Short generators without quantum computers: the case of multiquadratics

Christine van Vredendaal

Technische Universiteit Eindhoven

1 May 2017

Joint work with:
Jens Bauch & Daniel J. Bernstein & Henry de Valence & Tanja Lange

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* cryptosystems are secure?

How secure?

Multiple attack avenues showing progress

- Sieving asymptotics for dimension-N SVP
  - 2008 Nguyen–Vidick: $2^{(0.415+o(1))N}$
  - 2015 Becker–Ducas–Gama–Laarhoven: $2^{(0.292+o(1))N}$
  - 2014 Laarhoven–Mosca–van de Pol: Quantumly $2^{(0.268+o(1))N}$

How secure?

Multiple attack avenues showing progress

- Sieving asymptotics for dimension-N SVP
  - 2008 Nguyen–Vidick: $2^{(0.415+o(1))N}$
  - 2015 Becker–Ducas–Gama–Laarhoven: $2^{(0.292+o(1))N}$
  - 2014 Laarhoven–Mosca–van de Pol: Quantumly $2^{(0.268+o(1))N}$
- Pre-quantum attacks against cyclotomic ideal lattice problems
  - 2017 Biasse–Espitau–Fouque–Gélin–Kirchner: $L_{|\Delta|}(1/2)$ (see next talk)

How secure?

Multiple attack avenues showing progress

- Sieving asymptotics for dimension-N SVP
  - 2008 Nguyen–Vidick: $2^{(0.415+o(1))N}$
  - 2015 Becker–Ducas–Gama–Laarhoven: $2^{(0.292+o(1))N}$
  - 2014 Laarhoven–Mosca–van de Pol: Quantumly $2^{(0.268+o(1))N}$
- Pre-quantum attacks against cyclotomic ideal lattice problems
  - 2017 Biasse–Espitau–Fouque–Gélin–Kirchner: $L_{|\Delta|}(1/2)$ (see next talk)
- Quantum attacks against cyclotomic ideal lattice problems
  - 2015 Biasse–Song (using 2014 Campbell–Groves–Shepherd): poly-time quantum algorithm against short generators
  - 2016 Cramer–Ducas–Peikert–Regev: general analysis for *arbitrary* principal ideals (within an $e^{\tilde{O}(n^{1/2})}$ approximation factor)
  - 2016 Cramer–Ducas–Wesolowski: generalize to any ideal

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": 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": 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.

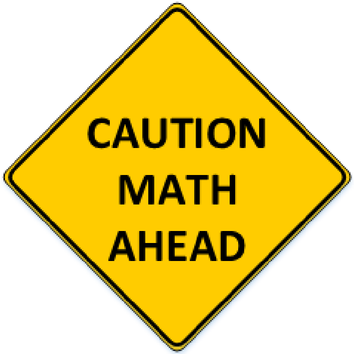
A reasonable multiquadratic cryptosystem

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.

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)

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 [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 $\mathrm{Log}\, ug$ 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

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

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.

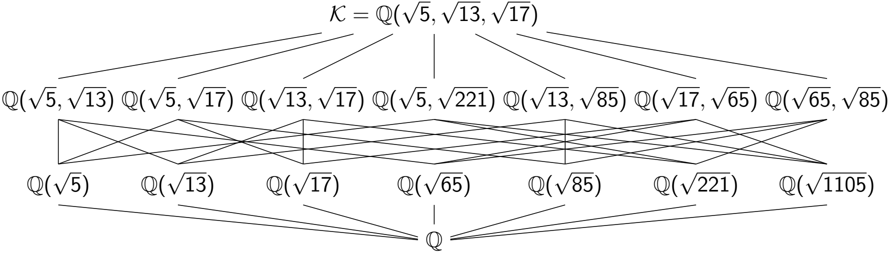
https://starecat.com/algorithm-word-used-by-programmers-when-they-do-not-want-to-explain-what-they-did/

Algorithm idea 1: subfields

Multiquadratic fields have a huge number of subfields

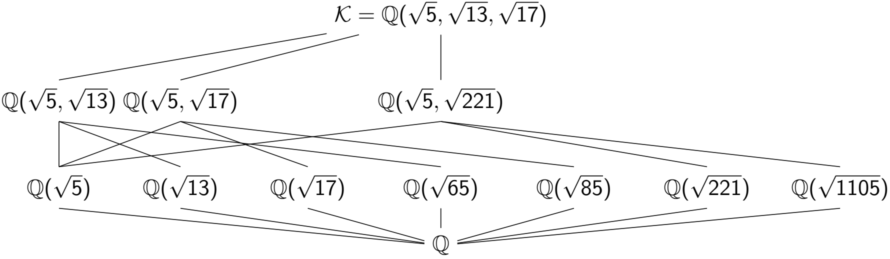
Algorithm idea 1: subfields

Multiquadratic fields have a huge number of subfields



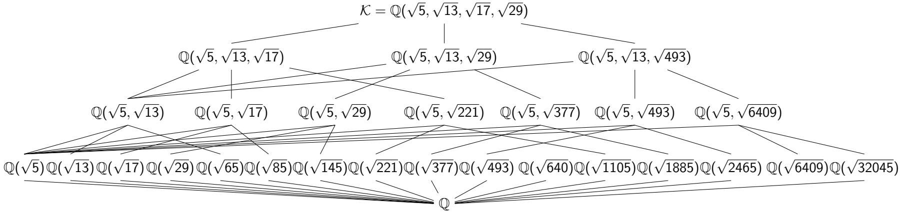
Algorithm idea 1: subfields

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 $\sqrt{d_j}$.

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

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 $\sqrt{d_j}$.

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

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 $\mathcal{O}_L^\times$.

We can do this—in polynomial time!

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 $\mathcal{O}_L^\times$.

We can do this—in 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 $h\mathcal{O}_L$.

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 $h\mathcal{O}_L$.

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 $h\mathcal{O}_L$.

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 $h\mathcal{O}_L$.

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 $h\mathcal{O}_L$.

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.

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

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 $\mathcal{I}$

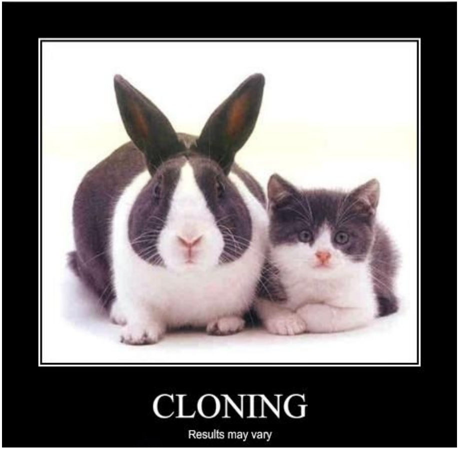
if $[L : \mathbb{Q}] = 2$ **then**
    return $\mathrm{QPIP}(L, \mathcal{I})$
$\sigma, \tau \leftarrow \mathrm{Gal}(L/\mathbb{Q})$
for $\ell \in \{\sigma, \tau, \sigma\tau\}$ **do**
    Set K_ℓ so that $\mathrm{Gal}(L/K_\ell) = \langle \ell \rangle$
    $\mathcal{I}_\ell \leftarrow (\mathcal{I} \cdot \sigma_\ell(\mathcal{I})) \cap K_\ell = N_\ell(\mathcal{I})$
    $g_\ell, U_\ell \leftarrow \mathrm{MQPIP}(K_\ell, \mathcal{I}_\ell)$
$\mathcal{O}_L^\times, X \leftarrow \mathrm{UnitsGivenSubgroup}(U_\ell)$
$h \leftarrow g_\sigma g_\tau \sigma(g_{\sigma\tau}^{-1})$
$g' \leftarrow \mathrm{IdealSqrt}(h, \mathcal{O}_L^\times, X)$
$g \leftarrow \mathrm{ShortenGen}(g', \mathcal{O}_L^\times)$
return $g, \mathcal{O}_L^\times$

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

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 $\mathcal{I}$

if $[L : \mathbb{Q}] = 2$ **then**
  return $\mathrm{QPIP}(L, \mathcal{I})$ ▷ $O(NB)$
$\sigma, \tau \leftarrow \mathrm{Gal}(L/\mathbb{Q})$
for $\ell \in \{\sigma, \tau, \sigma\tau\}$ **do**
  Set K_ℓ so that $\mathrm{Gal}(L/K_\ell) = \langle \ell \rangle$
  $\mathcal{I}_\ell \leftarrow (\mathcal{I} \cdot \sigma_\ell(\mathcal{I})) \cap K_\ell = N_\ell(\mathcal{I})$
  $g_\ell, U_\ell \leftarrow \mathrm{MQPIP}(K_\ell, \mathcal{I}_\ell)$
$\mathcal{O}_L^\times, X \leftarrow \mathrm{UnitsGivenSubgroup}(U_\ell)$ ▷ $O(N^7)$ (exp. $O(N^{2+\log_2 3} B)$)
$h \leftarrow g_\sigma g_\tau \sigma(g_{\sigma\tau}^{-1})$ ▷ $O(N^2 B)$
$g' \leftarrow \mathrm{IdealSqrt}(h, \mathcal{O}_L^\times, X)$ ▷ $O(N^3 + N^2 B)$
$g \leftarrow \mathrm{ShortenGen}(g', \mathcal{O}_L^\times)$ ▷ $O(N^2 B)$
return $g, \mathcal{O}_L^\times$

Part IV: Results



Attack Speed Results (in seconds)

2^n	tower	absolute	new	new2	attack	attack2
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 unit group of $\mathbb{Q}(\sqrt{d_1}, \ldots, \sqrt{d_n})$; and to find a generator for the public key in the cryptosystem.

Attack Success Results

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?