Improvements to ideal class group and regulator computation in real quadratic number fields

Jean-François Biasse\textsuperscript{1}, Michael J. Jacobson Jr\textsuperscript{2}
\textsuperscript{1}École polytechnique,
\textsuperscript{2}University of Calgary

ANTS IX
Let $\mathbb{K}$ be a quadratic number field of discriminant $\Delta$ and maximal order $\mathcal{O}_\Delta$. We are interested in

- Computing the group structure of $\text{Cl}(\Delta) := \text{Cl}(\mathcal{O}_\Delta)$. 

Motivations

Let $\mathbb{K}$ be a quadratic number field of discriminant $\Delta$ and maximal order $\mathcal{O}_\Delta$. We are interested in

- Computing the group structure of $\text{Cl}(\Delta) := \text{Cl}(\mathcal{O}_\Delta)$.
- Computing the regulator $R_\Delta$ of $\mathbb{K}$. 
Let $\mathbb{K}$ be a quadratic number field of discriminant $\Delta$ and maximal order $\mathcal{O}_\Delta$. We are interested in

- Computing the group structure of $\text{Cl}(\Delta) := \text{Cl}(\mathcal{O}_\Delta)$.
- Computing the regulator $R_\Delta$ of $\mathbb{K}$.
- Computing a compact representation of the fundamental unit $\varepsilon_\Delta$.

We provide practical improvements to the classical subexponential algorithms. We achieve the computation of $\text{Cl}(\Delta)$ and $R_\Delta$ for a 110-digit discriminant.
Motivations

Let $K$ be a quadratic number field of discriminant $\Delta$ and maximal order $\mathcal{O}_\Delta$. We are interested in

- Computing the group structure of $\text{Cl}(\Delta) := \text{Cl}(\mathcal{O}_\Delta)$.
- Computing the regulator $R_\Delta$ of $K$.
- Computing a compact representation of the fundamental unit $\varepsilon_\Delta$.

We provide practical improvements to the classical subexponential algorithms.
Motivations

Let $\mathbb{K}$ be a quadratic number field of discriminant $\Delta$ and maximal order $\mathcal{O}_\Delta$. We are interested in

- Computing the group structure of $\text{Cl}(\Delta) := \text{Cl}(\mathcal{O}_\Delta)$.
- Computing the regulator $R_\Delta$ of $\mathbb{K}$.
- Computing a compact representation of the fundamental unit $\varepsilon_\Delta$.

We provide practical improvements to the classical subexponential algorithms.

We achieve the computation of $\text{Cl}(\Delta)$ and $R_\Delta$ for a 110-digit discriminant.
1 Introduction

2 Classical Algorithms

3 Practical improvements
1 Introduction

2 Classical Algorithms

3 Practical improvements
Ideals

- We work in $\mathbb{K}$ that satisfies $[\mathbb{K} : \mathbb{Q}] = 2$. 
We work in $K$ that satisfies $[K : \mathbb{Q}] = 2$.

Let $\mathcal{O}_\Delta$ be the ring of integers of $K$ and $\Delta$ its discriminant.
Ideals

- We work in $K$ that satisfies $[K : \mathbb{Q}] = 2$.
- Let $\mathcal{O}_\Delta$ be the ring of integers of $K$ and $\Delta$ its discriminant.
- If $\Delta < 0$: imaginary case. If $\Delta > 0$: real case.
Ideals

- We work in $\mathbb{K}$ that satisfies $[\mathbb{K} : \mathbb{Q}] = 2$.
- Let $\mathcal{O}_\Delta$ be the ring of integers of $\mathbb{K}$ and $\Delta$ its discriminant.
- If $\Delta < 0$: **imaginary** case. If $\Delta > 0$: **real case**.
- The *fractional ideals* $\alpha$ are the sets of the form
  \[ \frac{1}{d} \alpha', \quad | \quad d \in \mathbb{K}, \quad \alpha' \text{ is an ideal of } \mathcal{O}_\Delta. \]
Ideal class group

Let $\mathcal{I}(\Delta)$ be the invertible fractional ideals and $\mathcal{P}$ the principal ideals, then

$$\text{Cl}(\Delta) := \mathcal{I}(\Delta)/\mathcal{P}.$$
Let \( \mathcal{I}(\Delta) \) be the invertible fractional ideals and \( \mathcal{P} \) the principal ideals, then

\[
\text{Cl}(\Delta) := \mathcal{I}(\Delta)/\mathcal{P}.
\]

\textbf{Cl}(\Delta) is finite of cardinality \( h(\Delta) \).
Ideal class group

- Let $\mathcal{I}(\Delta)$ be the invertible fractional ideals and $\mathcal{P}$ the principal ideals, then

$$\text{Cl}(\Delta) := \mathcal{I}(\Delta)/\mathcal{P}.$$ 

- $\text{Cl}(\Delta)$ is finite of cardinality $h(\Delta)$.
- $h(\Delta)$ is essentially as “hard” to compute as $\text{Cl}(\Delta)$. 

J-F. Biasse, M. J. Jacobson Jr
Improvements in class group and regulator computation
Ideal class group

- Let $\mathcal{I}(\Delta)$ be the invertible fractional ideals and $\mathcal{P}$ the principal ideals, then

$$\text{Cl}(\Delta) := \mathcal{I}(\Delta)/\mathcal{P}.$$ 

- $\text{Cl}(\Delta)$ is finite of cardinality $h(\Delta)$.
- $h(\Delta)$ is essentially as “hard” to compute as $\text{Cl}(\Delta)$.

Let $a, b \in \mathcal{I}(\Delta)$, then we denote by $a \sim b$ :

$$[a] = [b] \in \text{Cl}(\Delta) \iff \exists \alpha \in \mathbb{K}, \ a = (\alpha)b.$$
Regulator

We assume that $\Delta > 0$. 
We assume that $\Delta > 0$.

- Elements of $\mathbb{K}$ such that $\mathcal{N}(x) = \pm 1$ are *units*. 
We assume that $\Delta > 0$.

- Elements of $\mathbb{K}$ such that $\mathcal{N}(x) = \pm 1$ are \textit{units}.
- Every unit $\epsilon$ can be written as $\epsilon = \pm \epsilon_{\Delta}^n$, where $\epsilon_{\Delta}$ is the \textit{fundamental unit} of $\mathbb{K}$. 
We assume that $\Delta > 0$.

- Elements of $K$ such that $\mathcal{N}(x) = \pm 1$ are *units*.
- Every unit $\varepsilon$ can be written as $\varepsilon = \pm \varepsilon_\Delta^n$, where $\varepsilon_\Delta$ is the *fundamental unit* of $K$.

The *regulator* of $K$ is

$$R_\Delta = \log \varepsilon_\Delta.$$
We assume that $\Delta > 0$.

- Elements of $K$ such that $\mathcal{N}(x) = \pm 1$ are units.
- Every unit $\varepsilon$ can be written as $\varepsilon = \pm \varepsilon_\Delta^n$, where $\varepsilon_\Delta$ is the fundamental unit of $K$.

The regulator of $K$ is

$$R_\Delta = \log \varepsilon_\Delta.$$

Every unit $\varepsilon$ satisfies $\exists n$, $\log |\varepsilon| = nR_\Delta$. 
The algorithms for solving our problems follow the same pattern. Let $B = \{p_1, \ldots, p_N\}$ be a generating set of $\text{Cl}(\Delta)$. 

Every time a relation is found, $[e_1, \ldots, e_N]$ is added as a row of the relation matrix $M$. 

Perform a linear algebra phase on $M$. 

J-F. Biasse, M. J. Jacobson Jr

Improvements in class group and regulator computation
General strategy

The algorithms for solving our problems follow the same pattern. Let $\mathcal{B} = \{p_1, \ldots, p_N\}$ be a generating set of $\text{Cl}(\Delta)$.

1. Find relations of the form

$$ (\alpha) = p_1^{e_1} \cdots p_N^{e_N}, $$

that is $\prod_i [p_i]^{e_i} = [1]$.
The algorithms for solving our problems follow the same pattern. Let $B = \{p_1, \ldots, p_N\}$ be a generating set of $Cl(\Delta)$.

1. Find relations of the form

$$ (\alpha) = p_1^{e_1} \cdots p_N^{e_N}, $$

that is $\prod_i [p_i]^{e_i} = \{1\}$

2. Every time a relation is found, $[e_1, \ldots, e_N]$ is added as a row of the *relation matrix* $M$. 
General strategy

The algorithms for solving our problems follow the same pattern. Let $\mathcal{B} = \{p_1, \ldots, p_N\}$ be a generating set of $\text{Cl}(\Delta)$.

1. Find relations of the form

$$\alpha = p_1^{e_1} \cdots p_N^{e_N},$$

that is $\prod_i [p_i]^{e_i} = [1]$

2. Every time a relation is found, $[e_1, \ldots, e_N]$ is added as a row of the relation matrix $M$

3. Perform a linear algebra phase on $M$. 

We define the *subexponential* function by

\[
L_\Delta(\alpha, \beta) = e^{\beta \log |\Delta|^{\alpha} \log \log |\Delta|^{1-\alpha}}.
\]
We define the \textit{subexponential} function by

\[
L_\Delta(\alpha, \beta) = e^{\beta \log |\Delta|^\alpha \log \log |\Delta|^{1-\alpha}}.
\]

For $\alpha \in [0, 1]$, $L_\Delta(\alpha, \beta)$ is between exponential and polynomial in $\log |\Delta|$ since

\[
L_\Delta(0, \beta) = \log |\Delta|^\beta,
\]
\[
L_\Delta(1, \beta) = |\Delta|^\beta.
\]
Complexity

We define the *subexponential* function by

\[ L_\Delta(\alpha, \beta) = e^{\beta \log |\Delta|^\alpha \log \log |\Delta|^{1-\alpha}}. \]

For \( \alpha \in [0, 1] \), \( L_\Delta(\alpha, \beta) \) is between exponential and polynomial in \( \log |\Delta| \) since

\[ L_\Delta(0, \beta) = \log |\Delta|^{\beta}, \]
\[ L_\Delta(1, \beta) = |\Delta|^{\beta}. \]

Our problems for quadratic number fields have complexity

\[ L_\Delta(1/2, c), \]

where \( c \) depends on the linear algebra phase.
The factor base

We fill the factor base with invertible prime ideals \( p \). There is \( p \) prime such that

\[
p \cap \mathbb{Z} = (p) \quad \text{and} \quad \mathcal{N}(p) = p.
\]
The factor base

We fill the factor base with invertible prime ideals \( p \). There is a prime such that

\[ p \cap \mathbb{Z} = (p) \quad \text{and} \quad \mathcal{N}(p) = p. \]

Let \( B \) a bound, we define

\[ B := \{ p \text{ invertible prime} \mid \mathcal{N}(p) \leq B \} = \{ p_1, \ldots, p_N \}. \]
The factor base

We fill the factor base with invertible prime ideals $p$. There is $p$ prime such that

$$p \cap \mathbb{Z} = (p) \text{ and } \mathcal{N}(p) = p.$$ 

Let $B$ a bound, we define

$$\mathcal{B} := \{p \text{ invertible prime} \mid \mathcal{N}(p) \leq B\} = \{p_1, \ldots, p_N\}.$$ 

Under ERH, if $B > 6 \log^2 |\Delta|$, then $\mathcal{B}$ generates $\text{Cl}(\Delta)$, and the lattice $\mathcal{L}$ of the relations satisfies

$$\text{Cl}(\Delta) \cong \mathbb{Z}^N / \mathcal{L}.$$
The factor base

We fill the factor base with invertible prime ideals \( p \). There is \( p \) prime such that

\[
\mathfrak{p} \cap \mathbb{Z} = (p) \quad \text{and} \quad \mathcal{N}(\mathfrak{p}) = p.
\]

Let \( B \) a bound, we define

\[
\mathcal{B} := \{ \mathfrak{p} \text{ invertible prime} \mid \mathcal{N}(\mathfrak{p}) \leq B\} = \{p_1, \ldots, p_N\}.
\]

Under ERH, if \( B > 6 \log^2 |\Delta| \), then \( \mathcal{B} \) generates \( \text{Cl}(\Delta) \), and the lattice \( \mathcal{L} \) of the relations satisfies

\[
\text{Cl}(\Delta) \simeq \mathbb{Z}^N / \mathcal{L}.
\]

We have \((\alpha \text{ is } \mathcal{B}\text{-smooth}) \iff (\mathcal{N}(\alpha) \text{ is } B\text{-smooth})\).
Invertible operations on rows lead to the **Hermite Normal Form** $H$ of $M$:

$$
H = \begin{bmatrix}
  h_{1,1} & \cdots & 0 \\
  \vdots & \ddots & \vdots \\
  * & \cdots & h_{l,l} \\
\end{bmatrix}
\begin{bmatrix}
  (0) \\
  1 & (0) \\
  (0) & 1 \\
\end{bmatrix},
$$

where $\forall i > j : 0 \leq h_{ij} < h_{jj}$. 
Invertible operations on rows lead to the **Hermite Normal Form** $H$ of $M$:

$$H = \begin{pmatrix}
  h_{1,1} & \cdots & 0 \\
  \vdots & \ddots & \vdots \\
  * & \cdots & h_{l,l} \\
\end{pmatrix}
\begin{pmatrix}
  (0) \\
  (0) \\
  (0) \\
\end{pmatrix}
\begin{pmatrix}
  1 \\
  \vdots \\
  1 \\
\end{pmatrix},
$$

where $\forall i > j : 0 \leq h_{ij} < h_{jj}$.

Upper left : **Essential part**
Any matrix $A \in \mathbb{Z}^{n \times n}$ with non zero determinant can be written as:

$$A = V^{-1} \begin{pmatrix} d_1 & 0 & \ldots & 0 \\ 0 & d_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \ldots & 0 & d_n \end{pmatrix} U^{-1}$$

where $\forall i$ such that $1 \leq i < n : d_{i+1} | d_i$. 
Any matrix $A \in \mathbb{Z}^{n \times n}$ with non zero determinant can be written as:

$$A = V^{-1} \begin{pmatrix} d_1 & 0 & \ldots & 0 \\ 0 & d_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \ldots & 0 & d_n \end{pmatrix} U^{-1}$$

where $\forall i$ such that $1 \leq i < n : d_{i+1} | d_i$.

If $(d_i)$ are the diagonal coefficients of the SNF of the essential part of $H$ then

$$Cl(\Delta) = \bigoplus_{1 \leq i \leq n} (\mathbb{Z}/d_i\mathbb{Z})$$
Several implementation of HNF algorithm exist: Pari, Kash, Sage, Magma, NTL ... We used an NTL/Linbox-based strategy.
Several implementation of HNF algorithm exist: Pari, Kash, Sage, Magma, NTL... We used an NTL/Linbox-based strategy.

Let $M \in \mathbb{Z}^{N' \times N}$ be the relation matrix.

1. Extract two random $N \times N$ full-rank submatrices $M_1$ and $M_2$ of $M$. 

2. Compute $h_1 \leftarrow \text{det}(M_1)$ and $h_2 \leftarrow \text{det}(M_2)$ with function det of linbox.

3. Let $h := \gcd(h_1, h_2)$. It is a multiple of $h \Delta$.

4. Call the implementation of DomKanTro87 modular HNF algorithm with $(M, h)$.

In fact MAGMA is much faster :(

⇒ room for improvement.
Computing the HNF in practice

Several implementation of HNF algorithm exist: Pari, Kash, Sage, Magma, NTL... We used an NTL/Linbox-based strategy.

Let $M \in \mathbb{Z}^{N' \times N}$ be the relation matrix.

1. Extract two random $N \times N$ full-rank submatrices $M_1$ and $M_2$ of $M$.

2. Compute $h_1 \leftarrow \det(M_1)$ and $h_2 \leftarrow \det(M_2)$ with function $\det$ of linbox.
Computing the HNF in practice

Several implementation of HNF algorithm exist: Pari, Kash, Sage, Magma, NTL ... We used an NTL/Linbox-based strategy.

Let \( M \in \mathbb{Z}^{N' \times N} \) be the relation matrix.

1. Extract two random \( N \times N \) full-rank submatrices \( M_1 \) and \( M_2 \) of \( M \).
2. Compute \( h_1 \leftarrow \det(M_1) \) and \( h_2 \leftarrow \det(M_2) \) with function \( \det \) of linbox.
3. Let \( h := \gcd(h_1, h_2) \). It is a multiple of \( h_\Delta \).
Several implementation of HNF algorithm exist : Pari, Kash, Sage, Magma, NTL ... We used an NTL/Linbox-based strategy.

Let $M \in \mathbb{Z}^{N' \times N}$ be the relation matrix.

1. Extract two random $N \times N$ full-rank submatrices $M_1$ and $M_2$ of $M$.
2. Compute $h_1 \leftarrow \det(M_1)$ and $h_2 \leftarrow \det(M_2)$ with function $\det$ of linbox.
3. Let $h := \gcd(h_1, h_2)$. It is a multiple of $h_\Delta$.
4. Call the implementation of DomKanTro87 modular HNF algorithm with $(M, h)$. 

In fact MAGMA is much faster :(
Several implementation of HNF algorithm exist: Pari, Kash, Sage, Magma, NTL... We used an NTL/Linbox-based strategy.

Let $M \in \mathbb{Z}^{N' \times N}$ be the relation matrix.

1. Extract two random $N \times N$ full-rank submatrices $M_1$ and $M_2$ of $M$.
2. Compute $h_1 \leftarrow \text{det}(M_1)$ and $h_2 \leftarrow \text{det}(M_2)$ with function \text{det} of linbox.
3. Let $h := \gcd(h_1, h_2)$. It is a multiple of $h_\Delta$.
4. Call the implementation of DomKanTro87 modular HNF algorithm with $(M, h)$.

In fact MAGMA is much faster :( ⇒ room for improvement.
Let $M = (m_{ij}) \in \mathbb{Z}^{N' \times N}$, be the relation matrix, $\mathcal{B} = \{p_1, \ldots, p_N\}$, and

$$(\alpha_i) = p_1^{m_{i1}} \cdots p_N^{m_{iN}}.$$
Regulator computation

Let $M = (m_{ij}) \in \mathbb{Z}^{N' \times N}$, be the relation matrix, $\mathcal{B} = \{p_1, \ldots, p_N\}$, and

$$(\alpha_i) = p_1^{m_{i1}} \cdots p_N^{m_{iN}}.$$ 

Let $X = (x_i), \ i \leq N'$ be a kernel vector of $M$. Then

$$\gamma := \alpha_1^{x_1} \cdots \alpha_{N'}^{x_{N'}}$$

is a unit since $(\gamma) = \prod_i \alpha_i^{x_i} = (1)$. 

Let $M = (m_{ij}) \in \mathbb{Z}^{N' \times N}$, be the relation matrix, $\mathcal{B} = \{p_1, \ldots, p_N\}$, and

$$(\alpha_i) = p_1^{m_{i1}} \cdots p_N^{m_{iN}}.$$ 

Let $X = (x_i), i \leq N'$ be a kernel vector of $M$. Then

$$\gamma := \alpha_1^{x_1} \cdots \alpha_{N'}^{x_{N'}}$$

is a unit since $(\gamma) = \prod_i \alpha_i^{x_i} = (1)$. There is $n$ such that

$$\log |\gamma| = nR_{\Delta}$$
Regulator computation

Let $M = (m_{ij}) \in \mathbb{Z}^{N' \times N}$, be the relation matrix, $B = \{p_1, \ldots, p_N\}$, and

$$(\alpha_i) = p_1^{m_{i1}} \cdots p_N^{m_{iN}}.$$ 

Let $X = (x_i), i \leq N'$ be a kernel vector of $M$. Then

$$\gamma := \alpha_1^{x_1} \cdots \alpha_{N'}^{x_{N'}}$$

is a unit since $(\gamma) = \prod_i \alpha_i^{x_i} = (1)$. 

There is $n$ such that

$$\log |\gamma| = nR_\Delta$$

Each kernel vector of $M$ yields a multiple of $R_\Delta$. We recover $R_\Delta$ by successive real-GCD computation.
Relation collection via sieving

Let $\mathfrak{a}$ be an ideal. There is $\mathfrak{a}' \sim \mathfrak{a}$ of the form $\mathfrak{a}' = a\mathbb{Z} + \frac{(b+\sqrt{\Delta})}{2}\mathbb{Z}$. 
Relation collection via sieving

Let $\mathfrak{a}$ be an ideal. There is $\mathfrak{a}' \sim \mathfrak{a}$ of the form $\mathfrak{a}' = a\mathbb{Z} + \frac{(b+\sqrt{\Delta})}{2}\mathbb{Z}$. Then for each $x, y$ we have

$$\gamma := ax + y \left( \frac{b + \sqrt{\Delta}}{2} \right) \in \mathfrak{a}'. $$
Relation collection via sieving

Let $a$ be an ideal. There is $a' \sim a$ of the form $a' = a\mathbb{Z} + \frac{(b+\sqrt{\Delta})}{2}\mathbb{Z}$. Then for each $x, y$ we have

$$
\gamma := ax + y \left( \frac{b + \sqrt{\Delta}}{2} \right) \in a'.
$$

(JacWil09) There is an ideal $b$ such that $(\gamma) = a'b$ (that is $a \cdot b \sim 1$) and

$$\mathcal{N}(b) = ax^2 + bxy + cy^2.$$
Relation collection via sieving

Let \( \alpha \) be an ideal. There is \( \alpha' \sim \alpha \) of the form \( \alpha' = \alpha \mathbb{Z} + \frac{(b + \sqrt{\Delta})}{2} \mathbb{Z} \).
Then for each \( x, y \) we have

\[
\gamma := ax + y \left( \frac{b + \sqrt{\Delta}}{2} \right) \in \alpha'.
\]

(JacWil09) There is an ideal \( b \) such that \( (\gamma) = \alpha' b \) (that is \( \alpha \cdot b \sim 1 \)) and

\[
\mathcal{N}(b) = ax^2 + bxy + cy^2.
\]

1. Start with \( \alpha := \prod_i p_i^{e_i} \) which is \( B \)-smooth.
Relation collection via sieving

Let $a$ be an ideal. There is $a' \sim a$ of the form $a' = a\mathbb{Z} + \frac{(b+\sqrt{\Delta})}{2}\mathbb{Z}$. Then for each $x, y$ we have

$$\gamma := ax + y\left(\frac{b + \sqrt{\Delta}}{2}\right) \in a'.$$

(JacWil09) There is an ideal $b$ such that $(\gamma) = a'b$ (that is $a \cdot b \sim 1$) and

$$\mathcal{N}(b) = ax^2 + bxy + cy^2.$$

1. Start with $a := \prod_i p_i^{e_i}$ which is $B$-smooth.
2. Find $x, y$ such that $\phi_a(x, y) := ax^2 + bxy + cy^2$ is $B$-smooth.
Relation collection via sieving

Let $a$ be an ideal. There is $a' \sim a$ of the form $a' = a\mathbb{Z} + \frac{(b+\sqrt{\Delta})}{2}\mathbb{Z}$. Then for each $x, y$ we have

$$\gamma := ax + y \left( \frac{b + \sqrt{\Delta}}{2} \right) \in a'.$$

(JacWil09) There is an ideal $b$ such that $\gamma = a'b$ (that is $a \cdot b \sim 1$) and

$$\mathcal{N}(b) = ax^2 + bxy + cy^2.$$

1. Start with $a := \prod_i p_i^{e_i}$ which is $B$-smooth.
2. Find $x, y$ such that $\phi_a(x, y) := ax^2 + bxy + cy^2$ is $B$-smooth.
3. Deduce $B$-smooth ideal $b$ such that $a \cdot b \sim 1$. 
The quadratic sieve

Let \( \phi_a(X, Y) = aX^2 + bXY + cY^2 \) and \( B \) defining \( \mathcal{B} \). We look for \( B \)-smooth values of \( \phi_a(X, Y) \). (Jac99) : use the \textbf{quadratic sieve}
The quadratic sieve

Let $\phi_a(X, Y) = aX^2 + bXY + cY^2$ and $B$ defining $B$. We look for $B$-smooth values of $\phi_a(X, Y)$. (Jac99) : use the **quadratic sieve**

We look for $x \in [-M, M]$ such that $\phi_a(x, 1)$ is $B$-smooth. We do not want to test them all.
The quadratic sieve

Let \( \phi_a(X, Y) = aX^2 + bXY + cY^2 \) and \( B \) defining \( \mathcal{B} \). We look for \( B \)-smooth values of \( \phi_a(X, Y) \). (Jac99) : use the quadratic sieve

We look for \( x \in [-M, M] \) such that \( \phi_a(x, 1) \) is \( B \)-smooth. We do not want to test them all.

1. We compute the roots \( r_p \) of \( \phi_a(X, 1) \mod p \) for \( p \leq B \).
The quadratic sieve

Let \( \phi_a(X, Y) = aX^2 + bXY + cY^2 \) and \( B \) defining \( \mathcal{B} \). We look for \( B \)-smooth values of \( \phi_a(X, Y) \). (Jac99) : use the **quadratic sieve**

We look for \( x \in [-M, M] \) such that \( \phi_a(x, 1) \) is \( B \)-smooth. We do not want to test them all.

1. We compute the roots \( r_p \) of \( \phi_a(X, 1) \mod p \) for \( p \leq B \).
2. We initialize \( S \) of length \( 2M + 1 \) to 0.
The quadratic sieve

Let \( \phi_a(X, Y) = aX^2 + bXY + cY^2 \) and \( B \) defining \( B \). We look for \( B \)-smooth values of \( \phi_a(X, Y) \). (Jac99) : use the **quadratic sieve**

We look for \( x \in [-M, M] \) such that \( \phi_a(x, 1) \) is \( B \)-smooth. We do not want to test them all.

1. We compute the roots \( r_p \) of \( \phi_a(X, 1) \mod p \) for \( p \leq B \).
2. We initialize \( S \) of length \( 2M + 1 \) to 0.
3. For \( x = r_p + kp \in [-M, M] \) do \( S[x] \leftarrow S[x] + \log p \) because

\[
\phi_a(x, 1) = \phi_a(r_p + kp, 1) \equiv \phi_a(r_p, 1) \equiv 0 \mod p.
\]
The quadratic sieve

Let $\phi_a(X, Y) = aX^2 + bXY + cY^2$ and $B$ defining $\mathcal{B}$. We look for $B$-smooth values of $\phi_a(X, Y)$. (Jac99) : use the **quadratic sieve**

We look for $x \in [-M, M]$ such that $\phi_a(x, 1)$ is $B$-smooth. We do not want to test them all.

1. We compute the roots $r_p$ of $\phi_a(X, 1) \mod p$ for $p \leq B$.
2. We initialize $S$ of length $2M + 1$ to 0.
3. For $x = r_p + kp \in [-M, M]$ do $S[x] \leftarrow S[x] + \log p$ because

   $$\phi_a(x, 1) = \phi_a(r_p + kp, 1) \equiv \phi_a(r_p, 1) \equiv 0 \mod p.$$ 

4. For “large” $S[x]$, test the smoothness of $\phi_a(x, 1)$.
Large prime variants

We speed-up the relation collection phase by considering \( p \) such that 
\[ B \leq \mathcal{N}(p) \leq B_2. \]
Large prime variants

We speed-up the relation collection phase by considering \( p \) such that
\[ B \leq \mathcal{N}(p) \leq B_2. \]

- **Single large prime variant.** We authorize relations of the form
  \[ a = \underbrace{p_1 \ldots p_n}_\in B p, \]
  where \( B \leq \mathcal{N}(p) \leq B_2. \)
Large prime variants

We speed-up the relation collection phase by considering $p$ such that $B \leq \mathcal{N}(p) \leq B_2$.

- **Single large prime variant.** We authorize relations of the form

  $$a = \underbrace{p_1 \ldots p_n}_{\in \mathcal{B}} p,$$

  where $B \leq \mathcal{N}(p) \leq B_2$.

- **Double large prime variant.** We authorise relations of the form

  $$a = \underbrace{p_1 \ldots p_n}_{\in \mathcal{B}} pp',$$

  where $B \leq \mathcal{N}(p), \mathcal{N}(p') \leq B_2$. 
Batch smoothness test

Quadratic sieve: for large $S[x]$, we test the smoothness of $\phi_a(x, 1)$. 
Batch smoothness test

Quadratic sieve: for large $S[x]$, we test the smoothness of $\phi_a(x, 1)$. This can be done by trial division.
Batch smoothness test

Quadratic sieve: for large $S[x]$, we test the smoothness of $\phi_a(x, 1)$.

This can be done by trial division.

We used an algorithm due to Berstein.
Batch smoothness test

Quadratic sieve: for large $S[x]$, we test the smoothness of $\phi_a(x, 1)$. This can be done by trial division.

We used an algorithm due to Berstein.

- Takes non negative $x_1, \ldots, x_K$ and primes $p_1, \ldots, p_N$. 
Batch smoothness test

Quadratic sieve: for large $S[x]$, we test the smoothness of $\phi_a(x, 1)$. This can be done by trial division.

We used an algorithm due to Berstein.
- Takes non-negative $x_1, \ldots, x_K$ and primes $p_1, \ldots, p_N$.
- Returns the $\{p_1, \ldots, p_N\}$-smooth part of each $x_i$. 
Batch smoothness test

Quadratic sieve: for large $S[x]$, we test the smoothness of $\phi_a(x, 1)$.

This can be done by trial division.

We used an algorithm due to Berstein.

- Takes non negative $x_1, \ldots, x_K$ and primes $p_1, \ldots, p_N$.
- returns the $\{p_1, \ldots, p_N\}$-smooth part of each $x_i$.
- Test is simultaneous.
Batch smoothness test

Quadratic sieve: for large $S[x]$, we test the smoothness of $\phi_a(x, 1)$. This can be done by trial division.

We used an algorithm due to Berstein.

- Takes non-negative $x_1, \ldots, x_K$ and primes $p_1, \ldots, p_N$.
- returns the $\{p_1, \ldots, p_N\}$-smooth part of each $x_i$.
- Test is simultaneous.
- uses a tree structure.
Relation collection timings

**Tab.:** Comparison of the relation collection time for $\Delta = -4(10^n + 1)$

<table>
<thead>
<tr>
<th>$n$</th>
<th>0LP</th>
<th>1LP</th>
<th>2LP</th>
<th>2LP Batch</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>0.69</td>
<td>0.56</td>
<td>0.59</td>
<td>0.66</td>
</tr>
<tr>
<td>45</td>
<td>7.25</td>
<td>3.77</td>
<td>3.83</td>
<td>4.41</td>
</tr>
<tr>
<td>50</td>
<td>18.82</td>
<td>9.30</td>
<td>9.84</td>
<td>6.82</td>
</tr>
<tr>
<td>55</td>
<td>152.28</td>
<td>74.78</td>
<td>55.99</td>
<td>36.49</td>
</tr>
<tr>
<td>60</td>
<td>333.26</td>
<td>166.88</td>
<td>140.79</td>
<td>83.06</td>
</tr>
<tr>
<td>65</td>
<td>2033.97</td>
<td>853.27</td>
<td>478.57</td>
<td>368.31</td>
</tr>
<tr>
<td>70</td>
<td>2828.92</td>
<td>1277.94</td>
<td>822.39</td>
<td>670.63</td>
</tr>
<tr>
<td>75</td>
<td>14811.70</td>
<td>6033.89</td>
<td>3324.61</td>
<td>2732.68</td>
</tr>
</tbody>
</table>
Eliminating columns

Sparse large matrix. Especially with the large primes.
Sparse large matrix. Especially with the large primes.

We want to eliminate columns to reduce its dimension and apply algorithms for dense matrices.
Eliminating columns

Sparse large matrix. Especially with the large primes.

We want to eliminate columns to reduce its dimension and apply algorithms for dense matrices.

We can use the standard Gaussian elimination. It consists of pivoting with an arbitrary row.
Eliminating columns

Sparse large matrix. Especially with the large primes.

We want to eliminate columns to reduce its dimension and apply algorithms for dense matrices.

We can use the standard Gaussian elimination. It consists of pivoting with an arbitrary row.

Two problems encountered:

1. \( R_3 \) can have Hamming weight \( w(R_3) = w(R_1) + w(R_2) \).

2. The coefficients might grow dramatically.

We describe a method for managing the growth of the density and the size of the coefficients during the elimination.
Eliminating columns

Sparse large matrix. Especially with the large primes.

We want to eliminate columns to reduce its dimension and apply algorithms for dense matrices.

We can use the standard Gaussian elimination. It consists of pivoting with an arbitrary row.

Two problems encountered:

1. $R_3$ can have Hamming weight $w(R_3) = w(R_1) + w(R_2)$. 
Eliminating columns

Sparse large matrix. Especially with the large primes.

We want to eliminate columns to reduce its dimension and apply algorithms for dense matrices.

We can use the standard Gaussian elimination. It consists of pivoting with an arbitrary row.

Two problems encountered:

1. $R_3$ can have Hamming weight $w(R_3) = w(R_1) + w(R_2)$.
2. The coefficients might grow dramatically.
Eliminating columns

Sparse large matrix. Especially with the large primes.

We want to eliminate columns to reduce its dimension and apply algorithms for dense matrices.

We can use the standard Gaussian elimination. It consists of pivoting with an arbitrary row.

Two problems encountered:

1. $R_3$ can have Hamming weight $w(R_3) = w(R_1) + w(R_2)$.
2. The coefficients might grow dramatically.

We describe a method for managing the growth of the density and the size of the coefficients during the elimination.
Row $R \rightarrow$ cost function $COST(R)$ taking into account:

1. Hamming weight of $R$
2. Size of its coefficients
Structured Elimination

Row $R \rightarrow$ cost function $COST(R)$ taking into account:

1. Hamming weight of $R$
Structured Elimination

Row $R \rightarrow$ cost function $COST(R)$ taking into account:

1. Hamming weight of $R$
2. Size of its coefficients
Structured Elimination

Row $R \rightarrow$ cost function $COST(R)$ taking into account:

1. Hamming weight of $R$
2. Size of its coefficients

For a given column involving rows $R_1, \ldots, R_k$ we construct the complete graph $G$:
Structured Elimination

Row $R \rightarrow$ cost function $COST(R)$ taking into account:

1. Hamming weight of $R$
2. Size of its coefficients

For a given column involving rows $R_1, \ldots, R_k$ we construct the complete graph $G$:

4. vertices $R_i$
Structured Elimination

Row $R \rightarrow$ cost function $COST(R)$ taking into account:

1. Hamming weight of $R$
2. Size of its coefficients

For a given column involving rows $R_1, \ldots, R_k$ we construct the complete graph $G$:

1. vertices $R_i$
2. edges labeled with the cost of the recombination $C_{ij} = COST(RECOMB(R_i, R_j))$
Structured Elimination

Row $R \rightarrow$ cost function $COST(R)$ taking into account:

1. Hamming weight of $R$
2. Size of its coefficients

For a given column involving rows $R_1, \ldots, R_k$ we construct the complete graph $G$:

1. vertices $R_i$
2. edges labeled with the cost of the recombination $C_{ij} = COST(RECOMB(R_i, R_j))$

We then construct the minimum spanning tree of $G$ and eliminate rows from the leaves to the root.
Minimum spanning tree on Alberta’s map
Minimum spanning tree on Alberta’s map
Minimum spanning tree on Alberta’s map
Minimum spanning tree on Alberta’s map
Minimum spanning tree on Alberta’s map
Minimum spanning tree on Alberta’s map

Jasper
Edmonton
Red Deer
Calgary

5 h
3 h
5 h
Minimum spanning tree on Alberta’s map

Jasper Edmonton
Red Deer
Calgary

3 h

J-F. Biasse, M. J. Jacobson Jr

Improvements in class group and regulator computation
Minimum spanning tree on Alberta’s map

Jasper
Edmonton
Red Deer
Calgary

2 h
2 h
3 h
5 h
5 h
5 h

J-F. Biasse, M. J. Jacobson Jr
Improvements in class group and regulator computation
Minimum spanning tree on Alberta’s map

Jasper  Edmonton  Red Deer  Calgary

3 h    2 h

J-F. Biasse, M. J. Jacobson Jr  Improvements in class group and regulator computation
Minimum spanning tree on Alberta’s map
Minimum spanning tree on Alberta’s map
Timings Gaussian elimination for $\Delta = 4(10^{60} + 3)$

<table>
<thead>
<tr>
<th>$i$</th>
<th>Col Nb</th>
<th>HNF time</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>1067</td>
<td>357.9</td>
</tr>
<tr>
<td>10</td>
<td>799</td>
<td>184.8</td>
</tr>
<tr>
<td>50</td>
<td>596</td>
<td>93.7</td>
</tr>
<tr>
<td>125</td>
<td>542</td>
<td>73.8</td>
</tr>
<tr>
<td>160</td>
<td>533</td>
<td>72.0</td>
</tr>
<tr>
<td>170</td>
<td>532</td>
<td>222.4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$i$</th>
<th>Col Nb</th>
<th>HNF time</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>1078</td>
<td>368.0</td>
</tr>
<tr>
<td>10</td>
<td>806</td>
<td>187.2</td>
</tr>
<tr>
<td>50</td>
<td>580</td>
<td>84.3</td>
</tr>
<tr>
<td>125</td>
<td>515</td>
<td>63.4</td>
</tr>
<tr>
<td>160</td>
<td>497</td>
<td>56.9</td>
</tr>
<tr>
<td>170</td>
<td>493</td>
<td>192.6</td>
</tr>
</tbody>
</table>
Regulator computation

We want to avoid kernel computation and use fewer vectors. Idea due to Vollmer
Regulator computation

We want to avoid kernel computation and use fewer vectors. Idea due to Vollmer

1. We find $k$ extra relations $\vec{r}_i$. 

$\vec{x}_i$
Regulator computation

We want to avoid kernel computation and use fewer vectors. Idea due to Vollmer

1. We find \( k \) extra relations \( \vec{r}_i \).
2. We solve the \( k \) linear systems \( \vec{x}_i M = \vec{r}_i \).
Regulator computation

We want to avoid kernel computation and use fewer vectors. Idea due to Vollmer

1. We find $k$ extra relations $\vec{r}_i$.
2. We solve the $k$ linear systems $\vec{x}_i M = \vec{r}_i$.
3. We augment the matrix $M$ with the $k$ extra rows

$$M' := \begin{pmatrix} M & \vdots \vdots \\ \vec{r}_i & \vdots \vdots \end{pmatrix} \quad \vec{x}_i' := \begin{pmatrix} \vec{x}_i \\ 0 \ldots 0 \ -1 \ 0 \ldots 0 \end{pmatrix}.$$
Regulator computation

We want to avoid kernel computation and use fewer vectors. Idea due to Vollmer

1. We find $k$ extra relations $\vec{r}_i$.
2. We solve the $k$ linear systems $\vec{x}_i M = \vec{r}_i$.
3. We augment the matrix $M$ with the $k$ extra rows

$$
M' := \begin{pmatrix}
M \\
......... \\
\vec{r}_i
\end{pmatrix}
\vec{x}_i' := \begin{pmatrix} \vec{x}_i, 0...0, -1, 0...0 \end{pmatrix}.
$$

The $\vec{x}_i'$ are kernel vectors of the new relation matrix $M'$. 
Timings regulator computation

Kernel computation in $O(L_\Delta(1/2, \sqrt{2}))$. 

System solving in $O(L_\Delta(1/2, 3/\sqrt{8}))$. 

<table>
<thead>
<tr>
<th>n</th>
<th>Kernel computation</th>
<th>System solving</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>40</td>
<td>3.4</td>
</tr>
<tr>
<td>5</td>
<td>45</td>
<td>6.1</td>
</tr>
<tr>
<td>6</td>
<td>50</td>
<td>18.2</td>
</tr>
<tr>
<td>7</td>
<td>55</td>
<td>53.0</td>
</tr>
<tr>
<td>8</td>
<td>60</td>
<td>140.0</td>
</tr>
<tr>
<td>9</td>
<td>65</td>
<td>320.2</td>
</tr>
<tr>
<td>10</td>
<td>70</td>
<td>791.1</td>
</tr>
<tr>
<td>11</td>
<td>75</td>
<td>1775.8</td>
</tr>
</tbody>
</table>

J-F. Biasse, M. J. Jacobson Jr
Timings regulator computation

Kernel computation in $O(L_\Delta(1/2, \sqrt{2}))$.
System solving in $O(L_\Delta(1/2, 3/\sqrt{8}))$. 
Timings regulator computation

Kernel computation in $O(L_\Delta(1/2, \sqrt{2}))$.
System solving in $O(L_\Delta(1/2, 3/\sqrt{8}))$.

Tab.: Regulator computation time for $\Delta = 4(10^n + 3)$

<table>
<thead>
<tr>
<th>$n$</th>
<th>kernel computation</th>
<th>system solving</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>9.7</td>
<td>3.4</td>
</tr>
<tr>
<td>45</td>
<td>17.6</td>
<td>6.1</td>
</tr>
<tr>
<td>50</td>
<td>39.9</td>
<td>18.2</td>
</tr>
<tr>
<td>55</td>
<td>126.7</td>
<td>53.0</td>
</tr>
<tr>
<td>60</td>
<td>424.1</td>
<td>140.0</td>
</tr>
<tr>
<td>65</td>
<td>514.8</td>
<td>320.2</td>
</tr>
<tr>
<td>70</td>
<td>2728.5</td>
<td>791.1</td>
</tr>
<tr>
<td>75</td>
<td>8587.8</td>
<td>1775.8</td>
</tr>
</tbody>
</table>
Overall time comparison

Discriminants of the form $\Delta = 4(10^n + 3)$

<table>
<thead>
<tr>
<th>$n$</th>
<th>Old</th>
<th>New</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>35.6</td>
<td>15.5</td>
</tr>
<tr>
<td>45</td>
<td>107.0</td>
<td>57.0</td>
</tr>
<tr>
<td>50</td>
<td>224.0</td>
<td>119.0</td>
</tr>
<tr>
<td>55</td>
<td>756.0</td>
<td>271.0</td>
</tr>
<tr>
<td>60</td>
<td>1535.0</td>
<td>655.0</td>
</tr>
<tr>
<td>65</td>
<td>24607.0</td>
<td>3125.0</td>
</tr>
<tr>
<td>70</td>
<td>38818.0</td>
<td>9991.0</td>
</tr>
</tbody>
</table>
In the imaginary case, let $\Delta_n = -4(10^n + 1)$
In the imaginary case, let $\Delta_n = -4(10^n + 1)$

\[
\text{Cl}_{\Delta_{100}} \cong C(2)^7 \times C(1462491779472195274571694315857495335176880879072) \\
\text{Cl}_{\Delta_{110}} \cong C(2)^{11} \times C(8576403641950292891121955131452148838284294200071440)
\]
In the imaginary case, let $\Delta_n = -4(10^n + 1)$

\[
\text{Cl}_{\Delta_{100}} \cong C(2)^7 \times C(146249177947219527457169431585749 \\
5335176880879072)
\]

\[
\text{Cl}_{\Delta_{110}} \cong C(2)^{11} \times C(857640364195029289112195513145214 \\
8838284294200071440)
\]

In the real case, let $\Delta_{110} = 4(10^{110} + 3)$
Heroic computations

In the imaginary case, let $\Delta_n = -4(10^n + 1)$

$$\text{Cl}_{\Delta_{100}} \cong C(2)^7 \times C(1462491779472195274571694315857495335176880879072)$$

$$\text{Cl}_{\Delta_{110}} \cong C(2)^{11} \times C(8576403641950292891121955131452148838284294200071440)$$

In the real case, let $\Delta_{110} = 4(10^{110} + 3)$

$$\text{Cl}_{\Delta_{110}} \cong \mathbb{Z}/12\mathbb{Z} \times \mathbb{Z}/2\mathbb{Z}$$
In the imaginary case, let $\Delta_n = -4(10^n + 1)$

$$\text{Cl}_{\Delta_{100}} \cong C(2)^7 \times C(1462491779472195274571694315857495335176880879072)$$

$$\text{Cl}_{\Delta_{110}} \cong C(2)^{11} \times C(8576403641950292891121955131452148838284294200071440)$$

In the real case, let $\Delta_{110} = 4(10^{110} + 3)$

$$\text{Cl}_{\Delta_{110}} \cong \mathbb{Z}/12\mathbb{Z} \times \mathbb{Z}/2\mathbb{Z}$$

$$R_{\Delta_{110}} \approx 70795074091059722608293227655184666748799878533480399.67302$$
Thank you for your attention