

Algorithms for computing class groups (and some motivation from lattice-based cryptanalysis)

Alice Pellet-Mary

CNRS and Univ. Bordeaux

Partly based on a joint work with Koen de Boer and Benjamin Wesolowski

NuTMiC 2026

Katowice



Number field: $K = \mathbb{Q}[X]/f(X)$ (f irreducible)

degree of $K := \text{degree of } f = \text{dimension of } K \text{ as } \mathbb{Q}\text{-vector space}$

Number field: $K = \mathbb{Q}[X]/f(X)$ (f irreducible)

degree of $K := \text{degree of } f = \text{dimension of } K \text{ as } \mathbb{Q}\text{-vector space}$

Class group: object related to K , denoted $\text{Cl}(K)$

Our objective: given K , compute $\text{Cl}(K)$

Number field: $K = \mathbb{Q}[X]/f(X)$ (f irreducible)

degree of $K := \text{degree of } f = \text{dimension of } K \text{ as } \mathbb{Q}\text{-vector space}$

Class group: object related to K , denoted $\text{Cl}(K)$

Our objective: given K , compute $\text{Cl}(K)$

Other related problems:

- ▶ computing units
- ▶ computing a generator of a principal ideal
- ▶ class-group discrete logarithm problem
- ▶ computing S -units

- ▶ The Castagnos-Laguillaumie (CL) framework [CL15]
 - ▶ class groups over quadratic fields provide groups with hard discrete log but easy in a subgroup

[CL15] Castagnos and Laguillaumie. Linearly homomorphic encryption from DDH. CT-RSA.

- ▶ The Castagnos-Laguillaumie (CL) framework [CL15]
 - ▶ class groups over quadratic fields provide groups with hard discrete log but easy in a subgroup
- ▶ some (old) isogeny-based cryptosystems
 - ▶ CSI-FiSh required computation of class-group of quadratic field with 154-digit discriminant [BKV19]

[BKV19] Beullens, Kleinjung, and Vercauteren. CSI-FiSh: Efficient isogeny based signatures through class group computations. Asiacrypt.

Lattice-based cryptography

- ▶ The Castagnos-Laguillaumie (CL) framework [CL15]
 - ▶ class groups over quadratic fields provide groups with hard discrete log but easy in a subgroup
- ▶ some (old) isogeny-based cryptosystems
 - ▶ CSI-FiSh required computation of class-group of quadratic field with 154-digit discriminant [BKV19]
- ▶ computing mildly short vectors in ideal lattices
 - ▶ [cyclotomic fields] requires solving class-group discrete log [CDW21]
 - ▶ [arbitrary fields] requires class-group discrete log + a short basis of the S -units [PHS19]

[CDW21] Cramer, Ducas, and Wesolowski. Mildly short vectors in cyclotomic ideal lattices [...]. Journal of the ACM.

[PHS19] Pellet-Mary, Hanrot, Stehlé. Approx-SVP in ideal lattices with pre-processing. Eurocrypt.

Lattice-based cryptography

- ▶ The Castagnos-Laguillaumie (CL) framework [CL15]
 - ▶ class groups over quadratic fields provide groups with hard discrete log but easy in a subgroup
- ▶ some (old) isogeny-based cryptosystems
 - ▶ CSI-FiSh required computation of class-group of quadratic field with 154-digit discriminant [BKV19]
- ▶ computing mildly short vectors in ideal lattices
 - ▶ [cyclotomic fields] requires solving class-group discrete log [CDW21]
 - ▶ [arbitrary fields] requires class-group discrete log + a short basis of the S -units [PHS19]
- ▶ cryptanalysis of Hawk signature scheme
 - ▶ computing units in a quaternion algebra would break Hawk

[CME+25] Chevignard, Mureau, Espitau, Pellet-Mary, Pliatsok, Wallet. A reduction from Hawk to the principal ideal problem in a quaternion algebra. Eurocrypt.

Lattice-based cryptography

- ▶ The Castagnos-Laguillaumie (CL) framework [CL15]
 - ▶ class groups over quadratic fields provide groups with hard discrete log but easy in a subgroup
- ▶ some (old) isogeny-based cryptosystems
 - ▶ CSI-FiSh required computation of class-group of quadratic field with 154-digit discriminant [BKV19]

Mostly quadratic fields

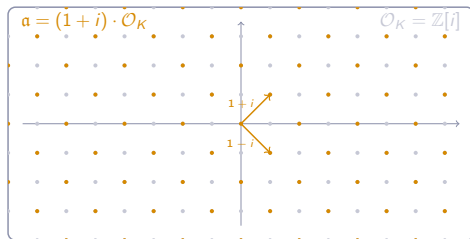
- ▶ computing mildly short vectors in ideal lattices
 - ▶ [cyclotomic fields] requires solving class-group discrete log [CDW21]
 - ▶ [arbitrary fields] requires class-group discrete log + a short basis of the S -units [PHS19]
- ▶ cryptanalysis of Hawk signature scheme
 - ▶ computing units in a quaternion algebra would break Hawk

Mostly fields of high degree (e.g., 512)

[CME+25] Chevignard, Mureau, Espitau, Pellet-Mary, Pliatsok, Wallet. A reduction from Hawk to the principal ideal problem in a quaternion algebra. Eurocrypt.

1. Some definitions and pictures
2. The quantum algorithm
3. The classical heuristic algorithms
4. Removing the heuristics

Some definitions and pictures



For this talk

- ▶ $\zeta \in \mathbb{C}$ primitive $2n$ -th root of unity for some $n = 2^k$
- ▶ $K = \mathbb{Q}(\zeta)$ is a **power-of-two cyclotomic field**
 - ▶ $K = \mathbb{Q}(i) = \{a + bi \mid a, b \in \mathbb{Q}\}$ (for $\zeta = i$)

For this talk

- ▶ $\zeta \in \mathbb{C}$ primitive $2n$ -th root of unity for some $n = 2^k$
- ▶ $K = \mathbb{Q}(\zeta)$ is a **power-of-two cyclotomic field**
 - ▶ $K = \mathbb{Q}(i) = \{a + bi \mid a, b \in \mathbb{Q}\}$ (for $\zeta = i$)
- ▶ $\mathcal{O}_K = \mathbb{Z}[\zeta]$ is the **ring of integers** of K
 - ▶ $\mathcal{O}_K = \mathbb{Z}[i] = \{a + bi \mid a, b \in \mathbb{Z}\}$ (for $\zeta = i$)

For this talk

- ▶ $\zeta \in \mathbb{C}$ primitive $2n$ -th root of unity for some $n = 2^k$
- ▶ $K = \mathbb{Q}(\zeta)$ is a **power-of-two cyclotomic field**
 - ▶ $K = \mathbb{Q}(i) = \{a + bi \mid a, b \in \mathbb{Q}\}$ (for $\zeta = i$)
- ▶ $\mathcal{O}_K = \mathbb{Z}[\zeta]$ is the **ring of integers** of K
 - ▶ $\mathcal{O}_K = \mathbb{Z}[i] = \{a + bi \mid a, b \in \mathbb{Z}\}$ (for $\zeta = i$)
- ▶ n is the **degree** of K (e.g., $n = 512$)
 - ▶ $K = \mathbb{Q} + \mathbb{Q}\zeta + \cdots + \mathbb{Q}\zeta^{n-1}$
 - ▶ $\mathcal{O}_K = \mathbb{Z} + \mathbb{Z}\zeta + \cdots + \mathbb{Z}\zeta^{n-1}$

An **integral ideal** $\mathfrak{a} \subseteq \mathcal{O}_K$ satisfies

- ▶ $x + y \in \mathfrak{a}$ for all $x, y \in \mathfrak{a}$
- ▶ $x \cdot t \in \mathfrak{a}$ for all $x \in \mathfrak{a}$ and $t \in \mathcal{O}_K$

An **integral ideal** $\mathfrak{a} \subseteq \mathcal{O}_K$ satisfies

- ▶ $x + y \in \mathfrak{a}$ for all $x, y \in \mathfrak{a}$
- ▶ $x \cdot t \in \mathfrak{a}$ for all $x \in \mathfrak{a}$ and $t \in \mathcal{O}_K$

Example

$$\mathfrak{a} = \alpha \cdot \mathcal{O}_K = \{\alpha \cdot t \mid t \in \mathcal{O}_K\} \text{ for some } \alpha \in \mathcal{O}_K$$

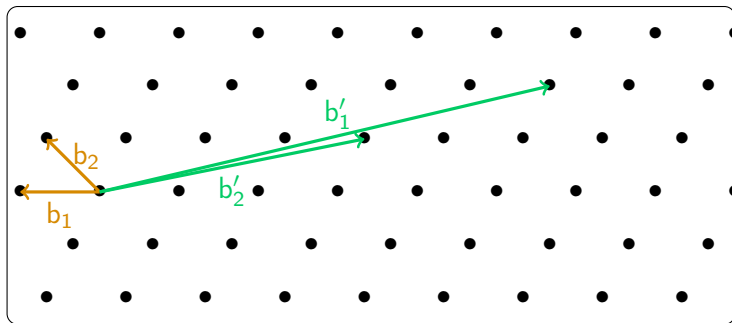
An **integral ideal** $\mathfrak{a} \subseteq \mathcal{O}_K$ satisfies

- ▶ $x + y \in \mathfrak{a}$ for all $x, y \in \mathfrak{a}$
- ▶ $x \cdot t \in \mathfrak{a}$ for all $x \in \mathfrak{a}$ and $t \in \mathcal{O}_K$

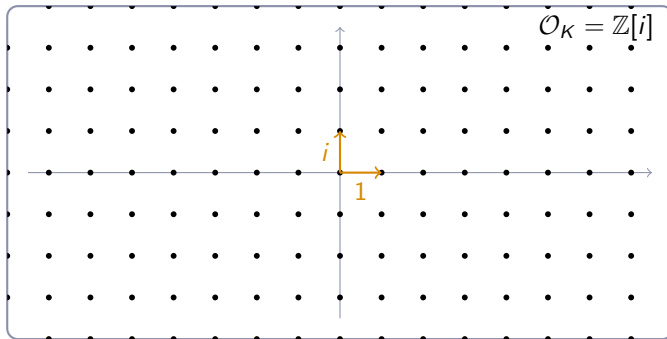
Example

$$\mathfrak{a} = \alpha \cdot \mathcal{O}_K = \{\alpha \cdot t \mid t \in \mathcal{O}_K\} \text{ for some } \alpha \in \mathcal{O}_K$$

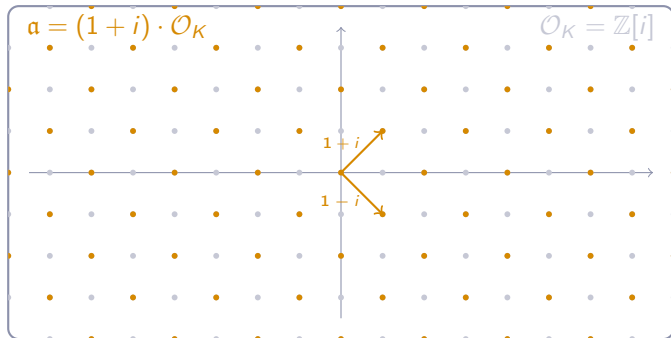
Integral ideals in $\mathcal{O}_K$	Integers in $\mathbb{Z}$
stable by multiplication $\mathfrak{a} \cdot \mathfrak{b}$ is a integral ideal	stable by multiplication $a \cdot b$ is an integer
prime ideals (unique factorization) $\mathfrak{a} = \prod_p \mathfrak{p}^{v_p}$	prime integers (unique factorization) $a = \prod_p p^{v_p}$
fractional ideals $\mathfrak{a} \cdot \mathfrak{b}^{-1}$	rational numbers $a \cdot b^{-1}$



- ▶ $\mathcal{L} = \{\sum_{i=1}^n x_i b_i \mid \forall i, x_i \in \mathbb{Z}\}$ is a **lattice**
- ▶ $(b_1, \dots, b_n) =: B \in \text{GL}_n(\mathbb{R})$ is a **basis**
- ▶ n is the **rank** (or dimension)
- ▶ $\det(\mathcal{L}) := |\det(B)|$ is the **volume**

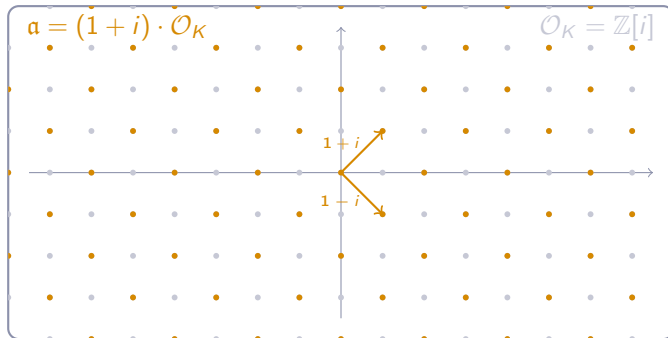


- $\mathcal{O}_K \simeq \mathbb{Z}^n$ is a lattice



- ▶ $\mathcal{O}_K \simeq \mathbb{Z}^n$ is a lattice
- ▶ $\mathfrak{a}$ is a sublattice of $\mathcal{O}_K$

Ideals are lattices



Norm of an ideal

- ▶ $\mathcal{O}_K \simeq \mathbb{Z}^n$ is a lattice
- ▶ $\mathfrak{a}$ is a sublattice of $\mathcal{O}_K$
- ▶ $\mathcal{N}(\mathfrak{a}) = \det(\mathfrak{a})$ (seen as a lattice)
- ▶ $\mathcal{N}(\mathfrak{a} \cdot \mathfrak{b}) = \mathcal{N}(\mathfrak{a}) \cdot \mathcal{N}(\mathfrak{b})$

Principal ideal: $\mathfrak{a} = \alpha \cdot \mathcal{O}_K = \{\alpha \cdot t \mid t \in \mathcal{O}_K\}$ for some $\alpha \in K$
 $\Rightarrow \alpha$ is called a generator of $\mathfrak{a}$

Principal ideal: $\mathfrak{a} = \alpha \cdot \mathcal{O}_K = \{\alpha \cdot t \mid t \in \mathcal{O}_K\}$ for some $\alpha \in K$

$\Rightarrow \alpha$ is called a generator of $\mathfrak{a}$

Class group: $\text{Cl}(K) := \{\text{fractional ideals}\} \text{ modulo } \{\text{principal ideals}\}$

Principal ideal: $\mathfrak{a} = \alpha \cdot \mathcal{O}_K = \{\alpha \cdot t \mid t \in \mathcal{O}_K\}$ for some $\alpha \in K$
 $\Rightarrow \alpha$ is called a generator of $\mathfrak{a}$

Class group: $\text{Cl}(K) := \{\text{fractional ideals}\} \text{ modulo } \{\text{principal ideals}\}$

Principal ideal: $\mathfrak{a} = \alpha \cdot \mathcal{O}_K = \{\alpha \cdot t \mid t \in \mathcal{O}_K\}$ for some $\alpha \in K$
 $\Rightarrow \alpha$ is called a generator of $\mathfrak{a}$

Class group: $\text{Cl}(K) := \{\text{fractional ideals}\} \text{ modulo } \{\text{principal ideals}\}$

Notations: $\mathcal{S}$ is a finite set of prime ideals of $\mathcal{O}_K$

Definition

The $\mathcal{S}$ -class-group lattice* is

$$\begin{aligned}\Lambda_{\mathcal{S}} &:= \left\{ (v_{\mathfrak{p}})_{\mathfrak{p} \in \mathcal{S}} \in \mathbb{Z}^{|\mathcal{S}|} \mid \prod_{\mathfrak{p} \in \mathcal{S}} \mathfrak{p}^{v_{\mathfrak{p}}} \text{ is a principal ideal} \right\} \\ &= \left\{ (v_{\mathfrak{p}})_{\mathfrak{p} \in \mathcal{S}} \in \mathbb{Z}^{|\mathcal{S}|} \mid \exists x \in K \text{ s.t. } \prod_{\mathfrak{p} \in \mathcal{S}} \mathfrak{p}^{v_{\mathfrak{p}}} = x \cdot \mathcal{O}_K \right\}\end{aligned}$$

* this is not a usual name

Notations:

- ▶ $\mathcal{S}$ is a finite set of prime ideals of $\mathcal{O}_K$
- ▶ $\Lambda_{\mathcal{S}} = \left\{ (v_{\mathfrak{p}})_{\mathfrak{p} \in \mathcal{S}} \in \mathbb{Z}^{|\mathcal{S}|} \mid \exists x \in K \text{ s.t. } \prod_{\mathfrak{p} \in \mathcal{S}} \mathfrak{p}^{v_{\mathfrak{p}}} = x \cdot \mathcal{O}_K \right\}$

Notations:

- ▶ $\mathcal{S}$ is a finite set of prime ideals of $\mathcal{O}_K$
- ▶ $\Lambda_{\mathcal{S}} = \left\{ (v_{\mathfrak{p}})_{\mathfrak{p} \in \mathcal{S}} \in \mathbb{Z}^{|\mathcal{S}|} \mid \exists x \in K \text{ s.t. } \prod_{\mathfrak{p} \in \mathcal{S}} \mathfrak{p}^{v_{\mathfrak{p}}} = x \cdot \mathcal{O}_K \right\}$

Properties:

- ▶ $\Lambda_{\mathcal{S}} \subseteq \mathbb{Z}^{|\mathcal{S}|}$ is a lattice of rank $|\mathcal{S}|$

Notations:

- ▶ $\mathcal{S}$ is a finite set of prime ideals of $\mathcal{O}_K$
- ▶ $\Lambda_{\mathcal{S}} = \left\{ (v_{\mathfrak{p}})_{\mathfrak{p} \in \mathcal{S}} \in \mathbb{Z}^{|\mathcal{S}|} \mid \exists x \in K \text{ s.t. } \prod_{\mathfrak{p} \in \mathcal{S}} \mathfrak{p}^{v_{\mathfrak{p}}} = x \cdot \mathcal{O}_K \right\}$

Properties:

- ▶ $\Lambda_{\mathcal{S}} \subseteq \mathbb{Z}^{|\mathcal{S}|}$ is a lattice of rank $|\mathcal{S}|$
- ▶ for well chosen sets $\mathcal{S}$,

knowing $\Lambda_{\mathcal{S}} \Rightarrow$ knowing the class group of K

Notations:

- ▶ $\mathcal{S}$ is a finite set of prime ideals of $\mathcal{O}_K$
- ▶ $\Lambda_{\mathcal{S}} = \left\{ (v_{\mathfrak{p}})_{\mathfrak{p} \in \mathcal{S}} \in \mathbb{Z}^{|\mathcal{S}|} \mid \exists x \in K \text{ s.t. } \prod_{\mathfrak{p} \in \mathcal{S}} \mathfrak{p}^{v_{\mathfrak{p}}} = x \cdot \mathcal{O}_K \right\}$

Properties:

- ▶ $\Lambda_{\mathcal{S}} \subseteq \mathbb{Z}^{|\mathcal{S}|}$ is a lattice of rank $|\mathcal{S}|$
- ▶ for well chosen sets $\mathcal{S}$,

knowing $\Lambda_{\mathcal{S}} \Rightarrow$ knowing the class group of K

Objective

Given $\mathcal{S}$, compute a basis of $\Lambda_{\mathcal{S}}$.

Existing algorithms for computing class groups

Notation: $L_x(\alpha) = \exp(O(\log(x)^\alpha \cdot \log \log(x)^{1-\alpha}))$

Instantiated with power-of-two cyclotomic fields

all algorithms assume ERH

	Number fields	Complexity	Non heuristic
[HM89]	$\mathbb{Q}(i)$	poly	✓

[HM89] Hafner, McCurley. A rigorous subexponential algorithm for computation of class groups. Journal of the American mathematical society.

Existing algorithms for computing class groups

Notation: $L_x(\alpha) = \exp(O(\log(x)^\alpha \cdot \log \log(x)^{1-\alpha}))$

Instantiated with power-of-two cyclotomic fields

all algorithms assume ERH

	Number fields	Complexity	Non heuristic
[HM89]	$\mathbb{Q}(i)$	poly	✓
[Buc88]	fixed degree n	poly	✗

[Buc88] Buchmann. A subexponential algorithm for the determination of class groups and regulators of algebraic number fields.

Séminaire de théorie des nombres.

Existing algorithms for computing class groups

Notation: $L_x(\alpha) = \exp(O(\log(x)^\alpha \cdot \log \log(x)^{1-\alpha}))$

Instantiated with power-of-two cyclotomic fields

all algorithms assume ERH

	Number fields	Complexity	Non heuristic
[HM89]	$\mathbb{Q}(i)$	poly	✓
[Buc88]	fixed degree n	poly	✗
[BF14]	arbitrary degree n	$L_{n^n}(2/3)$	✗

[BF14] Biasse, Fieker. Subexponential class group and unit group computation in large degree number fields. LMS Journal of Computation and Mathematics.

Existing algorithms for computing class groups

Notation: $L_x(\alpha) = \exp(O(\log(x)^\alpha \cdot \log \log(x)^{1-\alpha}))$

Instantiated with power-of-two cyclotomic fields

all algorithms assume ERH

	Number fields	Complexity	Non heuristic
[HM89]	$\mathbb{Q}(i)$	poly	✓
[Buc88]	fixed degree n	poly	✗
[BF14]	arbitrary degree n	$L_{n^n}(2/3)$	✗
[BEF+17]	arbitrary degree n	$L_{n^n}(1/2)$	✗

[BEF+17] Biasse, Espitau, Fouque, Gélín, Kirchner. Computing generator in cyclotomic integer rings. Eurocrypt.

Existing algorithms for computing class groups

Notation: $L_x(\alpha) = \exp(O(\log(x)^\alpha \cdot \log \log(x)^{1-\alpha}))$

Instantiated with power-of-two cyclotomic fields

all algorithms assume ERH

	Number fields	Complexity	Non heuristic
[HM89]	$\mathbb{Q}(i)$	poly	✓
[Buc88]	fixed degree n	poly	✗
[BF14]	arbitrary degree n	$L_{n^n}(2/3)$	✗
[BEF+17]	arbitrary degree n	$L_{n^n}(1/2)$	✗
[BPW25]	arbitrary degree n	$L_{n^n}(2/3)$	✓

[BPW25] de Boer, Pellet-Mary, Wesolowski. Rigorous Methods for Computational Number Theory. ePrint, eprint.iacr.org/2025/1514.

Existing algorithms for computing class groups

Notation: $L_x(\alpha) = \exp(O(\log(x)^\alpha \cdot \log \log(x)^{1-\alpha}))$

Instantiated with power-of-two cyclotomic fields

all algorithms assume ERH

	Number fields	Complexity	Non heuristic
[HM89]	$\mathbb{Q}(i)$	poly	✓
[Buc88]	fixed degree n	poly	✗
[BF14]	arbitrary degree n	$L_{n^n}(2/3)$	✗
[BEF+17]	arbitrary degree n	$L_{n^n}(1/2)$	✗
[BPW25]	arbitrary degree n	$L_{n^n}(2/3)$	✓
[BS16]	arbitrary degree n	quantum poly	✓

[BS16] Biasse and Song. A polynomial time quantum algorithm for computing class groups and solving the principal ideal problem in arbitrary degree number fields. SODA.

The quantum algorithm

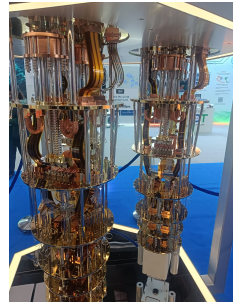


Image by Dev Jadiya — CC BY-SA 4.0

Notation: $\Lambda_S = \left\{ (v_p)_{p \in S} \in \mathbb{Z}^{|S|} \mid \exists x \in K \text{ s.t. } \prod_{p \in S} \mathfrak{p}^{v_p} = x \cdot \mathcal{O}_K \right\}$

$$\begin{aligned} f : \quad \mathbb{Z}^{|S|} &\rightarrow \text{Cl}(K) = \{\text{fractional ideals}\} \text{ modulo } \{\text{principal ideals}\} \\ (v_p)_{p \in S} &\mapsto \left[\prod_{p \in S} \mathfrak{p}^{v_p} \right] = \prod_{p \in S} \mathfrak{p}^{v_p} \text{ mod } \{\text{principal ideals}\} \end{aligned}$$

[BS16] Biasse and Song. A polynomial time quantum algorithm for computing class groups [...]. SODA.

[BF26] de Boer and Felderhoff. Quantumly Computing S-unit Groups in Quantified Polynomial Time and Space. Crypto.

Notation: $\Lambda_S = \left\{ (v_p)_{p \in S} \in \mathbb{Z}^{|S|} \mid \exists x \in K \text{ s.t. } \prod_{p \in S} \mathfrak{p}^{v_p} = x \cdot \mathcal{O}_K \right\}$

$$\begin{aligned} f : \quad \mathbb{Z}^{|S|} &\rightarrow \text{Cl}(K) = \{\text{fractional ideals}\} \text{ modulo } \{\text{principal ideals}\} \\ (v_p)_{p \in S} &\mapsto \left[\prod_{p \in S} \mathfrak{p}^{v_p} \right] = \prod_{p \in S} \mathfrak{p}^{v_p} \text{ mod } \{\text{principal ideals}\} \end{aligned}$$

By definition of Λ_S ,

$$v \in \Lambda_S \iff f(v) = [\mathcal{O}_K] = f(0)$$

[BS16] Biasse and Song. A polynomial time quantum algorithm for computing class groups [...]. SODA.

[BF26] de Boer and Felderhoff. Quantumly Computing S-unit Groups in Quantified Polynomial Time and Space. Crypto.

Notation: $\Lambda_S = \left\{ (v_p)_{p \in S} \in \mathbb{Z}^{|S|} \mid \exists x \in K \text{ s.t. } \prod_{p \in S} \mathfrak{p}^{v_p} = x \cdot \mathcal{O}_K \right\}$

$$f : \quad \mathbb{Z}^{|S|} \rightarrow \text{Cl}(K) = \{\text{fractional ideals}\} \text{ modulo } \{\text{principal ideals}\}$$

$$(v_p)_{p \in S} \mapsto \left[\prod_{p \in S} \mathfrak{p}^{v_p} \right] = \prod_{p \in S} \mathfrak{p}^{v_p} \text{ mod } \{\text{principal ideals}\}$$

By definition of Λ_S ,

$$v \in \Lambda_S \iff f(w + v) = f(w), \forall w \in \mathbb{Z}^{|S|}$$

[BS16] Biasse and Song. A polynomial time quantum algorithm for computing class groups [...]. SODA.

[BF26] de Boer and Felderhoff. Quantumly Computing S-unit Groups in Quantified Polynomial Time and Space. Crypto.

Notation: $\Lambda_S = \left\{ (v_p)_{p \in S} \in \mathbb{Z}^{|S|} \mid \exists x \in K \text{ s.t. } \prod_{p \in S} p^{v_p} = x \cdot \mathcal{O}_K \right\}$

$$\begin{aligned} f : \quad \mathbb{Z}^{|S|} &\rightarrow \text{Cl}(K) = \{\text{fractional ideals}\} \text{ modulo } \{\text{principal ideals}\} \\ (v_p)_{p \in S} &\mapsto \left[\prod_{p \in S} p^{v_p} \right] = \prod_{p \in S} p^{v_p} \text{ mod } \{\text{principal ideals}\} \end{aligned}$$

By definition of Λ_S ,

$$v \in \Lambda_S \iff f(w + v) = f(w), \forall w \in \mathbb{Z}^{|S|}$$

Take-away: f is a **periodic** function with period Λ_S

[BS16] Biasse and Song. A polynomial time quantum algorithm for computing class groups [...]. SODA.

[BF26] de Boer and Felderhoff. Quantumly Computing S-unit Groups in Quantified Polynomial Time and Space. Crypto.

Reminder: $f : \mathbb{Z}^{|\mathcal{S}|} \rightarrow \text{Cl}(K)$ is a **periodic** function with period $\Lambda_{\mathcal{S}}$

Algorithm

[EHKS14] Eisenträger, Hallgren, Kitaev, and Song. A quantum algorithm for computing the unit group [...]. STOC.

[BDF20] de Boer, Ducas, and Fehr. On the Quantum Complexity of the Continuous Hidden Subgroup Problem. Eurocrypt.

Reminder: $f : \mathbb{Z}^{|\mathcal{S}|} \rightarrow \text{Cl}(K)$ is a **periodic** function with period $\Lambda_{\mathcal{S}}$

Algorithm

$$\sum_{\mathbf{v} \in \mathbb{Z}^{|\mathcal{S}|}} f(\mathbf{v}) |\mathbf{v}\rangle$$

[EHKS14] Eisenträger, Hallgren, Kitaev, and Song. A quantum algorithm for computing the unit group [...]. STOC.

[BDF20] de Boer, Ducas, and Fehr. On the Quantum Complexity of the Continuous Hidden Subgroup Problem. Eurocrypt.

Reminder: $f : \mathbb{Z}^{|\mathcal{S}|} \rightarrow \text{Cl}(K)$ is a **periodic** function with period $\Lambda_{\mathcal{S}}$

Algorithm

$$\sum_{v \in \mathbb{Z}^{|\mathcal{S}|}} f(v) |v\rangle \xrightarrow{\text{QFT}} \sum_{w \in \mathbb{Z}^{|\mathcal{S}|}} \hat{f}(w) |w\rangle$$

[EHKS14] Eisenträger, Hallgren, Kitaev, and Song. A quantum algorithm for computing the unit group [...]. STOC.

[BDF20] de Boer, Ducas, and Fehr. On the Quantum Complexity of the Continuous Hidden Subgroup Problem. Eurocrypt.

Reminder: $f : \mathbb{Z}^{|\mathcal{S}|} \rightarrow \text{Cl}(K)$ is a **periodic** function with period $\Lambda_{\mathcal{S}}$

Algorithm

$$\sum_{v \in \mathbb{Z}^{|\mathcal{S}|}} f(v) |v\rangle \xrightarrow{\text{QFT}} \sum_{w \in \mathbb{Z}^{|\mathcal{S}|}} \hat{f}(w) |w\rangle$$

Fact: f periodic $\implies \begin{cases} \hat{f}(w) = 0 & \text{if } w \notin \Lambda_{\mathcal{S}} \\ \hat{f}(w) \neq 0 & \text{if } w \in \Lambda_{\mathcal{S}} \end{cases}$

[EHKS14] Eisenträger, Hallgren, Kitaev, and Song. A quantum algorithm for computing the unit group [...]. STOC.

[BDF20] de Boer, Ducas, and Fehr. On the Quantum Complexity of the Continuous Hidden Subgroup Problem. Eurocrypt.

Reminder: $f : \mathbb{Z}^{|\mathcal{S}|} \rightarrow \text{Cl}(K)$ is a **periodic** function with period $\Lambda_{\mathcal{S}}$

Algorithm

$$\sum_{v \in \mathbb{Z}^{|\mathcal{S}|}} f(v) |v\rangle \xrightarrow{\text{QFT}} \sum_{w \in \mathbb{Z}^{|\mathcal{S}|}} \hat{f}(w) |w\rangle \xrightarrow{\text{measure}} w \in \Lambda_{\mathcal{S}}$$

Fact: f periodic $\implies \begin{cases} \hat{f}(w) = 0 & \text{if } w \notin \Lambda_{\mathcal{S}} \\ \hat{f}(w) \neq 0 & \text{if } w \in \Lambda_{\mathcal{S}} \end{cases}$

[EHKS14] Eisenträger, Hallgren, Kitaev, and Song. A quantum algorithm for computing the unit group [...]. STOC.

[BDF20] de Boer, Ducas, and Fehr. On the Quantum Complexity of the Continuous Hidden Subgroup Problem. Eurocrypt.

Reminder: $f : \mathbb{Z}^{|S|} \rightarrow \text{Cl}(K)$ is a **periodic** function with period Λ_S

Algorithm

$$\sum_{v \in \mathbb{Z}^{|S|}} f(v) |v\rangle \xrightarrow{\text{QFT}} \sum_{w \in \mathbb{Z}^{|S|}} \hat{f}(w) |w\rangle \xrightarrow{\text{measure}} w \in \Lambda_S$$

Fact: f periodic $\implies \begin{cases} \hat{f}(w) = 0 & \text{if } w \notin \Lambda_S \\ \hat{f}(w) \neq 0 & \text{if } w \in \Lambda_S \end{cases}$

Some lie here: need to compute the units at the same time $\Rightarrow$ real coordinates (very annoying)

[EHKS14] Eisenträger, Hallgren, Kitaev, and Song. A quantum algorithm for computing the unit group [...]. STOC.

[BDF20] de Boer, Ducas, and Fehr. On the Quantum Complexity of the Continuous Hidden Subgroup Problem. Eurocrypt.

Reminder: $f : \mathbb{Z}^{|S|} \rightarrow \text{Cl}(K)$ is a **periodic** function with period Λ_S

$$\sum_{v \in \mathbb{Z}^{|S|}} f(v) \quad \begin{array}{c} \text{Algorithm} \\ \text{Quantum poly time algorithm} \end{array} \quad w \in \Lambda_S$$

Fact: f periodic $\implies \begin{cases} \hat{f}(w) = 0 & \text{if } w \notin \Lambda_S \\ \hat{f}(w) \neq 0 & \text{if } w \in \Lambda_S \end{cases}$

Some lie here: need to compute the units at the same time $\Rightarrow$ real coordinates (very annoying)

[EHKS14] Eisenträger, Hallgren, Kitaev, and Song. A quantum algorithm for computing the unit group [...]. STOC.

[BDF20] de Boer, Ducas, and Fehr. On the Quantum Complexity of the Continuous Hidden Subgroup Problem. Eurocrypt.

The classical heuristic algorithms

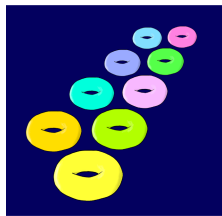


Image by Koen de Boer – PhD thesis

$\mathcal{S}$ -class-group lattice

- ▶ $\mathcal{S}$ a set of prime ideals
- ▶ $\Lambda_{\mathcal{S}} = \left\{ (v_{\mathfrak{p}})_{\mathfrak{p} \in \mathcal{S}} \in \mathbb{Z}^{|\mathcal{S}|} \mid \right.$
 $\left. \exists x \in K \text{ s.t. } \prod_{\mathfrak{p} \in \mathcal{S}} \mathfrak{p}^{v_{\mathfrak{p}}} = x \cdot \mathcal{O}_K \right\}$

$\mathcal{S}$ -class-group lattice

- ▶ $\mathcal{S}$ a set of prime ideals
- ▶ $\Lambda_{\mathcal{S}} = \left\{ (v_{\mathfrak{p}})_{\mathfrak{p} \in \mathcal{S}} \in \mathbb{Z}^{|\mathcal{S}|} \mid \right.$
 $\left. \exists x \in K \text{ s.t. } \prod_{\mathfrak{p} \in \mathcal{S}} \mathfrak{p}^{v_{\mathfrak{p}}} = x \cdot \mathcal{O}_K \right\}$

Analogy with factoring

- ▶ $\mathcal{S}$ a set of prime integers
- ▶ $\Lambda_{\mathcal{S}} = \left\{ (v_p)_{p \in \mathcal{S}} \in \mathbb{Z}^{|\mathcal{S}|} \mid \right.$
 $\left. \prod_{p \in \mathcal{S}} p^{v_p} = 1 \bmod N \right\}$

$\mathcal{S}$ -class-group lattice

- ▶ $\mathcal{S}$ a set of prime ideals
- ▶ $\Lambda_{\mathcal{S}} = \left\{ (v_{\mathfrak{p}})_{\mathfrak{p} \in \mathcal{S}} \in \mathbb{Z}^{|\mathcal{S}|} \mid \right.$
 $\left. \exists x \in K \text{ s.t. } \prod_{\mathfrak{p} \in \mathcal{S}} \mathfrak{p}^{v_{\mathfrak{p}}} = x \cdot \mathcal{O}_K \right\}$

Analogy with factoring

- ▶ $\mathcal{S}$ a set of prime integers
- ▶ $\Lambda_{\mathcal{S}} = \left\{ (v_p)_{p \in \mathcal{S}} \in \mathbb{Z}^{|\mathcal{S}|} \mid \right.$
 $\left. \prod_{p \in \mathcal{S}} p^{v_p} = 1 \bmod N \right\}$

Strategy:

- ▶ compute one “relation”
- ▶ repeat until you have a full rank system (and then solve with linear algebra)

Computing a vector of $\Lambda_{\mathcal{S}}$ – main idea

Reminder: $\Lambda_{\mathcal{S}} = \left\{ (v_{\mathfrak{p}})_{\mathfrak{p} \in \mathcal{S}} \in \mathbb{Z}^{|\mathcal{S}|} \mid \exists x \in K \text{ s.t. } \prod_{\mathfrak{p} \in \mathcal{S}} \mathfrak{p}^{v_{\mathfrak{p}}} = x \cdot \mathcal{O}_K \right\}$

Definition: $\mathcal{S} = \{\text{prime } \mathfrak{p} \mid \mathcal{N}(\mathfrak{p}) \leq B\}$ (for some B to be determined)

Computing a vector of $\Lambda_{\mathcal{S}}$ – main idea

Reminder: $\Lambda_{\mathcal{S}} = \left\{ (v_{\mathfrak{p}})_{\mathfrak{p} \in \mathcal{S}} \in \mathbb{Z}^{|\mathcal{S}|} \mid \exists x \in K \text{ s.t. } \prod_{\mathfrak{p} \in \mathcal{S}} \mathfrak{p}^{v_{\mathfrak{p}}} = x \cdot \mathcal{O}_K \right\}$

Definition: $\mathcal{S} = \{\text{prime } \mathfrak{p} \mid \mathcal{N}(\mathfrak{p}) \leq B\}$ (for some B to be determined)

Objective: compute a vector of $\Lambda_{\mathcal{S}}$

Computing a vector of Λ_S – main idea

Reminder: $\Lambda_S = \left\{ (v_p)_{p \in S} \in \mathbb{Z}^{|S|} \mid \exists x \in K \text{ s.t. } \prod_{p \in S} p^{v_p} = x \cdot \mathcal{O}_K \right\}$

Definition: $S = \{\text{prime } p \mid \mathcal{N}(p) \leq B\}$ (for some B to be determined)

Objective: find $(v_p)_{p \in S} \in \mathbb{Z}^{|S|}$ and $x \in K$ such that $\prod_{p \in S} p^{v_p} = x \cdot \mathcal{O}_K$

Computing a vector of Λ_S – main idea

Reminder: $\Lambda_S = \left\{ (v_p)_{p \in S} \in \mathbb{Z}^{|S|} \mid \exists x \in K \text{ s.t. } \prod_{p \in S} p^{v_p} = x \cdot \mathcal{O}_K \right\}$

Definition: $S = \{\text{prime } p \mid \mathcal{N}(p) \leq B\}$ (for some B to be determined)

Objective: find $(v_p)_{p \in S} \in \mathbb{Z}^{|S|}$ and $x \in K$ such that $\prod_{p \in S} p^{v_p} = x \cdot \mathcal{O}_K$

Two possible strategies:

1. try random $(v_p)_{p \in S} \in \mathbb{Z}^{|S|}$ until $\prod_{p \in S} p^{v_p} = x \cdot \mathcal{O}_K$ for some $x \in K$
2. try random $x \in \mathcal{O}_K$ until $x \cdot \mathcal{O}_K = \prod_{p \in S} p^{v_p}$ for some $(v_p)_{p \in S} \in \mathbb{Z}^{|S|}$

Computing a vector of Λ_S – main idea

Reminder: $\Lambda_S = \left\{ (v_p)_{p \in S} \in \mathbb{Z}^{|S|} \mid \exists x \in K \text{ s.t. } \prod_{p \in S} p^{v_p} = x \cdot \mathcal{O}_K \right\}$

Definition: $S = \{\text{prime } p \mid \mathcal{N}(p) \leq B\}$ (for some B to be determined)

Objective: find $(v_p)_{p \in S} \in \mathbb{Z}^{|S|}$ and $x \in K$ such that $\prod_{p \in S} p^{v_p} = x \cdot \mathcal{O}_K$

Two possible strategies:

1. try random $(v_p)_{p \in S} \in \mathbb{Z}^{|S|}$ until $\prod_{p \in S} p^{v_p} = x \cdot \mathcal{O}_K$ for some $x \in K$
2. try random $x \in \mathcal{O}_K$ until $x \cdot \mathcal{O}_K = \prod_{p \in S} p^{v_p}$ for some $(v_p)_{p \in S} \in \mathbb{Z}^{|S|}$

Remark:

- ▶ testing principality is harder than testing B -smoothness
- ▶ density of smooth ideals $\geq$ density of principal ideals
(for cyclotomic fields, if B is large enough)

Computing a vector of Λ_S – main idea

Reminder: $\Lambda_S = \left\{ (v_p)_{p \in S} \in \mathbb{Z}^{|S|} \mid \exists x \in K \text{ s.t. } \prod_{p \in S} p^{v_p} = x \cdot \mathcal{O}_K \right\}$

Definition: $S = \{\text{prime } p \mid \mathcal{N}(p) \leq B\}$ (for some B to be determined)

Objective: find $(v_p)_{p \in S} \in \mathbb{Z}^{|S|}$ and $x \in K$ such that $\prod_{p \in S} p^{v_p} = x \cdot \mathcal{O}_K$

Two possible strategies:

1. try random $(v_p)_{p \in S} \in \mathbb{Z}^{|S|}$ until $\prod_{p \in S} p^{v_p} = x \cdot \mathcal{O}_K$ for some $x \in K$
2. try random $x \in \mathcal{O}_K$ until $x \cdot \mathcal{O}_K = \prod_{p \in S} p^{v_p}$ for some $(v_p)_{p \in S} \in \mathbb{Z}^{|S|}$

Remark:

- ▶ testing principality is harder than testing B -smoothness
- ▶ density of smooth ideals $\geq$ density of principal ideals
(for cyclotomic fields, if B is large enough)

Reminder: $\mathcal{S} = \{\text{prime } \mathfrak{p} \mid \mathcal{N}(\mathfrak{p}) \leq B\}$ (for some B to be determined)

Reminder: $\mathcal{S} = \{\text{prime } \mathfrak{p} \mid \mathcal{N}(\mathfrak{p}) \leq B\}$ (for some B to be determined)

Algorithm SampleVector

- 1: **repeat**
 - 2: Sample random $x \in \mathcal{O}_K$
 - 3: **until** $x\mathcal{O}_K = \prod_{\mathfrak{p} \in \mathcal{S}} \mathfrak{p}^{v_{\mathfrak{p}}}$ for some $v_{\mathfrak{p}} \in \mathbb{Z}$
 - 4: **return** $v := (v_{\mathfrak{p}})_{\mathfrak{p}}$
-

Reminder: $S = \{\text{prime } \mathfrak{p} \mid \mathcal{N}(\mathfrak{p}) \leq B\}$ (for some B to be determined)

Algorithm SampleVector

- 1: **repeat**
 - 2: Sample random $x \in \mathcal{O}_K$
 - 3: **until** $x\mathcal{O}_K = \prod_{\mathfrak{p} \in S} \mathfrak{p}^{v_{\mathfrak{p}}}$ for some $v_{\mathfrak{p}} \in \mathbb{Z}$
 - 4: **return** $v := (v_{\mathfrak{p}})_{\mathfrak{p}}$
-

Correctness: ✓ ($v \in \Lambda_S$)

Reminder: $\mathcal{S} = \{\text{prime } \mathfrak{p} \mid \mathcal{N}(\mathfrak{p}) \leq B\}$ (for some B to be determined)

Algorithm SampleVector

- 1: **repeat**
 - 2: Sample random $x \in \mathcal{O}_K$
 - 3: **until** $x\mathcal{O}_K = \prod_{\mathfrak{p} \in \mathcal{S}} \mathfrak{p}^{v_{\mathfrak{p}}}$ for some $v_{\mathfrak{p}} \in \mathbb{Z}$
 - 4: **return** $v := (v_{\mathfrak{p}})_{\mathfrak{p}}$
-

Correctness: ✓ ($v \in \Lambda_S$)

Complexity: $O\left((T_{\text{sample}} + |\mathcal{S}|) \cdot p_{\text{smooth}}^{-1}\right)$

- ▶ T_{sample} : time to sample x
- ▶ $|\mathcal{S}| = O(B)$: time to test smoothness
- ▶ p_{smooth} : probability that $x\mathcal{O}_K$ is smooth

Reminder: $\mathcal{S} = \{\text{prime } \mathfrak{p} \mid \mathcal{N}(\mathfrak{p}) \leq B\}$ (for some B to be determined)

Algorithm SampleVector

- 1: **repeat**
 - 2: Sample random $x \in \mathcal{O}_K$
 - 3: **until** $x\mathcal{O}_K = \prod_{\mathfrak{p} \in \mathcal{S}} \mathfrak{p}^{v_{\mathfrak{p}}}$ for some $v_{\mathfrak{p}} \in \mathbb{Z}$
 - 4: **return** $v := (v_{\mathfrak{p}})_{\mathfrak{p}}$
-

Correctness: ✓ ($v \in \Lambda_S$)

Complexity: $O\left((T_{\text{sample}} + |\mathcal{S}|) \cdot p_{\text{smooth}}^{-1}\right)$

- ▶ T_{sample} : time to sample x
- ▶ $|\mathcal{S}| = O(B)$: time to test smoothness
- ▶ p_{smooth} : probability that $x\mathcal{O}_K$ is smooth

BKZ algorithm

- ▶ **parameter:** blocksize $\beta \in \{2, \dots, n\}$
- ▶ **output:** $x \in \mathcal{O}_K$ with $\mathcal{N}(x \cdot \mathcal{O}_K) \leq n^{n^2/\beta}$
- ▶ **running time:** $2^{O(\beta)}$

BKZ algorithm

- ▶ **parameter:** blocksize $\beta \in \{2, \dots, n\}$
- ▶ **output:** $x \in \mathcal{O}_K$ with $\mathcal{N}(x \cdot \mathcal{O}_K) \leq n^{n^2/\beta}$
- ▶ **running time:** $2^{O(\beta)}$

Buchmann

- ▶ $\beta = n$
- ▶ $\mathcal{N}(x \cdot \mathcal{O}_K) \leq n^n$
- ▶ time $2^{O(n)}$ (exponential time)

[Buc88] Buchmann. A subexponential algorithm for the determination of class groups and regulators of algebraic number fields.

Séminaire de théorie des nombres.

How to sample x ?

BKZ algorithm

- ▶ **parameter:** blocksize $\beta \in \{2, \dots, n\}$
- ▶ **output:** $x \in \mathcal{O}_K$ with $\mathcal{N}(x \cdot \mathcal{O}_K) \leq n^{n^2/\beta}$
- ▶ **running time:** $2^{O(\beta)}$

Buchmann

- ▶ $\beta = n$
- ▶ $\mathcal{N}(x \cdot \mathcal{O}_K) \leq n^n$
- ▶ time $2^{O(n)}$ (exponential time)

Biasse-Fieker

- ▶ $\beta = n^{2/3}$
- ▶ $\mathcal{N}(x \cdot \mathcal{O}_K) \leq n^{n^{4/3}}$
- ▶ time $2^{O(n^{2/3})}$ (sub-exponential time)

[BF14] Biasse, Fieker. Subexponential class group and unit group computation in large degree number fields. LMS Journal of Computation and Mathematics.

$$\mathcal{S} := \{\text{prime } \mathfrak{p} \mid \mathcal{N}(\mathfrak{p}) \leq B\}$$

$$\beta \in \{1, \dots, n\}$$

Algorithm SampleVector

- 1: **repeat**
 - 2: Sample random $x \in \mathcal{O}_K$ using BKZ_β // $\mathcal{N}(x \cdot \mathcal{O}_K) \leq n^{n^2/\beta}$
 - 3: **until** $x\mathcal{O}_K = \prod_{\mathfrak{p} \in \mathcal{S}} \mathfrak{p}^{v_{\mathfrak{p}}}$ for some $v_{\mathfrak{p}} \in \mathbb{Z}$
 - 4: **return** $v := (v_{\mathfrak{p}})_{\mathfrak{p}}$
-

$$\mathcal{S} := \{\text{prime } \mathfrak{p} \mid \mathcal{N}(\mathfrak{p}) \leq B\}$$

$$\beta \in \{1, \dots, n\}$$

Algorithm SampleVector

- 1: **repeat**
 - 2: Sample random $x \in \mathcal{O}_K$ using BKZ_β // $\mathcal{N}(x \cdot \mathcal{O}_K) \leq n^{n^2/\beta}$
 - 3: **until** $x\mathcal{O}_K = \prod_{\mathfrak{p} \in \mathcal{S}} \mathfrak{p}^{v_{\mathfrak{p}}}$ for some $v_{\mathfrak{p}} \in \mathbb{Z}$
 - 4: **return** $v := (v_{\mathfrak{p}})_{\mathfrak{p}}$
-

Complexity:

$$O\left((T_{\text{sample}} + |\mathcal{S}|) \cdot p_{\text{smooth}}^{-1}\right) = (2^{O(\beta)} + B) \cdot p_{\text{smooth}}^{-1}$$

$$\mathcal{S} := \{\text{prime } \mathfrak{p} \mid \mathcal{N}(\mathfrak{p}) \leq B\}$$

$$\beta \in \{1, \dots, n\}$$

Algorithm SampleVector

- 1: **repeat**
 - 2: Sample random $x \in \mathcal{O}_K$ using BKZ_β // $\mathcal{N}(x \cdot \mathcal{O}_K) \leq n^{n^2/\beta}$
 - 3: **until** $x\mathcal{O}_K = \prod_{\mathfrak{p} \in \mathcal{S}} \mathfrak{p}^{v_{\mathfrak{p}}}$ for some $v_{\mathfrak{p}} \in \mathbb{Z}$
 - 4: **return** $v := (v_{\mathfrak{p}})_{\mathfrak{p}}$
-

Complexity:

$$O\left((T_{\text{sample}} + |\mathcal{S}|) \cdot p_{\text{smooth}}^{-1}\right) = (2^{O(\beta)} + B) \cdot p_{\text{smooth}}^{-1}$$

Notation: $p_{y,B}$ = proba that a uniformly random ideal of norm $\leq y$ is B -smooth

Notation: $p_{y,B}$ = proba that a uniformly random ideal of norm $\leq y$ is B -smooth

Asymptotically: $p_{y,B} \sim \rho(u) \approx u^{-u}$ (ρ Dickman function, $u = \log y / \log B$)

Notation: $p_{y,B}$ = proba that a uniformly random ideal of norm $\leq y$ is B -smooth

Asymptotically: $p_{y,B} \sim \rho(u) \approx u^{-u}$ (ρ Dickman function, $u = \log y / \log B$)

⚠ K fixed and B, u tending to infinity

Notation: $p_{y,B}$ = proba that a uniformly random ideal of norm $\leq y$ is B -smooth

Asymptotically: $p_{y,B} \sim \rho(u) \approx u^{-u}$ (ρ Dickman function, $u = \log y / \log B$)

⚠ K fixed and B, u tending to infinity

Heuristic: $p_{\text{smooth}} \approx u^{-u}$ where $u = \frac{\log \mathcal{N}(x \cdot \mathcal{O}_K)}{\log B}$

Notation: $p_{y,B}$ = proba that a uniformly random ideal of norm $\leq y$ is B -smooth

Asymptotically: $p_{y,B} \sim \rho(u) \approx u^{-u}$ (ρ Dickman function, $u = \log y / \log B$)

⚠ K fixed and B, u tending to infinity

Heuristic: $p_{\text{smooth}} \approx u^{-u}$ where $u = \frac{\log \mathcal{N}(x \cdot \mathcal{O}_K)}{\log B}$

2 assumptions hidden:

- ▶ the provable asymptotic bounds require **huge** B to be effective (roughly $B \gtrsim 2^{2^n}$)
- ▶ $x \cdot \mathcal{O}_K$ is **not** a random ideal of bounded norm

Algorithm SampleVector

- 1: **repeat**
 - 2: Sample random $x \in \mathcal{O}_K$ using BKZ_β // $\mathcal{N}(x \cdot \mathcal{O}_K) \leq n^{n^2/\beta}$
 - 3: **until** $x\mathcal{O}_K = \prod_{\mathfrak{p} \in \mathcal{S}} \mathfrak{p}^{v_{\mathfrak{p}}}$ for some $v_{\mathfrak{p}} \in \mathbb{Z}$
 - 4: **return** $v := (v_{\mathfrak{p}})_{\mathfrak{p}}$
-

Complexity (heuristic): $(2^{O(\beta)} + B) \cdot u^u$ with $u = \frac{\log \mathcal{N}(x \cdot \mathcal{O}_K)}{\log B} \leq \frac{n^2 \log n}{\beta \cdot \log B}$

Algorithm SampleVector

- 1: **repeat**
 - 2: Sample random $x \in \mathcal{O}_K$ using BKZ_β // $\mathcal{N}(x \cdot \mathcal{O}_K) \leq n^{n^2/\beta}$
 - 3: **until** $x\mathcal{O}_K = \prod_{\mathfrak{p} \in \mathcal{S}} \mathfrak{p}^{v_{\mathfrak{p}}}$ for some $v_{\mathfrak{p}} \in \mathbb{Z}$
 - 4: **return** $v := (v_{\mathfrak{p}})_{\mathfrak{p}}$
-

Complexity (heuristic): $(2^{O(\beta)} + B) \cdot u^u$ with $u = \frac{\log \mathcal{N}(x \cdot \mathcal{O}_K)}{\log B} \leq \frac{n^2 \log n}{\beta \cdot \log B}$

Instantiating:

- ▶ $\beta = n^{2/3}$ (Biasse-Fieker)
- ▶ $B = L_{n^n}(2/3)$

Total complexity: $L_{n^n}(2/3)$

Fact: $\mathcal{L}_1 \subseteq \mathcal{L}_2 \quad + \quad \text{rank}(\mathcal{L}_1) = \text{rank}(\mathcal{L}_2) \quad + \quad \det(\mathcal{L}_1) = \det(\mathcal{L}_2) \implies \mathcal{L}_1 = \mathcal{L}_2$

Fact: $\mathcal{L}_1 \subseteq \mathcal{L}_2 \quad + \quad \text{rank}(\mathcal{L}_1) = \text{rank}(\mathcal{L}_2) \quad + \quad \det(\mathcal{L}_1) = \det(\mathcal{L}_2) \implies \mathcal{L}_1 = \mathcal{L}_2$

For $\Lambda_{\mathcal{S}}$:

- ▶ $\text{rank}(\Lambda_{\mathcal{S}}) = |\mathcal{S}|$ (known)
- ▶ assume that we know $\det(\Lambda_{\mathcal{S}})$

Fact: $\mathcal{L}_1 \subseteq \mathcal{L}_2 \quad + \quad \text{rank}(\mathcal{L}_1) = \text{rank}(\mathcal{L}_2) \quad + \quad \det(\mathcal{L}_1) = \det(\mathcal{L}_2) \implies \mathcal{L}_1 = \mathcal{L}_2$

For $\Lambda_{\mathcal{S}}$:

- ▶ $\text{rank}(\Lambda_{\mathcal{S}}) = |\mathcal{S}|$ (known)
- ▶ assume that we know $\det(\Lambda_{\mathcal{S}})$

Algorithm ComputeClassGroup

```
1: repeat  
2:    $z \leftarrow \text{SampleVector}()$   
3:   Compute a basis  $C$  of  $\mathcal{L}$  (lattice generated by all the  $z$  sampled so far)  
4: until  $\mathcal{L}$  is a lattice with rank  $|\mathcal{S}|$  and determinant  $\det(\Lambda_{\mathcal{S}})$   
5: return  $C$ 
```

Fact: $\mathcal{L}_1 \subseteq \mathcal{L}_2 \quad + \quad \text{rank}(\mathcal{L}_1) = \text{rank}(\mathcal{L}_2) \quad + \quad \det(\mathcal{L}_1) = \det(\mathcal{L}_2) \implies \mathcal{L}_1 = \mathcal{L}_2$

For $\Lambda_{\mathcal{S}}$:

- ▶ $\text{rank}(\Lambda_{\mathcal{S}}) = |\mathcal{S}|$ (known)
- ▶ assume that we know $\det(\Lambda_{\mathcal{S}})$

Algorithm ComputeClassGroup

```
1: repeat  
2:    $z \leftarrow \text{SampleVector}()$   
3:   Compute a basis  $C$  of  $\mathcal{L}$  (lattice generated by all the  $z$  sampled so far)  
4: until  $\mathcal{L}$  is a lattice with rank  $|\mathcal{S}|$  and determinant  $\det(\Lambda_{\mathcal{S}})$   
5: return  $C$ 
```

Heuristic: $O(B)$ vectors generate $\Lambda_{\mathcal{S}}$ with good probability $(\text{rk}(\Lambda_{\mathcal{S}}) = O(B))$

Fact: $\mathcal{L}_1 \subseteq \mathcal{L}_2 \quad + \quad \text{rank}(\mathcal{L}_1) = \text{rank}(\mathcal{L}_2) \quad + \quad \det(\mathcal{L}_1) = \det(\mathcal{L}_2) \implies \mathcal{L}_1 = \mathcal{L}_2$

For Λ_S :

- ▶ $\text{rank}(\Lambda_S) = |\mathcal{S}|$ (known)
- ▶ assume that we know $\det(\Lambda_S)$

Algorithm ComputeClassGroup

```

1: repeat
2:    $z \leftarrow \text{SampleVector}(\Lambda_S)$ 
3:   Compute a basis  $C$  of  $\langle z \rangle$  (sampled so far)
4: until  $\mathcal{L}$  is a lattice with  $\det(\mathcal{L}) = \det(\Lambda_S)$ 
5: return  $C$ 
    
```

Heuristic algorithm with
complexity $L_{n^n}(2/3)$

Heuristic: $O(B)$ vectors generate Λ_S with good probability ($\text{rk}(\Lambda_S) = O(B)$)

Removing the heuristics

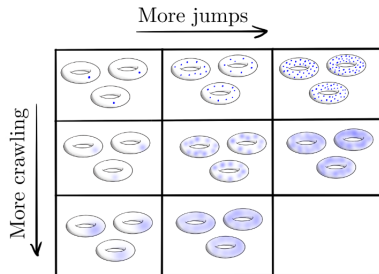


Image by Koen de Boer – PhD thesis

Heuristic 1: $p_{\text{smooth}} \approx u^{-u}$ where $u = \frac{\log \mathcal{N}(x \cdot \mathcal{O}_K)}{\log B}$

Heuristic 1: $p_{\text{smooth}} \approx u^{-u}$ where $u = \frac{\log \mathcal{N}(x \cdot \mathcal{O}_K)}{\log B}$

1.1. the asymptotic bound holds even for smallish B 's

1.2. $x \cdot \mathcal{O}_K$ behaves like a uniform ideal of bounded norm

Heuristic 1: $p_{\text{smooth}} \approx u^{-u}$ where $u = \frac{\log \mathcal{N}(x \cdot \mathcal{O}_K)}{\log B}$

1.1. the asymptotic bound holds even for smallish B 's

1.2. $x \cdot \mathcal{O}_K$ behaves like a uniform ideal of bounded norm

Heuristic 2: $O(B)$ vectors generate the full lattice Λ_S with good probability

Heuristic 1: $p_{\text{smooth}} \approx u^{-u}$ where $u = \frac{\log \mathcal{N}(x \cdot \mathcal{O}_K)}{\log B}$

1.1. the asymptotic bound holds even for smallish B 's

1.2. $x \cdot \mathcal{O}_K$ behaves like a uniform ideal of bounded norm

Heuristic 2: $O(B)$ vectors generate the full lattice Λ_S with good probability

Theorem (ERH, informal)

There exists an algorithm `ProvableSample` that

- ▶ runs in time $2^{O(\beta)}$ ($\beta \in \{2, \dots, n\}$ some parameter)
- ▶ outputs $x \in \mathcal{O}_K$ such that

$$x \cdot \mathcal{O}_K = \mathfrak{a} \cdot \prod_{\mathfrak{p} \in S} \mathfrak{p}^{v_{\mathfrak{p}}},$$

where $\mathfrak{a}$ is a **uniform** integral ideal of norm $\mathcal{N}(\mathfrak{a}) \leq n^{n^2/\beta}$
 (and $S = \{\mathfrak{p} \text{ prime ideal} \mid \mathcal{N}(\mathfrak{p}) \leq B\}$ for some $B = \text{poly}(n)$)

Theorem (ERH, informal)

There exists an algorithm `ProvableSample` that

- ▶ runs in time $2^{O(\beta)}$ ($\beta \in \{2, \dots, n\}$ some parameter)
- ▶ outputs $x \in \mathcal{O}_K$ such that

$$x \cdot \mathcal{O}_K = \mathfrak{a} \cdot \prod_{\mathfrak{p} \in \mathcal{S}} \mathfrak{p}^{v_{\mathfrak{p}}},$$

where $\mathfrak{a}$ is a **uniform** integral ideal of norm $\mathcal{N}(\mathfrak{a}) \leq n^{n^2/\beta}$
 (and $\mathcal{S} = \{\mathfrak{p} \text{ prime ideal} \mid \mathcal{N}(\mathfrak{p}) \leq B\}$ for some $B = \text{poly}(n)$)

Applications: ▶ $\Pr(x \cdot \mathcal{O}_K \text{ is } B\text{-smooth}) = \Pr(\mathfrak{a} \text{ is } B\text{-smooth})$ for a **uniform** $\mathfrak{a}$

Theorem (ERH, informal)

There exists an algorithm `ProvableSample` that

- ▶ runs in time $2^{O(\beta)}$ ($\beta \in \{2, \dots, n\}$ some parameter)
- ▶ outputs $x \in \mathcal{O}_K$ such that

$$x \cdot \mathcal{O}_K = \mathfrak{a} \cdot \prod_{\mathfrak{p} \in \mathcal{S}} \mathfrak{p}^{v_{\mathfrak{p}}},$$

where $\mathfrak{a}$ is a **uniform** integral ideal of norm $\mathcal{N}(\mathfrak{a}) \leq n^{n^2/\beta}$
 (and $\mathcal{S} = \{\mathfrak{p} \text{ prime ideal} \mid \mathcal{N}(\mathfrak{p}) \leq B\}$ for some $B = \text{poly}(n)$)

- Applications:**
- ▶ $\Pr(x \cdot \mathcal{O}_K \text{ is } B\text{-smooth}) = \Pr(\mathfrak{a} \text{ is } B\text{-smooth})$ for a **uniform** $\mathfrak{a}$
 - ▶ $\Pr(x \cdot \mathcal{O}_K \text{ is near-prime}) = \Pr(\mathfrak{a} \text{ is near-prime})$ for a **uniform** $\mathfrak{a}$

Algorithm SampleVector

- 1: **repeat**
 - 2: Sample random $x \in \mathcal{O}_K$ using BKZ_β // $\mathcal{N}(x \cdot \mathcal{O}_K) \leq n^{n^2/\beta}$
 - 3: **until** $x\mathcal{O}_K = \prod_{\mathfrak{p} \in \mathcal{S}} \mathfrak{p}^{v_{\mathfrak{p}}}$ for some $v_{\mathfrak{p}} \in \mathbb{Z}$
 - 4: **return** $v := (v_{\mathfrak{p}})_{\mathfrak{p}}$
-

Heuristic 1.2: $x \cdot \mathcal{O}_K$ behaves like a uniform ideal of bounded norm

Algorithm SampleVector

- 1: **repeat**
 - 2: Sample random $x \in \mathcal{O}_K$ using **ProvableSample** $_\beta$
 - 3: **until** $x\mathcal{O}_K = \prod_{\mathfrak{p} \in \mathcal{S}} \mathfrak{p}^{v_{\mathfrak{p}}}$ for some $v_{\mathfrak{p}} \in \mathbb{Z}$
 - 4: **return** $v := (v_{\mathfrak{p}})_{\mathfrak{p}}$
-

Heuristic 1.2: $x \cdot \mathcal{O}_K$ behaves like a uniform ideal of bounded norm

Algorithm SampleVector

- 1: **repeat**
 - 2: Sample random $x \in \mathcal{O}_K$ using $\text{ProvableSample}_\beta$
 - 3: **until** $x\mathcal{O}_K = \prod_{\mathfrak{p} \in \mathcal{S}} \mathfrak{p}^{v_{\mathfrak{p}}}$ for some $v_{\mathfrak{p}} \in \mathbb{Z}$
 - 4: **return** $v := (v_{\mathfrak{p}})_{\mathfrak{p}}$
-

Heuristic 1.2: $x \cdot \mathcal{O}_K$ behaves like a uniform ideal of bounded norm

Theorem [ERH]: $\Pr(x \cdot \mathcal{O}_K \text{ is } B\text{-smooth}) = p_{y,B}$

where

► $y = n^{n^2/\beta}$

► $p_{y,B}$ = probability that a uniform ideal of norm $\leq y$ is B -smooth

Heuristic 1.1: $p_{y,B} \approx u^{-u}$ even for smallish B 's where $u = \frac{\log y}{\log B}$

Conjecture 1.1: $p_{y,B} \approx u^{-u}$ even for smallish B 's where $u = \frac{\log y}{\log B}$

Conjecture 1.1: $p_{y,B} \approx u^{-u}$ even for smallish B 's where $u = \frac{\log y}{\log B}$

For cyclotomic fields: proven for $B \geq \Omega(n^4)$ ✓

A few words about Heuristic 1.1

Conjecture 1.1: $p_{y,B} \approx u^{-u}$ even for smallish B 's where $u = \frac{\log y}{\log B}$

For cyclotomic fields: proven for $B \geq \Omega(n^4)$ ✓

For arbitrary fields: best proven lower bound so far is $p_{y,B} \gtrsim u^{-u} \cdot \rho_K^{-1}$ ✗
(ρ_K residue at one of the zeta function of K)

Conclusion

All the same: computing class group, units, S -units, ...

All the same: computing class group, units, S -units, ...

Quantum algorithm: polynomial classical algorithm: $\rho_K \cdot (L_{\Delta_K}(1/2) + L_{n^n}(2/3))$

All the same: computing class group, units, S -units, ...

Quantum algorithm: polynomial classical algorithm: $\rho_K \cdot (L_{\Delta_K}(1/2) + L_{n^n}(2/3))$

Some open problems:

- ▶ from number fields to quaternion algebras
probably hard, would break Hawk signature scheme

All the same: computing class group, units, S -units, ...

Quantum algorithm: polynomial classical algorithm: $\rho_K \cdot (L_{\Delta_K}(1/2) + L_{n^n}(2/3))$

Some open problems:

- ▶ from number fields to quaternion algebras
probably hard, would break Hawk signature scheme
- ▶ is there really a ρ_K in the complexity?
numerical experiments? better provable smoothness bound?

All the same: computing class group, units, S -units, ...

Quantum algorithm: polynomial classical algorithm: $\rho_K \cdot (L_{\Delta_K}(1/2) + L_{n^n}(2/3))$

Some open problems:

- ▶ from number fields to quaternion algebras
probably hard, would break Hawk signature scheme
- ▶ is there really a ρ_K in the complexity?
numerical experiments? better provable smoothness bound?
- ▶ better algorithms for some number fields?
(already known for some fields, e.g., with norm relations)

All the same: computing class group, units, S -units, ...

Quantum algorithm: polynomial classical algorithm: $\rho_K \cdot (L_{\Delta_K}(1/2) + L_{n^n}(2/3))$

Some open problems:

- ▶ from number fields to quaternion algebras
probably hard, would break Hawk signature scheme
- ▶ is there really a ρ_K in the complexity?
numerical experiments? better provable smoothness bound?
- ▶ better algorithms for some number fields?
(already known for some fields, e.g., with norm relations)

Thank you