Algorithms for multiquadratic number fields

D. J. Bernstein

Jens Bauch, Daniel J. Bernstein, Henry de Valence, Tanja Lange, Christine van Vredendaal. "Short generators without quantum computers: the case of multiquadratics." Eurocrypt 2017.

Paper and software:
https://multiquad.cr.yp.to

Breakthrough STOC 2009 Gentry cryptosystem “Fully homomorphic encryption using ideal lattices” was broken several years later, under reasonable assumptions.

Breakthrough STOC 2009 Gentry cryptosystem "Fully homomorphic encryption using ideal lattices" was broken several years later, under reasonable assumptions.

Assumption 1: User chooses a ("small h^+") cyclotomic field as the underlying number field.

Breakthrough STOC 2009 Gentry cryptosystem "Fully homomorphic encryption using ideal lattices" was broken several years later, under reasonable assumptions.

Assumption 1: User chooses a ("small h^+") cyclotomic field as the underlying number field.

Assumption 2: Attacker has a large quantum computer.

Breakthrough STOC 2009 Gentry cryptosystem "Fully homomorphic encryption using ideal lattices" was broken several years later, under reasonable assumptions.

Assumption 1: User chooses a ("small h^+") cyclotomic field as the underlying number field.

Assumption 2: Attacker has a large quantum computer.

Can other fields be attacked?
Are there non-quantum attacks?
What about other cryptosystems?

Compare to 2013 Lyubashevsky–Peikert–Regev: "All of the algebraic and algorithmic tools (including quantum computation) that we employ . . . can also be brought to bear against SVP and other problems on ideal lattices. Yet despite considerable effort, no significant progress in attacking these problems has been made. The best known algorithms for ideal lattices perform essentially no better than their generic counterparts, both in theory and in practice."

Secret key in Gentry's system: short element g of R.

R: e.g., ring of integers $\mathcal{O}_K$ of a cyclotomic field K.

Public key: ideal gR.

Secret key in Gentry's system:
short element g of R.

R: e.g., ring of integers $\mathcal{O}_K$
of a cyclotomic field K.

Public key: ideal gR.

Attack stage 1, quantum:
SODA 2016 Biasse–Song
finds some generator of gR.
Builds on Eisenträger–Hallgren–
Kitaev–Song algorithm for R^*.

Secret key in Gentry's system: short element g of R.

R: e.g., ring of integers $\mathcal{O}_K$ of a cyclotomic field K.

Public key: ideal gR.

Attack stage 1, quantum: SODA 2016 Biasse–Song finds some generator of gR. Builds on Eisenträger–Hallgren–Kitaev–Song algorithm for R^*.

Attack stage 2, cyclotomic: simple reduction algorithm from 2014 Campbell–Groves–Shepherd.

Standard algebraic-number-theory view of all generators of gR, i.e., all ug where $u \in R^*$: $\operatorname{Log} u$ ranges over Dirichlet's log-unit lattice; $\operatorname{Log} ug = \operatorname{Log} u + \operatorname{Log} g$.

Standard algebraic-number-theory view of all generators of gR, i.e., all ug where $u \in R^*$: $\operatorname{Log} u$ ranges over Dirichlet's log-unit lattice; $\operatorname{Log} ug = \operatorname{Log} u + \operatorname{Log} g$.

Given any generator ug, try to find short $\operatorname{Log} g$ by finding lattice vector $\operatorname{Log} u$ close to $\operatorname{Log} ug$.

Standard algebraic-number-theory view of all generators of gR, i.e., all ug where $u \in R^*$:
$\operatorname{Log} u$ ranges over Dirichlet's log-unit lattice;
$\operatorname{Log} ug = \operatorname{Log} u + \operatorname{Log} g$.

Given any generator ug, try to find short $\operatorname{Log} g$ by finding lattice vector $\operatorname{Log} u$ close to $\operatorname{Log} ug$.

Apply, e.g., embedding or Babai, starting from basis for $\operatorname{Log} R^*$?
Hard to find short enough basis, unless g is extremely short.

For cyclotomic fields,
often u is a "cyclotomic unit".
Known textbook basis for
cyclotomic units is a short basis.

For cyclotomic fields,
often u is a "cyclotomic unit".
Known textbook basis for
cyclotomic units is a short basis.

Take, e.g., $\zeta = \exp(2\pi i/1024)$;
field $\mathbf{Q}(\zeta)$; ring $R = \mathbf{Z}[\zeta]$.

For cyclotomic fields,
often u is a “cyclotomic unit”.
Known textbook basis for
cyclotomic units is a short basis.

Take, e.g., $\zeta = \exp(2\pi i/1024)$;
field $\mathbf{Q}(\zeta)$; ring $R = \mathbf{Z}[\zeta]$.

$(\zeta^3 - 1)/(\zeta - 1)$ is a unit:
directly invert, or apply $\zeta \mapsto \zeta^3$
automorphism to factors of $\zeta - 1$.

For cyclotomic fields,
often u is a "cyclotomic unit".
Known textbook basis for
cyclotomic units is a short basis.

Take, e.g., $\zeta = \exp(2\pi i/1024)$;
field $\mathbf{Q}(\zeta)$; ring $R = \mathbf{Z}[\zeta]$.

$(\zeta^3 - 1)/(\zeta - 1)$ is a unit:
directly invert, or apply $\zeta \mapsto \zeta^3$
automorphism to factors of $\zeta - 1$.

$(\zeta^9 - 1)/(\zeta^3 - 1)$ is a unit.
$(\zeta^{27} - 1)/(\zeta^9 - 1)$ is a unit.
Et cetera. Obtain short basis.

For cyclotomic fields,
often u is a "cyclotomic unit".
Known textbook basis for
cyclotomic units is a short basis.

Take, e.g., $\zeta = \exp(2\pi i/1024)$;
field $\mathbf{Q}(\zeta)$; ring $R = \mathbf{Z}[\zeta]$.

$(\zeta^3 - 1)/(\zeta - 1)$ is a unit:
directly invert, or apply $\zeta \mapsto \zeta^3$
automorphism to factors of $\zeta - 1$.

$(\zeta^9 - 1)/(\zeta^3 - 1)$ is a unit.
$(\zeta^{27} - 1)/(\zeta^9 - 1)$ is a unit.
Et cetera. Obtain short basis.

Now embedding easily finds g.

Are you a lattice salesman?

Try to dismiss lattice attacks.

Ask: Do attacks against

- the $gR \mapsto g$ problem,
- Gentry's original FHE system,
- the original Garg–Gentry–Halevi multilinear maps, . . .

really matter for users?

Are you a lattice salesman?

Try to dismiss lattice attacks.

Ask: Do attacks against

- the $gR \mapsto g$ problem,
- Gentry's original FHE system,
- the original Garg–Gentry–Halevi multilinear maps, . . .

really matter for users?

My response to the salesman:
Maybe not—but this problem is a natural starting point for studying other lattice problems that we certainly care about.

"Canary in the coal mine."

"Exact Ideal-SVP":

$I \mapsto$ shortest nonzero vector in I.

"Approximate Ideal-SVP":

$I \mapsto$ short nonzero vector in I.

"Exact Ideal-SVP":
$I \mapsto$ shortest nonzero vector in I.

"Approximate Ideal-SVP":
$I \mapsto$ short nonzero vector in I.

Attack is against ideal I
with a *short generator*.

"Exact Ideal-SVP":
$I \mapsto$ shortest nonzero vector in I.

"Approximate Ideal-SVP":
$I \mapsto$ short nonzero vector in I.

Attack is against ideal I
with a *short generator*.

2015 Peikert says idea 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."

2015 Peikert also says idea is limited to principal ideals: "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."

Actually, the idea produces attacks far beyond this case.

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

Actually, the idea produces attacks far beyond this case.

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

Can techniques be pushed to smaller approx factors? Can techniques be adapted to break, e.g., Ring-LWE?

NIST post-quantum competition

69 submissions (5 withdrawn), including 20 lattice-based enc.

NIST post-quantum competition

69 submissions (5 withdrawn), including 20 lattice-based enc.

Most lattice-based enc systems use power-of-2 cyclotomics. Some non-power-of-2 cyclotomics: LIMA has Φ_{1019} option, “more conservative choice of field”; NTRU-HRSS-KEM uses Φ_{701}; NTRUEncrypt uses Φ_{743} etc.

NIST post-quantum competition

69 submissions (5 withdrawn), including 20 lattice-based enc.

Most lattice-based enc systems use power-of-2 cyclotomics.
Some non-power-of-2 cyclotomics:
LIMA has Φ_{1019} option, “more conservative choice of field”;
NTRU-HRSS-KEM uses Φ_{701};
NTRUEncrypt uses Φ_{743} etc.

Can cyclotomic attacks on Gentry be extended to these systems?

Some systems avoid cyclotomics.

FrodoKEM-640, 9616-byte key: relies on matrix rings; says that commutative rings "have the potential for weaknesses due to the extra structure".

Some systems avoid cyclotomics.

FrodoKEM-640, 9616-byte key: relies on matrix rings; says that commutative rings "have the potential for weaknesses due to the extra structure".

Titanium-lite, 14720-byte key: uses "middle product" to "hedge against the weakness of specific polynomial rings".

Some systems avoid cyclotomics.

FrodoKEM-640, 9616-byte key: relies on matrix rings; says that commutative rings "have the potential for weaknesses due to the extra structure".

Titanium-lite, 14720-byte key: uses "middle product" to "hedge against the weakness of specific polynomial rings".

Streamlined NTRU Prime 4591^{761}, 1218-byte key: see Tanja's talk later today.

Two theories of lattice safety

Theory 1: Best choices of field F are choices where we know proofs "attack against cryptosystem C_F $\Rightarrow$ attack against problem L_F", where L_F is a "lattice problem".

Two theories of lattice safety

Theory 1: Best choices of field F are choices where we know proofs "attack against cryptosystem C_F $\Rightarrow$ attack against problem L_F", where L_F is a "lattice problem".

Intuitive flaw in theory 1: Maybe these choices make L_F weak!

Two theories of lattice safety

Theory 1: Best choices of field F are choices where we know proofs "attack against cryptosystem C_F $\Rightarrow$ attack against problem L_F", where L_F is a "lattice problem".

Intuitive flaw in theory 1: Maybe these choices make L_F weak!

Theory 2: Safety of field F is damaged by extra automorphisms, extra subfields, etc. Similar situation to discrete-log crypto.

Two theories of lattice safety

Theory 1: Best choices of field F are choices where we know proofs "attack against cryptosystem C_F $\Rightarrow$ attack against problem L_F", where L_F is a "lattice problem".

Intuitive flaw in theory 1: Maybe these choices make L_F weak!

Theory 2: Safety of field F is damaged by extra automorphisms, extra subfields, etc. Similar situation to discrete-log crypto.

What's a good test case for F?

Multiquadratic fields

Assumptions: $n \in \{0, 1, 2, \ldots\}$;
squarefree $d_1, \ldots, d_n \in \mathbf{Z}$;
$\prod_{j \in J} d_j$ non-square for each
nonempty subset $J \subseteq \{1, \ldots, n\}$.

$K = \mathbf{Q}(\sqrt{d_1}, \ldots, \sqrt{d_n})$:
smallest subfield of $\mathbf{C}$
containing $\sqrt{d_1}, \ldots, \sqrt{d_n}$.

K is a degree-2^n number field.
Basis: $\prod_{j \in J} d_j$ for each
subset $J \subseteq \{1, \ldots, n\}$.

e.g. $\mathbf{Q}(\sqrt{2}, \sqrt{3}) =$
$\mathbf{Q} \oplus \mathbf{Q}\sqrt{2} \oplus \mathbf{Q}\sqrt{3} \oplus \mathbf{Q}\sqrt{6}$.

This field is Galois:
has 2^n automorphisms.

e.g. automorphisms of $\mathbf{Q}(\sqrt{2}, \sqrt{3})$
map $a + b\sqrt{2} + c\sqrt{3} + d\sqrt{6}$ to
$a + b\sqrt{2} + c\sqrt{3} + d\sqrt{6}$;
$a - b\sqrt{2} + c\sqrt{3} - d\sqrt{6}$;
$a + b\sqrt{2} - c\sqrt{3} - d\sqrt{6}$;
$a - b\sqrt{2} - c\sqrt{3} + d\sqrt{6}$.

This field is Galois:
has 2^n automorphisms.

e.g. automorphisms of $\mathbf{Q}(\sqrt{2}, \sqrt{3})$
map $a + b\sqrt{2} + c\sqrt{3} + d\sqrt{6}$ to
$a + b\sqrt{2} + c\sqrt{3} + d\sqrt{6}$;
$a - b\sqrt{2} + c\sqrt{3} - d\sqrt{6}$;
$a + b\sqrt{2} - c\sqrt{3} - d\sqrt{6}$;
$a - b\sqrt{2} - c\sqrt{3} + d\sqrt{6}$.

About $2^{n^2/4}$ subfields.

e.g. subfields of $\mathbf{Q}(\sqrt{2}, \sqrt{3})$:
$\mathbf{Q}(\sqrt{2}, \sqrt{3})$,
$\mathbf{Q}(\sqrt{2})$, $\mathbf{Q}(\sqrt{3})$, $\mathbf{Q}(\sqrt{6})$,
$\mathbf{Q}$.

Gentry for multiquadratics

Use optimizations from
PKC 2010 Smart–Vercauteren,
Eurocrypt 2011 Gentry–Halevi.

Gentry for multiquadratics

Use optimizations from
PKC 2010 Smart–Vercauteren,
Eurocrypt 2011 Gentry–Halevi.

F: monic irreducible polynomial.
Ring $R = \mathbf{Z}[x]/F$; not required
to be ring of integers of $\mathbf{Q}[x]/F$.

Gentry for multiquadratics

Use optimizations from
PKC 2010 Smart–Vercauteren,
Eurocrypt 2011 Gentry–Halevi.

F: monic irreducible polynomial.
Ring $R = \mathbf{Z}[x]/F$; not required
to be ring of integers of $\mathbf{Q}[x]/F$.

Multiquadratics: take, e.g.,

$$F = (x - \sqrt{2} - \sqrt{3}) \cdot (x + \sqrt{2} - \sqrt{3}) \cdot (x - \sqrt{2} + \sqrt{3}) \cdot (x + \sqrt{2} + \sqrt{3}).$$

Note $\mathbf{Q}(\sqrt{2} + \sqrt{3}) = \mathbf{Q}(\sqrt{2}, \sqrt{3})$.

Smart–Vercauteren keygen:

Take short random $g \in R$.

Compute q, absolute norm of g.

Start over if q is not prime.

Smart–Vercauteren keygen:
Take short random $g \in R$.
Compute q, absolute norm of g.
Start over if q is not prime.

Compute root r of g in $\mathbf{Z}/q$.
Public key $gR = qR + (x - r)R$
is represented as (q, r).

Smart–Vercauteren keygen:
Take short random $g \in R$.
Compute q, absolute norm of g.
Start over if q is not prime.

Compute root r of g in $\mathbf{Z}/q$.
Public key $gR = qR + (x - r)R$
is represented as (q, r).

(We implemented multiquadratic adaptation of Gentry–Halevi cyclotomic keygen speedup: instead of requiring prime q, require $\gcd\{b, q\} > 1$ for each relative norm $a + b\sqrt{d_i}$ of g. Any squarefree q will work.)

Smart–Vercauteren encryption:

Take short $m \in \mathbf{Z}[x]/F$.

Ciphertext is $m(r) \in \mathbf{Z}/q$.

Smart–Vercauteren encryption:
Take short $m \in \mathbf{Z}[x]/F$.
Ciphertext is $m(r) \in \mathbf{Z}/q$.

Homomorphic operations:
add/multiply ciphertexts $m(r)$
to add/multiply messages m.

Smart–Vercauteren encryption:
Take short $m \in \mathbf{Z}[x]/F$.
Ciphertext is $m(r) \in \mathbf{Z}/q$.

Homomorphic operations:
add/multiply ciphertexts $m(r)$
to add/multiply messages m.

Decryption:
given $c \in \{0, 1, \ldots, q-1\}$,
compute $c/g \in \mathbf{Q}[x]/F$,
round to element of $\mathbf{Z}[x]/F$,
multiply by g, subtract from c.

Smart–Vercauteren encryption:
Take short $m \in \mathbf{Z}[x]/F$.
Ciphertext is $m(r) \in \mathbf{Z}/q$.

Homomorphic operations:
add/multiply ciphertexts $m(r)$
to add/multiply messages m.

Decryption:
given $c \in \{0, 1, \ldots, q-1\}$,
compute $c/g \in \mathbf{Q}[x]/F$,
round to element of $\mathbf{Z}[x]/F$,
multiply by g, subtract from c.

Decryption works if
each coefficient of $m/g \in \mathbf{Q}[x]/F$
is in $(-1/2, 1/2)$.

Gentry says “computational complexity of all of these algorithms must be polynomial in security parameter”.

Flaw in Smart–Vercauteren: for some choices of F, keygen time is not polynomial in security parameter.

Gentry says "computational complexity of all of these algorithms must be polynomial in security parameter".

Flaw in Smart–Vercauteren: for some choices of F, keygen time is not polynomial in security parameter.

For multiquadratic F, keygen is disastrously slow: far too many tries to find prime q. (Adaptation of Gentry–Halevi speedup gives only a polynomial improvement.)

Why this happens: Fix prime p.
Take field k of size p^2.

Why this happens: Fix prime p.
Take field k of size p^2.

$d_1, \ldots, d_n$ are squares in k,
so F splits completely in $k[x]$.
$\deg h \in \{1, 2\}$ for each
irred factor h of F in $\mathbf{F}_p[x]$.

Why this happens: Fix prime p.
Take field k of size p^2.

$d_1, \ldots, d_n$ are squares in k,
so F splits completely in $k[x]$.
$\deg h \in \{1, 2\}$ for each
irred factor h of F in $\mathbf{F}_p[x]$.

Heuristic: for most $p \le 2^n$, have
$\Theta(p)$ distinct linear factors h.

Why this happens: Fix prime p.
Take field k of size p^2.

$d_1, \ldots, d_n$ are squares in k,
so F splits completely in $k[x]$.
$\deg h \in \{1, 2\}$ for each
irred factor h of F in $\mathbf{F}_p[x]$.

Heuristic: for most $p \le 2^n$, have
$\Theta(p)$ distinct linear factors h.

For each linear factor h:
with probability $\approx 1/p$,
h divides g in $\mathbf{F}_p[x]$,
forcing p^2 to divide norm of g
if any d_i is non-square in $\mathbf{F}_p$.

Our multiquadratic tweaks to Smart–Vercauteren (including adaptation of Gentry–Halevi):

1. Generalize cryptosystem to support n polynomial variables. Use $R = \mathbf{Z}[\sqrt{d_1}, \ldots, \sqrt{d_n}]$.

Our multiquadratic tweaks to Smart–Vercauteren (including adaptation of Gentry–Halevi):

1. Generalize cryptosystem to support n polynomial variables. Use $R = \mathbf{Z}[\sqrt{d_1}, \ldots, \sqrt{d_n}]$.

2. Subroutine: Construct uniform random invertible element of R/p.

Our multiquadratic tweaks to Smart–Vercauteren (including adaptation of Gentry–Halevi):

1. Generalize cryptosystem to support n polynomial variables. Use $R = \mathbf{Z}[\sqrt{d_1}, \ldots, \sqrt{d_n}]$.

2. Subroutine: Construct uniform random invertible element of R/p.

3. Choose $y \in \Theta(2^n/n)$. Force g to be invertible mod all primes $p \le y$. Heuristically, good chance of squarefree norm.

Computing units

Fix positive non-square $d \in \mathbf{Z}$.
Assume d quasipoly in 2^n;
i.e., $\log d \in n^{O(1)}$.

Computing units

Fix positive non-square $d \in \mathbf{Z}$.
Assume d quasipoly in 2^n;
i.e., $\log d \in n^{O(1)}$.

$\{\ldots, \pm\varepsilon^{-2}, \pm\varepsilon^{-1}, \pm 1, \pm\varepsilon, \pm\varepsilon^2, \ldots\}$
is unit group of ring of integers of
$\mathbf{Q}(\sqrt{d})$ for a unique $\varepsilon > 1$, the
normalized fundamental unit.
$\log \varepsilon < \sqrt{d}(2 + \log 4d)$; quasipoly.

Computing units

Fix positive non-square $d \in \mathbf{Z}$.
Assume d quasipoly in 2^n;
i.e., $\log d \in n^{O(1)}$.

$\{\ldots, \pm\varepsilon^{-2}, \pm\varepsilon^{-1}, \pm 1, \pm\varepsilon, \pm\varepsilon^2, \ldots\}$
is unit group of ring of integers of
$\mathbf{Q}(\sqrt{d})$ for a unique $\varepsilon > 1$, the
normalized fundamental unit.
$\log \varepsilon < \sqrt{d}(2 + \log 4d)$; quasipoly.

Standard algorithms compute
$a, b \in \mathbf{Q}$ with $\varepsilon = a + b\sqrt{d}$
in time $(\log \varepsilon)^{1+o(1)}$; quasipoly.
(Can save time by instead
representing ε as product.)

Take a multiquadratic field
$K = \mathbf{Q}(\sqrt{d_1}, \ldots, \sqrt{d_n})$.
Assume $n > 0$ and all $d_i > 0$.

The set of **multiquadratic units** is the group generated by units of all $2^n - 1$ quadratic subfields. Analogous to cyclotomic units.

Compute this group by computing all normalized fundamental units.

Take a multiquadratic field
$K = \mathbf{Q}(\sqrt{d_1}, \ldots, \sqrt{d_n})$.
Assume $n > 0$ and all $d_i > 0$.

The set of **multiquadratic units** is the group generated by units of all $2^n - 1$ quadratic subfields.
Analogous to cyclotomic units.

Compute this group by computing all normalized fundamental units.

We go beyond this: compute $\mathcal{O}_K^*$.
Could use Eisenträger–Hallgren–Kitaev–Song, but we don't want to wait for quantum computers.

1966 Wada: exponential-time $\mathcal{O}_K^*$ algorithm for multiquadratics.

1966 Wada: exponential-time $\mathcal{O}_K^*$ algorithm for multiquadratics.

First step: Recursively compute unit groups for three proper subfields $K_\sigma, K_\tau, K_{\sigma\tau}$ of K.
Base cases: $\mathbf{Q}$; $\mathbf{Q}(\sqrt{d})$.

σ, τ: distinct non-identity automorphisms of K.
$K_\sigma = \{x \in K : \sigma(x) = x\}$.

1966 Wada: exponential-time $\mathcal{O}_K^*$ algorithm for multiquadratics.

First step: Recursively compute unit groups for three proper subfields $K_\sigma, K_\tau, K_{\sigma\tau}$ of K.
Base cases: $\mathbf{Q}$; $\mathbf{Q}(\sqrt{d})$.

σ, τ: distinct non-identity automorphisms of K.
$K_\sigma = \{x \in K : \sigma(x) = x\}$.

e.g. $K = \mathbf{Q}(\sqrt{2}, \sqrt{3}, \sqrt{5})$, appropriate σ, τ: have
$K_\sigma = \mathbf{Q}(\sqrt{2}, \sqrt{3})$;
$K_\tau = \mathbf{Q}(\sqrt{2}, \sqrt{5})$;
$K_{\sigma\tau} = \mathbf{Q}(\sqrt{2}, \sqrt{15})$.

Second step:

Compute $U = \mathcal{O}^*_{K_\sigma} \mathcal{O}^*_{K_\tau} \sigma(\mathcal{O}^*_{K_{\sigma\tau}})$.

Second step:

Compute $U = \mathcal{O}^*_{K_\sigma} \mathcal{O}^*_{K_\tau} \sigma(\mathcal{O}^*_{K_{\sigma\tau}})$.

Fact: $U \le \mathcal{O}^*_K$.

Second step:

Compute $U = \mathcal{O}^*_{K_\sigma} \mathcal{O}^*_{K_\tau} \sigma(\mathcal{O}^*_{K_{\sigma\tau}})$.

Fact: $U \le \mathcal{O}^*_K$.

Fact: $(\mathcal{O}^*_K)^2 \le U$.

Second step:

Compute $U = \mathcal{O}^*_{K_\sigma} \mathcal{O}^*_{K_\tau} \sigma(\mathcal{O}^*_{K_{\sigma\tau}})$.

Fact: $U \le \mathcal{O}^*_K$.

Fact: $(\mathcal{O}^*_K)^2 \le U$.

Proof:

If $u \in \mathcal{O}^*_K$ then
$u\sigma(u) \in \mathcal{O}^*_{K_\sigma}$;
$u\tau(u) \in \mathcal{O}^*_{K_\tau}$;
$u\sigma(\tau(u)) \in \mathcal{O}^*_{K_{\sigma\tau}}$; so
$u\sigma(u)u\tau(u)/\sigma(u\sigma(\tau(u))) \in U$.

Second step:

Compute $U = \mathcal{O}^*_{K_\sigma} \mathcal{O}^*_{K_\tau} \sigma(\mathcal{O}^*_{K_{\sigma\tau}})$.

Fact: $U \le \mathcal{O}^*_K$.

Fact: $(\mathcal{O}^*_K)^2 \le U$.

Proof:

If $u \in \mathcal{O}^*_K$ then
$u\sigma(u) \in \mathcal{O}^*_{K_\sigma}$;
$u\tau(u) \in \mathcal{O}^*_{K_\tau}$;
$u\sigma(\tau(u)) \in \mathcal{O}^*_{K_{\sigma\tau}}$; so
$u\sigma(u)u\tau(u)/\sigma(u\sigma(\tau(u))) \in U$.
In other words, $u^2 \in U$.

Third step:
identify $(\mathcal{O}_K^*)^2$ inside U by trying to compute square roots of products of generators of U.

Third step:
identify $(\mathcal{O}_K^*)^2$ inside U by trying to compute square roots of products of generators of U.

$2^{\Theta(2^n)}$ products.

Third step:
identify $(\mathcal{O}_K^*)^2$ inside U by
trying to compute square roots
of products of generators of U.

$2^{\Theta(2^n)}$ products.

We do much better using
an NFS idea from 1991 Adleman.

Third step:
identify $(\mathcal{O}_K^*)^2$ inside U by
trying to compute square roots
of products of generators of U.

$2^{\Theta(2^n)}$ products.

We do much better using
an NFS idea from 1991 Adleman.

$\alpha_1^{e_1} \cdots \alpha_k^{e_k}$ square $\Rightarrow$
$\chi(\alpha_1)^{e_1} \cdots \chi(\alpha_k)^{e_k} = 1$
for any quadratic character χ
with $\chi(\alpha_1), \ldots, \chi(\alpha_k) \in \{-1, 1\}$.

Third step:
identify $(\mathcal{O}_K^*)^2$ inside U by trying to compute square roots of products of generators of U.

$2^{\Theta(2^n)}$ products.

We do much better using an NFS idea from 1991 Adleman.

$\alpha_1^{e_1} \cdots \alpha_k^{e_k}$ square $\Rightarrow$
$\chi(\alpha_1)^{e_1} \cdots \chi(\alpha_k)^{e_k} = 1$
for any quadratic character χ with $\chi(\alpha_1), \ldots, \chi(\alpha_k) \in \{-1, 1\}$.

Linear equation, usually reducing $\dim\{e\}$ by 1. Use many such χ.

Computing generators

Main goal: Find g given gR,
where $R = \mathbf{Z}[\sqrt{d_1}, \ldots, \sqrt{d_n}]$.

Computing generators

Main goal: Find g given gR,
where $R = \mathbf{Z}[\sqrt{d_1}, \ldots, \sqrt{d_n}]$.

Strategy: Reuse the equation
$g^2 = g\sigma(g)g\tau(g)/\sigma(g\sigma(\tau(g)))$.
Square root of g^2 is $\pm g$.

Computing generators

Main goal: Find g given gR,
where $R = \mathbf{Z}[\sqrt{d_1}, \ldots, \sqrt{d_n}]$.

Strategy: Reuse the equation
$g^2 = g\sigma(g)g\tau(g)/\sigma(g\sigma(\tau(g)))$.
Square root of g^2 is $\pm g$.

How to compute $g\sigma(g)$?

Computing generators

Main goal: Find g given gR,
where $R = \mathbf{Z}[\sqrt{d_1}, \ldots, \sqrt{d_n}]$.

Strategy: Reuse the equation
$g^2 = g\sigma(g)g\tau(g)/\sigma(g\sigma(\tau(g)))$.
Square root of g^2 is $\pm g$.

How to compute $g\sigma(g)$?

First compute relative norm
of ideal gR from K to K_σ.
Obtain ideal generated by $g\sigma(g)$.

Computing generators

Main goal: Find g given gR,
where $R = \mathbf{Z}[\sqrt{d_1}, \ldots, \sqrt{d_n}]$.

Strategy: Reuse the equation
$g^2 = g\sigma(g)g\tau(g)/\sigma(g\sigma(\tau(g)))$.
Square root of g^2 is $\pm g$.

How to compute $g\sigma(g)$?

First compute relative norm
of ideal gR from K to K_σ.
Obtain ideal generated by $g\sigma(g)$.

Recursively compute a generator
of this ideal: probably not $g\sigma(g)$.
Some $ug\sigma(g)$ with $u \in \mathcal{O}^*_{K_\sigma}$.

Unit multiple of $g\sigma(g)$,
unit multiple of $g\tau(g)$,
unit multiple of $g\sigma(\tau(g))$
$\Rightarrow$ some ug^2 with $u \in \mathcal{O}_K^*$.

Unit multiple of $g\sigma(g)$,
unit multiple of $g\tau(g)$,
unit multiple of $g\sigma(\tau(g))$
$\Rightarrow$ some ug^2 with $u \in \mathcal{O}_K^*$.

Use quadratic characters
(with values ± 1 on g)
to identify $v \in \mathcal{O}_K^*$
such that vug^2 is a square.

Unit multiple of $g\sigma(g)$,
unit multiple of $g\tau(g)$,
unit multiple of $g\sigma(\tau(g))$
$\Rightarrow$ some ug^2 with $u \in \mathcal{O}_K^*$.

Use quadratic characters
(with values ± 1 on g)
to identify $v \in \mathcal{O}_K^*$
such that vug^2 is a square.

Now compute square root:
some unit multiple of g,
i.e., some g' with $g'\mathcal{O}_K = g\mathcal{O}_K$.

Unit multiple of $g\sigma(g)$,
unit multiple of $g\tau(g)$,
unit multiple of $g\sigma(\tau(g))$
$\Rightarrow$ some ug^2 with $u \in \mathcal{O}_K^*$.

Use quadratic characters
(with values ± 1 on g)
to identify $v \in \mathcal{O}_K^*$
such that vug^2 is a square.

Now compute square root:
some unit multiple of g,
i.e., some g' with $g'\mathcal{O}_K = g\mathcal{O}_K$.

All of this takes quasipoly time.

Computing short generators

Assume $d_1, \ldots, d_n \geq 2^{1.03n}$.
(More work seems to push bound to $<n^2$; see paper and software.)

Computing short generators

Assume $d_1, \ldots, d_n \geq 2^{1.03n}$.
(More work seems to push bound to $<n^2$; see paper and software.)

Find multiquadratic (MQ) units.
Find all units.
Find some generator ug.

Computing short generators

Assume $d_1, \ldots, d_n \geq 2^{1.03n}$.
(More work seems to push bound to $<n^2$; see paper and software.)

Find multiquadratic (MQ) units.
Find all units.
Find some generator ug.

Heuristic: For most $d_1, \ldots, d_n$, all regulators $\log \varepsilon$ are larger than $2^{0.51n}$; so coefficients of $2^n \operatorname{Log} g$ on MQ unit basis are almost certainly in $(-0.1, 0.1)$.

u^{2^n} is an MQ unit.
$\operatorname{Log} u^{2^n} = 2^n \operatorname{Log} u$ is
closest vector to $2^n \operatorname{Log} ug$.

u^{2^n} is an MQ unit.
$\operatorname{Log} u^{2^n} = 2^n \operatorname{Log} u$ is
closest vector to $2^n \operatorname{Log} ug$.

MQ unit lattice is orthogonal.
Round $2^n \operatorname{Log} ug$ to find $2^n \operatorname{Log} u$
and $2^n \operatorname{Log} g$. Deduce $\pm g^{2^n}$.

u^{2^n} is an MQ unit.
$\operatorname{Log} u^{2^n} = 2^n \operatorname{Log} u$ is
closest vector to $2^n \operatorname{Log} ug$.

MQ unit lattice is orthogonal.
Round $2^n \operatorname{Log} ug$ to find $2^n \operatorname{Log} u$
and $2^n \operatorname{Log} g$. Deduce $\pm g^{2^n}$.

Use quadratic character: g^{2^n}.

u^{2^n} is an MQ unit.
$\operatorname{Log} u^{2^n} = 2^n \operatorname{Log} u$ is
closest vector to $2^n \operatorname{Log} ug$.

MQ unit lattice is orthogonal.
Round $2^n \operatorname{Log} ug$ to find $2^n \operatorname{Log} u$
and $2^n \operatorname{Log} g$. Deduce $\pm g^{2^n}$.

Use quadratic character: g^{2^n}.
Square root: $\pm g^{2^{n-1}}$.

u^{2^n} is an MQ unit.
$\operatorname{Log} u^{2^n} = 2^n \operatorname{Log} u$ is
closest vector to $2^n \operatorname{Log} ug$.

MQ unit lattice is orthogonal.
Round $2^n \operatorname{Log} ug$ to find $2^n \operatorname{Log} u$
and $2^n \operatorname{Log} g$. Deduce $\pm g^{2^n}$.

Use quadratic character: g^{2^n}.
Square root: $\pm g^{2^{n-1}}$.
Use quadratic character: $g^{2^{n-1}}$.
Square root: $\pm g^{2^{n-2}}$.

u^{2^n} is an MQ unit.
$\operatorname{Log} u^{2^n} = 2^n \operatorname{Log} u$ is
closest vector to $2^n \operatorname{Log} ug$.

MQ unit lattice is orthogonal.
Round $2^n \operatorname{Log} ug$ to find $2^n \operatorname{Log} u$
and $2^n \operatorname{Log} g$. Deduce $\pm g^{2^n}$.

Use quadratic character: g^{2^n}.
Square root: $\pm g^{2^{n-1}}$.
Use quadratic character: $g^{2^{n-1}}$.
Square root: $\pm g^{2^{n-2}}$.
$\vdots$
Square root: $\pm g$. Done!
MQ cryptosystem is broken
for all of these fields.

Slightly simpler:

Find MQ units,
but skip finding all units.

Slightly simpler:

Find MQ units,
but skip finding all units.

Recursively find $ug^{2^{n-1}}$
where u is an MQ unit; i.e.,
skip square-root computations.

Slightly simpler:

Find MQ units,
but skip finding all units.

Recursively find $ug^{2^{n-1}}$
where u is an MQ unit; i.e.,
skip square-root computations.

Take logs: $\operatorname{Log} ug^{2^{n-1}}$.
Round: $\operatorname{Log} u$.

Slightly simpler:

Find MQ units,
but skip finding all units.

Recursively find $ug^{2^{n-1}}$
where u is an MQ unit; i.e.,
skip square-root computations.

Take logs: $\operatorname{Log} ug^{2^{n-1}}$.
Round: $\operatorname{Log} u$.

Deduce $\pm g^{2^{n-1}}$.
Use quadratic character: $g^{2^{n-1}}$.
Square root: $\pm g^{2^{n-2}}$.
$\vdots$
Square root: $\pm g$.