial-time computable functions and NP denotes all languages accepted by polynomial-time nondeterministic Turing machines. Let $\mathbb{Z}$ denote integers. A function $f : \Sigma^* \to \mathbb{Z}$ is said to be *gap-definable* if there is an NP machine M (i.e. a nondeterministic polynomial time Turing machine M) such that, for each $x \in \Sigma^*$, $f(x)$ is the difference between the number of accepting paths and the number of rejecting paths of M on input x. Let GapP denote the class of gap-definable functions [10]. For each NP machine M let gap_M denote the GapP function defined by it.

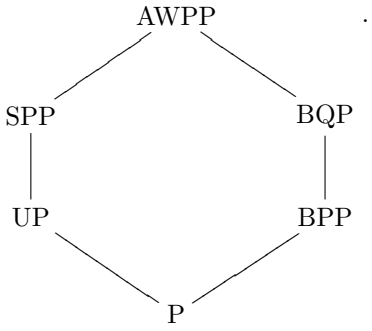
A language L is in UP if there is an NP machine M accepting L such that M has at most one accepting path on any input. The class UP was defined by Valiant and it captures the complexity of 1-way functions. The complexity class SPP is defined as follows. A language L is in SPP if there is an NP machine M such that $x \in L$ implies $\text{gap}_M(x) = 1$ and $x \notin L$ implies $\text{gap}_M(x) = 0$. In this case we say that L is *accepted* by the machine M. Note that the class SPP is essentially a GapP analogue of the class UP and UP $\subseteq$ SPP.

We say that f is in GapP^A, for oracle $A \in \Sigma^*$, if there is an NP^A machine

M^A such that, for each $x \in \Sigma^*$, $f(x)$ is the difference between the number of accepting paths and the number of rejecting paths of M^A on input x. For an oracle A, we can now define the class SPP^A.

The class PP is defined as follows: a language L is in PP if there is an $f \in \mathrm{GapP}$ such that $x \in L$ if and only if $f(x) > 0$. PP is a hard counting class: by Toda's theorem we know that $\mathrm{PH} \subseteq \mathrm{P}^{\mathrm{PP}}$. We say that a language $A \in \Sigma^*$ is *low* for PP if $\mathrm{PP}^A = \mathrm{PP}$. Characterizing the class of languages low for PP is an intriguing open question in structural complexity.

In [10] it is shown that every language in SPP is low for PP. Additionally, SPP has nice closure properties [10]: $\mathrm{P}^{\mathrm{SPP}} = \mathrm{SPP}^{\mathrm{SPP}} = \mathrm{SPP}$. Another class that is low for PP [14] is BPP (the class of languages with polynomial-time randomized algorithms with error probability bounded by, say, 1/3.) Subsequently, the complexity class AWPP was introduced[1] in [9]. The class AWPP generalizes both BPP and SPP, and it is shown that every language in AWPP is low for PP. To complete the picture relating these classes, Fortnow and Rogers in [11] show that BQP is contained in AWPP. It is interesting to note that $\mathrm{NP} \cap \mathrm{coNP}$ is *not* known to be low for PP. Here is a diagram that shows the containments between the complexity classes discussed here.



Although no containment is known between BQP and SPP, it is interesting to compare these classes in terms of natural problems they contain. Important problems known to be in SPP are Graph Isomorphism and the hidden subgroup problem for permutation groups [1]. These problems have resisted efficient deterministic or randomized algorithms, but are considered potential candidates for quantum algorithms. On the other hand, $\mathrm{FP}^{\mathrm{SPP}}$ contains Integer Factoring and Discrete Log that have polynomial-time quantum algorithms.[2]

1.2 *The New Results and the Methods*

We now state the main results of the paper.

[1] For the definition see [9].

[2] In fact, these problems are even in $\mathrm{FP}^{\mathrm{UP}}$. Also, as $\mathrm{P}^{\mathrm{SPP}} = \mathrm{SPP}$, notice that the class $\mathrm{FP}^{\mathrm{SPP}}$ is essentially SPP: for $f \in \mathrm{FP}^{\mathrm{SPP}}$ and input x, the bits of $f(x)$ can be computed in SPP. A similar closure property holds for BQP.

(a) Given a number field K (by its minimal polynomial as input), the problem of computing a fundamental system of units is in $\mathrm{FP}^{\mathrm{SPP}}$, assuming that K is a constant degree extension of $\mathbb{Q}$. As a consequence finding the regulator of K upto polynomially many bits of approximation is also in $\mathrm{FP}^{\mathrm{SPP}}$. As a corollary the PELL′S EQUATION problem is in $\mathrm{FP}^{\mathrm{SPP}}$.

(b) Given a constant-degree number field K and an ideal A of the ring $\mathcal{O}_K$, testing if A is a principal ideal is in SPP.

(c) Given a constant-degree number field K (by its minimal polynomial as input), the problem of computing the class group of K (by finding a generator-relator presentation for it) and finding the class number of K is in $\mathrm{FP}^{\mathrm{SPP}}$, assuming GRH.

In particular, PELL′S EQUATION is also in $\mathrm{FP}^{\mathrm{SPP}}$. Thus, we add to the list of natural problems that are in both SPP and BQP. A brief outline of the methods used to show the above results is given below.

Let M be an oracle Turing machine. For a language A in NP, we say that M^A makes *UP-like queries* to A if there is an NP machine N accepting A such that on all inputs x, $M^A(x)$ makes *only* such queries y for which $N(y)$ has *at most* one accepting path. Effectively, it is like M having access to a UP oracle. We state a useful variant of a result from [15].

Theorem 1.1 ([15]). *Let M be a nondeterministic polynomial-time oracle machine with oracle $A \in \mathrm{NP}$ such that M^A makes UP-like queries to A then the function $h(x) = \mathrm{gap}_{M^A}(x)$ is in* GapP.

Next, we recall an important property of the class SPP shown in [10].

Theorem 1.2 ([10]). *If L is in SPP^A for some oracle $A \in \mathrm{SPP}$ then $L \in \mathrm{SPP}$. I.e. $\mathrm{SPP}^{\mathrm{SPP}} = \mathrm{SPP}$.*

The following lemma, which is a straightforward consequence of Theorem 1.1 and of Theorem 1.2, is in a form useful for this paper.

Lemma 1.3.

- *Suppose L is in SPP^A accepted by the nondeterministic polynomial-time oracle machine M^A with oracle $A \in \mathrm{NP}$ (i.e. $x \in L$ implies that $\mathrm{gap}_{M^A}(x) = 1$, and $x \notin L$ implies that $\mathrm{gap}_{M^A}(x) = 0$), such that the machine M^A makes UP-like queries to A, then L is in* SPP.
- *Suppose a function $f : \Sigma^* \to \Sigma^*$ is in FP^A (i.e. f is computed by a polynomial-time oracle transducer M^A) where $A \in \mathrm{NP}$, such that the machine M^A makes UP-like queries to A, then f is in $\mathrm{FP}^{\mathrm{SPP}}$.*

Lemma 1.3 is a crucial tool in obtaining the $\mathrm{FP}^{\mathrm{SPP}}$ upper bounds. For computing a fundamental system of units in $\mathrm{FP}^{\mathrm{SPP}}$ we first show that a bound $B \in \mathbb{Q}$ can be computed in $\mathrm{FP}^{\mathrm{SPP}}$ such that the regulator R_K of K lies between B and $2B$. Once such a bound is computed, we again apply an algorithm based

on Lemma 1.3 to compute a *canonical* fundamental system of units in $\mathrm{FP}^{\mathrm{SPP}}$. This notion of canonical fundamental system of units is developed and explained in Sections 2 and 3, where we show how to transform an arbitrary fundamental system of units to the canonical set.

Once we have the $\mathrm{FP}^{\mathrm{SPP}}$ upper bound for computing fundamental units, we can design an SPP algorithm for principal ideal testing. If we assume the generalized Riemann hypothesis then, by a result of Bach [2], we can apply our SPP algorithm for principal ideal testing and give an $\mathrm{FP}^{\mathrm{SPP}}$ algorithm for computing the class group $Cl(K)$.

1.3 *Comparison with previous results*

As mentioned, Thiel [17] has shown that principal ideal testing is in NP. Thiel also shows, assuming the GRH, that principal ideal testing and verifying class number are in $\mathrm{NP} \cap \mathrm{coNP}$. On the other hand, our results on principal ideal testing and computing a fundamental system of units are *unconditional*, but applicable to only number fields of constant degree. The $\mathrm{FP}^{\mathrm{SPP}}$ upper bound for the class group problem depends on the GRH.

No containment relation is known between SPP and $\mathrm{NP} \cap \mathrm{coNP}$ or BQP and $\mathrm{NP} \cap \mathrm{coNP}$. Furthermore, we remark here that $\mathrm{NP} \cap \mathrm{coNP}$ is not known to be low for PP. Thus, the results of Thiel [17] are incomparable to our results.

An important computational aspect in all our results is the notion of *compact representation* as explained by Thiel [17], based on Buchmann's earlier papers [3, 4]. We need compact representations to succinctly express units as well as the generating element of a principal ideal in $\mathcal{O}_K$.

2 Compact representation

Let K be a number field of degree n and let $\mathcal{O}$ be the ring of integer of K. Let D be the discriminant of K. For an element $\alpha \in K$ by $\mathrm{N}(\alpha)$ we mean the norm $\mathrm{N}_{\mathbb{Q}}^{K}(\alpha)$. Without loss of generality we assume that the input to the algorithm is $\mathcal{O}$ presented as a $\mathbb{Z}$-module with basis $\omega_1, \ldots \omega_n$ and constants c_{ijk} such that $\omega_i \omega_j = \sum_k c_{ijk} \omega_k$. For, computing the maximal order from a given order is reducible to the problem of finding the square free part of an integer which can be done in $\mathrm{FP}^{\mathrm{SPP}}$, as factoring integers is in $\mathrm{FP}^{\mathrm{SPP}}$. By size of $\mathcal{O}$ we mean $\sum size(c_{ijk})$. The constants c_{ijk} will be the called the explicit data for K.

Fractional ideals $\mathfrak{a}$ of $\mathcal{O}$ will be presented by giving a $\mathbb{Z}$-basis for $\mathfrak{a}$. Let $\alpha_i = \sum_j a_{ij} \omega_j$, $1 \leq i \leq n$, be a basis of $\mathfrak{a}$ then by $\mathrm{HNF}(\mathfrak{a})$ we mean the Hermite normal form of the matrix (a_{ij}). Once ω_i's are fixed, for every ideal $\mathfrak{a}$, $\mathrm{HNF}(\mathfrak{a})$ is unique. Since the Hermite normal form of a matrix can be computed in polynomial time this gives a polynomial time algorithm for testing whether two ideals are equal.

Let $\sigma_1, \ldots, \sigma_r$ be all the r real embeddings and $\sigma_{r+1}, \overline{\sigma}_{r+1}, \ldots, \sigma_{r+s}, \overline{\sigma}_{r+s}$ be all the $2s$ complex embeddings of K. Define the $r+s$ absolute values on K

as follows.

$$|\alpha|_i = \begin{cases} |\sigma_i(\alpha)| & \text{if } 1 \le i \le r \\ |\sigma_i(\alpha)|^2 & \text{if } r+1 \le i \le r+s \end{cases}$$

For $\alpha \in K$, by height of α, denoted by $\mathrm{H}(\alpha)$, we mean $\max\{|\alpha|_i : 1 \le i \le r+s\}$.

Lemma 2.1. *Given $\mathcal{O}$, we have $\mathrm{H}(\omega_i) \le n2^{size(\mathcal{O})}$, and $\lg D \le n(2\lg n + size(\mathcal{O}))$.*

Proof. Let l be such that for $1 \le i \le n$ $\mathrm{H}(\omega_l) \ge \mathrm{H}(\omega_i)$. Then we have

$$\mathrm{H}(\omega_l\omega_l) \le \sum_k |c_{llk}|\,\mathrm{H}(\omega_k) \le n2^{size(\mathcal{O})}\mathrm{H}(\omega_l).$$

Hence $\mathrm{H}(\omega_l) \le n2^{size(\mathcal{O})}$. Also since $D = \sum_{g\in S_n}\prod_{i=1}^{n}\sigma_{g(i)}(\omega_i)$ we have $D \le n!.\mathrm{H}(\omega_l)^n$. □

Fix a basis for $\mathcal{O}$. For any $\alpha \in \mathcal{O}$ there is a unique set of integers $a_1, \ldots, a_n$ such that $\alpha = \sum a_i\omega_i$. By giving the vector of integers a_i's the algebraic integer α is completely specified. Following [17] this is the *standard representation* of α which is unique for a fixed $\mathbb{Z}$-basis for $\mathcal{O}$. The size of α in standard representation is $size_s(\alpha) = \sum size(a_i)$.

The following result from [17] describes a *compact representation* of algebraic integers.

Theorem 2.2. *For $\alpha \in \mathcal{O}$ there exists $k \le \lg(\lg(D) + (n-1)\lg \mathrm{H}(\alpha)) + 2$ and $\gamma, \alpha_i \in \mathcal{O}$ and $d_i \in \mathbb{Z}$, $1 \le i \le k$, with $\mathrm{H}(\gamma) \le \mathrm{N}(\alpha)^{2/n}$, $\mathrm{H}(\alpha_i) \le D^{\frac{3}{4}(m+2)}$ and $0 < d_i \le \sqrt{D}$ such that*

$$\alpha = \gamma\prod_{i=1}^{k}\left(\frac{\alpha_i}{d_i}\right)^{2^{k-i}}.$$

Moreover, for $1 \le j \le k$ the ideal $\prod_{i=1}^{j}\left(\frac{\alpha_i}{d_i}\right)^{2^{k-i}}\mathcal{O}$ is a reduced ideal.

A product of this form can be presented as a tuple $\langle k, \gamma, \langle\alpha_i, d_i\rangle_{i=1}^{k}\rangle$. For a given compact representation of α the size of the representation is the sum of the sizes of the integers d_i and the sizes of algebraic integers γ and α_i's in their standard representation. Compact representations are not unique for a given α even for a given $\mathbb{Z}$-basis. Let $size_c(\alpha)$ denotes the maximum of sizes of all compact representation of α. Using Lemma 2.1 and [17, Corollary 15] we have the following theorem on compact representations.

Theorem 2.3. [17] *For nonzero $\alpha \in \mathcal{O}$*

$$\begin{aligned} size_s(\alpha) &\le (n\lg \mathrm{H}(\alpha)\cdot size(\mathcal{O}))^{O(1)}. \\ size_c(\alpha) &\le \left(n^2\lg^2(n)\cdot size(\mathcal{O}).\lg(size(\mathcal{O}))\cdot \mathrm{N}(\alpha)\cdot \lg\lg \mathrm{H}(\alpha)\right)^{O(1)} \end{aligned}$$

Furthermore, given the compact representation $\langle k, \gamma, \langle \alpha_i, d_i \rangle_{i=1}^{k} \rangle$ *of* α*, there is a polynomial time algorithm that computes the Hermite Normal Form for the ideal* $\alpha\mathcal{O}$*.*

Conversely the following proposition from [17] gives a bound on the height of the algebraic number based on the size of their representation.

Proposition 2.4. *Let* α *be any algebraic integer of* K*, a number field of discriminant* D*. Then we have:*

1. *For all* j*,* $\mathrm{H}(\alpha) \leq n^2 2^{size_s(\alpha)+size(\mathcal{O})}$*.*
2. *For all* j*,* $\ln \mathrm{H}(\alpha) \leq \ln \mathrm{N}(\alpha) + n^3 \lg n \cdot size_c(\alpha) \cdot size(\mathcal{O}) \cdot 2^{size_c \alpha}$*.*

3 Minimal bases for lattices

For a set of linearly independent vectors $\mathbf{a}_1, \mathbf{a}_2, \ldots \mathbf{a}_n$ let $\mathbf{a}_1^*, \mathbf{a}_2^*, \ldots, \mathbf{a}_n^*$ denote the corresponding GSO (Gram-Schmidt) basis. Given a lattice $\Lambda = \sum_{i=1}^{n} \mathbb{Z}\mathbf{b}_i$ in $\mathbb{R}^n$ with basis $\mathbf{b}_i = \sum_{j=1}^{n} b_{ij}\mathbf{e}_i$. Let $M = (\mu_{ij})$ be the matrix that transforms the GSO basis given by $\mathbf{b}_i^*$'s to the basis given by the $\mathbf{b}_i$'s. We say that the basis is *proper* if for every $i < j \leq n$ we have $-\frac{1}{2} \leq \mu_{ij} < \frac{1}{2}$. The following holds for any lattice.

Lemma 3.1. *Given a lattice* Λ *with basis* $\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_n$*, a new basis* $\mathbf{b}_1, \mathbf{b}_2, \ldots \mathbf{b}_n$ *can be computed in polynomial time such that* $\mathbf{b}_i$*'s form a proper basis and* $\mathbf{b}_i^* = \mathbf{a}_i^*$ *(i.e. the GSO basis of both vectors are the same).*

Proof. Here is the algorithm.

```
   b_1 := a_1;
1  for i = 2 to n do
       Let a_i = a_i^* + sum_{j=1}^{i-1} mu_ij a_j^*;
       b_i := a_i;
2      for j = i - 1 downto 1 do
           if mu_ij >= 1/2 or mu_ij < -1/2 then
               Let n be the nearest integer to mu_ij;
3              b_i := b_i - n b_j;
           end
       end
   end
```

The invariant for the loop in step 1 is $-\frac{1}{2} \leq \mu_{kj} < \frac{1}{2}$ for all $1 \leq j < k \leq i-1$. If the invariant is violated at i for some j then the loop in step 2 fixes it. For a given k note that step 3 does not affect any of the μ_{ij} for $j > k$. It is also clear that step 3 does not affect the GSO of the basis. □

Given the vector space $W = U \oplus V$ such that U and V are orthogonal, for $\mathbf{w} \in W$, if $\mathbf{w} = \mathbf{u} + \mathbf{v}$, $\mathbf{u} \in U$ and $\mathbf{v} \in V$, then $\mathbf{w}/U$ denotes the vector $\mathbf{v}$ (i.e.

the component of $\mathbf{w}$ orthogonal to the space U). For a lattice Λ, Λ/V is the lattice $\{\mathbf{v}/V : \mathbf{v} \in \Lambda\}$. If $\mathbf{b}_1, \mathbf{b}_2 \ldots \mathbf{b}_n$ forms a basis for Λ then any vector of Λ/V can be expressed as an integer linear combination of $\mathbf{b}_i/V$'s.

Given a lattice Λ, a basis $\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n$ is called a *minimal basis* if it is proper and it satisfies the following conditions:

1. $\mathbf{b}_1$ is a ℓ_1-shortest vector in Λ.
2. For all i, if V_{i-1} is the span of the vectors $\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_{i-1}$ then $\mathbf{b}_i/V_{i-1}$ is the vector of least ℓ_1 norm in the lattice Λ/V_{i-1}.

To find a canonical basis for the lattice Λ one can define a total order on the set of all minimal basis of Λ and choose the least basis under that order. For two vectors $\mathbf{u} = \sum u_i \mathbf{e}_i$ and $\mathbf{v} = \sum v_i \mathbf{e}_i$, $\mathbf{u} \prec \mathbf{v}$ if $\| \mathbf{u} \|_1 < \| \mathbf{v} \|_1$ or if $\| \mathbf{u} \|_\mathbf{1} = \| \mathbf{v} \|_\mathbf{1}$ then there is an $1 \leq i \leq n$ such that $u_j = v_j$ for all $1 \leq j < i$ and $u_i < v_i$.

Consider two minimal basis $A = \{\mathbf{a}_i\}_{i=1}^n$ and $B = \{\mathbf{b}_i\}_{i=1}^n$ for the lattice Λ. Let $A^* = \{\mathbf{a}_i^*\}_{i=1}^n$ and $B^* = \{\mathbf{b}_i^*\}_{i=1}^n$ be their GSO basis respectively. For two such minimal basis $A \prec B$ if there is an i such that for all $j < i$, $\mathbf{a}_j = \mathbf{b}_j$ and $\mathbf{a}_i^* \prec \mathbf{b}_i^*$.

Theorem 3.2. *On the set of minimal bases, the relation $\prec$ forms a total order.*

Proof. Suppose $\prec$ does not form a total order then we have two minimal basis A and B and an index i such that $\mathbf{a}_j = \mathbf{b}_j$ for all $j < i$ and $\mathbf{a}_i^* = \mathbf{b}_i^*$ yet $\mathbf{a}_i \neq \mathbf{b}_i$. Expressing $\mathbf{a}_i$ and $\mathbf{b}_i$ in the respective GSO basis we have

$$\mathbf{a}_i = \sum_{j=1}^{i} \alpha_j \mathbf{a}_j^* \qquad \mathbf{b}_i = \sum_{j=1}^{i} \beta_j \mathbf{b}_j^*.$$

Let k be the index such that for all $j > k$, $\alpha_j = \beta_j$ and $\alpha_k \neq \beta_k$. Clearly $k < i$. Since the basis A and B are proper we have α_j and β_j lie in the interval $\left[-\frac{1}{2}, \frac{1}{2}\right)$. Consider the vector $\mathbf{u} = \mathbf{a}_i - \mathbf{b}_i$. Since $\mathbf{a}_j^* = \mathbf{b}_j^*$ for all $1 \leq j \leq i$ we have $\mathbf{u} = (\alpha_k - \beta_k)\mathbf{a}_k^* + \mathbf{u}'$ where $\mathbf{u}'$ lies in the vector space V_{k-1} that is spanned by $\{\mathbf{a}_j\}_{j=1}^{k-1}$. But then $\| \mathbf{u}/V_{k-1} \|_1 = |\alpha_k - \beta_k| \, \| \mathbf{a_k}^* \|_1 < \| \mathbf{a_k}/V_{k-1} \|_1$ which contradicts the fact that A and B are minimal. □

Given an algorithm for finding the $\prec$-least vector of a lattice, Theorem 3.2 suggests the following algorithm for finding $\prec$-minimum basis for a lattice Λ.

To complete this section we give an algorithm for finding the $\prec$-minimum element of a lattice Λ. We apply Lenstra's algorithm for integer linear programming [13] that runs in polynomial time if the number of variables is constant.

Lemma 3.3. *If $\Lambda \subset \mathbb{R}^n$ is a lattice of rank r then assuming n is a constant there is a polynomial time algorithm for finding the $\prec$-minimum element of Λ.*

Proof. Let $\Lambda = \mathbb{Z}[\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_r]$ where the basis $\mathbf{b}_i$'s are given by $\mathbf{b}_i = \sum b_{ij} \mathbf{e}_j$. The algorithm goes in two steps. The first step is to find a vector with least ℓ_1 norm. In the subsequent steps this solution is refined further till we get the $\prec$-minimum vector.

Input: A set of linearly independent vectors $\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n$.
Output: The $\prec$-minimum basis for Λ.
Let Λ be the lattice $\mathbb{Z}[\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n]$;
Find the $\prec$-least element $\mathbf{u}$ of the lattice Λ;
Let $\{\mathbf{u}_k\}_{k=1}^n$ be a basis for Λ such that $\mathbf{u}_1 = \mathbf{u}$;
Let $\mathbf{u}_k^* = \mathbf{u}_k - \frac{\langle \mathbf{u}, \mathbf{u}_k \rangle}{\langle \mathbf{u}, \mathbf{u} \rangle} \mathbf{u}$ $1 < k \leq n$;
Find the $\prec$-minimum basis for the lattice Λ^* generated by $\{\mathbf{u}_k^*\}_{k=2}^n$ recursively;
Let this basis be $\{\mathbf{b}_i' = \sum_{j=2}^n x_{ij} \mathbf{u}_j^*\}_{i=2}^n$;
Consider the basis $\{\mathbf{a}_i\}_{i=1}^n$ defined by $\mathbf{a}_1 = \mathbf{u}$ and $\mathbf{a}_i = \sum_{j=2}^n x_{ij} \mathbf{u}_j$, $2 \leq i \leq n$;
Converting this basis to a proper basis using Lemma 3.1 completes that algorithm;

Algorithm 1: Computing $\prec$-minimal basis

To find a vector with least ℓ_1 norm we have the following integer programming problem in r variables $x_1, x_2, \ldots, x_r$: Minimize the expression $f(\mathbf{x}) = \sum_{i=1}^n |\sum_{j=1}^r b_{ji} x_j|$ given $f(\mathbf{x}) > 0$. Although this is not an integer linear programming problem one can use the algorithm of Lenstra here as follows: For the 2^n different vector $\mathbf{c} \in \{-1, 1\}^n$, solve the following integer linear programming problem and pick the best among those 2^n different solutions.

$$\text{Minimize } \sum_{i=1}^r \left(\sum_{j=1}^n b_{ij} c_j \right) x_i,$$

under the constraints

$$\sum_{i=1}^r \left(\sum_{j=1}^n b_{ij} c_j \right) x_i > 0 \tag{1}$$

$$0 \leq c_j \sum_{i=1}^r b_{ij} x_i \leq B, \qquad 1 \leq j \leq n. \tag{2}$$

The first constraints expresses the fact that the solution should be nonzero. The second set of constraints express the fact that we are choosing the right c_i's.

The B in the equation is an upper bound on the ℓ_∞ of the shortest vector. The ℓ_1 norm of any particular vector in the basis will be a suitable value for B.

Having obtained a solution for the ℓ_1-shortest vector say $\mathbf{u}$ one has to refine the solution to get a $\prec$-minimum solution. Let $\mathbf{u}_j$ denote a ℓ_1-shortest vector which agrees with the $\prec$-minimum vector on all coordinates less than or equal to j, then $\mathbf{u}_n$ is the desired solution. Let $\mathbf{u}_0 = \mathbf{u}$. Having got the vector $\mathbf{u}_j$ to compute $\mathbf{u}_{j+1}$ we minimize $\sum_{i=1}^r x_i b_{ij+1}$ under the constraints $\| \sum_{i=1}^r x_i \mathbf{b}_i \|_1 = \| \mathbf{u}_j \|_1$ and $\sum_{i=1}^r x_i b_{ik} = u_{jk}$ for $1 \leq k \leq j$. Here u_{jk} denotes the component of $\mathbf{u}_j$ in the direction $\mathbf{e}_k$. One can use the same trick as before to convert this to an integer linear programming problem and use Lenstra's algorithm. Since the dimension n is bounded, the running time is polynomial. □

Combining Lemma 3.3 and algorithm 1 we have the following result.

Theorem 3.4. *Given a lattice a basis $\{\mathbf{b}_i\}_{i=1}^r$ of a rank r lattice $\Lambda \subseteq \mathbb{R}^n$ there is a polynomial time algorithm to compute the $\prec$-minimal basis of Λ assuming n to be a constant*

4 Units of a number field

Let K be a number field of degree n and let $\mathcal{O}$ be the set of algebraic integers of K. If K has r real embeddings and $2s$ complex embeddings then by Dirichlet's theorem (see, e.g. [8]) there exists a set of $m = r + s - 1$ units $\{\varepsilon_i\}_{i=1}^m$, called a fundamental system of units, such that every unit of $\mathcal{O}$ can be expressed as $\zeta\varepsilon_1^{x_1} \dots \varepsilon_m^{x_m}$, $x_i \in \mathbb{Z}$, where ζ is a root of unity in K. Consider the map $\mathrm{Log} : K \mapsto \mathbb{R}^m$ defines as follows:

$$\mathrm{Log}\,(\alpha) = \langle \ln|\alpha|_1, \ln|\alpha|_2, \dots, \ln|\alpha|_m \rangle$$

From Dirichlet's theorem it follows that the set $\mathrm{Log}\,(\mathcal{O}^*)$ is a lattice in $\mathbb{R}^m$ with basis $\mathrm{Log}\,(\varepsilon_i)_{i=1}^m$. Often it is necessary to work with vectors in this lattice whose coordinates are in general irrationals. We will use rational approximations of these vectors instead.

An important algorithmic task that will be useful in the $\mathrm{FP}^{\mathrm{SPP}}$ algorithm is to compute a canonical fundamental system of units from a given fundamental system of units $U = \{\varepsilon_i\}_{i=1}^m$. In this section we give a polynomial time algorithm for the above task assuming that the degree $[K : \mathbb{Q}]$ is constant. The next theorem is a re-statement of [17, Lemma 16] using the bound in Lemma 2.1.

Theorem 4.1. [17] *There exists a fundamental system of units $\{\varepsilon_i\}_{i=1}^m$ for $\mathcal{O}$ such that $size_c(\varepsilon_i) = (n.size(\mathcal{O}))^{O(1)}$ for all $1 \le i \le m$.*

Consider the fundamental system of units $\{\eta_i\}_{i=1}^m$ that corresponds to $\prec$-minimal basis of the lattice $\mathrm{Log}\,(\mathcal{O}^*)$. We have the following observation.

Lemma 4.2. *For all $1 \le i \le m$, $size_c(\eta_i) = (n.size(\mathcal{O}))^{O(1)}$.*

Proof. The basis $\mathbf{b}_i = \mathrm{Log}\,(\eta_i)$ is the $\prec$-minimal basis of $\Lambda = \mathrm{Log}\,(\mathcal{O}^*)$. Let $\mathbf{b}_i^*$ denote the corresponding GSO basis and let $V_i = \mathrm{Span}\{\mathbf{b}_i\}_{i=1}^m$. Let $\{\varepsilon_i\}_{i=1}^m$ be any fundamental system of units satisfying the condition in Theorem 4.1 then $\mathbf{a}_i = \mathrm{Log}\,(\varepsilon_i)$ spans the lattice Λ. Without loss of generality we may assume that $\mathbf{a}_i \notin V_i$. Since $\mathbf{b}_i$'s form the $\prec$-minimal basis of Λ we have $\| \mathbf{b}_i^* \|_\infty \le \| \mathbf{b}_i^* \|_1 \le \| \mathbf{a}_i / V_{i-1} \|_1 \le m \| \mathbf{a}_i \|_\infty$. Hence we have $\| \mathbf{b}_i \|_\infty \le \| \mathbf{b}_i^* \|_\infty + \frac{1}{2} \sum_{j=1}^{i-1} \| \mathbf{b}_j^* \| \le m.A.(i-1)/2$, where A is an upper bound on $\| \mathbf{a}_i \|_\infty$. From Theorem 4.1 and Proposition 2.4 we have $A \le (n \cdot size(\mathcal{O}) + 2^{size_c(\varepsilon_i)})^{O(1)}$ where i is such that $\mathrm{Log}\,(\varepsilon_i)$ has the largest ℓ_∞ norm. Hence for every i we have $\ln \mathrm{H}\,(\eta_i) \le (n \cdot size(\mathcal{O}) + 2^{size_c(\varepsilon_i)})^{O(1)}$. Together with Theorem 2.3 we have the result. □

We now describe how a canonical fundamental system of units can be computed given an arbitrary fundamental system of units. The following theorem, based on a remark in [17], will be useful.

Theorem 4.3. [17] *Assuming that* $[K : \mathbb{Q}]$ *is constant, there is a polynomial time algorithm that takes as input a principal ideal* $\mathfrak{a} = \alpha\mathcal{O}$ *by its Hermite Normal Form and a good rational approximation for* $\mathrm{Log}\,(\alpha)$*, and outputs a compact representation for* $\zeta\alpha$ *where* ζ *is a root of unity in* K*.*

Remark 4.4. The point to note in the above theorem is that only $\mathrm{Log}\,(\alpha)$ is given (as a rational approximation) and not the compact representation of α. Also notice that $\mathrm{Log}\,(\alpha)$ is unique only upto multiplication by roots of unity in K. The theorem promises that one such element $\zeta\alpha$, which depends only on $\mathrm{HNF}(\mathfrak{a})$ and $\mathrm{Log}\,(\alpha)$, is computable in polynomial time.

Theorem 4.5. *Assuming* $[K : \mathbb{Q}]$ *is a constant, there is a polynomial time deterministic algorithm that takes as input a fundamental system of units (as compact representations) and outputs another fundamental system of units* $\{\eta_i\}_{i=1}^m$ *(as compact representations) corresponding to the* $\prec$*-minimal basis for* $\mathrm{Log}\,(\mathcal{O}^*)$*. Furthermore,* $\{\eta_i\}_{i=1}^m$ *is canonical in the sense that it does not depend on the input fundamental system of units.*

Proof. Given a fundamental system of units $\{\varepsilon_i\}_{i=1}^m$ compute $\{\mathrm{Log}\,(\varepsilon_i)\}_{i=1}^m$ to the desired approximation. Compute the $\prec$-minimal basis for the lattice generated by $\{\mathrm{Log}\,(\varepsilon_i)\}_{i=1}^m$ using algorithm in Theorem 3.4. Let it be $\{\mathbf{b}_i\}_{i=1}^m$. Use algorithm in Theorem 4.3 to compute compact representations of units $\{\eta_i\}_{i=1}^m$ corresponding to the vectors $\{\mathbf{b}_i\}_{i=1}^m$. Since the basis $\{\mathbf{b}_i\}_{i=1}^m$ is unique (upto approximation) and since all the algorithms involved are polynomial time deterministic algorithm the output generated is independent of the fundamental system of units that was given as input. □

Given any set of units $\{\varepsilon_i\}$, we now analyze the approximation of $\mathrm{Log}\,(\varepsilon_i)$ required in order to accurately compute the canonical fundamental system of units.

Let $\{\eta_i\}$ the canonical fundamental system of units and let $\mathbf{a}_i = \mathrm{Log}\,(\eta_i)$. Let $\mathbf{b}_i = \mathrm{Log}\,(\varepsilon_i)$. Consider the matrices $A = (a_{ij})$ and $B = (b_{ij})$ (recall that for a vector $\mathbf{v}$, v_i denotes its i^{th} component). Since $\mathbf{a}_i$'s and $\mathbf{b}_i$'s span the same lattice $\Lambda = \mathrm{Log}\,(\mathcal{O}^*)$, there is a unimodular transformation $U \in \mathrm{SL}_m(\mathbb{Z})$ such that $A = UB$. Note that the determinant of B is the regulator which is at least 0.2 and hence it can be shown that each entries of U is of size bounded by a polynomial in the sized of entries in A and B. Let B_q denote the q bit approximations of B and let $A_q = UB_q$. We have

$$\| A_q - A \|_\infty = \| U(B_q - B) \|_\infty \leq m \| U \|_\infty 2^{-q}.$$

If we take q large enough so that $\| A_q - A \|_\infty$ is small enough for us to recover back the compact representation of η_i's we are through. It is easy to see that a q that is bounded by a polynomial in the sizes of entries of A and B is sufficient for this purpose.

Lemma 4.6. *In the algorithm of Theorem 4.5, it suffices to approximate* $\mathrm{Log}\,(\varepsilon_i)$ *to an error of* 2^{-q} *where* $q \leq (\lg(\| A \|_\infty) \lg(\| B \|_\infty))^{O(1)}$*.*

5 Computing Units is in $\mathrm{FP}^{\mathrm{SPP}}$

In this section we give an $\mathrm{FP}^{\mathrm{SPP}}$ algorithm for computing a fundamental system of units for a number field K. The algorithm is in two stages. In the first stage it computes a number B such that the regulator R_K lies in the range $[B, 2B)$. Notice that, having computed such a bound B, we can test in deterministic polynomial time if an arbitrary set of m algebraic numbers is a fundamental system of units. Given this value of B, in the second stage the $\mathrm{FP}^{\mathrm{SPP}}$ algorithm computes a fundamental system of units of K. The first stage is described in the following lemma.

Lemma 5.1. *Given a number field K, there is an $\mathrm{FP}^{\mathrm{SPP}}$ algorithm to compute a constant B such that the regulator R_K of K, lies in the interval $[B, 2B)$.*

Proof. We give a polynomial time algorithm that makes UP-like queries to an NP oracle. Consider the following NP language:

$$A = \{\langle x, \mathcal{O}_K\rangle \mid \text{there is a subgroup of index } y \text{ in } \mathcal{O}_K^* \ : \ x \leq yR_K < 2x\}.$$

We consider the following nondeterministic procedure that accepts A.

Input: A rational x and basis for the ring of integers, $\mathcal{O}$ of a number field K
Output: "Yes" if there is a subgroup of $\mathcal{O}^*$ of index y such that $x \leq yR_K < 2x$; "No" otherwise.

1 Guess the polynomial sized compact representations of m units of $\mathcal{O}$ say $\{\varepsilon\}_{i=1}^m$;
Compute rational approximations of $\{\mathrm{Log}\,(\varepsilon_i)\}_{i=1}^m$ and check if they form a linearly independent set. If not reject;
Compute the volume of the parallelepiped formed by the $\{\mathrm{Log}\,(\varepsilon_i)\}_{i=1}^m$ and check if it lies in the interval $[x, 2x)$. If not reject;
2 Use the algorithm in Theorem 4.5 to compute a canonical fundamental system of units say η_i's;
Check the whether the compact representations obtained in step 2 is same as the guessed compact representations. If yes accept else reject;

We now explain Step 1. First guess m (polynomial sized) compact representations of m algebraic integers $\{\alpha_i\}_{i=1}^m$. Applying Theorem 2.3 it is possible to compute $\alpha_i\mathcal{O}$ and check whether $\alpha_i\mathcal{O} = \mathcal{O}$ in polynomial time.

In the above NP machine if x is such that $x \leq R_K < 2x$ then there will be only one accepting path. This is because any set of m units that was guessed in step 1 will indeed be a fundamental system of units. For each of these accepting paths, step 2 will give a unique compact representation of a fundamental system of units— those units that in the Log map gives the $\prec$-minimum basis of $\mathrm{Log}\,(\mathcal{O}^*)$. Hence the only path that will accept is that which guessed that unique compact representation of units corresponding to the $\prec$-minimal basis of $\mathrm{Log}\,(\mathcal{O}^*)$.

It is known that the regulator of any number field is at least 0.2 [8]. We now describe the procedure that computes the required bound B:

Input: A $\mathbb{Z}$-basis for the ring of integers, $\mathcal{O}$, of a number field K
Output: A rational B such that $B \leq R_K < 2B$
$B := 0.2$;
while true do
  if $\langle B, \mathcal{O}_K \rangle \in A$ **then return** B;
  $B := 2B$;
end

Since this procedure makes UP-like queries to the NP language A we can convert it into a $\mathrm{FP}^{\mathrm{SPP}}$ algorithm by Lemma 1.3. □

Lemma 5.2. *Given a constant B such that $B \leq R_K < 2B$, a fundamental system of units can be computed in* $\mathrm{FP}^{\mathrm{SPP}}$.

Proof. First, consider the following nondeterministic polynomial time machine M. The machine M first guesses a set of m algebraic integers in their compact representation and then verifies in polynomial time that the guessed algebraic integers indeed form a fundamental system of units by first checking whether they are indeed units (check if $\alpha\mathcal{O} = \mathcal{O}$) and then calculating the volume of the parallelepiped (in the Log map) formed by the vectors corresponding to the guessed units, by a determinant computation. If this volume does not lie between B and $2B$, the machine M rejects on this computation path. Otherwise, applying Theorem 4.5 along with the guessed fundamental system of units as input, the machine M now computes a canonical fundamental system of units and checks if it coincides with the guessed fundamental system of units. If they do coincide the machine M accepts along this computation path. It is clear from the above description that the nondeterministic machine M has a unique accepting path. Applying Lemma 1.3, we can now design from M an $\mathrm{FP}^{\mathrm{SPP}}$ algorithm that will compute a fundamental system of units, if it is additionally given B such that $B \leq R_K < 2B$. □

Now, combining Lemmas 5.1 and 5.2 we immediately obtain the following.

Theorem 5.3. *There is a* $\mathrm{FP}^{\mathrm{SPP}}$ *algorithm to compute a fundamental system of units of the ring of integers of a number field K assuming that the degree $[K : \mathbb{Q}]$ is a constant.*

6 Principle Ideal Testing is in SPP

Given a number field K and a $\mathbb{Z}$-basis for its ring of integer $\mathcal{O}$, the *principal ideal testing* problem (denoted by PrI) problem is to check if an $\mathfrak{a}$ of $\mathcal{O}$ is a principal ideal. We show that this problem is in SPP.

Theorem 6.1. *Given a number field K with ring of integers $\mathcal{O}$ and the $\mathbb{Z}$ basis of a ideal $\mathfrak{a}$, checking whether $\mathfrak{a}$ is principal is in* SPP*, assuming $[K : \mathbb{Q}]$ is a constant.*

Proof. Without loss of generality we can assume that $\mathfrak{a}$ is an integral ideal. First compute a fundamental system of units $\{\varepsilon_i\}_{i=1}^m$ of $\mathcal{O}$ using the algorithm in Theorem 5.3. Guess the compact representation of an algebraic integer α. Check if $\alpha O = \mathfrak{a}$ if not reject. Check if $\mathrm{Log}(\alpha)$ lies in the fundamental parallelepiped $\{\mathbf{x} \in \mathbb{R}^m : \mathbf{x} = \sum \alpha_i \mathrm{Log}(\varepsilon_i), \alpha_i \in [0,1]\}$. If not reject. Next, apply Theorem 4.3 to obtain a compact representation of α' from $\mathrm{Log}(\alpha)$ and $\mathfrak{a}$, such that $\mathrm{Log}(\alpha) = \mathrm{Log}(\alpha')$. If the compact representations of α and α' coincide we accept on that computation path and reject otherwise. The correctness of the easily follows from Theorem 4.3 and the fact that for every $\alpha \in \mathcal{O}$ there is a unique associate in the fundamental parallelepiped. □

6.1 *Computing the Class Number*

Finally, we show that if we assume the generalized Riemann hypothesis (GRH) then finding the class number and a presentation for the class group are in $\mathrm{FP}^{\mathrm{SPP}}$. It is shown by Bach [2] that if the GRH is true then the class group of any number field K is generated by the ideal classes of all non-inert prime ideals of norm less $L = 12\ln^2|D|$. Let $\mathfrak{p}_1, \ldots, \mathfrak{p}_N$ be the (polynomially many) ideal classes of all non-inert prime ideals of norm less $L = 12\ln^2|D|$. We can compute these ideals $\mathfrak{p}_i$ in polynomial time as explained, for example in [8, Section 6.2.5] or [5].

Our goal is to compute the class number and a generator-relator presentation of the class group of K. Let G_i denote the subgroup of $Cl(K)$ generated by $\{\mathfrak{p}_1, \ldots, \mathfrak{p}_i\}$ and let $G_0 = \{id\}$. For each i let t_i be the least positive integer such that $\mathfrak{p}_i^{t_i} \in G_{i-1}$. Let s_{ij}, $1 \le i \le N$ and $1 \le j < i$ be integers such that $0 \le s_{ij} < t_j$ and such that $\mathfrak{p}_i^{t_i} \sim \prod_j^{i-1} \mathfrak{p}_j^{s_{ij}}$. The set of relators defined by

$$R = \left\{ \mathfrak{p}_i^{t_i} = \prod_{j=1}^{i-1} \mathfrak{p}_j^{s_{ij}} : 1 \le i \le N \right\}$$

together with the generator set $\{\mathfrak{p}_1, \ldots, \mathfrak{p}_N\}$ gives a generator-relator presentation of $Cl(K)$. Furthermore, notice that the s_{ij}'s are unique in the range $0 \le s_{ij} < t_j$. Also, t_1 is the order of the ideal class of $\mathfrak{p}_1$ in $Cl(K)$. We will describe an $\mathrm{FP}^{\mathrm{SPP}}$ procedure for computing the set R inductively as follows.

Assume that the set of relators

$$R_i = \left\{ \mathfrak{p}_j^{t_j} = \prod_{k=1}^{j-1} \mathfrak{p}_k^{s_{jk}} : 1 \le j \le i \right\}$$

is already computed (where $R_0 = \emptyset$). It suffices to give an $\mathrm{FP}^{\mathrm{SPP}}$ procedure for computing R_{i+1}. To this end, we define an $\mathrm{NP}^{\mathrm{SPP}}$ language A as follows: A consists of the set of tuples $\langle x, y, \{\mathfrak{a}_j\}_{j=1}^i, \{(m_j, n_j)\}_{j=1}^{i-1} \rangle$ such that there is a $x \le t < y$ and $m_j \le s_j < n_j$ such that $\mathfrak{a}_i^t \sim \prod_{j=1}^{i-1} \mathfrak{a}_j^{s_j}$, where $\mathfrak{a}_j$'s are ideals in $\mathcal{O}_K$ given by their HNFs. It is easy to see that the language A is in $\mathrm{NP}^{\mathrm{SPP}}$:

guess t and the s_j's and verify the class group identity by applying the SPP algorithm for principal ideal testing in Theorem 6.1.

The following code is a polynomial-time oracle computation (with oracle A) that computes R_{i+1} from R_i.

Let $T := 1$;
while true do
  if $\langle T, 2T, \{\mathfrak{p}_j\}_{j=1}^{i+1}, \{(1, t_j)\}_{j=1}^{i}\rangle \in A$ **then** break **else** $T := 2T$
end
Do a binary search for t_{i+1} in the range $[T, 2T)$ using A as oracle.;
Next, do a binary search to compute the s_j's. (* These s_j's are actually the s_{i+1j} in the definition of R. *)

It is easy to see that in the above algorithm only UP-like queries are made to A. More precisely, the queries will be of the form $\langle x, y, \{\mathfrak{p}_j\}_{j=1}^{i}, \{(m_j, n_j)\}_{j=1}^{i-1}\rangle$, with parameters such that the $\mathrm{NP}^{\mathrm{SPP}}$ machine for A will have at most one accepting path. The queries are UP-like as R_i is a set of relators for G_i. Now, using closure properties of SPP and Lemma 1.3 we can transform the above algorithm to an $\mathrm{FP}^{\mathrm{SPP}}$ procedure that inductively computes the generator-relator presentation of the class group $Cl(K)$. Observe that the class number is given by $\prod_{i=1}^{N} t_i$. Hence we have the following theorem.

Theorem 6.2. *Assuming the GRH, the class number and a generator-relator presentation for the class group of a constant-degree number field can be computed in* $\mathrm{FP}^{\mathrm{SPP}}$.

Acknowledgment. We thank the referees for their useful comments.

References

[1] V. Arvind and P. P. Kurur. Graph Isomorphism is in SPP. In 43rd *Annual Symposium of Foundations of Computer Science*, pages 743–750. IEEE, November 2002.

[2] E. Bach. Explicit bounds for primality testing and related problems. *Mathematics of Computation*, 55:355–380, 1990.

[3] J. Buchmann. On computation of units and class number by a generalization of Lagrange's algorithm. *Journal of Number Theory*, 26:8–30, 1987.

[4] J. Buchmann. On the period length of the generalized Lagrange algorithm. *Journal of Number Theory*, 26:31–37, 1987.

[5] J. Buchmann and H. W. Lenstra Jr. Computing maximal orders and decomposing primes in number fields. preprint.

[6] J. Buchmann and H. C. Williams. On the infrastructure of the principal ideal class of an algebraic number field of unit rank one. *Mathematics of Computation*, 50:569–579, 1988.

[7] J. Buchmann and H. C. Williams. On the existence of a short proof for the value of the class number and regulator of a real quadratic field. *Number Theory and Applications*, 265:327–345, 1989.

[8] H. Cohen. *A Course in Computational Algebraic Number Theory*. Springer-Verlag, Berlin, 1993.

[9] S. Fenner, L. Fortnow, S. A. Kurtz, and L. Li. An oracle builder's toolkit. In *SCT: Annual Conference on Structure in Complexity Theory*, pages 120–131, 1993.

[10] S. A. Fenner, L. Fortnow, and S. A. Kurtz. Gap-definable counting classes. In *Structure in Complexity Theory Conference*, pages 30–42, 1991.

[11] L. Fortnow and J. D. Rogers. Complexity limitations on quantum computation. In *IEEE Conference on Computational Complexity*, pages 202–209, 1998.

[12] S. Hallgren. Polynomial-time quantum algorithms for pell's equation and the principal ideal problem. In *Proceedings of the* 34^{th} *ACM Symposium on Theory of Computing*, pages 653–658. IEEE, 2002.

[13] H. W. Lenstra Jr. Integer programming with a fixed number of variables. *Mathematics of Operations Research*, 8:538–548, 1983.

[14] J. Kobler, U. Schoning, S. Toda, and J. Toran. Turing machines with few accepting computations and low sets for PP. *Journal of Computer and System Sciences*, 44(2):272–286, 1992.

[15] J. Köbler, U. Schöning, and J. Torán. *The Graph Isomorphism Problem: Its Structural Complexity*. Birkhauser, 1993.

[16] K. S. McCurley. Cryptographic key distribution and computation in class groups. In *NATO Advanced Science Institute Series C*, volume 256, pages 459–479. Kluwer, Dordrecht, 1989.

[17] C. Thiel. Under the assumption of the Generalized Riemann Hyothesis verifying the class number belongs to NP $\cap$ co-NP. In *Algorithmic Number Theory, First International Symposium, ANTS-I*, volume 877 of *Lecture Notes in Computer Science*, pages 234–247. Springer, 1994.