УДК 511.23 DOI 10.17223/20710410/58/3

ON IDEAL CLASS GROUP COMPUTATION OF IMAGINARY MULTIQUADRATIC FIELDS[1]

S. A. Novoselov

Immanuel Kant Baltic Federal University, Kaliningrad, Russia

E-mail: snovoselov@kantiana.ru

In the paper, we extend Biasse — van Vredendaal (OBS, 2019, vol. 2) implementation and experiments of the class group computation from real to imaginary multiquadratic fields. The implementation is optimized by introducing an explicit prime ideal lift operation and by using LLL reduction instead of HNF computation. We provide examples of class group computation of the imaginary multiquadratic fields of degree 64 and 128, that has been previously unreachable.

Keywords: *multiquadratic field, ideal class group.*

О ВЫЧИСЛЕНИИ ГРУППЫ КЛАССОВ ИДЕАЛОВ МНИМЫХ МУЛЬТИКВАДРАТИЧНЫХ ПОЛЕЙ

С. А. Новоселов

БФУ им. И. Канта, г. Калининград, Россия

Расширены эксперименты Биассе — ван Вредендал (OBS, 2019, vol. 2) по вычислению группы классов идеалов с действительных мультиквадратичных полей на мнимые мультиквадратичные поля. Представлена адаптированная и оптимизированная для работы с мнимыми полями реализация алгоритма Биассе — ван Вредендал. Оптимизации включают в себя введение и использование явных формул для подъёма простых идеалов и замену вычислений эрмитовой нормальной формы на LLL-редукцию. Представлены примеры вычисления группы классов для мнимых мультквадратичных полей степени 64 и 128, недостижимые ранее.

Ключевые слова: *мультквадратичное числовое поле, группа классов идеалов.*

1. Introduction

A multiquadratic field of degree 2^n is defined as

$$K = \mathbb{Q}(\sqrt{d_1}, \ldots, \sqrt{d_n}),$$

where $d_1, \ldots, d_n$ are square-free integers. The field K is called real if all $d_1, \ldots, d_n$ are positive, otherwise it is called imaginary. The class group Cl_K of the multiquadratic field is a factor group of fractional ideals of K modulo principal ideals. The class group computation of a number field is a classical hard problem in algebraic number theory. It is an essential tool in cryptanalysis of the cryptosystems based on arithmetic operations in number fields [1–3] and ideal lattices [4, 5]. Security of the first ones relies on hardness of the short principal

[1]The research was funded by the Russian Science Foundation (project No. 22-41-04411, `https://rscf.ru/en/project/22-41-04411/`).

ideal problem (SPIP) and the discrete logarithm problem. For real multiquadratic fields, the solution to SPIP is provided by the paper [6]. The discrete logarithm problem in number fields can be solved by factoring target ideal into product of prime ideals, adding these obtained primes to the factor base, and running the class group computation algorithm. Cryptosystems based on ideal lattices rely on the hardness of approximate shortest vector problem. The class group computation is also used in [7, 8] as part of the algorithm for solving this problem.

The common way to compute the ideal class group of a number field is to use algorithms [9–11]. These algorithms are subexponential in $\log \Delta_K$, where Δ_K is the discriminant of the field. Biasse and van Vredendaal proposed [12] an algorithm that reduces the problem of the ideal class group computation of a multiquadratic field K to the quadratic subfields of K. Therefore, the time complexity was reduced to subexponential time in $\log D$, where $D = d_1 \cdot \ldots \cdot d_n$ is the largest discriminant of the quadratic subfield. In practice, this allowed Biasse and van Vredendaal to compute class groups of real multiquadratic fields of degrees up to 128. However, provided implementation does not support imaginary multiquadratic fields. In addition, the latter case requires optimizations since the class groups become much larger.

Our contributions:

1) We extend Biasse — van Vredendaal [12] implementation for the class group computation of real multiquadratic fields to imaginary multiquadratic fields[2].
2) For both imaginary and real multiquadratic fields, we provide an optimized algorithm for computing the splitting trees of prime ideals over subfields of the field K. These trees are used in the algorithm to lift solution from subfields to the base field.
3) To reduce memory consumption when computing large class groups, we have replaced Hermite normal form computations with LLL-reduction [13].
4) Using results above, we computed the ideal class group for previously unreachable imaginary fields of degrees 64 and 128. For degree 256, partial information is computed — the result is correct up to factor of size $\leqslant 32208$.

Organization of the paper. In section 2 we give necessary notations. Section 3.1 briefly describes a general approach from [9, 10] for the class group computation and its improvement by Biasse and van Vredendaal [12]. In 3.3 we describe the choice of norm bounds for prime ideals in the factor base. Section 3.4 describes the verification procedure for the result of the class group computation. Sections 4 and 5 contain main results — improvements for the Biasse — van Vredendaal implementation and experiments on the class group computation.

2. Notation

We use the following notations:

— $K = \mathbb{Q}(\sqrt{d_1}, \ldots, \sqrt{d_n})$ is a n-quadratic field where all d_i are pairwise coprime and square-free, $\mathcal{O}_K$ is its ring of integers;
— $\mathrm{Cl}_K = \mathrm{I}_K \,/\, \mathrm{Princ}_K$ — the class group of K, i.e., the factor group of fractional ideals I_K of K modulo principal ideals of $\mathcal{O}_K$;
— h is the order of Cl_K;
— $G_K = \mathrm{Gal}(K/\mathbb{Q})$ is the absolute Galois group of K;
— K_σ is a subfield, that is invariant under the action of an automorphism $\sigma \in G_K$;
— Δ_K is the discriminant of K;

[2]Available here: `https://github.com/novoselov-sa/multiclass-im`.

— $N(I)$, $N(a)$ is the absolute norm of an ideal I or $a \in K$, respectively;
— $N_{L/K}(\alpha)$ is the relative norm of $\alpha \in L$;
— $\mathcal{C}_m$ is a cyclic group of order m.

3. Preliminaries

3.1. General algorithm for class group computation

Assume that we have a factor base $S = \{\mathfrak{p}_1, \ldots, \mathfrak{p}_d\}$ — a set of all prime ideals of $\mathcal{O}_K$ that generate the class group Cl_K. To build this factor base, we can take all prime ideals of norm $\leqslant 12 \log^2 \Delta_K$, see [14]. In practice, this general bound is very pessimistic and a better heuristic bound can be computed instead using Grenié — Molteni algorithm [15].

The computation of the ideal class group of a number field K can be done by collecting enough relations $(\alpha, e) \in K \times \mathbb{Z}^d$ of the form

$$\alpha \mathcal{O}_K = \prod_{i=1}^{d} \mathfrak{p}_i^{e_i}.$$

One relation is computed by taking a random element $\alpha \in K$ that splits over the factor base S. After collecting enough relations, we form a matrix of relations A, the rows of which are vectors e. The Smith Normal Form $A = UBV$ of this matrix gives us the group structure and the generators of the class group. In particular, we have $B = \mathrm{diag}(b_1, \ldots, b_k)$ and the ideal class group is the following product of cyclic groups:

$$\mathrm{Cl}_K \simeq \mathcal{C}_{b_1} \times \ldots \times \mathcal{C}_{b_k} \simeq \langle \mathfrak{g}_1 \rangle \times \ldots \times \langle \mathfrak{g}_k \rangle ,$$

where $\mathfrak{g}_i = \prod\limits_{j=1}^{d} \mathfrak{p}_j^{v'_{i,j}}$ for $V^{-1} = (v'_{i,j})$. In addition, we have $\mathfrak{p}_i = \prod\limits_{j=1}^{k} \mathfrak{g}_j^{v_{i,j}}$. It is well known that the whole process of computing the ideal class group takes subexponential time [9] in $\log \Delta_K$.

3.2. Algorithm of Biasse and van Vredendaal

For an arbitrary number field, the complexity of the class group computation is subexponential in $\log \Delta_K$. However, as it was shown by Biasse and van Vredendaal [12], for multiquadratic fields the problem of the class group computation can be efficiently reduced to finding relations in certain quadratic subfields of K, which is a much simpler task, and then lifting them to K. It results in an algorithm of complexity subexponential in $\log D$, where $D = d_1 \cdot \ldots \cdot d_n$ is the largest discriminant of quadratic subfields, which is always smaller than Δ_K.

3.3. Choice of norm bound

Choosing the correct bound for the norms of the prime ideals that constitute the factor base is also a non-trivial task. Taking all primes satisfying Bach's bound [14] or the bound computed using the Grenié — Molteni script [15] appears to be impractical due to high memory consumption (the factor base turns out to be too large for both real and imaginary cases). Instead, in our experiments, we used the following idea. Since Biasse — van Vredendaal algorithm lifts the relations from quadratic subfields of K back to the base field K, choosing primes with norms greater than the corresponding bounds for these quadratic subfields does not give us new relations to be lifted. So we first compute the bounds for all quadratic subfields with Grenié — Molteni script [15]. We take the maximum of these bounds plus a small constant suggested by the experiments (these are of orders 200 for $n \leqslant 5$).

3.4. Verification

To verify class group computation, we checked the obtained class number following the approach from [10, Step D]. Assuming Generalized Riemann Hypothesis (GRH), one can verify the result of class group computation for the field K in polynomial time (in $\log \Delta_K$ and $\deg K$) by computing the product hR_K with enough precision, where R_K is a regulator [16, Def. 4.9.8] of K. This product can be computed using the following class number formula [16, Th. 4.9.12]:

$$\lim_{s\to 1}(s-1)\zeta_K(s) = 2^{r_1}(2\pi)^{r_2}\frac{hR_K}{w_K\sqrt{|\Delta_K|}}.$$

Here $\zeta_K(s) = \prod_{\mathfrak{p}} \frac{1}{1-(N(\mathfrak{p}))^{-s}}$ is the Dedekind zeta function of K, w_K is a number of roots of unity, r_1 is a number of real embeddings of K, and r_2 is such that $2r_2$ is a number of complex embeddings of K. The left side of this formula (Euler product) can be computed in polynomial time (assuming GRH) following [17, 18]. The other parts of the class number formula (with exception of h and R_K) can be efficiently computed for multiquadratic fields. This gives us the (approximated) product εhR_K. The term ε represents error in the computation of Euler product.

To verify the class group computation, we compare the computed product εhR_K with the tentative product h^*R^*, where h^* is the tentative class number — result of our class number computation, and R^* is the approximation of regulator computed using [6]. If $|\varepsilon| < \sqrt{2}$, then it remains to check that $h^*R^*_K < \sqrt{2}\varepsilon hR_K$, since $h^*R^*_K$ is an integral multiple of hR. In our experiments we computed Euler products using the method zeta_log_residue from Hecke package [19] for Julia [20]. This method implements the approximation of Euler products from [18].

4. Optimizations and improvements for the Biasse — van Vredendaal algorithm

4.1. The case of imaginary multiquadratic fields

The algorithm of Biasse — van Vredendaal and its implementation [12] is given for real multiquadratic fields. However, as claimed in the paper [12, p. 104], the algorithm works for any fields, where we can choose two different automorphisms of order 2. Indeed, after cumbersome, but rather technical fixes in the Biasse — van Vredendaal implementation, we were able to run the class group computation algorithm for imaginary multiquadratic fields. To be more precise, in our implementation we take into account a different rank of the unit group as in imaginary multiquadratic field it is half of a real multiquadratic field of the same degree. Additionally, we augment the implementation with a procedure that generates trees describing the splitting of a prime ideal over the subfields — a missing piece of the implementation from [12]. This is the subject of the next section. After running algorithm for imaginary fields, we were unable to compute the class groups for high degree fields due to high memory consumption. This was caused by HNF computations. So we replaced HNF computation with LLL [13] reduction.

4.2. Generation of splitting trees

The main drawback of the Biasse — van Vredendaal implementation [12] is the lack of code that computes trees, describing splitting of prime ideals over subfields. These trees contain for each prime ideal $\mathfrak{P}$ of a subfield K_γ of the field K the exponent vector $\mathbf{x} = (x_1, \ldots, x_d)$ such that $\mathfrak{P}\mathcal{O}_K = \prod_{i=1}^{d} \mathfrak{p}_i^{x_i}$. This information allows us efficiently lift prime ideals and thus prime ideal products from K_γ to K. The prime ideals in these trees are also used as a factor base for computations in the subfields. Our implementation produces the splitting picture for prime ideals using a certain representation of these ideals, as well as the operation for lifting ideal from the subfields to the base field. We give the details in the next section.

4.3. Representation of prime ideals

Let $\theta = \sqrt{d_1} + \ldots + \sqrt{d_n}$ and $f = \prod\limits_{\gamma \in G_K} (x - \gamma(\theta))$. Let $K = \mathbb{Q}(\sqrt{d_1}, \ldots, \sqrt{d_n}) = \mathbb{Q}(x)/f\mathbb{Q}(x)$. If we consider only primes p such that $p \nmid [\mathcal{O}_K : \mathbb{Z}[\theta]]$, then every prime ideal $\mathfrak{p} \mid p\mathcal{O}_K$ may be written in one of the two following forms:

1) $(p, \theta + \overline{\gamma(\theta)})$ for some $\gamma \in G_K$ if all $d_1, \ldots, d_n$ are quadratic residues modulo p;
2) $(p, (\theta - \overline{\gamma(\theta')})^2 - \overline{\gamma(\theta'')^2})$ for $\gamma \in G_K$, $\theta' = \sum\limits_{\sqrt{d_i} \in \mathbb{F}_p} \sqrt{d_i}$, and $\theta'' = \sum\limits_{\sqrt{d_i} \notin \mathbb{F}_p} \sqrt{d_i}$.

Here $\overline{z}$ denotes the reduction of z modulo p. This representation of primes is unique and there is one-to-one correspondence between primes and automorphisms G_K for the case 1 and for the automorphisms in $G_K/\pm 1$ for the case 2. The representation follows from the standard prime decomposition theorem [16, Th. 4.8.13] and from the fact that the multiquadratic fields are Galois extensions of $\mathbb{Q}$.

4.4. Fast ideal above operation

For $K = \mathbb{Q}(\sqrt{d_1}, \ldots, \sqrt{d_n})$, the class group computation algorithm of Biasse — van Vredendaal [12] uses only three subfields $K_\sigma = \mathbb{Q}(\sqrt{d_1}, \ldots, \sqrt{d_{n-1}})$, $K_\tau = \mathbb{Q}(\sqrt{d_1}, \ldots, \sqrt{d_{n-2}}, \sqrt{d_n})$, and $K_{\sigma\tau} = \mathbb{Q}(\sqrt{d_1}, \ldots, \sqrt{d_{n-2}}, \sqrt{d_{n-1}}\sqrt{d_n})$ fixed by the automorphisms:

1) $\sigma : \sqrt{d_n} \mapsto -\sqrt{d_n}$;
2) $\tau : \sqrt{d_{n-1}} \mapsto -\sqrt{d_{n-1}}$;
3) $\sigma\tau : \sqrt{d_n} + \sqrt{d_{n-1}} \mapsto -\sqrt{d_n} - \sqrt{d_{n-1}}$.

For fast generation of the trees describing splitting of prime ideals over the subfields, we used an explicit ideal above (Lift) operation for prime ideals presented in Table 1. It is defined for prime ideals that are given in the form as described in Section 4.3. For the field $K_{\tau\sigma}$ we omitted the cases $\sqrt{d_n} \in \mathbb{F}_p, \sqrt{d_{n-1}} \notin \mathbb{F}_p$ or $\sqrt{d_n} \notin \mathbb{F}_p, \sqrt{d_{n-1}} \in \mathbb{F}_p$. They can be given in analogous way and differ from the presented case by signs only. Table 1 was derived using the fact that for an unramified ideal $\mathfrak{p}$ of $\mathcal{O}_{K_\gamma}$ and the ideal $\mathfrak{P} = (p, \beta)$ of $\mathcal{O}_K$ such that $\mathfrak{P} \mid \mathfrak{p}\mathcal{O}_K$ we have [21, Alg. 2.5.3] that $\mathfrak{p} = (p, \mathrm{N}_{K/K_\gamma}(\beta))$. Notations in the Table 1: $\mathfrak{p} \mid (p)$, $\mathfrak{p} = (p, \alpha)$ is a prime ideal of $\mathcal{O}_{K_\gamma}$ for some prime number p; $\mathfrak{p}\mathcal{O}_K = \mathfrak{P}_1\mathfrak{P}_2$, in the case $\mathfrak{P}_2 = (1)$ we omit $\mathfrak{P}_2$; $\mathfrak{P}_1 = (p, \beta_1)$, $\mathfrak{P}_2 = (p, \beta_2)$.

The described Lift operation allows us to do the fast generation of the splitting trees for primes $\mathfrak{p} \mid (p)$, where $p \nmid [\mathcal{O}_K : \mathbb{Z}[\theta]]$ and $p \nmid [\mathcal{O}_{K_\gamma} : \mathbb{Z}[\theta_\gamma]]$ for each subfield K_γ of K.

Table 1

Explicit lift of prime ideals from subfields K_γ of K

Subfield	α	Conditions	β_1, β_2
K_σ	$\theta_\sigma + \overline{\mu(\theta_\sigma)}$	$\sqrt{d}_n \in \mathbb{F}_p$	$\theta + \overline{\mu(\theta_\sigma)} + \overline{\sqrt{d_n}}$, $\theta + \overline{\mu(\theta_\sigma)} - \overline{\sqrt{d_n}}$
K_σ	$\theta_\sigma + \overline{\mu(\theta_\sigma)}$	$\sqrt{d}_n \notin \mathbb{F}_p$	$(\theta + \overline{\mu(\theta_\sigma)})^2 - \overline{d_n}$
K_σ	$(\theta_\sigma - \overline{\mu(\theta'_\sigma)})^2 - \overline{\mu(\theta''_\sigma)^2}$	$\sqrt{d}_n \in \mathbb{F}_p$	$(\theta_\sigma - \overline{\mu(\theta'_\sigma) + \sqrt{d_n}})^2 - \overline{\mu(\theta''_\sigma)^2}$, $(\theta_\sigma - \overline{\mu(\theta'_\sigma) - \sqrt{d_n}})^2 - \overline{\mu(\theta''_\sigma)^2}$
K_σ	$(\theta_\sigma - \overline{\mu(\theta'_\sigma)})^2 - \overline{\mu(\theta''_\sigma)^2}$	$\sqrt{d}_n \notin \mathbb{F}_p$	$(\theta_\sigma - \overline{\mu(\theta'_\sigma)})^2 - \overline{(\mu(\theta''_\sigma) + \sqrt{d_n})^2}$, $(\theta_\sigma - \overline{\mu(\theta'_\sigma)})^2 - \overline{(\mu(\theta''_\sigma) - \sqrt{d_n})^2}$
K_τ	$\theta_\tau + \overline{\mu(\theta_\tau)}$	$\sqrt{d_{n-1}} \in \mathbb{F}_p$	$\theta + \overline{\mu(\theta_\tau)} + \overline{\sqrt{d_{n-1}}}$, $\theta + \overline{\mu(\theta_\tau)} - \overline{\sqrt{d_{n-1}}}$
K_τ	$\theta_\tau + \overline{\mu(\theta_\tau)}$	$\sqrt{d_{n-1}} \notin \mathbb{F}_p$	$(\theta + \overline{\mu(\theta_\tau)})^2 - \overline{d_{n-1}}$
K_τ	$(\theta_\tau - \overline{\mu(\theta'_\tau)})^2 - \overline{\mu(\theta''_\tau)^2}$	$\sqrt{d_{n-1}} \in \mathbb{F}_p$	$(\theta_\sigma - \overline{\mu(\theta'_\tau) + \sqrt{d_{n-1}}})^2 - \overline{\mu(\theta''_\tau)^2}$, $(\theta_\tau - \overline{\mu(\theta'_\tau) - \sqrt{d_{n-1}}})^2 - \overline{\mu(\theta''_\tau)^2}$
K_τ	$(\theta_\tau - \overline{\mu(\theta'_\tau)})^2 - \overline{\mu(\theta''_\tau)^2}$	$\sqrt{d_{n-1}} \notin \mathbb{F}_p$	$(\theta_\tau - \overline{\mu(\theta'_\tau)})^2 - \overline{(\mu(\theta''_\tau) + \sqrt{d_{n-1}})^2}$, $(\theta_\tau - \overline{\mu(\theta'_\tau)})^2 - \overline{(\mu(\theta''_\tau) - \sqrt{d_{n-1}})^2}$
$K_{\sigma\tau}$	$\theta_{\sigma\tau} + \overline{\mu(\theta_{\sigma\tau})}$	$\sqrt{d}_n, \sqrt{d_{n-1}} \in \mathbb{F}_p$, $\mu(\sqrt{d_{n-1}d_n}) = \sqrt{d_{n-1}d_n}$	$\theta + \overline{\mu(\theta_{\sigma\tau})} - \overline{\mu(\sqrt{d_{n-1}d_n})} + \overline{\sqrt{d_{n-1}}} - \overline{\sqrt{d_n}}$ $\theta + \overline{\mu(\theta_{\sigma\tau})} - \overline{\mu(\sqrt{d_{n-1}d_n})} - \overline{\sqrt{d_{n-1}}} + \overline{\sqrt{d_n}}$
$K_{\sigma\tau}$	$\theta_{\sigma\tau} + \overline{\mu(\theta_{\sigma\tau})}$	$\sqrt{d}_n, \sqrt{d_{n-1}} \in \mathbb{F}_p$, $\mu(\sqrt{d_{n-1}d_n}) = -\sqrt{d_{n-1}d_n}$	$\theta + \overline{\mu(\theta_{\sigma\tau})} - \overline{\mu(\sqrt{d_{n-1}d_n})} + \overline{\sqrt{d_{n-1}}} + \overline{\sqrt{d_n}}$ $\theta + \overline{\mu(\theta_{\sigma\tau})} - \overline{\mu(\sqrt{d_{n-1}d_n})} - \overline{\sqrt{d_{n-1}}} - \overline{\sqrt{d_n}}$
$K_{\sigma\tau}$	$\theta_{\sigma\tau} + \overline{\mu(\theta_{\sigma\tau})}$	$\sqrt{d}_n, \sqrt{d}_{n-1} \notin \mathbb{F}_p$, $\mu(\sqrt{d_{n-1}d_n}) = \sqrt{d_{n-1}d_n}$	$(\theta + \overline{\mu(\theta_{\sigma\tau}) - \sqrt{d_{n-1}d_n}})^2 - \overline{(d_{n-1} + d_n)^2}$
$K_{\sigma\tau}$	$\theta_{\sigma\tau} + \overline{\mu(\theta_{\sigma\tau})}$	$\sqrt{d}_{n-1}, \sqrt{d}_n \notin \mathbb{F}_p$, $\mu(\sqrt{d_{n-1}d_n}) = -\sqrt{d_{n-1}d_n}$	$(\theta + \overline{\mu(\theta_{\sigma\tau}) + \sqrt{d_{n-1}d_n}})^2 - \overline{(d_{n-1} + d_n)^2}$
$K_{\sigma\tau}$	$(\theta_{\sigma\tau} - \overline{\mu(\theta'_{\sigma\tau})})^2 - \overline{\mu(\theta''_{\sigma\tau})^2}$	$\sqrt{d}_n, \sqrt{d_{n-1}} \in \mathbb{F}_p$ $\mu(\sqrt{d_{n-1}d_n}) = \sqrt{d_{n-1}d_n}$	$(\theta - \overline{\mu(\theta'_{\sigma\tau}) - \sqrt{d_{n-1}d_n} + \sqrt{d}_n + \sqrt{d}_{n-1}})^2 - \overline{\mu(\theta''_{\sigma\tau})^2}$ $(\theta - \overline{\mu(\theta'_{\sigma\tau}) - \sqrt{d_{n-1}d_n} - \sqrt{d}_n - \sqrt{d}_{n-1}})^2 - \overline{\mu(\theta''_{\sigma\tau})^2}$
$K_{\sigma\tau}$	$(\theta_{\sigma\tau} - \overline{\mu(\theta'_{\sigma\tau})})^2 - \overline{\mu(\theta''_{\sigma\tau})^2}$	$\sqrt{d}_n, \sqrt{d_{n-1}} \in \mathbb{F}_p$ $\mu(\sqrt{d_{n-1}d_n}) = -\sqrt{d_{n-1}d_n}$	$(\theta - \overline{\mu(\theta'_{\sigma\tau}) + \sqrt{d_{n-1}d_n} + \sqrt{d}_n - \sqrt{d}_{n-1}})^2 - \overline{\mu(\theta''_{\sigma\tau})^2}$ $(\theta - \overline{\mu(\theta'_{\sigma\tau}) + \sqrt{d_{n-1}d_n} - \sqrt{d}_n + \sqrt{d}_{n-1}})^2 - \overline{\mu(\theta''_{\sigma\tau})^2}$
$K_{\sigma\tau}$	$(\theta_{\sigma\tau} - \overline{\mu(\theta_{\sigma\tau})})^2 - \overline{\mu(\theta''_{\sigma\tau})^2}$	$\sqrt{d}_n, \sqrt{d_{n-1}} \notin \mathbb{F}_p$ $\mu(\sqrt{d_{n-1}d_n}) = \sqrt{d_{n-1}d_n}$	$(\theta - \overline{\mu(\theta'_{\sigma\tau}) - \sqrt{d_{n-1}d_n}})^2 - \overline{(\mu(\theta''_{\sigma\tau}) + \sqrt{d}_n + \sqrt{d}_{n-1})^2}$ $(\theta - \overline{\mu(\theta'_{\sigma\tau}) - \sqrt{d_{n-1}d_n}})^2 - \overline{(\mu(\theta''_{\sigma\tau}) - \sqrt{d}_n - \sqrt{d_{n-1}})^2}$
$K_{\sigma\tau}$	$(\theta_{\sigma\tau} - \overline{\mu(\theta_{\sigma\tau})})^2 - \overline{\mu(\theta''_{\sigma\tau})^2}$	$\sqrt{d}_n, \sqrt{d_{n-1}} \notin \mathbb{F}_p$ $\mu(\sqrt{d_{n-1}d_n}) = -\sqrt{d_{n-1}d_n}$	$(\theta - \overline{\mu(\theta'_{\sigma\tau}) + \sqrt{d_{n-1}d_n}})^2 - \overline{(\mu(\theta''_{\sigma\tau}) + \sqrt{d}_n - \sqrt{d}_{n-1})^2}$ $(\theta - \overline{\mu(\theta'_{\sigma\tau}) + \sqrt{d_{n-1}d_n}})^2 - \overline{(\mu(\theta''_{\sigma\tau}) - \sqrt{d_n} + \sqrt{d_{n-1}})^2}$

5. Experiments

For experiments, we used implementation of [12], which we adapted and optimized for imaginary multiquadratic fields as described above. In our experiments we used the fields of the form

$$K = \mathbb{Q}(\sqrt{-3}, \sqrt{-7}, \sqrt{-11}, \sqrt{-19}, \sqrt{-23}, \ldots),$$

where $d_1, \ldots, d_n$ are primes such that $d_i \equiv 1 \pmod 4$ for all i. Results of computations are presented in Table 2. Computations were done on one core of Xeon Silver 4210R processor clocked at 2.4 GHz on computer with 629 GB RAM. The experiments were run on Sage 9.4 [22].

Table 2

Class group computation experiments

$[K:\mathbb{Q}]$	Multiclass-im	Sage	Cl_K
8	25	0.188	$\mathcal{C}_3$
16	175	0.378	$\mathcal{C}_8 \times \mathcal{C}_{48}$
32	$2.67 \cdot 10^3$	30	$\mathcal{C}_2 \times \mathcal{C}_4^3 \times \mathcal{C}_{24} \times \mathcal{C}_{48}^2 \times \mathcal{C}_{3360}$
64	$4.484 \cdot 10^4$	$> 5.898 \cdot 10^4$	$\mathcal{C}_2^2 \times \mathcal{C}_4^9 \times \mathcal{C}_8^3 \times \mathcal{C}_{16} \times \mathcal{C}_{48} \times \mathcal{C}_{96}^2 \times \mathcal{C}_{192}^2 \times$ $\times \mathcal{C}_{6720}^2 \times \mathcal{C}_{927360}$
128	$6.279 \cdot 10^4$		$\mathcal{C}_2^6 \times \mathcal{C}_4^{11} \times \mathcal{C}_8^{15} \times \mathcal{C}_{16}^9 \times \mathcal{C}_{32} \times \mathcal{C}_{96}^4 \times \mathcal{C}_{192}^2 \times$ $\times \mathcal{C}_{960} \times \mathcal{C}_{1920} \times \mathcal{C}_{13440}^5 \times \mathcal{C}_{443520} \times$ $\times \mathcal{C}_{20401920} \times \mathcal{C}_{554955114954240}$
256	$3.428 \cdot 10^5$		$\mathcal{C}_2^{15} \times \mathcal{C}_4^{33} \times \mathcal{C}_8^{19} \times \mathcal{C}_{16}^{15} \times \mathcal{C}_{48}^{10} \times \mathcal{C}_{96}^{19} \times \mathcal{C}_{480} \times$ $\times \mathcal{C}_{960}^7 \times \mathcal{C}_{6720} \times \mathcal{C}_{13440}^4 \times \mathcal{C}_{40320} \times \mathcal{C}_{80640}^2 \times$ $\times \mathcal{C}_{11531520}^3 \times \mathcal{C}_{219098880} \times \mathcal{C}_{438197760} \times \mathcal{C}_{514005972480} \times$ $\times \mathcal{C}_{16648653448627200} \times \mathcal{C}_{27884713764834445351472640 0}$

Timings for Sage in Table 2 are given for the class_group(proof = False) method. Similar to the case of real multiquadratic fields [12, Table 1], we have slower computations for small degrees. But algorithm of Biasse — van Vredendaal allows us to compute class groups for fields of degrees $\geqslant 64$, while Sage does not.

Computation for $\deg K = 64$

In this case, we were not able to compute class group using built-in Sage methods. For the factor base, we took prime ideals above the rational primes $3, \ldots, 173$. The result was verified using the class number formula by the method from Section 3.4 as follows. The tentative class number is $h^* = 586628182540838348608924916121 6000$. The logarithm of Euler product (from class number formula) computed using Hecke [19] is equal to 11.2 ± 0.0136. We have $\left|\log \dfrac{h^* R^*}{\varepsilon h R}\right| \approx 0.00347$, $\dfrac{h^* R^*}{\varepsilon h R} \in [0.996, 1.003]$, so $\dfrac{h^* R^*}{\varepsilon h R} < \sqrt{2}$ as desired for verification of the class number.

Computation for $\deg K = 128$

To form the factor base, we took only prime ideals above rational primes p such that $p \nmid [\mathcal{O}_M : \mathbb{Z}[\theta_M]]$ for subfields M of K that were used in the class group computation algorithm. Note that the algorithm does not use all subfields of the field M, but a subset of subfields. Namely, we used prime ideals that are above the following rational primes:

$$p \in \{197, 223, 239, 257, 263, 271, 277, 331, 347, 353, 359, 367, 373, 379, 409\}.$$

Since the factor base is relatively small, we did not expect to obtain $h^* = h$. Nevertheless, the verification test was successfully passed. The logarithm of Euler product obtained using Hecke [19] is 24 ± 0.261, $\left|\log \dfrac{h^* R^*}{\varepsilon h R}\right| \approx 0.16$, $\dfrac{h^* R^*}{\varepsilon h R} \in [0.85, 1.17] < \sqrt{2}$. Thus, our computation returned the correct class number.

Computation for $\deg K = 256$

Here, we used prime ideals above the following rational primes

$$p \in \{233, 263, 347, 353, 359, 373, 421, 443, 467, 503, 509\}$$

in the factor base. At this point the verification fails and we report in the Table 2 only a multiple of the class number. Euler product computed using Hecke [19] is equal to 51±0.116. This leads to $\left|\log \frac{h^*R^*}{\varepsilon hR}\right| \approx 10.38$ and so h^* contains an extra factor of size $\leqslant 32208$.

REFERENCES

1. *Buchmann J. and Paulus S.* A one way function based on ideal arithmetic in number fields. LNCS, 1997, vol. 1294, pp. 385–394.
2. *Biehl I., Buchmann J., Hamdy S., and Meyer A.* A signature scheme based on the intractability of computing roots. Des. Codes Cryptogr., 2002, vol. 25, pp. 223–236.
3. *Meyer A., Neis S., and Pfahler T.* First implementation of cryptographic protocols based on algebraic number fields. LNCS, 2001, vol. 2119, pp. 84–103.
4. *Lyubashevsky V., Peikert C., and Regev O.* On ideal lattices and learning with errors over rings. LNCS, 2010, vol. 6110, pp. 1–23.
5. *Lyubashevsky V. and Micciancio D.* Generalized compact knapsacks are collision resistant. LNCS, 2006, vol. 4052, pp. 144–155.
6. *Bauch J., Bernstein D. J., de Valence H., et al.* Short generators without quantum computers: the case of multiquadratics. Ann. Intern. Conf. on the Theory and Applications of Cryptographic Techniques, 2017, vol. 84, no. 291, pp. 27–59. Source code: `https://multiquad.cr.yp.to/software.html`.
7. *Pellet-Mary A., Hanrot G., and Stehlé D.* Approx-SVP in ideal lattices with pre-processing. LNCS, 2019, vol. 11477, pp. 685–716.
8. *Bernard O. and Roux-Langlois A.* Twisted-PHS — Using the product formula to solve Approx-SVP in ideal lattices. LNCS, 2020, vol. 12492, pp. 349–380.
9. *Buchmann J.* A subexponential algorithm for the determination of class groups and regulators of algebraic number fields. Séminaire de Théorie des Nombres (Paris 1988/1989), Progr. Math., Birkhäuser, Boston, 1990, no. 91, pp. 27–41.
10. *Cohen H., Diaz Y Diaz F., and Olivier M.* Subexponential algorithms for class group and unit computations. J. Symbolic Computation, 1997, vol. 24, pp. 433–441.
11. *Biasse J.-F.* An L(1/3) Algorithm for Discrete Logarithm Computation and Principality Testing in Certain Number Fields. arXiv preprint, 2012. `https://arxiv.org/abs/1204.1292`.
12. *Biasse J.-F. and van Vredendaal C.* Fast multiquadratic S-unit computation and application to the calculation of class groups. The Open Book Series, 2019, vol. 2, pp. 103–118. Source code: `https://scarecryptow.org/publications/multiclass.html`.
13. *Lenstra A. K., Lenstra H. W., and Lovasz L.* Factoring polynomials with rational coefficients. Math. Ann., 1982, vol. 261, pp. 515–534.
14. *Bach E.* Explicit bounds for primality testing and related problems. Mathematics of Computation, 1990, vol. 55, no. 191, pp. 355–380.
15. *Grenié L. and Molteni G.* Explicit bounds for primality testing and related problems. Mathematics of Computation, 2018, no. 313, pp. 2483–2511.
16. *Cohen H.* A Course in Computational Algebraic Number Theory. Berlin, Springer Verlag, 1993.
17. *Bach E.* Improved approximations for Euler products. Number Theory, CMS Conf. Proc., 1995, vol. 15, pp. 13–28.
18. *Belabas K. and Friedman E.* Computing the residue of the Dedekind zeta function. Mathematics of Computation, 2015, vol. 84, no. 291, pp. 357–369.

19. *Fieker C., Hart W., Hofmann T., and Johansson F.* Nemo/Hecke: computer algebra and number theory packages for the Julia programming language. Proc. ISSAC'17, 2017, pp. 157–164. https://www.thofma.com/Hecke.jl/dev/.

20. *Bezanson J., Edelman A., Karpinski S., and Shah V. B.* Julia: A fresh approach to numerical computing. SIAM Review, 2017, vol. 59, no. 1, pp. 65–98.

21. *Cohen H.* Advanced Topics in Computational Number Theory. N.Y., Springer Science + Business Media, 2000.

22. *The Sage Developers.* SageMath, the Sage Mathematics Software System (Version 9.4). 2022, https://www.sagemath.org.