Short generators without quantum computers: the case of multiquadratics

Daniel J. Bernstein

University of Illinois at Chicago

11 July 2017

Joint work with:
Jens Bauch & Henry de Valence & Tanja Lange & Christine van Vredendaal

`https://multiquad.cr.yp.to`

Part I: Introduction

"*Lattice-based crypto is secure because lattice problems are hard.*"

— Everyone who works on lattice-based crypto

"*Lattice-based crypto is secure because lattice problems are hard.*"

— Everyone who works on lattice-based crypto

Really? How hard are they? Which problems are broken in time $<2^{100}$?
Which *cryptosystems* are broken in time $<2^{100}$?

"*Lattice-based crypto is secure because lattice problems are hard.*"

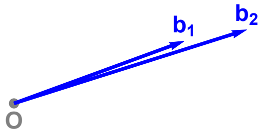
— Everyone who works on lattice-based crypto

Really? How hard are they? Which problems are broken in time $<2^{100}$? Which *cryptosystems* are broken in time $<2^{100}$?

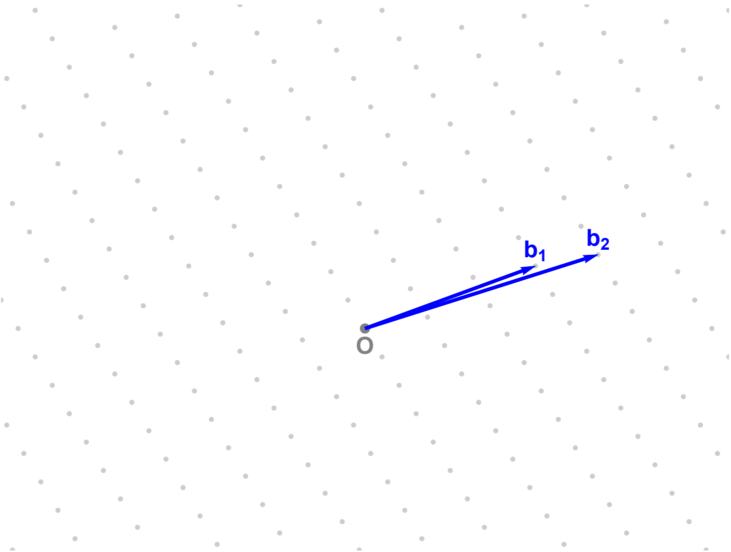
2006 Silverman: "Lattices, SVP and CVP, have been intensively studied for more than 100 years, both as intrinsic mathematical problems and for applications in pure and applied mathematics, physics and cryptography."

So SVP is a hard problem? How hard is it?

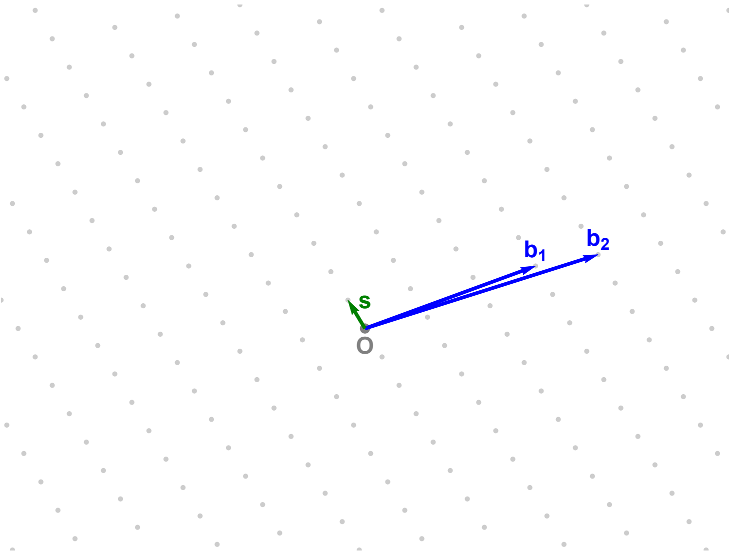
SVP: find minimum-length nonzero vector in lattice



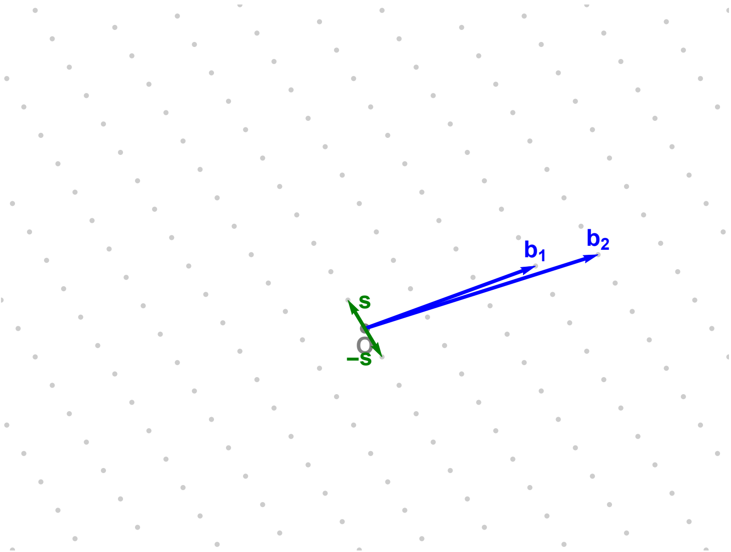
SVP: find minimum-length nonzero vector in lattice



SVP: find minimum-length nonzero vector in lattice



SVP: find minimum-length nonzero vector in lattice



How secure is SVP?

Best SVP algorithms known at the end of the 20th century:
time $2^{\Theta(N \log N)}$ for almost all dimension-N lattices.

How secure is SVP?

Best SVP algorithms known at the end of the 20th century:
time $2^{\Theta(N \log N)}$ for almost all dimension-N lattices.

Best SVP algorithms known today: $2^{\Theta(N)}$, asymptotically much faster.
Some algorithms taking time $2^{(c+o(1))N}$, under plausible assumptions:

$c \approx 0.415$: 2008 Nguyen–Vidick.
$c \approx 0.415$: 2010 Micciancio–Voulgaris.

How secure is SVP?

Best SVP algorithms known at the end of the 20th century:
time $2^{\Theta(N \log N)}$ for almost all dimension-N lattices.

Best SVP algorithms known today: $2^{\Theta(N)}$, asymptotically much faster.
Some algorithms taking time $2^{(c+o(1))N}$, under plausible assumptions:

$c \approx 0.415$: 2008 Nguyen–Vidick.
$c \approx 0.415$: 2010 Micciancio–Voulgaris.
$c \approx 0.384$: 2011 Wang–Liu–Tian–Bi.

How secure is SVP?

Best SVP algorithms known at the end of the 20th century:
time $2^{\Theta(N \log N)}$ for almost all dimension-N lattices.

Best SVP algorithms known today: $2^{\Theta(N)}$, asymptotically much faster.
Some algorithms taking time $2^{(c+o(1))N}$, under plausible assumptions:

$c \approx 0.415$: 2008 Nguyen–Vidick.
$c \approx 0.415$: 2010 Micciancio–Voulgaris.
$c \approx 0.384$: 2011 Wang–Liu–Tian–Bi.
$c \approx 0.378$: 2013 Zhang–Pan–Hu.

How secure is SVP?

Best SVP algorithms known at the end of the 20th century:
time $2^{\Theta(N \log N)}$ for almost all dimension-N lattices.

Best SVP algorithms known today: $2^{\Theta(N)}$, asymptotically much faster.
Some algorithms taking time $2^{(c+o(1))N}$, under plausible assumptions:

$c \approx 0.415$: 2008 Nguyen–Vidick.
$c \approx 0.415$: 2010 Micciancio–Voulgaris.
$c \approx 0.384$: 2011 Wang–Liu–Tian–Bi.
$c \approx 0.378$: 2013 Zhang–Pan–Hu.
$c \approx 0.337$: 2014 Laarhoven.

How secure is SVP?

Best SVP algorithms known at the end of the 20th century:
time $2^{\Theta(N \log N)}$ for almost all dimension-N lattices.

Best SVP algorithms known today: $2^{\Theta(N)}$, asymptotically much faster.
Some algorithms taking time $2^{(c+o(1))N}$, under plausible assumptions:

$c \approx 0.415$: 2008 Nguyen–Vidick.
$c \approx 0.415$: 2010 Micciancio–Voulgaris.
$c \approx 0.384$: 2011 Wang–Liu–Tian–Bi.
$c \approx 0.378$: 2013 Zhang–Pan–Hu.
$c \approx 0.337$: 2014 Laarhoven.
$c \approx 0.298$: 2015 Laarhoven–de Weger.

How secure is SVP?

Best SVP algorithms known at the end of the 20th century:
time $2^{\Theta(N \log N)}$ for almost all dimension-N lattices.

Best SVP algorithms known today: $2^{\Theta(N)}$, asymptotically much faster.
Some algorithms taking time $2^{(c+o(1))N}$, under plausible assumptions:

$c \approx 0.415$: 2008 Nguyen–Vidick.
$c \approx 0.415$: 2010 Micciancio–Voulgaris.
$c \approx 0.384$: 2011 Wang–Liu–Tian–Bi.
$c \approx 0.378$: 2013 Zhang–Pan–Hu.
$c \approx 0.337$: 2014 Laarhoven.
$c \approx 0.298$: 2015 Laarhoven–de Weger.
$c \approx 0.292$: 2015 Becker–Ducas–Gama–Laarhoven.

How secure is SVP?

Best SVP algorithms known at the end of the 20th century:
time $2^{\Theta(N \log N)}$ for almost all dimension-N lattices.

Best SVP algorithms known today: $2^{\Theta(N)}$, asymptotically much faster.
Some algorithms taking time $2^{(c+o(1))N}$, under plausible assumptions:

$c \approx 0.415$: 2008 Nguyen–Vidick.
$c \approx 0.415$: 2010 Micciancio–Voulgaris.
$c \approx 0.384$: 2011 Wang–Liu–Tian–Bi.
$c \approx 0.378$: 2013 Zhang–Pan–Hu.
$c \approx 0.337$: 2014 Laarhoven.
$c \approx 0.298$: 2015 Laarhoven–de Weger.
$c \approx 0.292$: 2015 Becker–Ducas–Gama–Laarhoven.
$c \approx 0.268$ quantum algorithm: 2014 Laarhoven–Mosca–van de Pol.

How secure is SVP?

Best SVP algorithms known at the end of the 20th century:
time $2^{\Theta(N \log N)}$ for almost all dimension-N lattices.

Best SVP algorithms known today: $2^{\Theta(N)}$, asymptotically much faster.
Some algorithms taking time $2^{(c+o(1))N}$, under plausible assumptions:

$c \approx 0.415$: 2008 Nguyen–Vidick.
$c \approx 0.415$: 2010 Micciancio–Voulgaris.
$c \approx 0.384$: 2011 Wang–Liu–Tian–Bi.
$c \approx 0.378$: 2013 Zhang–Pan–Hu.
$c \approx 0.337$: 2014 Laarhoven.
$c \approx 0.298$: 2015 Laarhoven–de Weger.
$c \approx 0.292$: 2015 Becker–Ducas–Gama–Laarhoven.
$c \approx 0.268$ quantum algorithm: 2014 Laarhoven–Mosca–van de Pol.

Who thinks this is the end of the story?
Is $2^{(0.1+o(1))N}$ possible? $2^{\Theta(N/\log N)}$? $2^{N^{1/2+o(1)}}$?

How secure is approx SVP?

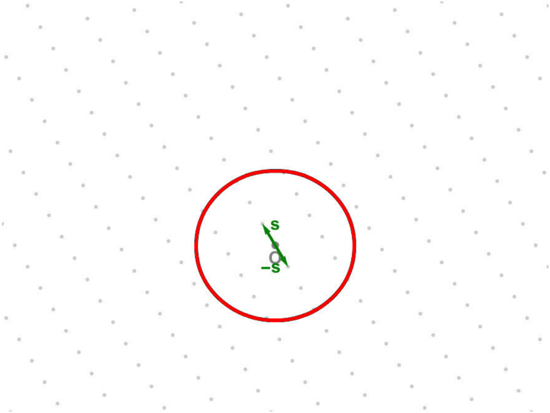
Maybe still reasonable to conjecture that SVP takes exponential time.

But is SVP the problem used in cryptography?
No! Cryptographic problems actually have approximation factors.

How secure is approx SVP?

Maybe still reasonable to conjecture that SVP takes exponential time.

But is SVP the problem used in cryptography?
No! Cryptographic problems actually have approximation factors.



How secure is approx SVP?

2002 Micciancio–Goldwasser (emphasis added): "To date, the best known polynomial time (possibly randomized) approximation algorithms for SVP and CVP achieve worst-case (over the choice of the input) approximation factors $\gamma(n)$ that are **essentially exponential** in the rank *n*."

2007 Regev:

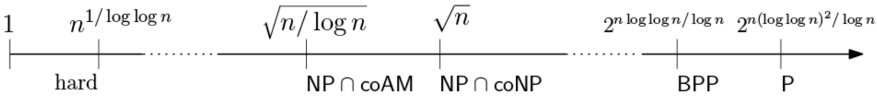


Figure 1: The complexity of lattice problems (some constants omitted)

2013 Micciancio: "Smooth trade-off between running time and approximation: $\gamma \approx 2^{O(n \log\log T/\log T)}$"

Quantum attacks against cyclotomic lattice problems

STOC 2014 Eisenträger–Hallgren–Kitaev–Song:
poly-time quantum algorithm for $K \mapsto \mathcal{O}_K^\times$.

K: number field.
$\mathcal{O}_K$: ring of algebraic integers in K.
$\mathcal{O}_K^\times$: group of units in $\mathcal{O}_K$.

Quantum attacks against cyclotomic lattice problems

STOC 2014 Eisenträger–Hallgren–Kitaev–Song:
poly-time quantum algorithm for $K \mapsto \mathcal{O}_K^\times$.

K: number field.
$\mathcal{O}_K$: ring of algebraic integers in K.
$\mathcal{O}_K^\times$: group of units in $\mathcal{O}_K$.

2015 (and SODA 2016) Biasse–Song,
also using an idea from 2014 Campbell–Groves–Shepherd:
poly-time quantum algorithm for $K, g\mathcal{O}_K \mapsto \zeta_m^j g$ for some j,
assuming cyclotomic $K = \mathbf{Q}(\zeta_m)$, small h_m^+, very short g.

Quantum attacks against cyclotomic lattice problems

STOC 2014 Eisenträger–Hallgren–Kitaev–Song:
poly-time quantum algorithm for $K \mapsto \mathcal{O}_K^\times$.

K: number field.
$\mathcal{O}_K$: ring of algebraic integers in K.
$\mathcal{O}_K^\times$: group of units in $\mathcal{O}_K$.

2015 (and SODA 2016) Biasse–Song,
also using an idea from 2014 Campbell–Groves–Shepherd:
poly-time quantum algorithm for $K, g\mathcal{O}_K \mapsto \zeta_m^j g$ for some j,
assuming cyclotomic $K = \mathbf{Q}(\zeta_m)$, small h_m^+, very short g.

This recovers secret keys in, e.g.,
STOC 2009 Gentry homomorphic-encryption system using cyclotomics,
Eurocrypt 2013 Garg–Gentry–Halevi multilinear-map system, etc.

Is the attack idea limited to very short generators?

More lattice problems of interest:
$I \mapsto$ shortest nonzero vector in I. ("Exact Ideal-SVP".)
$I \mapsto$ close to shortest nonzero vector in I. ("Approximate Ideal-SVP".)

Attack is against principal I with a *very short generator*.

Is the attack idea limited to very short generators?

More lattice problems of interest:
$I \mapsto$ shortest nonzero vector in I. ("Exact Ideal-SVP".)
$I \mapsto$ close to shortest nonzero vector in I. ("Approximate Ideal-SVP".)

Attack is against principal I with a *very short generator*.

2015 Peikert says technique is "useless" for more general principal ideals. ("We simply hadn't realized that the added guarantee of a short generator would transform the technique from useless to devastatingly effective.")

Is the attack idea limited to very short generators?

More lattice problems of interest:
$I \mapsto$ shortest nonzero vector in I. ("Exact Ideal-SVP".)
$I \mapsto$ close to shortest nonzero vector in I. ("Approximate Ideal-SVP".)

Attack is against principal I with a *very short generator*.

2015 Peikert says technique is "useless" for more general principal ideals. ("We simply hadn't realized that the added guarantee of a short generator would transform the technique from useless to devastatingly effective.")

Counterargument: attack is poly time against **arbitrary principal ideals** for approx factor $2^{N^{1/2+o(1)}}$ in degree-N cyclotomics, assuming small h^+. See, e.g., 2016 Cramer–Ducas–Peikert–Regev.

Is the attack idea limited to principal ideals?

2015 Peikert:
"Although cyclotomics have a lot of structure, nobody has yet found a way to exploit it in attacking Ideal-SVP/BDD ... For commonly used rings, principal ideals are an extremely small fraction of all ideals. ... The weakness here is not so much due to the structure of cyclotomics, but rather to the extra structure of principal ideals that have short generators."

Is the attack idea limited to principal ideals?

2015 Peikert:
"Although cyclotomics have a lot of structure, nobody has yet found a way to exploit it in attacking Ideal-SVP/BDD . . . For commonly used rings, principal ideals are an extremely small fraction of all ideals. . . . The weakness here is not so much due to the structure of cyclotomics, but rather to the extra structure of principal ideals that have short generators."

Counterargument, 2016 Cramer–Ducas–Wesolowski:
fast Ideal-SVP attack for approx factor $2^{N^{1/2+o(1)}}$ in degree-N cyclotomics, under plausible assumptions about class-group generators etc.
Starts from Biasse–Song, uses more features of cyclotomic fields.

This shreds the standard approx-Ideal-SVP tradeoff picture.

Non-cyclotomic lattice-based cryptography

Cyclotomics are scary. Let's explore alternatives:

- Eliminate the ideal structure.
  e.g., use LWE instead of Ring-LWE.
  But this limits the security achievable for key size K.

Non-cyclotomic lattice-based cryptography

Cyclotomics are scary. Let's explore alternatives:

- Eliminate the ideal structure.
  e.g., use LWE instead of Ring-LWE.
  But this limits the security achievable for key size K.
- 2016 Bernstein–Chuengsatiansup–Lange–van Vredendaal "NTRU Prime" (preliminary announcement 2014.02, *before* these attacks): as in discrete-log crypto, eliminate unnecessary ring morphisms.
  Use prime degree, large Galois group: e.g., $x^p - x - 1$.

Non-cyclotomic lattice-based cryptography

Cyclotomics are scary. Let's explore alternatives:

- Eliminate the ideal structure.
  e.g., use LWE instead of Ring-LWE.
  But this limits the security achievable for key size K.
- 2016 Bernstein–Chuengsatiansup–Lange–van Vredendaal "NTRU Prime" (preliminary announcement 2014.02, *before* these attacks): as in discrete-log crypto, eliminate unnecessary ring morphisms.
  Use prime degree, large Galois group: e.g., $x^p - x - 1$.
- This talk: Switch from cyclotomics to other Galois number fields.
  Another popular example in algebraic-number-theory textbooks: multiquadratics; e.g., $\mathbf{Q}(\sqrt{2}, \sqrt{3}, \sqrt{5}, \sqrt{7}, \sqrt{11}, \sqrt{13}, \sqrt{17}, \sqrt{19}, \sqrt{23})$.

A reasonable multiquadratic cryptosystem

Case study of a lattice-based cryptosystem
that was already defined in detail for arbitrary number fields:
2010 Smart–Vercauteren, optimized version of 2009 Gentry.

Parameter: $R = \mathbf{Z}[\alpha]$ for an algebraic integer α.
Secret key: very short $g \in R$.
Public key: gR.

A reasonable multiquadratic cryptosystem

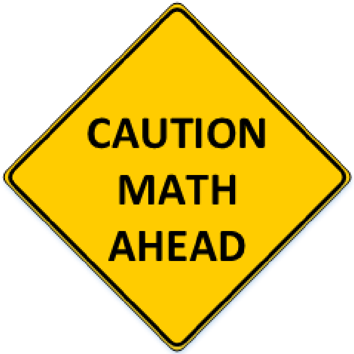
Case study of a lattice-based cryptosystem
that was already defined in detail for arbitrary number fields:
2010 Smart–Vercauteren, optimized version of 2009 Gentry.

Parameter: $R = \mathbf{Z}[\alpha]$ for an algebraic integer α.
Secret key: very short $g \in R$.
Public key: gR.

To handle multiquadratics better,
we generalized beyond $\mathbf{Z}[\alpha]$; fixed a keygen speed problem;
used twisted Hadamard transforms as replacement for FFTs;
adapted 2011 Gentry–Halevi cyclotomic speedups to multiquadratics.

Like Smart–Vercauteren, we took $N \in \lambda^{2+o(1)}$ for target security 2^λ.
Checked security against standard lattice attacks:
nothing better than exponential time.

Part II: Some preliminaries



Definition

A **number field** is a field L containing $\mathbb{Q}$ with finite dimension as a $\mathbb{Q}$-vector space. Its **degree** is this dimension.

Definition

The **ring of integers** $\mathcal{O}_L$ of a number field L is the set of algebraic integers in L. The invertible elements of this ring form the **unit group** $\mathcal{O}_L^\times$.

Problem

Recover a "small" $g \in \mathcal{O}_L$ (modulo roots of unity) given $g\mathcal{O}_L$.

Definition (for this talk—see paper for broader definition)

A **multiquadratic** field is a number field that can be written in the form $L = \mathbb{Q}(\sqrt{d_1}, \ldots, \sqrt{d_n})$, where $(d_1, \ldots, d_n)$ are distinct primes.

The degree of the multiquadratic field is $N = 2^n$.

General strategy to recover g

0. Compute the unit group $\mathcal{O}_L^\times$

General strategy to recover g

0. Compute the unit group $\mathcal{O}_L^\times$
1. Find *some* generator ug of principal ideal $g\mathcal{O}_L$
    - subexponential-time algorithm: see, e.g., 1990 Buchmann, 2014 Biasse–Fieker, 2014 Biasse
    - quantum poly-time algorithm: 2015/2016 Biasse–Song

General strategy to recover g

0. Compute the unit group $\mathcal{O}_L^\times$
1. Find *some* generator ug of principal ideal $g\mathcal{O}_L$
   - subexponential-time algorithm: see, e.g., 1990 Buchmann, 2014 Biasse–Fieker, 2014 Biasse
   - quantum poly-time algorithm: 2015/2016 Biasse–Song
2. Solve BDD for $\mathrm{Log}\, ug$ in the log-unit lattice to find $\mathrm{Log}\, u$
   - 2014 Campbell–Groves–Shepherd pointed out this was easy for cyclotomic fields with h^+ small
   - 2015 Schanck confirmed experimentally
   - 2015 Cramer–Ducas–Peikert–Regev proved pre-quantum polynomial time for these fields

(BDD: bounded-distance decoding; i.e., finding a lattice vector close to an input point.)

Definition

Fix a number field L of degree N and fix distinct complex embeddings $\sigma_1, \ldots, \sigma_N$ of L. The Dirichlet logarithm map is defined as

$$\begin{aligned} \mathrm{Log} : L^\times &\mapsto \mathbb{R}^N \\ x &\mapsto (\log|\sigma_1(x)|, \ldots, \log|\sigma_N(x)|) \end{aligned}$$

Definition

Fix a number field L of degree N and fix distinct complex embeddings $\sigma_1, \ldots, \sigma_N$ of L. The Dirichlet logarithm map is defined as

$$\begin{aligned} \mathrm{Log}\ : L^\times &\mapsto \mathbb{R}^N \\ x &\mapsto (\log|\sigma_1(x)|, \ldots, \log|\sigma_N(x)|) \end{aligned}$$

Theorem (Dirichlet Unit Theorem)

The kernel of $\mathrm{Log}|_{\mathcal{O}_L - \{0\}}$ *is the cyclic group of roots of unity in* $\mathcal{O}_L$. *Let* $\Lambda = \mathrm{Log}\,\mathcal{O}_L^\times \subset \mathbb{R}^N$. Λ *is a lattice of rank* $r + c - 1$, *where* r *is the number of real embeddings and* c *is the number of complex-conjugate pairs of non-real embeddings of* L.

Definition

Fix a number field L of degree N and fix distinct complex embeddings $\sigma_1, \ldots, \sigma_N$ of L. The Dirichlet logarithm map is defined as

$$\begin{aligned} \mathrm{Log} : L^\times &\mapsto \mathbb{R}^N \\ x &\mapsto (\log|\sigma_1(x)|, \ldots, \log|\sigma_N(x)|) \end{aligned}$$

Theorem (Dirichlet Unit Theorem)

The kernel of $\mathrm{Log}|_{\mathcal{O}_L - \{0\}}$ is the cyclic group of roots of unity in $\mathcal{O}_L$. Let $\Lambda = \mathrm{Log}\,\mathcal{O}_L^\times \subset \mathbb{R}^N$. Λ is a lattice of rank $r + c - 1$, where r is the number of real embeddings and c is the number of complex-conjugate pairs of non-real embeddings of L.

Fact

If $h\mathcal{O}_L = g\mathcal{O}_L$ and $g \neq 0$ then $h = ug$ for some $u \in \mathcal{O}_L^\times$, and

$$\mathrm{Log}\, g \in \mathrm{Log}\, h + \Lambda.$$

Part III: The algorithm

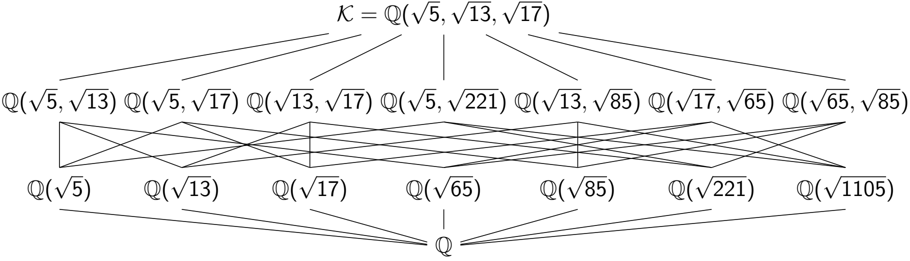
algorithm
noun

Word, used by programmers
When they do not want to
Explain what they did.

https://starecat.com/algorithm-word-used-by-programmers-when-they-do-not-want-to-explain-what-they-did/

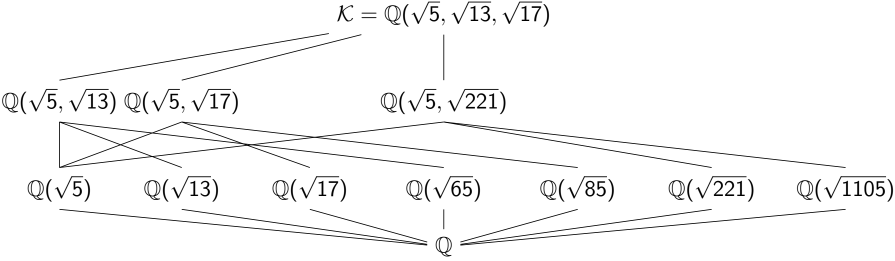
Algorithm idea 1: subfields

Multiquadratic fields have a huge number of subfields.



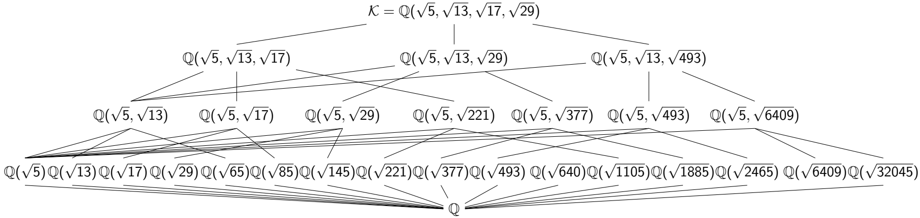
Algorithm idea 1: subfields

Multiquadratic fields have a huge number of subfields.
We use 3 specific subfields (plus recursion).



Algorithm idea 1: subfields

Multiquadratic fields have a huge number of subfields.
We use 3 specific subfields (plus recursion).



Algorithm idea 2: the subfield relation

Let σ be the automorphism of L that negates $\sqrt{d_n}$ and fixes other $\sqrt{d_j}$.

Define $K_\sigma = \{x \in L : \sigma(x) = x\}$ as the field fixed by σ.
The norm $N_\sigma(x)$ of $x \in L$ is defined as $x\sigma(x)$. Then $N_\sigma(x) \in K_\sigma$.

Algorithm idea 2: the subfield relation

Let σ be the automorphism of L that negates $\sqrt{d_n}$ and fixes other $\sqrt{d_j}$.

Define $K_\sigma = \{x \in L : \sigma(x) = x\}$ as the field fixed by σ.
The norm $N_\sigma(x)$ of $x \in L$ is defined as $x\sigma(x)$. Then $N_\sigma(x) \in K_\sigma$.

Let τ be the automorphism of L that negates $\sqrt{d_{n-1}}$ and fixes other $\sqrt{d_j}$.

$$\begin{aligned} N_\sigma(x) &= x\sigma(x) \\ N_\tau(x) &= x\tau(x) \\ \sigma(N_{\sigma\tau}(x)) &= \sigma(x\sigma(\tau(x))) \end{aligned}$$

Algorithm idea 2: the subfield relation

Let σ be the automorphism of L that negates $\sqrt{d_n}$ and fixes other $\sqrt{d_j}$.

Define $K_\sigma = \{x \in L : \sigma(x) = x\}$ as the field fixed by σ.
The norm $N_\sigma(x)$ of $x \in L$ is defined as $x\sigma(x)$. Then $N_\sigma(x) \in K_\sigma$.

Let τ be the automorphism of L that negates $\sqrt{d_{n-1}}$ and fixes other $\sqrt{d_j}$.

$$\begin{aligned} N_\sigma(x) &= x\sigma(x) \\ N_\tau(x) &= x\tau(x) \\ \sigma(N_{\sigma\tau}(x)) &= \sigma(x\sigma(\tau(x))) = \sigma(x)\tau(x) \end{aligned}$$

Algorithm idea 2: the subfield relation

Let σ be the automorphism of L that negates $\sqrt{d_n}$ and fixes other $\sqrt{d_j}$.

Define $K_\sigma = \{x \in L : \sigma(x) = x\}$ as the field fixed by σ.
The norm $N_\sigma(x)$ of $x \in L$ is defined as $x\sigma(x)$. Then $N_\sigma(x) \in K_\sigma$.

Let τ be the automorphism of L that negates $\sqrt{d_{n-1}}$ and fixes other $\sqrt{d_j}$.

$$
\begin{aligned}
N_\sigma(x) &= x\sigma(x) \\
N_\tau(x) &= x\tau(x) \\
\sigma(N_{\sigma\tau}(x)) &= \sigma(x\sigma(\tau(x))) = \sigma(x)\tau(x) \\
\frac{N_\sigma(x)N_\tau(x)}{\sigma(N_{\sigma\tau}(x))} &= x^2
\end{aligned}
$$

assuming $x \neq 0$.

Algorithm idea 3: computing units via subfields

Can use the subfield relation to find the unit group $\mathcal{O}_L^\times$:

$$u^2 = \frac{N_\sigma(u)N_\tau(u)}{\sigma(N_{\sigma\tau}(u))}$$

Algorithm idea 3: computing units via subfields

Can use the subfield relation to find the unit group $\mathcal{O}_L^\times$:

$$u^2 = \frac{N_\sigma(u)N_\tau(u)}{\sigma(N_{\sigma\tau}(u))}$$

If $U_L = \mathcal{O}_{K_\sigma}^\times \cdot \mathcal{O}_{K_\tau}^\times \cdot \sigma(\mathcal{O}_{K_{\sigma\tau}}^\times)$, then

$$(\mathcal{O}_L^\times)^2 \subseteq U_L \subseteq \mathcal{O}_L^\times$$

So if we can find a basis for $(\mathcal{O}_L^\times)^2$, taking square roots gives $\mathcal{O}_L^\times$.

Algorithm idea 3: computing units via subfields

Can use the subfield relation to find the unit group $\mathcal{O}_L^\times$:

$$u^2 = \frac{N_\sigma(u)N_\tau(u)}{\sigma(N_{\sigma\tau}(u))}$$

If $U_L = \mathcal{O}_{K_\sigma}^\times \cdot \mathcal{O}_{K_\tau}^\times \cdot \sigma(\mathcal{O}_{K_{\sigma\tau}}^\times)$, then

$$(\mathcal{O}_L^\times)^2 \subseteq U_L \subseteq \mathcal{O}_L^\times$$

So if we can find a basis for $(\mathcal{O}_L^\times)^2$, taking square roots gives $\mathcal{O}_L^\times$.

1966 Wada: We can do this—in exponential time!
Check which products of subsets of basis vectors for U_L are squares.

Algorithm idea 3: computing units via subfields

Can use the subfield relation to find the unit group $\mathcal{O}_L^\times$:

$$u^2 = \frac{N_\sigma(u)N_\tau(u)}{\sigma(N_{\sigma\tau}(u))}$$

If $U_L = \mathcal{O}_{K_\sigma}^\times \cdot \mathcal{O}_{K_\tau}^\times \cdot \sigma(\mathcal{O}_{K_{\sigma\tau}}^\times)$, then

$$(\mathcal{O}_L^\times)^2 \subseteq U_L \subseteq \mathcal{O}_L^\times$$

So if we can find a basis for $(\mathcal{O}_L^\times)^2$, taking square roots gives $\mathcal{O}_L^\times$.

1966 Wada: We can do this—in exponential time!
Check which products of subsets of basis vectors for U_L are squares.

Better: polynomial time, adapting 1991 Adleman idea from NFS.
Define many *quadratic characters* $\chi_i : \mathcal{O}_L^\times \to \mathbb{Z}/2\mathbb{Z}$.
Almost certainly $(\mathcal{O}_L^\times)^2 = U_L \cap (\bigcap_i \operatorname{Ker} \chi_i)$. Compute by linear algebra.

Algorithm idea 4: recovering generators via subfields

Fact

Can compute $N_\sigma(g)\mathcal{O}_{K_\sigma}$ quickly from $g\mathcal{O}_L$.

Apply algorithm recursively to find generator h_σ of $N_\sigma(g)\mathcal{O}_{K_\sigma}$.
i.e. $h_\sigma = u_\sigma N_\sigma(g)$ for some unit u_σ.

Algorithm idea 4: recovering generators via subfields

Fact

Can compute $N_\sigma(g)\mathcal{O}_{K_\sigma}$ quickly from $g\mathcal{O}_L$.

Apply algorithm recursively to find generator h_σ of $N_\sigma(g)\mathcal{O}_{K_\sigma}$.
i.e. $h_\sigma = u_\sigma N_\sigma(g)$ for some unit u_σ.

Similarly h_τ, $h_{\sigma\tau}$. Compute

$$h = \frac{h_\sigma h_\tau}{\sigma(h_{\sigma\tau})} = \frac{u_\sigma N_\sigma(g) u_\tau N_\tau(g)}{\sigma(u_{\sigma\tau})\sigma(N_{\sigma\tau}(g))}.$$

Subfield relation: $h = ug^2$ for some $u \in \mathcal{O}_L^\times$.

Algorithm idea 4: recovering generators via subfields

Fact

Can compute $N_\sigma(g)\mathcal{O}_{K_\sigma}$ quickly from $g\mathcal{O}_L$.

Apply algorithm recursively to find generator h_σ of $N_\sigma(g)\mathcal{O}_{K_\sigma}$.
i.e. $h_\sigma = u_\sigma N_\sigma(g)$ for some unit u_σ.

Similarly h_τ, $h_{\sigma\tau}$. Compute

$$h = \frac{h_\sigma h_\tau}{\sigma(h_{\sigma\tau})} = \frac{u_\sigma N_\sigma(g) u_\tau N_\tau(g)}{\sigma(u_{\sigma\tau})\sigma(N_{\sigma\tau}(g))}.$$

Subfield relation: $h = ug^2$ for some $u \in \mathcal{O}_L^\times$.

Problem: This is not necessarily a square!

Algorithm idea 4: recovering generators via subfields

Fact

Can compute $N_\sigma(g)\mathcal{O}_{K_\sigma}$ quickly from $g\mathcal{O}_L$.

Apply algorithm recursively to find generator h_σ of $N_\sigma(g)\mathcal{O}_{K_\sigma}$.
i.e. $h_\sigma = u_\sigma N_\sigma(g)$ for some unit u_σ.

Similarly h_τ, $h_{\sigma\tau}$. Compute

$$h = \frac{h_\sigma h_\tau}{\sigma(h_{\sigma\tau})} = \frac{u_\sigma N_\sigma(g) u_\tau N_\tau(g)}{\sigma(u_{\sigma\tau})\sigma(N_{\sigma\tau}(g))}.$$

Subfield relation: $h = ug^2$ for some $u \in \mathcal{O}_L^\times$.

Problem: This is not necessarily a square!
Solution: Use quadratic characters to find $v \in \mathcal{O}_L^\times$ with square vh.

Algorithm idea 4: recovering generators via subfields

Fact

Can compute $N_\sigma(g)\mathcal{O}_{K_\sigma}$ quickly from $g\mathcal{O}_L$.

Apply algorithm recursively to find generator h_σ of $N_\sigma(g)\mathcal{O}_{K_\sigma}$.
i.e. $h_\sigma = u_\sigma N_\sigma(g)$ for some unit u_σ.

Similarly h_τ, $h_{\sigma\tau}$. Compute

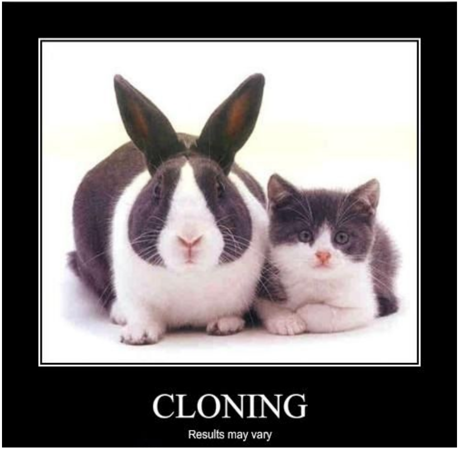
$$h = \frac{h_\sigma h_\tau}{\sigma(h_{\sigma\tau})} = \frac{u_\sigma N_\sigma(g) u_\tau N_\tau(g)}{\sigma(u_{\sigma\tau})\sigma(N_{\sigma\tau}(g))}.$$

Subfield relation: $h = ug^2$ for some $u \in \mathcal{O}_L^\times$.

Problem: This is not necessarily a square!
Solution: Use quadratic characters to find $v \in \mathcal{O}_L^\times$ with square vh.

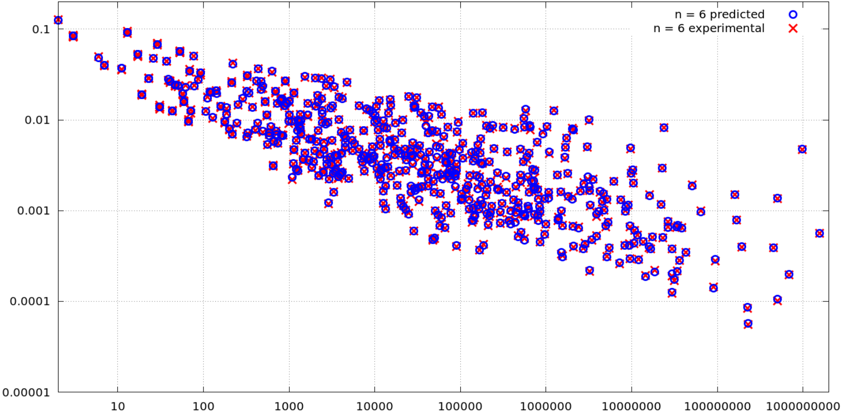
Last step is to shorten the generator $u'g = \sqrt{vh}$ by solving the BDD problem in the log-unit lattice.

Part IV: Results



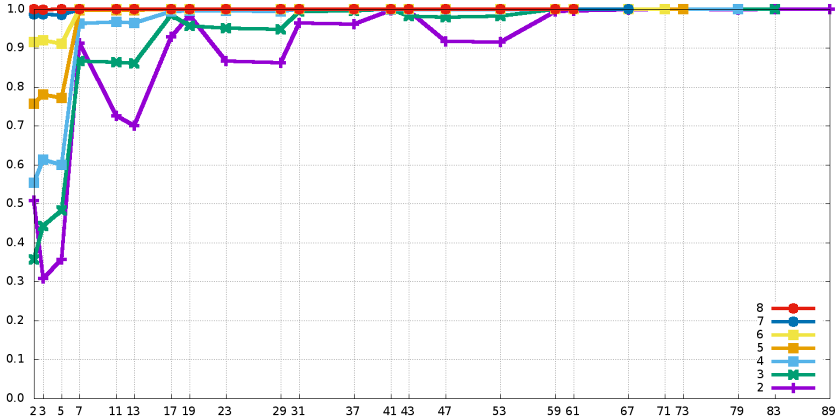
Coefficients for MQ lattice

Vertical axis: Average absolute coefficients of $\operatorname{Log} g$ on MQ basis.
Horizontal axis: $1.11/(2^{n/2}\log(u_D))$.



Success for MQ lattice

Vertical axis: Success probability of simple rounding (in the MQ lattice).
Horizontal axis: d_1, using n consecutive primes for $(d_1, \ldots, d_n)$.



Time (in seconds) to find full lattice and generator

2^n	Sage tower units	Sage absolute units	new units	new units2	new gen	new gen2
8	0.05	0.03	0.90	0.91	0.07	0.07
16	0.48	0.24	2.33	2.39	0.20	0.19
32	6.75	4.73	6.61	7.36	0.56	0.51
64	$>$700000	$>$700000	23.30	37.51	1.51	1.51
128			93.02	1560.49	4.95	7.29
256			463.91	31469.23	27.95	100.65

Table: Observed time to compute (once) the units of $\mathbb{Q}(\sqrt{d_1}, \ldots, \sqrt{d_n})$; and to find a generator for the public key in the cryptosystem.

Success at finding short generator of ideal

n	3	4	5	6	7	8
$p_{\mathrm{suc}}(L_1)$	0.122	0.137	0.132	0.036	0.001	0.000
$p_{\mathrm{suc}}(L_n)$	0.203	0.490	0.648	0.936	0.631	0.423
$p_{\mathrm{suc}}(L_{n^2})$	0.784	0.981	1.000	1.000	1.000	1.000

Table: Observed attack success probabilities for various multiquadratic fields.

Figure: A multitude of quads.

Questions?