

Short generators without quantum computers: the case of multiquadratics

Daniel J. Bernstein & Christine van Vredendaal

University of Illinois at Chicago
Technische Universiteit Eindhoven

19 January 2017

Joint work with Jens Bauch & Henry de Valence & Tanja Lange

Part I: Introduction



How secure is SVP?

“Lattice-based crypto is secure because lattice problems are hard.”

Really? How hard are they? *Which* cryptosystems are secure?

How secure is SVP?

“Lattice-based crypto is secure because lattice problems are hard.”

Really? How hard are they? *Which* cryptosystems are secure?

Consider, e.g., dimension- N SVP.

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?

“Lattice-based crypto is secure because lattice problems are hard.”

Really? How hard are they? *Which* cryptosystems are secure?

Consider, e.g., dimension- N SVP.

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?

“Lattice-based crypto is secure because lattice problems are hard.”

Really? How hard are they? *Which* cryptosystems are secure?

Consider, e.g., dimension- N SVP.

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?

“Lattice-based crypto is secure because lattice problems are hard.”

Really? How hard are they? *Which* cryptosystems are secure?

Consider, e.g., dimension- N SVP.

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?

“Lattice-based crypto is secure because lattice problems are hard.”

Really? How hard are they? *Which* cryptosystems are secure?

Consider, e.g., dimension- N SVP.

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?

“Lattice-based crypto is secure because lattice problems are hard.”

Really? How hard are they? *Which* cryptosystems are secure?

Consider, e.g., dimension- N SVP.

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?

“Lattice-based crypto is secure because lattice problems are hard.”

Really? How hard are they? *Which* cryptosystems are secure?

Consider, e.g., dimension- N SVP.

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?

“Lattice-based crypto is secure because lattice problems are hard.”

Really? How hard are they? *Which* cryptosystems are secure?

Consider, e.g., dimension- N SVP.

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.

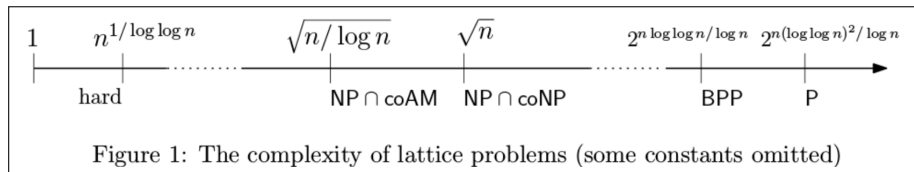
$c \approx 0.268$ quantum algorithm: 2014 Laarhoven–Mosca–van de Pol.

Who thinks this is the end of the story?

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:



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.,
2000 Buchmann–Maurer–Möller cryptosystem using cyclotomics,
STOC 2009 Gentry FHE 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:

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–Chuang–Satiansup–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, 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)

A **multiquadratic** field is a number field that can be written in the form $L = \mathbb{Q}(\sqrt{d_1}, \dots, \sqrt{d_n})$, where (d_1, \dots, 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 [1990 Buchmann, 2014 Biasse–Fieker, 2014 Biasse]
 - ▶ quantum poly-time algorithm [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 [1990 Buchmann, 2014 Biasse–Fieker, 2014 Biasse]
 - ▶ quantum poly-time algorithm [2016 Biasse–Song]
- 2 Solve BDD for $\text{Log } ug$ in the log-unit lattice to find $\text{Log } u$
 - ▶ 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 $\sigma_1, \dots, \sigma_N$ of L . The **Dirichlet logarithm map** is defined as

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

Definition

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

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

Theorem (Dirichlet Unit Theorem)

The kernel of $\text{Log}|_{\mathcal{O}_L - \{0\}}$ is the cyclic group of roots of unity in \mathcal{O}_L . Let $\Lambda = \text{Log } \mathcal{O}_L^\times \subset \mathbb{R}^N$. 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 $\sigma_1, \dots, \sigma_N$ of L . The **Dirichlet logarithm map** is defined as

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

Theorem (Dirichlet Unit Theorem)

The kernel of $\text{Log}|_{\mathcal{O}_L - \{0\}}$ is the cyclic group of roots of unity in \mathcal{O}_L . Let $\Lambda = \text{Log } \mathcal{O}_L^\times \subset \mathbb{R}^N$. 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

$$\text{Log } g \in \text{Log } h + \Lambda.$$

Part III: The algorithm

algorithm
noun

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

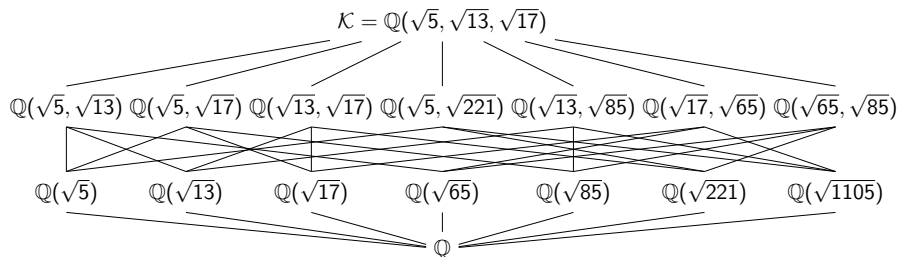
<https://starecat.com/algorithm-word-used-by-programmers-when-they-do-not-want-to-explain-what-they-did/>

Algorithm idea 1: subfields

Multiquadratic fields have a huge number of subfields

Algorithm idea 1: subfields

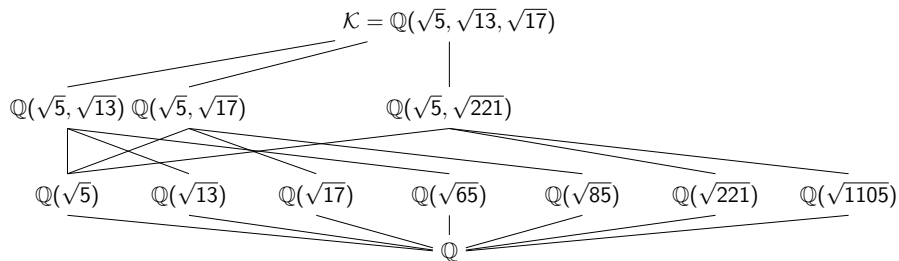
Multiquadratic fields have a huge number of subfields



Algorithm idea 1: subfields

Multiquadratic fields have a huge number of subfields

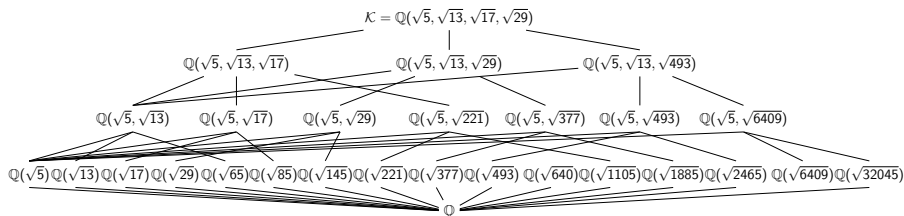
We use 3 specific ones (plus recursion)



Algorithm idea 1: subfields

Multiquadratic fields have a huge number of subfields

We use 3 specific ones (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 $\sqrt{d_j}$.

$$N_\sigma(x) = x\sigma(x)$$

$$N_\tau(x) = x\tau(x)$$

$$\sigma(N_{\sigma\tau}(x)) = \sigma(x\sigma(\tau(x)))$$

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 $\sqrt{d_j}$.

$$N_\sigma(x) = x\sigma(x)$$

$$N_\tau(x) = x\tau(x)$$

$$\sigma(N_{\sigma\tau}(x)) = \sigma(x\sigma(\tau(x))) = \sigma(x)\tau(x)$$

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 $\sqrt{d_j}$.

$$N_\sigma(x) = x\sigma(x)$$

$$N_\tau(x) = x\tau(x)$$

$$\sigma(N_{\sigma\tau}(x)) = \sigma(x\sigma(\tau(x))) = \sigma(x)\tau(x)$$

$$x^2 = N_\sigma(x)N_\tau(x)/\sigma(N_{\sigma\tau}(x))$$

Algorithm idea 3: computing units via subfields

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

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

Algorithm idea 3: computing units via subfields

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

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

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

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

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

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

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* $\chi_i : \mathcal{O}_L^\times \rightarrow \mathbb{Z}/2\mathbb{Z}$.

Almost certainly $(\mathcal{O}_L^\times)^2 = U_L \cap (\bigcap_i \text{Ker } \chi_i)$. 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 $h_\tau, h_{\sigma\tau}$. 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 $h_\tau, h_{\sigma\tau}$. 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 $h_\tau, h_{\sigma\tau}$. 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 $h_\tau, h_{\sigma\tau}$. 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.

Algorithm 1: MQPIP(L, \mathcal{I})

Input: Real multiquadratic field L and a basis matrix for a principal ideal \mathcal{I} of \mathcal{O}_L

Result: A short generator g for \mathcal{I}

```
1 if  $[L : \mathbb{Q}] = 2$  then
2   return QPIP( $L, \mathcal{I}$ )
3  $\sigma, \tau \leftarrow \text{Gal}(L/\mathbb{Q})$ 
4 for  $\ell \in \{\sigma, \tau, \sigma\tau\}$  do
5   Set  $K_\ell$  so that  $\text{Gal}(L/K_\ell) = \langle \ell \rangle$ 
6    $\mathcal{I}_\ell \leftarrow (\mathcal{I} \cdot \sigma_\ell(\mathcal{I})) \cap K_\ell = N_\ell(\mathcal{I})$ 
7    $g_\ell, U_\ell \leftarrow \text{MQPIP}(K_\ell, \mathcal{I}_\ell)$ 
8  $\mathcal{O}_L^\times, X \leftarrow \text{UnitGroupFromSubgroup}(U_\ell)$ 
9  $h \leftarrow g_\sigma g_\tau \sigma(g_{\sigma\tau}^{-1})$ 
10  $g' \leftarrow \text{IdealSqrt}(h, \mathcal{O}_L^\times, X)$ 
11  $g \leftarrow \text{ShortenGen}(g', \mathcal{O}_L^\times)$ 
12 return  $g, \mathcal{O}_L^\times$ 
```

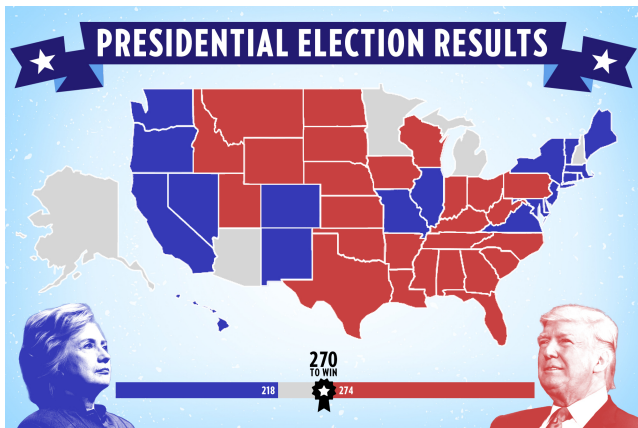
Algorithm 1: MQPIP(L, \mathcal{I})

Input: Real multiquadratic field L and a basis matrix for a principal ideal \mathcal{I} of \mathcal{O}_L

Result: A short generator g for \mathcal{I}

- 1 **if** $[L : \mathbb{Q}] = 2$ **then**
 - 2 **return** QPIP(L, \mathcal{I}) $\triangleright N^2 \cdot \exp((\ln |D|)^{1/2+o(1)})$
 - 3 $\sigma, \tau \leftarrow \text{Gal}(L/\mathbb{Q})$
 - 4 **for** $\ell \in \{\sigma, \tau, \sigma\tau\}$ **do**
 - 5 Set K_ℓ so that $\text{Gal}(L/K_\ell) = \langle \ell \rangle$
 - 6 $\mathcal{I}_\ell \leftarrow (\mathcal{I} \cdot \sigma_\ell(\mathcal{I})) \cap K_\ell = N_\ell(\mathcal{I})$
 - 7 $g_\ell, U_\ell \leftarrow \text{MQPIP}(K_\ell, \mathcal{I}_\ell)$
 - 8 $\mathcal{O}_L^\times, X \leftarrow \text{UnitGroupFromSubgroup}(U_\ell)$ $\triangleright O(N^5 B)$
 - 9 $h \leftarrow g_\sigma g_\tau \sigma(g_{\sigma\tau}^{-1})$ $\triangleright O(NB)$
 - 10 $g' \leftarrow \text{IdealSqrt}(h, \mathcal{O}_L^\times, X)$ $\triangleright O(N^4 B)$
 - 11 $g \leftarrow \text{ShortenGen}(g', \mathcal{O}_L^\times)$ $\triangleright O(N^5)$
 - 12 **return** g, \mathcal{O}_L^\times
-

Part IV: Results



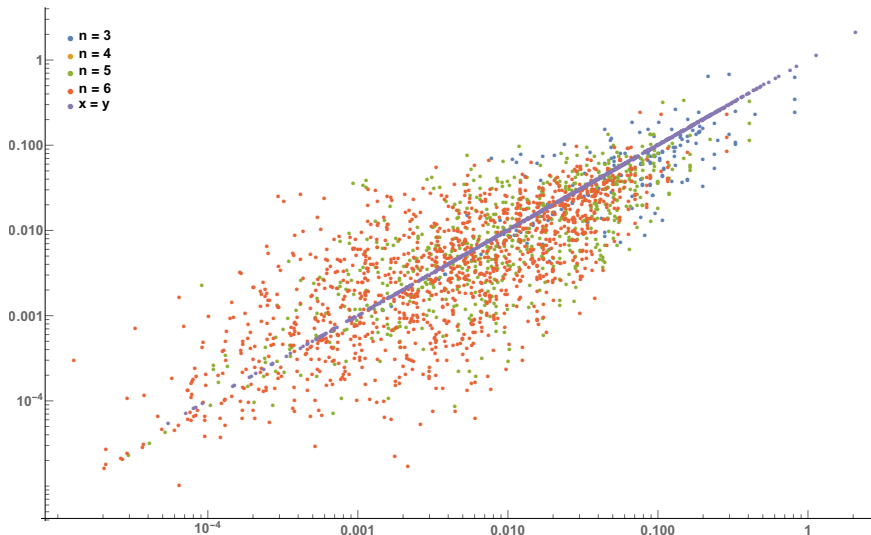
Speed Results (in seconds)

n	2^n	units			keygen	attack
		old tower	old absolute	new		
3	8	0.05	0.03	0.83	0.006	0.24
4	16	0.51	0.24	2.23	0.007	1.10
5	32	7.24	4.80	5.98	0.025	4.24
6	64	>700000.00	>700000.00	31.12	0.050	18.78
7	128			176.98	0.171	88.56
8	256			1855.05	0.834	452.75

Coefficient Results

Vertical axis: Average absolute coefficients of $\text{Log } g$ on MQ basis.

Horizontal axis: $1.11/(2^{n/2} \log(u_D))$.



Failure Results

Vertical axis: Failure probability of simple rounding (without enumeration).

Horizontal axis: d_1 , using n consecutive primes for (d_1, \dots, d_n) .

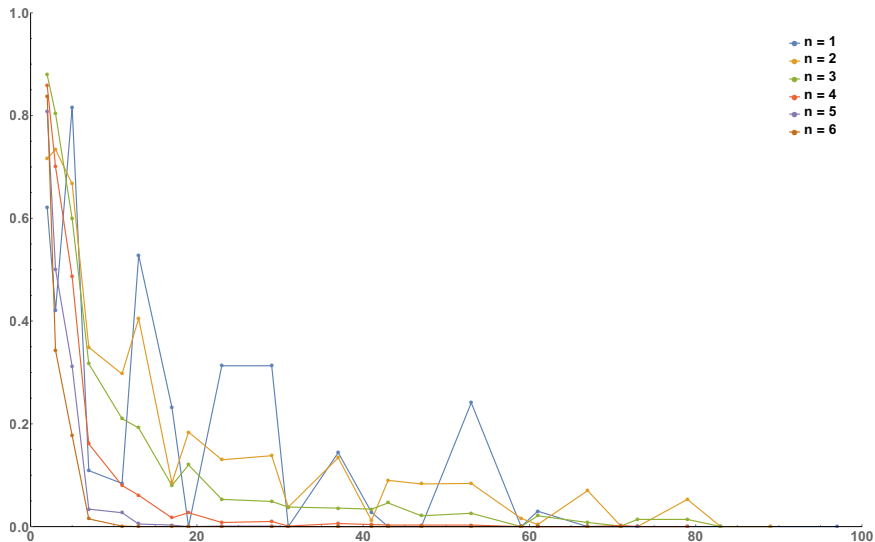




Figure: A multitude of quads.

Questions?