Algorithms for multiquadratic number fields

D. J. Bernstein

Jens Bauch, Daniel J. Bernstein, Henry de Valence, Tanja Lange, Christine van Vredendaal.


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.
Algorithms for multiquadratic number fields
D. J. Bernstein

Jens Bauch, Daniel J. Bernstein, Henry de Valence, Tanja Lange, Christine van Vredendaal.

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.

Assumption 1: User chooses a (“small $h^+$”) cyclotomic field as the underlying number field.
Algorithms for multiquadratic number fields
D. J. Bernstein

Jens Bauch, Daniel J. Bernstein, Henry de Valence, Tanja Lange, Christine van Vredendaal.

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.

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

Assumption 2: Attacker has a large quantum computer.
Algorithms for multiquadratic number fields
D. J. Bernstein


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.

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?
Algorithms for multiquadratic number fields

D. J. Bernstein

Jens Bauch, Daniel J. Bernstein, Henry de Valence, Tanja Lange, Christine van Vredendaal.


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.

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.”
Algorithms for multiquadratic number fields

D. J. Bernstein
Jens Bauch, Daniel J. Bernstein, Henry de Valence, Tanja Lange, Christine van Vredendaal.


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.

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 ... brought to bear against these 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 no better than their generic counterparts, both in theory and in practice.”
Algorithms for multiquadratic number fields
D. J. Bernstein, Jens Bauch, Daniel J. Bernstein, Henry de Valence, Tanja Lange, Christine van Vredendaal.


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

Assumption 1: User chooses a 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: short element \( g \) of \( R \), e.g., ring of integers \( \mathcal{O}_K \) of a cyclotomic field \( K \).

Public key: ideal \( gR \).
Gentry's fully homomorphic cryptosystem "Fully homomorphic encryption using ideal lattices" was broken several years later, under reasonable assumptions.

Assumption 1: User chooses a 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 $gR$. 

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

Assumption 1: User chooses a 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 \( gR \).
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 $gR$. 
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 $gR$.

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 $gR$.


Compare to 2013 Lyubashevsky–Peikert–Regev: “All of the algebraic and algorithmic tools (including quantum computation) we employ ... can also be brought to bear against SVP and other problems on ideal lattices. 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 $gR$.

Attack stage 1, quantum: SODA 2016 Biasse–Song finds some generator of $gR$.


Standard algebraic-number-theory view of all generators of $gR$, i.e., all $ug$ where $u \in R^*$: 

Log $u$ ranges over Dirichlet’s log-unit lattice; 

Log $ug = \log u + \log g$. 

All of the algebraic and algorithmic tools (including quantum computation) 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 $gR$.

Attack stage 1, quantum: SODA 2016 Biasse–Song finds some generator of $gR$.


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

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 $gR$.

Attack stage 1, quantum: SODA 2016 Biasse–Song finds some generator of $gR$.


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

Attack stage 2, cyclotomic:

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


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

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

\[
\text{Log } u \text{ ranges over Dirichlet’s log-unit lattice;} \\
\text{Log } ug = \text{Log } u + \text{Log } g. \\
\text{Given any generator } ug, \text{ try to find short Log } g \text{ by finding lattice vector Log } u \text{ close to Log } ug.
\]
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 $gR$.

Attack stage 2, cyclotomic:

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

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

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

Stage 1, quantum: SODA 2016 Biasse–Song finds some generator of $gR$.


Standard algebraic-number-theory view of all generators of $gR$, i.e., all $ug$ where $u \in R^*$:

$\log u$ ranges over Dirichlet’s log-unit lattice;

$\log ug = \log u + \log g$.

Given any generator $ug$, try to find short $\log g$ by finding lattice vector $\log u$ close to $\log ug$.

Apply, e.g., embedding or Babai, starting from basis for $\log R^*$?

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.
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: Biasse–Song finds some generator of \( gR \).

Builds on Eisenträger–Hallgren–Kitaev–Song algorithm for \( R^* \).


Standard algebraic-number-theory view of all generators of \( gR \), i.e., all \( ug \) where \( u \in R^* \):

\[ \log u \] ranges over Dirichlet’s log-unit lattice;

\[ \log ug = \log u + \log g. \]

Given any generator \( ug \), try to find short \( \log g \) by finding lattice vector \( \log u \) close to \( \log ug \).

Apply, e.g., embedding or Babai, starting from basis for \( \log R^* \)?

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

$R$: e.g., ring of integers $O_K$ of a cyclotomic field $K$.

Public key: ideal $gR$.

Attack stage 1, quantum:
SODA 2016 Biasse–Song finds some generator of $gR$.

Attack stage 2, cyclotomic:

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

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

Apply, e.g., embedding or Babai, starting from basis for $\text{Log } R^*$?
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.
Standard algebraic-number-theory view of all generators of $gR$, i.e., all $ug$ where $u \in R^*$:
Log $u$ ranges over Dirichlet’s log-unit lattice;
Log $ug = \text{Log } u + \text{Log } g$.

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

Apply, e.g., embedding or Babai, starting from basis for 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.
Standard algebraic-number-theory view of all generators of $g\mathcal{R}$, i.e., all $ug$ where $u \in \mathcal{R}^*$:

- Log $u$ ranges over Dirichlet’s log-unit lattice;
- $\log ug = \log u + \log g$.

Given any generator $ug$, try to find short $\log g$ by finding lattice vector $\log u$ close to $\log ug$.

Apply, e.g., embedding or Babai, starting from basis for $\log \mathcal{R}^*$?

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.

Take, e.g., $\zeta = \exp(2\pi i/1024)$; field $\mathbb{Q}(\zeta)$; ring $R = \mathbb{Z}[\zeta]$. 
Standard algebraic-number-theory view of all generators of $gR$, i.e., all $ug$ where $u \in R^*$:

Log $u$ ranges over Dirichlet’s log-unit lattice;

Log $ug = \text{Log} u + \text{Log} g$.

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

Apply, e.g., embedding or Babai, starting from basis for 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.

Take, e.g., $\zeta = \exp(2\pi i/1024)$;

field $\mathbb{Q}(\zeta)$; ring $R = \mathbb{Z}[\zeta]$.

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

(\zeta^3 - 1)/(\zeta - 1) is a unit:
Standard algebraic-number-theory view of all generators of $gR_R$, i.e., all $ug$ where $u \in R^*$:

\[
\log u \text{ ranges over Dirichlet's log-unit lattice; } \\
\log ug = \log u + \log g.
\]

Given any generator $ug$, try to find short $\log g$ by finding lattice vector $\log u$ close to $\log ug$.

Apply, e.g., embedding or Babai, starting from basis for $\log R^*$?

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.

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

$(\zeta^3 - 1)/(\zeta - 1)$ 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.
Standard algebraic-number-theory view of all generators of $gR$, i.e., all $ug$ where $u \in R^*$:

Log $u$ ranges over Dirichlet’s log-unit lattice;

Log $ug = \text{Log } u + \text{Log } g$.

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

Apply, e.g., embedding or Babai, starting from basis for 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.

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

$(\zeta^3 - 1)/(\zeta - 1)$ 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 $g$. 

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 $\mathbb{Q}(\zeta)$; ring $R = \mathbb{Z}[\zeta]$.

$(\zeta^3 - 1)/(\zeta - 1)$ 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 $g$. 

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 $\mathbb{Q}(\zeta)$; ring $R = \mathbb{Z}[\zeta]$.

$(\zeta^3 - 1)/(\zeta - 1)$ 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 $g$. 

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 $\mathbb{Q}(\zeta)$; ring $R = \mathbb{Z}[\zeta]$.

$(\zeta^3 - 1)/(\zeta - 1)$ 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 $g$. 
Standard algebraic-number-theory view of all generators of \( gR \), i.e., all \( ug \) where \( u \in R^* \): log ranges over \( \log \)’s log-unit lattice; \( \log u \) ranges over \( \log u + \log g \).

Try to find short \( \log g \) by finding lattice vector \( \log u \) close to \( \log ug \).

For any generator \( ug \), try to find short \( \log g \) by finding lattice vector \( \log u \) close to \( \log ug \).

Try, e.g., embedding or Babai, starting from basis for \( \log R^* \)?

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.

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

\( (\zeta^3 - 1)/(\zeta - 1) \) 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 \( g \).
Standard algebraic-number-theory view of all generators of $gR$, i.e., all $ug$ where $u \in R^*$:

Log $u$ ranges over Dirichlet's log-unit lattice; Log $ug$ = Log $u$ + Log $g$.

For $ug$, try to find short Log $g$ by finding lattice vector Log $u$ close to Log $ug$.

Embedding or Babai, for 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.

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

$(\zeta^3 - 1)/(\zeta - 1)$ 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?
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 $\mathbb{Q}(\zeta)$; ring $R = \mathbb{Z}[\zeta]$.

$(\zeta^3 - 1)/(\zeta - 1)$ 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?
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 $\mathbb{Q}(\zeta)$; ring $R = \mathbb{Z}[\zeta]$.

$(\zeta^3 - 1)/(\zeta - 1)$ 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?
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 $\mathbb{Q}(\zeta)$; ring $R = \mathbb{Z}[\zeta]$.

$(\zeta^3 - 1)/(\zeta - 1)$ 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?

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.”
For cyclotomic fields, often $u$ is a “cyclotomic unit”. Known textbook basis for cyclotomic units is a short basis. Take, e.g., $\zeta = \exp(2\pi i/1024)$; ring $R = \mathbb{Z}[\zeta]$.

$\mathbb{Z}[(\zeta - 1)]$ is a unit: directly invert, or apply $\zeta \mapsto \zeta^3$ automorphism to factors of $\zeta - 1$.

$\mathbb{Z}[(\zeta^3 - 1)]$ is a unit.

$\mathbb{Z}[(\zeta^9 - 1)]$ is a unit.

Obtain short basis. 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?

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 $I$. 
For cyclotomic fields, often \( u \) is a "cyclotomic unit". Known textbook basis for cyclotomic units is a short basis.

Take, e.g.,
\[
\exp(2\pi i / 1024); \\
\mathbb{Z}[\zeta].
\]

is a unit:

apply \( \zeta \mapsto \zeta^3 \) factors of \( \zeta - 1 \).

is a unit.

is a unit.

is a short basis.

easily finds \( g \).

Are you a lattice salesman? Try to dismiss lattice attacks. Ask: Do attacks against

- the \( g R \mapsto g \) 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 \( I \).
For cyclotomic fields, often \( u \) is a "cyclotomic unit". Known textbook basis for cyclotomic units is a short basis. Take, e.g., 
\[
\zeta = \exp(2\pi i \cdot 1024);
\]
field \( \mathbb{Q}(\zeta) \); ring \( R = \mathbb{Z}[\zeta] \).

\((\zeta^3 - 1) = (\zeta - 1) \) is a unit: directly invert, or apply \( \zeta \mapsto \zeta^3 \) 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 \( g R \mapsto g \) 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": \( I \mapsto \text{shortest nonzero vector in } I \).

"Approximate Ideal-SVP": \( I \mapsto \text{short nonzero vector in } I \).
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 $I$.

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

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.”
Are you a lattice salesman? Try to dismiss lattice attacks. Do attacks against
\( \mathbb{R} \mapsto g \) problem,
Gentry’s original FHE system,
original Garg–Gentry–Halevi
multilinear maps, ... 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”: \( I \mapsto \text{shortest nonzero vector in } I \).

“Approximate Ideal-SVP”: \( I \mapsto \text{short nonzero vector in } I \).

Attack is against ideal \( I \) with a \textit{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,
we simply hadn’t realized
that the added guarantee of a
short generator would transform
the technique from useless to
devastatingly effective.”

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...
Are you a lattice salesman? Try to dismiss lattice attacks. Ask: Do attacks against
• the $gR \rightarrow g$ 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 $I$.

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 attacking Ideal-SVP.
For commonly used principal ideals are extremely small fraction of all ideals. ... The weakness is not so much due to the structure of cyclotomics, but rather to the extra structure of principal ideals that have short generators.”
Are you a lattice salesman?
Try to dismiss lattice attacks.
Ask: Do attacks against
• the \( gR \rightarrow g \) 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."

---

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.”

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.”

"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 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.”
“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.”
"Ideal-SVP":
shortest nonzero vector in $I$.

"Approximate Ideal-SVP":
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:
"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."

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.
"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 also says idea is limited to principal ideals: “Although cyclotomies 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 cyclotomies, but rather to the extra structure of principal ideals that have short generators.”

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.
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 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. Start from Biasse–Song, use more features of cyclotomic.
2015 Peikert also says idea is limited to principal ideals: “Although cyclotomies 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 cyclotomies, 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-$N$ cyclotomies, under plausible assumptions about class-group generators etc. Start from Biasse–Song, use more features of cyclotomic fields.
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.

Can techniques be pushed to smaller approx factors? Can techniques be adapted to break, e.g., Ring-LWE?
Peikert also says idea is limited to principal ideals: although cyclotomics have a lot of structure, nobody has 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 $2N^{1/2+o(1)}$ 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.
Peikert also says idea is limited to principal ideals: "Although cyclotomics have a lot of structure, nobody has been able to exploit it in attacks on 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.

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.
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.
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.
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.

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

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

Can cyclotomic attacks on Gentry be extended to these systems?
Actually, the idea produces attacks far beyond this case.

2016 Cramer–Ducas–Wesolowski: Ideal-SVP attack for approx factor $2^{N_1}$ in deg-$N$ cyclotomics, under plausible assumptions about class-group generators etc. 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.

Most lattice-based enc systems use power-of-2 cyclotomics. Some non-power-of-2 cyclotomics:
LIMA has $\Phi_{1019}$ option, “more conservative choice of field”;
NTRU-HRSS-KEM uses $\Phi_{701}$;
NTRUEncrypt uses $\Phi_{743}$ 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”.
Actually, the idea produces attacks far beyond this case.

2016 Cramer–Ducas–Wesolowski: Ideal-SVP attack for approx factor \( 2^{N/1+o(1)} \) 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.

Most lattice-based enc systems use power-of-2 cyclotomics. Some non-power-of-2 cyclotomics: LIMA has \( \Phi_{1019} \) option, “more conservative choice of field”; NTRU-HRSS-KEM uses \( \Phi_{701} \); NTRUEncrypt uses \( \Phi_{743} \) 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.”
Actually, the idea produces attacks far beyond this case.

2016 Cramer–Ducas–Wesolowski: Ideal-SVP attack for approx factor \(2^{\omega(1)}\) in deg-

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 \(\Phi_{1019}\) option, “more conservative choice of field”;
- NTRU-HRSS-KEM uses \(\Phi_{701}\);
- NTRUEncrypt uses \(\Phi_{743}\) 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”.

Some systems avoid cyclotomics.
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 $\Phi_{1019}$ option, “more conservative choice of field”;
NTRU-HRSS-KEM uses $\Phi_{701}$;
NTRUEncrypt uses $\Phi_{743}$ 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”.
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 $\Phi_{1019}$ option, “more conservative choice of field”; NTRU-HRSS-KEM uses $\Phi_{701}$; NTRUEncrypt uses $\Phi_{743}$ 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”.
Titanium-lite, 14720-byte key: uses “middle product” to “hedge against the weakness of specific polynomial rings”.
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 $\Phi_{1019}$ option, “more conservative choice of field”;
- NTRU-HRSS-KEM uses $\Phi_{701}$;
- NTRUEncrypt uses $\Phi_{743}$ 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”.

- 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.
NIST post-quantum competition 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 $\Phi_{1019}$ option, “more conservative choice of field”; NTRU-HRSS-KEM uses $\Phi_{701}$; NTRUEncrypt uses $\Phi_{743}$ 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”.

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 $C_F$ ⇒ attack against problem $L_F$”, where $L_F$ is a “lattice problem”.

Theory 2: But are choices where $L_F$ is a “lattice problem” where $L_F$ is a “lattice problem”. 
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 $\Phi_{1019}$ option, “more conservative choice of field”;
- NTRU-HRSS-KEM uses $\Phi_{701}$;
- NTRUEncrypt uses $\Phi_{743}$ 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”.

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”.

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”.

Gentry 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”.

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”.
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\textsuperscript{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”.

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

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.
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”.

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 $F$?
Systems avoid cyclotomics.

FrodoKEM-640, 9616-byte key: matrix rings; says that commutative rings “have 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 $C_F$ ⇒ 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 $d_1; \ldots; d_n \in \mathbb{Z}$; $\mathbb{Q}\sqrt{d_j}$ for each nonempty subset $J \subseteq \{1; \ldots; n\}$.

$K = \mathbb{Q}(\sqrt{d_1}; \ldots; \sqrt{d_n})$: smallest subfield of $\mathbb{C}$ containing $\sqrt{d_j}$ for each subset $J$.

$K$ is a degree-$2^n$ number field.

Basis: $\mathbb{Q}\sqrt{d_j}$ for each subset $J$.

e.g. $\mathbb{Q}(\sqrt{2}; \sqrt{3}) = \mathbb{Q} \oplus \mathbb{Q}\sqrt{2} \oplus \mathbb{Q}\sqrt{3} \oplus \mathbb{Q}\sqrt{6}$.
Some systems avoid cyclotomics.

FrodoKEM-640, 9616-byte key:
relies on matrix rings; says that
commutative rings “have
weaknesses
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”.

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 $d_1,\ldots,\prod_{j \in J} d_j$ non-square for each
nonempty subset $J \subseteq \{1;\ldots;n\}$.

\[
K = \mathbb{Q}(\sqrt{d_1},\ldots,\sqrt{\prod_{j \in J} d_j})
\]

$K$ is a degree-$2^n$ number field.

Basis: $\prod_{j \in J} d_j$ for $J$, each
subset $J \subseteq \{1,\ldots,n\}$.

e.g. \(\mathbb{Q}(\sqrt{2},\sqrt{3}) = \mathbb{Q} \oplus \mathbb{Q}\sqrt{2} \oplus \mathbb{Q}\sqrt{3}\).
Two theories of lattice safety

Theory 1: Best choices of field $F$ are choices where we know proofs “attack against cryptosystem $C_F$ ⇒ 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 $d_1, \ldots, d_n \in \mathbb{Z}$; $\prod_{j \in J} d_j$ non-square for each nonempty subset $J \subseteq \{1, \ldots, n\}$.

$K = \mathbb{Q}(\sqrt{d_1}, \ldots, \sqrt{d_n})$: smallest subfield of $\mathbb{C}$ 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. $\mathbb{Q}(\sqrt{2}, \sqrt{3}) = \mathbb{Q} \oplus \mathbb{Q}\sqrt{2} \oplus \mathbb{Q}\sqrt{3} \oplus \mathbb{Q}\sqrt{6}$. 
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 $d_1, \ldots, d_n \in \mathbb{Z}$; $\prod_{j \in J} d_j$ non-square for each nonempty subset $J \subseteq \{1, \ldots, n\}$.

$K = \mathbb{Q}(\sqrt{d_1}, \ldots, \sqrt{d_n})$: smallest subfield of $\mathbb{C}$ 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. $\mathbb{Q}(\sqrt{2}, \sqrt{3}) = \mathbb{Q} \oplus \mathbb{Q}\sqrt{2} \oplus \mathbb{Q}\sqrt{3} \oplus \mathbb{Q}\sqrt{6}$.
Theories of lattice safety

Theory 1: Best choices of field $F$ are choices where we know proofs "attack against cryptosystem $C_F$⇒ attack against problem $L_F"$, $L_F$ 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: $n$ ∈ {0, 1, 2, ...}; squarefree $d_1, \ldots, d_n$ ∈ $\mathbb{Z}$; $\prod_{j \in J} d_j$ non-square for each nonempty subset $J \subseteq \{1, \ldots, n\}$.

$K = \mathbb{Q}(\sqrt{d_1}, \ldots, \sqrt{d_n})$: smallest subfield of $\mathbb{C}$ 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. $\mathbb{Q}(\sqrt{2}, \sqrt{3}) = \mathbb{Q} \oplus \mathbb{Q}\sqrt{2} \oplus \mathbb{Q}\sqrt{3} \oplus \mathbb{Q}\sqrt{6}$.

This field has $2^n$ automorphisms, e.g. automorphisms of $\mathbb{Q}(\sqrt{2}; \sqrt{3})$ map $a + b\sqrt{2} + c\sqrt{3} + d\sqrt{6}$ to $\ldots$. 

$a + b\sqrt{2} + c\sqrt{3} + d\sqrt{6}$; $a - b\sqrt{2} + c\sqrt{3} - d\sqrt{6}$; $a + b\sqrt{2} - c\sqrt{3} - d\sqrt{6}$; $a - b\sqrt{2} - c\sqrt{3} + d\sqrt{6}$. 
Two theories of lattice safety

Theory 1: Best choices of field $F$ are choices where we know proofs "attack against cryptosystem $C_F$ ⇒ 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 $d_1, \ldots, d_n \in \mathbb{Z}$; $\prod_{j \in J} d_j$ non-square for each nonempty subset $J \subseteq \{1, \ldots, n\}$.

$K = \mathbb{Q}(\sqrt{d_1}, \ldots, \sqrt{d_n})$: smallest subfield of $\mathbb{C}$ 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. $\mathbb{Q}(\sqrt{2}, \sqrt{3}) = \mathbb{Q} \oplus \mathbb{Q}\sqrt{2} \oplus \mathbb{Q}\sqrt{3} \oplus \mathbb{Q}\sqrt{6}$.

This field is Galois: has $2^n$ automorphisms.

e.g. automorphisms map $a + b\sqrt{2} + c\sqrt{3} + d\sqrt{6}$ to:

$a + b\sqrt{2} + c\sqrt{3} + d\sqrt{6}$;
$a - b\sqrt{2} + c\sqrt{3} - d\sqrt{6}$;
$a + b\sqrt{2} - c\sqrt{3} - d\sqrt{6}$;
$a - b\sqrt{2} - c\sqrt{3} + d\sqrt{6}$. 
Two theories of lattice safety

Theory 1: Best choices of field \( F \) are choices where we know proofs

\[
\text{"attack against cryptosystem } C \text{ against problem } L \quad F \Rightarrow \text{attack against problem } L.
\]

\( L \) is a "lattice problem".

Intuitive flaw in theory 1: Maybe these choices make \( L \) 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:

\[ n \in \{ 0, 1, 2, \ldots \} ; \]

\[ \text{squarefree} \quad d_1, \ldots, d_n \in \mathbb{Z} ; \]

\[ \prod_{j \in J} d_j \quad \text{non-square for each nonempty subset } J \subseteq \{ 1, \ldots, n \} . \]

\( K = \mathbb{Q}(\sqrt{d_1}, \ldots, \sqrt{d_n}) : \)

smallest subfield of \( \mathbb{C} \) 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. \( \mathbb{Q}(\sqrt{2}, \sqrt{3}) = \mathbb{Q} \oplus \mathbb{Q} \sqrt{2} \oplus \mathbb{Q} \sqrt{3} \oplus \mathbb{Q} \sqrt{6} \).

This field is Galois:

has \( 2^n \) automorphisms.

e.g. automorphisms of \( \mathbb{Q}(\sqrt{2}, \sqrt{3}) \) map

\[
a + b\sqrt{2} + c\sqrt{3} + d\sqrt{6}\]

\[
a - b\sqrt{2} + c\sqrt{3} - d\sqrt{6};
\]

\[
a + b\sqrt{2} - c\sqrt{3} - d\sqrt{6};
\]

\[
a - b\sqrt{2} - c\sqrt{3} + d\sqrt{6}.
\]
Multiquadratic fields

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

$K = \mathbb{Q}(\sqrt{d_1}, \ldots, \sqrt{d_n})$: smallest subfield of $\mathbb{C}$ 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. $\mathbb{Q}(\sqrt{2}, \sqrt{3}) = 
\mathbb{Q} \oplus \mathbb{Q}\sqrt{2} \oplus \mathbb{Q}\sqrt{3} \oplus \mathbb{Q}\sqrt{6}$.

This field is Galois: has $2^n$ automorphisms. 

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

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

\( K = \mathbb{Q}(\sqrt{d_1}, \ldots, \sqrt{d_n}) \): smallest subfield of \( \mathbb{C} \) 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 \( J \subseteq \{1, \ldots, n\} \).

\( \mathbb{Q}(\sqrt{2}, \sqrt{3}) = \mathbb{Q} \oplus \mathbb{Q}\sqrt{2} \oplus \mathbb{Q}\sqrt{3} \oplus \mathbb{Q}\sqrt{6} \).

This field is Galois:
has \( 2^n \) automorphisms.
e.g. automorphisms of \( \mathbb{Q}(\sqrt{2}, \sqrt{3}) \) 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 \( \mathbb{Q}(\sqrt{2}, \sqrt{3}) \):
\( \mathbb{Q}(\sqrt{2}, \sqrt{3}), \mathbb{Q}(\sqrt{2}), \mathbb{Q}(\sqrt{3}), \mathbb{Q}(\sqrt{6}), \mathbb{Q}. \)
Multiquadratic fields

Assumptions: \( n \in \{0, 1, 2, \ldots \} \); squarefree \( d_1, \ldots, d_n \in \mathbb{Z} \); non-square for each nonempty subset \( J \subseteq \{1, \ldots, n\} \).

\( \mathbb{Q}(\sqrt{d_1}, \ldots, \sqrt{d_n}) \): subfield of \( \mathbb{C} \) containing \( \sqrt{d_1}, \ldots, \sqrt{d_n} \).

Degree-\(2^n\) number field.

\( \prod_{j \in J} d_j \) for each subset \( J \subseteq \{1, \ldots, n\} \).

\( \mathbb{Q}(\sqrt{2}, \sqrt{3}) = \mathbb{Q}(\sqrt{2}) \oplus \mathbb{Q}\sqrt{3} \oplus \mathbb{Q}\sqrt{6} \).

This field is Galois:
has \(2^n\) automorphisms.
e.g. automorphisms of \( \mathbb{Q}(\sqrt{2}, \sqrt{3}) \) 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 \( \mathbb{Q}(\sqrt{2}, \sqrt{3}) \):
\( \mathbb{Q}(\sqrt{2}, \sqrt{3}) \),
\( \mathbb{Q}(\sqrt{2}), \mathbb{Q}(\sqrt{3}), \mathbb{Q}(\sqrt{6}), \mathbb{Q} \).

Gentry for multiquadratics

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

**Assumptions:**

- $n \in \{0, 1, 2, \ldots\}$;
- $d_n \in \mathbb{Z}$;
- for each nonempty subset $J \subseteq \{1, \ldots, n\}$.

**K** is a degree-$2^n$ number field. For each $J \subseteq \{1, \ldots, n\}$.

**Basis:** $Q_{J_d} \in J_{d_j}$ for each nonempty subset $J \subseteq \{1, \ldots, n\}$.

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

This field is Galois:

- has $2^n$ automorphisms.

**e.g.** automorphisms of $Q(\sqrt{2}, \sqrt{3})$

- 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 $Q(\sqrt{2}, \sqrt{3})$:

- $Q(\sqrt{2}, \sqrt{3})$,
- $Q(\sqrt{2})$, $Q(\sqrt{3})$, $Q(\sqrt{6})$, $Q$.

Gentry for multiquadrics

Use optimizations from PKC 2010 Smart–Vercauteren,
Eurocrypt 2011 Gentry–Halevi.
This field is Galois: has $2^n$ automorphisms.

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

$$
\begin{align*}
  a + b\sqrt{2} + c\sqrt{3} + d\sqrt{6} \\
  a - b\sqrt{2} + c\sqrt{3} - d\sqrt{6} \\
  a + b\sqrt{2} - c\sqrt{3} - d\sqrt{6} \\
  a - b\sqrt{2} - c\sqrt{3} + d\sqrt{6}.
\end{align*}
$$

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

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

$$
\begin{align*}
  \mathbb{Q}(\sqrt{2}, \sqrt{3}), \\
  \mathbb{Q}(\sqrt{2}), \mathbb{Q}(\sqrt{3}), \mathbb{Q}(\sqrt{6}), \mathbb{Q}.
\end{align*}
$$

Gentry for multiquadratics

Use optimizations from PKC 2010 Smart–Vercauteren, Eurocrypt 2011 Gentry–Halevi.
This field is Galois:
has $2^n$ automorphisms.
e.g. automorphisms of $\mathbb{Q}(\sqrt{2}, \sqrt{3})$
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 $\mathbb{Q}(\sqrt{2}, \sqrt{3})$:
$\mathbb{Q}(\sqrt{2}, \sqrt{3}),$
$\mathbb{Q}(\sqrt{2}), \mathbb{Q}(\sqrt{3}), \mathbb{Q}(\sqrt{6}),$
$\mathbb{Q}.$

Gentry for multiquadratics
Use optimizations from
PKC 2010 Smart–Vercauteren,
Eurocrypt 2011 Gentry–Halevi.
This field is Galois: has $2^n$ automorphisms.
e.g. automorphisms of $\mathbb{Q}(\sqrt{2}, \sqrt{3})$ 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 $\mathbb{Q}(\sqrt{2}, \sqrt{3})$:
   $\mathbb{Q}(\sqrt{2}, \sqrt{3})$,
   $\mathbb{Q}(\sqrt{2}), \mathbb{Q}(\sqrt{3}), \mathbb{Q}(\sqrt{6}), \mathbb{Q}$.  

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

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

This field is Galois: has $2^n$ automorphisms.
e.g. automorphisms of $\mathbb{Q}(\sqrt{2}, \sqrt{3})$
map $a + b\sqrt{2} + c\sqrt{3} + d\sqrt{6}$ to
\begin{align*}
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}.
\end{align*}
About $2^{n^2/4}$ subfields.
e.g. subfields of $\mathbb{Q}(\sqrt{2}, \sqrt{3})$: $\mathbb{Q}(\sqrt{2}, \sqrt{3}),$
$\mathbb{Q}(\sqrt{2}), \mathbb{Q}(\sqrt{3}), \mathbb{Q}(\sqrt{6}),$
$\mathbb{Q}.$

Gentry for multiquadratics
Use optimizations from
PKC 2010 Smart–Vercauteren,
Eurocrypt 2011 Gentry–Halevi.
$F$: monic irreducible polynomial.
Ring $R = \mathbb{Z}[x]/F$; not required
to be ring of integers of $\mathbb{Q}[x]/F$.
Multiquadratics: take, e.g.,
$F = (x - \sqrt{2} - \sqrt{3}) \cdot$
$\quad (x + \sqrt{2} - \sqrt{3}) \cdot$
$\quad (x - \sqrt{2} + \sqrt{3}) \cdot$
$\quad (x + \sqrt{2} + \sqrt{3}).$
Note $\mathbb{Q}(\sqrt{2} + \sqrt{3}) = \mathbb{Q}(\sqrt{2}, \sqrt{3}).$
This field is Galois:
has $2^n$ automorphisms.
e.g. automorphisms of $\mathbb{Q}(\sqrt{2}, \sqrt{3})$
map $a + b\sqrt{2} + c\sqrt{3} + d\sqrt{6}$ to
$a + b\sqrt{2} + c\sqrt{3} + d\sqrt{6};$
$a + b\sqrt{2} + c\sqrt{3} - d\sqrt{6};$
$a - b\sqrt{2} - c\sqrt{3} - d\sqrt{6};$
$a - b\sqrt{2} - c\sqrt{3} + d\sqrt{6}.$
$n^2/4$ subfields.

Fields of $\mathbb{Q}(\sqrt{2}, \sqrt{3})$:  
$\sqrt{3}),$
$\mathbb{Q}(\sqrt{3}), \mathbb{Q}(\sqrt{6}),$
$\mathbb{Q}(\sqrt{2} + \sqrt{3}).$
Note $\mathbb{Q}(\sqrt{2} + \sqrt{3}) = \mathbb{Q}(\sqrt{2}, \sqrt{3}).$

Gentry for multiquadratics
Use optimizations from
PKC 2010 Smart–Vercauteren,
Eurocrypt 2011 Gentry–Halevi.
$F$: monic irreducible polynomial.
Ring $R = \mathbb{Z}[x]/F$; not required
to be ring of integers of $\mathbb{Q}[x]/F$.

Multiquadratics: take, e.g.,
$F = (x - \sqrt{2} - \sqrt{3}) \cdot$
$(x + \sqrt{2} - \sqrt{3}) \cdot$
$(x - \sqrt{2} + \sqrt{3}) \cdot$
$(x + \sqrt{2} + \sqrt{3}).$

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

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

$F$: monic irreducible polynomial.

Ring $R = \mathbb{Z}[x]/F$; not required to be ring of integers of $\mathbb{Q}[x]/F$.

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

Smart–Vercauteren keygen:

Take short random $g \in R$.

Compute $q$, absolute norm of $g$.

Start over if $q$ is not prime.
This field is Galois:
has 2 automorphisms.
e.g. automorphisms of $\mathbb{Q}(\sqrt{2}, \sqrt{3})$ 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 $2^n = 4$ subfields.
e.g. subfields of $\mathbb{Q}(\sqrt{2}, \sqrt{3})$: $\mathbb{Q}(\sqrt{2}, \sqrt{3}), \mathbb{Q}(\sqrt{2}), \mathbb{Q}(\sqrt{3}), \mathbb{Q}(\sqrt{6}), \mathbb{Q}$.

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

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

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

Smart–Vercauteren keygen:
Take short random $g \in R$.
Compute $q$, absolute norm of $g$.
Start over if $q$ is not prime.
Gentry for multiquadratics
Use optimizations from PKC 2010 Smart–Vercauteren, Eurocrypt 2011 Gentry–Halevi.

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

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

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

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

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

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

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

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

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

$F$: monic irreducible polynomial.

Ring $R = \mathbb{Z}[x]/F$; not required to be ring of integers of $\mathbb{Q}[x]/F$.

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

Smart–Vercauteren keygen:

Take short random $g \in R$.

Compute $q$, absolute norm of $g$.

Start over if $q$ is not prime.

Compute root $r$ of $g$ in $\mathbb{Z}/q$.

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

monic irreducible polynomial. 
Ring \( R = \mathbb{Z}[x]/F \); not required 
ing of integers of \( \mathbb{Q}[x]/F \).

Multiquadratics: take, e.g.,
\[
(x - \sqrt{2} - \sqrt{3}) \cdot 
(x + \sqrt{2} - \sqrt{3}) \cdot 
(x + \sqrt{2} + \sqrt{3}) \cdot 
(x - \sqrt{2} + \sqrt{3}).
\]
\( (\sqrt{2} + \sqrt{3}) = \mathbb{Q}(\sqrt{2}, \sqrt{3}) \).

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

Compute root \( r \) of \( g \) in \( \mathbb{Z}/q \).
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.)
Gentry for multiquadratics
Use optimizations from PKC 2010 Smart–Vercauteren, Eurocrypt 2011 Gentry–Halevi.

**F**: monic irreducible polynomial.

**Ring** $\mathbb{R} = \mathbb{Z}[x]_F$; not required to be ring of integers of $\mathbb{Q}[x]/F$.

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 $\mathbb{Q}(\sqrt{2} + \sqrt{3}) = \mathbb{Q}(\sqrt{2}; \sqrt{3})$.

---

**Smart–Vercauteren keygen:**

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

Compute root $r$ of $g$ in $\mathbb{Z}/q$.
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 $m \in \mathbb{Z}[x]/F$.
Ciphertext is $m(r) \in \mathbb{Z}/q$. 

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

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

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 \( \mathbb{Q}(\sqrt{2} + \sqrt{3}) = \mathbb{Q}(\sqrt{2}; \sqrt{3}) \).

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

Compute root \( r \) of \( g \) in \( \mathbb{Z}/q \).
Public key \( g R = q R + (x - r) R \)
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 \( m \in \mathbb{Z}[x]/F \).
Ciphertext is \( m(r) \in \mathbb{Z}/q \).
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 $\mathbb{Z}/q$.
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 $m \in \mathbb{Z}[x]/F$.
Ciphertext is $m(r) \in \mathbb{Z}/q$. 
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 \( \mathbb{Z}/q \).
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 \( m \in \mathbb{Z}[x]/F \).
Ciphertext is \( m(r) \in \mathbb{Z}/q \).

Homomorphic operations:
add/multiply ciphertexts \( m(r) \)
to add/multiply messages \( m \).
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 \( \mathbb{Z}/q \).
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 \( m \in \mathbb{Z}[x]/F \).
Ciphertext is \( m(r) \in \mathbb{Z}/q \).

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

Decryption:
given \( c \in \{0, 1, \ldots, q - 1\} \),
compute \( c/g \in \mathbb{Q}[x]/F \),
round to element of \( \mathbb{Z}[x]/F \),
multiply by \( g \), subtract from \( c \).
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 \( \mathbb{Z}/q \).
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 \( m \in \mathbb{Z}[x]/F \).
Ciphertext is \( m(r) \in \mathbb{Z}/q \).

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

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

Decryption works if each coefficient of \( m/g \in \mathbb{Q}[x]/F \) is in \((-1/2, 1/2)\).
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 \( \mathbb{Z}/q \).
Key \( gR = qR + (x - r)R \) represented as \((q, r)\).
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 \( m \in \mathbb{Z}[x]/F \).
Ciphertext is \( m(r) \in \mathbb{Z}/q \).
Homomorphic operations: add/multiply ciphertexts \( m(r) \) to add/multiply messages \( m \).
Decryption:
given \( c \in \{0, 1, \ldots, q - 1\} \), compute \( c/g \in \mathbb{Q}[x]/F \),
round to element of \( \mathbb{Z}[x]/F \),
multiply by \( g \), subtract from \( c \).
Decryption works if each coefficient of \( m/g \in \mathbb{Q}[x]/F \)
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.
Smart–Vercauteren keygen:
Take short random \( g \in R \).
Compute absolute norm of \( g \).
Start over if \( q \) is not prime.

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 \( m \in \mathbb{Z}[x]/F \).
Ciphertext is \( m(r) \in \mathbb{Z}/q \).

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

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

Decryption works if each coefficient of \( m/g \in \mathbb{Q}[x]/F \)
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.
Smart–Vercauteren encryption: Take short \( m \in \mathbb{Z}[x]/F \).

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

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

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

Decryption works if each coefficient of \( m/g \in \mathbb{Q}[x]/F \) 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.
Smart–Vercauteren encryption:
Take short $m \in \mathbb{Z}[x]/F$.
Ciphertext is $m(r) \in \mathbb{Z}/q$.

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

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

Decryption works if each coefficient of $m/g \in \mathbb{Q}[x]/F$ 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.
Smart–Vercauteren encryption:
Take short \( m \in \mathbb{Z}[x]/F \).
Ciphertext is \( m(r) \in \mathbb{Z}/q \).

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

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

Decryption works if
each coefficient of \( m/g \in \mathbb{Q}[x]/F \)
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.

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.)
Smart–Vercauteren encryption:
Take short $m \in \mathbb{Z}[x]/F$.

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

Homomorphic operations:
- Multiply ciphertexts $m(r)$ to multiply messages $m$.

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

Decryption works if each coefficient of $m/g \in \mathbb{Q}[x]/F$ 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.

For multiquadratic $F$, keygen is disappointingly 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 $p^2$. 
Smart–Vercauteren encryption:
Take short $m \in \mathbb{Z}[x]/F$.
Ciphertext is $m(r) \in \mathbb{Z}/q$.

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

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

Decryption works if each coefficient of $m/g \in \mathbb{Q}[x]/F$
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.

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

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

Decryption:
given \( c \in \{0; 1; \ldots; q - 1\} \),
compute \( c = g \in \mathbb{Q}[x] = F \),
round to element of \( \mathbb{Z}[x] = F \),
multiply by \( g \), subtract from \( c \).
Decryption works if each coefficient of \( m = g \in \mathbb{Q}[x] = F \)
is in \((-1^2 = 2; 1^2 = 2)\).

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.

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 \( p^2 \).
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 $p^2$. 
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 $p^2$. $d_1, \ldots, d_n$ are squares in $k$, so $F$ splits completely in $k[x]$. $\deg h \in \{1, 2\}$ for each irred factor $h$ of $F$ in $\mathbf{F}_p[x]$. 
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 $p^2$. $d_1, \ldots, d_n$ 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 $p \leq 2^n$, have $\Theta(p)$ distinct linear factors $h$. 
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 $p^2$.

$d_1, \ldots, d_n$ are squares in $k$, so $F$ splits completely in $k[x]$.

$\deg h \in \{1, 2\}$ for each irred factor $h$ of $F$ in $F_p[x]$.

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

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

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 $\mathbb{F}_p[x]$.

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

For each linear factor $h$:

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

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

1. Generalize cryptosystem to support $n$ polynomial variables.

Use $R = \mathbb{Z}[\sqrt{d_1}, \ldots, \sqrt{d_n}]$. 
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 $p^2$.

$d_1, \ldots, d_n$ are squares in $k$, so $F$ splits completely in $k[x]$.

$\deg h \in \{1, 2\}$ for each irred factor $h$ of $F$ in $\mathbb{F}_p[x]$.

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

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

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

1. Generalize cryptosystem to support $n$ polynomial variables.
   Use $R = \mathbb{Z}[\sqrt{d_1}, \ldots, \sqrt{d_n}]$. 
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 $p^2$.

$d_1, \ldots, d_n$ are squares in $k$, so $F$ splits completely in $k[x]$.

$\deg h \in \{1, 2\}$ for each irred factor $h$ of $F$ in $\mathbb{F}_p[x]$.

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

For each linear factor $h$:

with probability $\approx 1/p$,

$h$ divides $g$ in $\mathbb{F}_p[x]$,

forcing $p^2$ to divide norm of $g$ if any $d_i$ is non-square in $\mathbb{F}_p$.

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

1. Generalize cryptosystem to support $n$ polynomial variables.

Use $R = \mathbb{Z}[\sqrt{d_1}, \ldots, \sqrt{d_n}]$. 

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 irreducible factor $h$ of $F$ in $\mathbb{F}_p[x]$.

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

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

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

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

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

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

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

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

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 irreducible factor $h$ of $F$ in $\mathbb{F}_p[x]$.

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

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

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

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


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

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

For each linear factor $h$: 

Probability $\approx 1/p$, $h$ divides $g$ in $\mathbb{F}_p[x]$, forcing $p^2$ to divide norm of $g$. 

If any $d_i$ is non-square in $\mathbb{F}_p$. 

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

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


3. Choose $y \in \Theta(2^n/n)$. 

Force $g$ to be invertible mod all primes $p \leq y$. Heuristically, good chance of squarefree norm.
Why this happens: Fix prime $p$. Take field $k$ of size $p^2$. D_1;:::;D_n are squares in $k$, so $F$ splits completely in $k[x]$. D_1;:::;D_n are squares in $k$, so $F$ splits completely in $k[x]$. $\text{deg } h \in \{1, 2\}$ for each irreducible factor $h$ of $F$ in $\mathbb{F}_p[x]$. $\text{deg } h \in \{1, 2\}$ for each irreducible factor $h$ of $F$ in $\mathbb{F}_p[x]$. For each linear factor $h$: with probability $\approx 1/p$, $h$ divides $g$ in $\mathbb{F}_p[x]$, forcing $p^2$ to divide norm of $g$ if any $D_i$ is non-square in $\mathbb{F}_p$.

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

1. Generalize cryptosystem to support $n$ polynomial variables. Use $R = \mathbb{Z}[\sqrt{D_1}, \ldots, \sqrt{D_n}]$.
3. Choose $y \in \Theta(2^n/n)$. Force $g$ to be invertible mod all primes $p \leq y$. Heuristically, good chance of squarefree norm.

Computing units
Fix positive non-square $D \in \mathbb{Z}$. Assume $D$ quasipoly in $2^n$; i.e., $\log D \in n^{O(1)}$. Fix positive non-square $D \in \mathbb{Z}$. Assume $D$ quasipoly in $2^n$; i.e., $\log D \in n^{O(1)}$.
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 $k[x]$.

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

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

---

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

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


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

---

Computing units

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

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

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


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

Computing units

Fix positive non-square $d \in \mathbb{Z}$. Assume $d$ quasipoly in $2^n$; i.e., $\log d \in n^{O(1)}$. 
Our multiquadratic tweaks to Smart–Vercauteren (including adaptation of Gentry–Halevi):

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


3. Choose $y \in \Theta(2^n/n)$.

Force $g$ to be invertible mod all primes $p \leq y$. Heuristically, good chance of squarefree norm.

Computing units

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

\[\{\ldots, \pm \epsilon^{-2}, \pm \epsilon^{-1}, \pm 1, \pm \epsilon, \pm \epsilon^2, \ldots \}\]

is unit group of ring of integers of $\mathbb{Q}(\sqrt{d})$ for a unique $\epsilon > 1$, the normalized fundamental unit. 

$\log \epsilon < \sqrt{d}(2 + \log 4d)$; quasipoly.
Our multiquadratic tweaks to Smart–Vercauteren (including adaptation of Gentry–Halevi):

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


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

Computing units

Fix positive non-square $d \in \mathbb{Z}$. 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 $\mathbb{Q}(\sqrt{d})$ for a unique $\varepsilon > 1$, the normalized fundamental unit.

$\log \varepsilon < \sqrt{d}(2 + \log 4d)$; quasipoly.

Standard algorithms compute $a, b \in \mathbb{Q}$ with $\varepsilon = a + b\sqrt{d}$ in time $(\log \varepsilon)^{1+o(1)}$; quasipoly. (Can save time by instead representing $\varepsilon$ as product.)
multiquadratic tweaks to Smart–Vercauteren (including adaptation of Gentry–Halevi):
1. Generalize cryptosystem to support \( n \) polynomial variables. Use \( \mathbb{R} = \mathbb{Z}[\sqrt{d_1}, \ldots, \sqrt{d_n}] \).
2. Subroutine: Construct uniform random invertible element of \( \mathbb{R}/p \).
3. Choose \( y \in \Theta(2^n/n) \).
   Force \( g \) to be invertible mod all primes \( p \leq y \). Heuristically, good chance of squarefree norm.

Computing units
Fix positive non-square \( d \in \mathbb{Z} \).
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 \( \mathbb{Q}(\sqrt{d}) \) for a unique \( \varepsilon > 1 \), the normalized fundamental unit.
\( \log \varepsilon < \sqrt{d}(2 + \log 4d) \); quasipoly.

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

Take a multiquadratic field \( K = \mathbb{Q}(\sqrt{d_1}, \ldots, \sqrt{d_n}) \).
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.
Our multiquadratic tweaks to
Smart–Vercauteren (including
adaptation of Gentry–Halevi):

1. Generalize cryptosystem to
support $n$ polynomial variables.

$R = \mathbb{Z}[\sqrt{d_1}, \ldots, \sqrt{d_n}]$.

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 $p \leq y$. Heuristically,
good chance of squarefree norm.

Computing units

Fix positive non-square $d \in \mathbb{Z}$.
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
$\mathbb{Q}(\sqrt{d})$ for a unique $\varepsilon > 1$, the
**normalized fundamental unit**.

$\log \varepsilon < \sqrt{d}(2 + \log 4d)$; quasipoly.

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

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

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

Compute this group by computing
all normalized fundamental units.
Computing units

Fix positive non-square $d \in \mathbb{Z}$. 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 $\mathbb{Q}(\sqrt{d})$ for a unique $\varepsilon > 1$, the normalized fundamental unit. $\log \varepsilon < \sqrt{d}(2 + \log 4d)$; quasipoly.

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

Take a multiquadratic field $K = \mathbb{Q}(\sqrt{d_1}, \ldots, \sqrt{d_n})$. 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.
Computing units

Fix positive non-square $d \in \mathbb{Z}$. 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 $\mathbb{Q}(\sqrt{d})$ for a unique $\varepsilon > 1$, the normalized fundamental unit. $\log \varepsilon < \sqrt{d}(2 + \log 4d)$; quasipoly.

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

Take a multiquadratic field $K = \mathbb{Q}(\sqrt{d_1}, \ldots, \sqrt{d_n})$. 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.
Computing units

Fix positive non-square $d \in \mathbb{Z}$. 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 $\mathbb{Q}(\sqrt{d})$ for a unique $\varepsilon > 1$, the normalized fundamental unit. 

$\log \varepsilon < \sqrt{d}(2 + \log 4d)$; quasipoly.

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

Take a multiquadratic field $K = \mathbb{Q}(\sqrt{d_1}, \ldots, \sqrt{d_n})$. 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 $O_K^\ast$. Could use Eisenträger–Hallgren–Kitaev–Song, but we don’t want to wait for quantum computers.
Computing units

Fix positive non-square $d \in \mathbb{Z}$.

Assume $d$ quasipoly in $2^n$; i.e., $\log d \in n^{O(1)}$.

Take a multiquadratic field $K = \mathbb{Q}(\sqrt{d_1}, \ldots, \sqrt{d_n})$.

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 $O^*_K$.

Could use Eisenträger–Hallgren–Kitaev–Song, but we don’t want to wait for quantum computers.
Computing units
Fix positive non-square \( d \in \mathbb{Z} \).
Assume \( d \) quasipoly in \( 2^n \);
i.e., \( \log d \in n \mathcal{O}(1) \).

\[ \mathcal{O} = \{ \pm 1, \pm \varepsilon, \pm \varepsilon^2, \ldots \} \]
\( \mathcal{O} \) is unit group of ring of integers of \( \mathbb{Q}(\sqrt{\varepsilon}) \) for a unique \( \varepsilon > 1 \), the normalized fundamental unit.

Standard algorithms compute \( \mathcal{O} \) in time \( (\log \mathcal{O})^{1+o(1)} \); quasipoly.

The set of **multi-quadratic 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 multi-quadratics.
Take a multiquadratic field $K = \mathbb{Q}(\sqrt{d_1}, \ldots, \sqrt{d_n})$. 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 $O_K^*$. Could use Eisenträger–Hallgren–Kitaev–Song, but we don’t want to wait for quantum computers.
Take a multiquadratic field \( K = \mathbb{Q}(\sqrt{d_1}, \ldots, \sqrt{d_n}) \).
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.
Take a multiquadratic field \( K = \mathbb{Q}(\sqrt{d_1}, \ldots, \sqrt{d_n}) \).
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 \( \mathcal{O}_K^* \).

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: \( \mathbb{Q}; \mathbb{Q}(\sqrt{d}) \).
\( \sigma, \tau \): distinct non-identity automorphisms of \( K \).

\( K_{\sigma} = \{ x \in K : \sigma(x) = x \} \).
Take a multiquadratic field $K = \mathbb{Q}(\sqrt{d_1}, \ldots, \sqrt{d_n})$.
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 $O^*_K$.

Could use Eisenträger–Hallgren–Kitaev–Song, but we don’t want to wait for quantum computers.


First step: Recursively compute unit groups for three proper subfields $K_\sigma, K_\tau, K_{\sigma\tau}$ of $K$.

Base cases: $\mathbb{Q}; \mathbb{Q}((\sqrt{d}))$.

$\sigma, \tau$: distinct non-identity automorphisms of $K$.

$K_\sigma = \{x \in K : \sigma(x) = x\}$.

e.g. $K = \mathbb{Q}(\sqrt{2}, \sqrt{3}, \sqrt{5})$, appropriate $\sigma, \tau$: have

$K_\sigma = \mathbb{Q}(\sqrt{2}, \sqrt{3})$;

$K_\tau = \mathbb{Q}(\sqrt{2}, \sqrt{5})$;

$K_{\sigma\tau} = \mathbb{Q}(\sqrt{2}, \sqrt{15})$. 

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

The set of \textbf{multiquadratic units} is the group generated by units in $n - 1$ quadratic subfields. An analogous relation holds to cyclotomic units. We can compute this group by computing all normalized fundamental units.

Beyond this: compute $\mathcal{O}_K^*$. We 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.

First step: Recursively compute unit groups for three proper subfields $K_\sigma, K_\tau, K_{\sigma\tau}$ of $K$.

Base cases: $\mathbb{Q}$; $\mathbb{Q}(\sqrt{d})$.

$\sigma, \tau$: distinct non-identity automorphisms of $K$.

$K_\sigma = \{x \in K : \sigma(x) = x\}$.

e.g. $K = \mathbb{Q}(\sqrt{2}, \sqrt{3}, \sqrt{5})$.

appropriate $\sigma, \tau$: have $K_\sigma = \mathbb{Q}(\sqrt{2}, \sqrt{3})$;
$K_\tau = \mathbb{Q}(\sqrt{2}, \sqrt{5})$;
$K_{\sigma\tau} = \mathbb{Q}(\sqrt{2}, \sqrt{15})$. 

Second step: Compute $U = \mathcal{O}_K^* \cap \mathcal{O}_{K_\sigma}^* \cap \mathcal{O}_{K_\tau}^* \cap \mathcal{O}_{K_{\sigma\tau}}^*$. 

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

**Multiquadratic units** are generated by units in quadratic subfields.

**Fundamental units** are computed by computing all normalized fundamental units.

Wada: exponential-time $O_K^\ast$ algorithm for multiquadratics.

First step: Recursively compute unit groups for three proper subfields $K_\sigma, K_\tau, K_{\sigma\tau}$ of $K$.

Base cases: $\mathbb{Q}; \mathbb{Q}(\sqrt{d})$.

$\sigma, \tau$: distinct non-identity automorphisms of $K$.

$K_\sigma = \{x \in K : \sigma(x) = x\}$.

e.g. $K = \mathbb{Q}(\sqrt{2}, \sqrt{3}, \sqrt{5})$, appropriate $\sigma, \tau$: have

$K_\sigma = \mathbb{Q}(\sqrt{2}, \sqrt{3})$;

$K_\tau = \mathbb{Q}(\sqrt{2}, \sqrt{5})$;

$K_{\sigma\tau} = \mathbb{Q}(\sqrt{2}, \sqrt{15})$.

Second step: Compute $U = O_K^\ast$.

First step: Recursively compute unit groups for three proper subfields $K_\sigma, K_\tau, K_{\sigma\tau}$ of $K$.

Base cases: $\mathbb{Q}; \mathbb{Q}(\sqrt{d})$.

$\sigma, \tau$: distinct non-identity automorphisms of $K$.

$K_\sigma = \{x \in K : \sigma(x) = x\}$.

e.g. $K = \mathbb{Q}(\sqrt{2}, \sqrt{3}, \sqrt{5})$, appropriate $\sigma, \tau$: have

$K_\sigma = \mathbb{Q}(\sqrt{2}, \sqrt{3})$;

$K_\tau = \mathbb{Q}(\sqrt{2}, \sqrt{5})$;

$K_{\sigma\tau} = \mathbb{Q}(\sqrt{2}, \sqrt{15})$. 
Take a multiquadratic field $\mathbb{K} = \mathbb{Q}(\sqrt{d_1}; \ldots; \sqrt{d_n})$. 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.

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

$\sigma, \tau$: distinct non-identity automorphisms of $K$.

$K_\sigma = \{x \in K : \sigma(x) = x\}$.

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

Second step: Compute $U = \mathcal{O}_{K_\sigma}^* \mathcal{O}_{K_\tau}^* \sigma(\mathcal{O}_{K_{\sigma\tau}}^*)$. 24

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

$\sigma, \tau$: distinct non-identity automorphisms of $K$.

$K_{\sigma} = \{ x \in K : \sigma(x) = x \}$.

e.g. $K = \mathbb{Q}(\sqrt{2}, \sqrt{3}, \sqrt{5})$.

appropriate $\sigma, \tau$: have

$K_{\sigma} = \mathbb{Q}(\sqrt{2}, \sqrt{3})$;
$K_{\tau} = \mathbb{Q}(\sqrt{2}, \sqrt{5})$;
$K_{\sigma\tau} = \mathbb{Q}(\sqrt{2}, \sqrt{15})$.

Second step: Compute $U = O_{K_{\sigma}}^* O_{K_{\tau}}^* \sigma(O_{K_{\sigma\tau}}^*)$. 

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

$\sigma, \tau$: distinct non-identity automorphisms of $K$.

$K_{\sigma} = \{x \in K : \sigma(x) = x\}$.

E.g. $K = \mathbb{Q}(\sqrt{2}, \sqrt{3}, \sqrt{5})$, appropriate $\sigma, \tau$: have

$K_{\sigma} = \mathbb{Q}(\sqrt{2}, \sqrt{3})$;
$K_{\tau} = \mathbb{Q}(\sqrt{2}, \sqrt{5})$;
$K_{\sigma\tau} = \mathbb{Q}(\sqrt{2}, \sqrt{15})$.

Second step:
Compute $U = O_{K_{\sigma}}^* O_{K_{\tau}}^* \sigma(O_{K_{\sigma\tau}}^*)$.

Fact: $U \leq O_K^*$. 

First step: Recursively compute unit groups for three proper subfields $K_\sigma, K_\tau, K_{\sigma\tau}$ of $K$.

Base cases: $\mathbb{Q}; \mathbb{Q}(\sqrt{d})$.

$\sigma, \tau$: distinct non-identity automorphisms of $K$.

$k_\sigma = \{x \in K : \sigma(x) = x\}$.

e.g. $K = \mathbb{Q}(\sqrt{2}, \sqrt{3}, \sqrt{5})$, appropriate $\sigma, \tau$: have

$K_\sigma = \mathbb{Q}(\sqrt{2}, \sqrt{3})$;

$K_\tau = \mathbb{Q}(\sqrt{2}, \sqrt{5})$;

$K_{\sigma\tau} = \mathbb{Q}(\sqrt{2}, \sqrt{15})$.

Second step:

Compute $U = O_{K_\sigma}^* O_{K_\tau}^* \sigma(O_{K_{\sigma\tau}}^*)$.

Fact: $U \leq O_K^*$.

Fact: $(O_K^*)^2 \leq U$. 

First step: Recursively compute unit groups for three proper subfields $K_\sigma, K_\tau, K_{\sigma\tau}$ of $K$.

Base cases: $\mathbb{Q}$; $\mathbb{Q}(\sqrt{d})$.

$\sigma, \tau$: distinct non-identity automorphisms of $K$.

$K_\sigma = \{x \in K : \sigma(x) = x\}$.

e.g. $K = \mathbb{Q}(\sqrt{2}, \sqrt{3}, \sqrt{5})$, appropriate $\sigma, \tau$: have

$K_\sigma = \mathbb{Q}(\sqrt{2}, \sqrt{3})$;
$K_\tau = \mathbb{Q}(\sqrt{2}, \sqrt{5})$;
$K_{\sigma\tau} = \mathbb{Q}(\sqrt{2}, \sqrt{15})$.

Second step:

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

Fact: $U \leq O^*_K$.

Fact: $(O^*_K)^2 \leq U$.

Proof:

If $u \in O^*_K$ then

$u\sigma(u) \in O^*_{K_\sigma}$;
$u\tau(u) \in O^*_{K_\tau}$;
$u\sigma(\tau(u)) \in O^*_{K_{\sigma\tau}}$; so

$u\sigma(u)u\tau(u)/\sigma(u\sigma(\tau(u))) \in U$. 

First step: Recursively compute unit groups for three proper subfields $K_\sigma, K_\tau, K_{\sigma\tau}$ of $K$.

Base cases: $\mathbb{Q}; \mathbb{Q}(\sqrt{d})$.

$\sigma, \tau$: distinct non-identity automorphisms of $K$.

$K_\sigma = \{x \in K : \sigma(x) = x\}$.

e.g. $K = \mathbb{Q}(\sqrt{2}, \sqrt{3}, \sqrt{5})$.

appropriate $\sigma, \tau$: have

$K_\sigma = \mathbb{Q}(\sqrt{2}, \sqrt{3})$;

$K_\tau = \mathbb{Q}(\sqrt{2}, \sqrt{5})$;

$K_{\sigma\tau} = \mathbb{Q}(\sqrt{2}, \sqrt{15})$.

Second step:

Compute $U = O_{K_\sigma}^* O_{K_\tau}^* \sigma(O_{K_{\sigma\tau}}^*)$.

Fact: $U \leq O_K^*$.

Fact: $(O_K^*)^2 \leq U$.

Proof:

If $u \in O_K^*$ then

$u\sigma(u) \in O_{K_\sigma}^*$;

$u\tau(u) \in O_{K_\tau}^*$;

$u\sigma(\tau(u)) \in O_{K_{\sigma\tau}}^*$; so

$u\sigma(u)u\tau(u)/\sigma(u\sigma(\tau(u))) \in U$.

In other words, $u^2 \in U$. 

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

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

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

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


First step: Recursively compute unit groups for three proper subfields $K_{\sigma}, K_{\tau}, K_{\sigma \tau}$ of $K$.

Cases: $Q, Q(\sqrt{d})$.

Distinct non-identity automorphisms of $K$.
$x \in K : \sigma(x) = x \}$.

\[K_{\sigma} = Q(\sqrt{2}, \sqrt{3}, \sqrt{5}),\]

\[K_{\tau} = Q(\sqrt{2}, \sqrt{3});\]

\[K_{\sigma \tau} = Q(\sqrt{2}, \sqrt{5});\]

$Q(\sqrt{2}, \sqrt{15})$. 


24


First step: Recursively compute unit groups for three proper subfields $K_{ff}$; $K_{fi}$; $K_{fffi}$ of $K$.

Base cases: $Q_{(\sqrt{d})}$. $ff;fi$ : distinct non-identity automorphisms of $K$.

$K_{ff} = \{ x \in K : \sigma(x) = x \}$. $\sqrt{2}; \sqrt{3}; \sqrt{5}$, have $K_{ff} = Q_{(\sqrt{2})}$; $K_{fi} = Q_{(\sqrt{2})}$, $K_{fffi} = Q_{(\sqrt{5})}$.

Second step: Compute $U = O^*_K \sigma(O^*_K \tau(O^*_K))$.

Fact: $U \leq O^*_K$.

Fact: $(O^*_K)^2 \leq U$.

Proof:
If $u \in O^*_K$ then
$u\sigma(u) \in O^*_K \sigma$;
$u\tau(u) \in O^*_K \tau$;
$u\sigma(\tau(u)) \in O^*_K \sigma \tau$; so
$u\sigma(u)\tau(u)/\sigma(u\sigma(\tau(u))) \in U$.

In other words, $u^2 \in U$.

25

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

26

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

Second step:
Compute \( U = \mathcal{O}_K^* \sigma \mathcal{O}_K^* \tau \sigma(\mathcal{O}_{K_{\sigma \tau}}^*) \).

Fact: \( U \leq \mathcal{O}_K^* \).

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

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, \( u^2 \in U \).
Second step:
Compute $U = \mathcal{O}_{K_\sigma}^* \mathcal{O}_{K_\tau}^* \sigma(\mathcal{O}_{K_{\sigma\tau}}^*)$.

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

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

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

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

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

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 $U$.
$2^{\Theta(2^n)}$ products.
Second step:
Compute $U = \mathcal{O}_{K_\sigma}^* \mathcal{O}_{K_\tau}^* \sigma(\mathcal{O}_{K_\sigma\tau}^*)$.

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

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

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 $U$.

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

We do much better using an NFS idea from 1991 Adleman.
Second step:
Compute \( U = \mathcal{O}_{K_\sigma}^* \mathcal{O}_{K_\tau}^* \sigma(\mathcal{O}_{K_\sigma \tau}^*) \).

Fact: \( U \leq \mathcal{O}_K^* \).

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

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 \( U \).
\( 2^{\Theta(2^n)} \) 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 \( \chi \) with \( \chi(\alpha_1), \ldots, \chi(\alpha_k) \in \{-1, 1\} \).
Second step:
Compute $U = \mathcal{O}_{K\sigma}^* \mathcal{O}_{K^T}^* \sigma(\mathcal{O}_{K_{\sigma T}}^*)$.

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

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

Proof:
If $u \in \mathcal{O}_K^*$ then
$u\sigma(u) \in \mathcal{O}_{K\sigma}^*$;
$u\tau(u) \in \mathcal{O}_{K^T}^*$;
$u\sigma(\tau(u)) \in \mathcal{O}_{K_{\sigma T}}^*$; 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 $U$.

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

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

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

Linear equation, usually reducing $\dim\{e\}$ by 1. Use many such $\chi$. 
Second step:
Compute \( U = \mathcal{O}^*_{K_\sigma} \mathcal{O}^*_{K_\tau} \sigma(\mathcal{O}^*_{K_{\sigma \tau}}) \).

\( \leq \mathcal{O}^*_K \).

\( (\mathcal{O}^*_K)^2 \leq U. \)

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

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

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

\( 2^{\Theta(2^n)} \) 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 \( \chi \) with \( \chi(\alpha_1), \ldots, \chi(\alpha_k) \in \{-1, 1\}. \)

Linear equation, usually reducing \( \dim\{e\} \) by 1. Use many such \( \chi. \)
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 \(\chi\)
with \(\chi(\alpha_1), \ldots, \chi(\alpha_k) \in \{-1, 1\}\).

Linear equation, usually reducing
\(\dim\{e\}\) by 1. Use many such \(\chi\).
Second step:
Compute $U = O^* \cdot K_{ff} \cdot O^* \cdot K_{ffi}$.

Fact: $U \leq O \cdot K$.

Fact: $(O^* \cdot K)^2 \leq U$.

Proof:
If $u \in O^* \cdot K$ then $u_{ff}(u) \in O^* \cdot K_{ff}$;
$v_{fi}(v) \in O^* \cdot K_{ffi}$; so $u_{ff}(u) v_{fi}(v) = ff(u_{ff}(v_{fi}(u))) \in U$.

In other words, $u^2 \in U$.

Third step:
identify $(O^*_K)^2$ 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 $\chi$
with $\chi(\alpha_1), \ldots, \chi(\alpha_k) \in \{-1, 1\}$.

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

Computing generators
Main goal: Find $g$ given $g_R$, where $R = \mathbb{Z}[\sqrt{d_1}, \ldots, \sqrt{d_n}]$. 
Third step:
identify \((O_K^*)^2\) 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 \(\chi\)
with \(\chi(\alpha_1), \ldots, \chi(\alpha_k) \in \{-1, 1\}\).

Linear equation, usually reducing \(\dim\{e\}\) by 1. Use many such \(\chi\).

---

Computing generators

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

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

Linear equation, usually reducing \(\dim\{e\}\) by 1. Use many such \(\chi\).

Computing generators

Main goal: Find \(g\) given \(g_R\), where \(R = \mathbb{Z}[\sqrt{d_1}, \ldots, \sqrt{d_n}]\).

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

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

Linear equation, usually reducing
\(\dim\{e\}\) by 1. Use many such \(\chi\).

Computing generators

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

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 \(g\sigma(g)\)?
Third step:
identify \((O_K^*)^2\) 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 \(\chi\)
with \(\chi(\alpha_1), \ldots, \chi(\alpha_k) \in \{-1, 1\}\).

Linear equation, usually reducing
\(\dim\{e\}\) by 1. Use many such \(\chi\).

Computing generators

Main goal: Find \(g\) given \(g_R\),
where \(R = \mathbb{Z}[\sqrt{d_1}, \ldots, \sqrt{d_n}]\).

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 \(g_R\) from \(K\) to \(K_\sigma\).
Obtain ideal generated by \(g\sigma(g)\).
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.

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

Linear equation, usually reducing \(\dim\{e\}\) by 1. Use many such \(\chi\).

---

Computing generators

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

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 \(K_\sigma\).

Obtain ideal generated by \(g\sigma(g)\).

Recursively compute a generator of this ideal: probably not \(g\sigma(g)\).

Some \(ug\sigma(g)\) with \(u \in \mathcal{O}_K^*\).
Step:

Identify \((O^*_K)^2\) inside \(U\) by trying to compute square roots of products of generators of \(U\).

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

\[
\chi \cdot e^k \quad \text{square} \Rightarrow \chi(\alpha_k)^{e_k} = 1
\]

for a quadratic character \(\chi\) on \(\{\alpha_1, \ldots, \alpha_k\} \in \{-1, 1\}\).

Linear equation, usually reducing \(\dim \{e\}\) by 1. Use many such \(\chi\).

Computing generators

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

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 \(K_\sigma\).

Obtain ideal generated by \(g\sigma(g)\).

Recursively compute a generator of this ideal: probably not \(g\sigma(g)\).

Some \(ug\sigma(g)\) with \(u \in O^*_K\).

Unit multiple of \(g\sigma(g)\), unit multiple of \(g\tau(g)\), unit multiple of \(g\sigma(\tau(g))\) \(\Rightarrow\) some \(ug^2\) with \(u \in O^*_K\).
Computing generators

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

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 $K_\sigma$.

Obtain ideal generated by $g\sigma(g)$.

Recursively compute a generator of this ideal: probably not $g\sigma(g)$.

Some $ug\sigma(g)$ with $u \in \mathcal{O}_{K_\sigma}^*$.
Computing generators

Main goal: Find \( g \) given \( gR \), where \( R = \mathbb{Z}[^\sqrt{d_1}, \ldots, \sqrt{d_n}] \).

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 \( K_\sigma \).

Obtain ideal generated by \( g\sigma(g) \).

Recursively compute a generator of this ideal: probably not \( g\sigma(g) \).

Some \( ug\sigma(g) \) with \( u \in O_{K_\sigma}^* \).

Unit multiple of \( g\sigma(g) \), unit multiple of \( g\tau(g) \), unit multiple of \( g\sigma(\tau(g)) \)
\( \Rightarrow \) some \( ug^2 \) with \( u \in O_K^* \).

Third step: identify \( (O^*K)^2 \) inside \( U \) by trying to compute square roots of products of generators of \( U \).

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

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

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 $K_{\sigma}$.

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 $g\sigma(\tau(g))$ ⇒ some $ug^2$ with $u \in \mathcal{O}_{K_{\sigma}}^*$. 
Computing generators

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

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 $K_\sigma$.
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 $u \in \mathcal{O}_K^*$.

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

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

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 $K_\sigma$. 
Obtain ideal generated by $g\sigma(g)$.

Recursively compute a generator of this ideal: probably not $g\sigma(g)$.

Some $ug\sigma(g)$ with $u \in \mathcal{O}_{K_\sigma}^\ast$.

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

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

Now compute square root: some unit multiple of $g$, i.e., some $g'$ with $g'O_K = gO_K$. 

Some $ug\sigma(g)$ with $u \in \mathcal{O}_{K_\sigma}^\ast$. 

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

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

Now compute square root: some unit multiple of $g$, i.e., some $g'$ with $g'O_K = gO_K$. 

Some $ug\sigma(g)$ with $u \in \mathcal{O}_{K_\sigma}^\ast$. 
Computing generators

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

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 $K_\sigma$.
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 $u \in \mathcal{O}_K^*$.

Use quadratic characters (with values $\pm 1$ 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 generators

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

Strategy: Reuse the equation $g^2 = g\sigma(g)\tau(g)/\sigma(g\sigma(\tau(g)))$.

Broken down:
- Square root of $g^2$ is $\pm g$.
- How to compute $g\sigma(g)$?
  - Compute relative norm of ideal $gR$ from $K$ to $K_\sigma$.
  - Obtain ideal generated by $g\sigma(g)$.
  - Recursively compute a generator of this ideal: probably not $g\sigma(g)$.
  - Some $u\sigma(g)$ with $u \in \mathcal{O}_{K_\sigma}^*$.

Unit multiple of $g\sigma(\tau(g))$:
- $u\sigma(g)$ with $u \in \mathcal{O}_{K_\sigma}^*$.

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

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

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 generators

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

Strategy: Reuse the equation $g^2 = g^\sigma(g^\tau(g))$.

Square root of $g^2$ is $\pm g$.

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

First compute relative norm of ideal $gR$ from $K$ to $K^\sigma$.

Obtain ideal generated by $g^\sigma(g)$.

Recursively compute a generator of this ideal: probably not $g^\sigma(g)$.

Some $u g^2$ with $u \in O_K^*$.

Use quadratic characters (with values $\pm 1$ on $g$) to identify $v \in O_{K^\sigma}^*$ such that $vug^2$ is a square.

Now compute square root: some unit multiple of $g$, i.e., some $g'$ with $g'O_K = gO_K$.

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 generators

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

Strategy: Reuse the equation $g^2 = g\sigma(g)f(g)$.

Square root of $g^2$ is $\pm g$.

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

First compute relative norm of ideal $gR$ from $K$ to $K$.

Obtain ideal generated by $g\sigma(g)$.

Recursively compute a generator of this ideal: probably not $g\sigma(g)$.

Some $ug\sigma(g)$ with $u \in \mathcal{O}_K^\ast$.

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

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

Now compute square root: some unit multiple of $g$, i.e., some $g'$ with $g'O_K = gO_K$.

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.)
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 $\pm 1$ 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 $g'O_K = gO_K$.

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.)
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 $\pm 1$ 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.)

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

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

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 $\pm 1$ 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 $g'O_K = gO_K$.

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.)

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 $2^n \log g$ on MQ unit basis are almost certainly in $(-0.1, 0.1)$. 
Unit multiple of $g\sigma(g)$, unit multiple of $g\tau(g)$, unit multiple of $g\sigma(\tau(g))$ $ug^2$ with $u \in \mathcal{O}_K^*$.

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

Compute square root: find multiple of $g$, i.e., some $g'$ with $g'\mathcal{O}_K = g\mathcal{O}_K$. 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.)

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 $2^n \log g$ on MQ unit basis are almost certainly in $(-0.1, 0.1)$.

$u^2^n$ is an MQ unit.

$\Log u^2^n$ is closest vector to $2^n \Log u g$.
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 \( 2^n \log g \) on MQ unit basis are almost certainly in \((-0.1, 0.1)\).

\( u^{2n} \) is an MQ unit. \( \log u^{2n} = 2^n \log u \) is closest vector to \( 2^n \log u \).
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 $2^n \log g$
on MQ unit basis are
almost certainly in $(-0.1, 0.1)$.

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

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

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

MQ unit lattice is orthogonal.
Round $2^n \log ug$ to find $2^n \log u$ and $2^n \log g$. Deduce $±g^{2n}$.
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 \( 2^n \log g \) on MQ unit basis are almost certainly in \((-0.1, 0.1)\).

\( u^{2^n} \) is an MQ unit.
\( \log u^{2^n} = 2^n \log u \) is closest vector to \( 2^n \log ug \).

MQ unit lattice is orthogonal.
Round \( 2^n \log ug \) to find \( 2^n \log u \) and \( 2^n \log g \). Deduce \( \pm g^{2^n} \).

Use quadratic character: \( g^{2^n} \).
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 $\log \epsilon$
are larger than $2^{0.51n}$;
so coefficients of $2^n \log g$
on MQ unit basis are
almost certainly in $(-0.1, 0.1)$.

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

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

Use quadratic character: $g^{2^n}$.
Square root: $\pm g^{2^n-1}$.
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 \( 2^n \log g \) on MQ unit basis are almost certainly in \((-0.1, 0.1)\).

\( u^{2^n} \) is an MQ unit.
\( \log u^{2^n} = 2^n \log u \) is closest vector to \( 2^n \log ug \).

MQ unit lattice is orthogonal.
Round \( 2^n \log ug \) to find \( 2^n \log u \) and \( 2^n \log g \). 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: \( \pm g^{2^n-2} \).
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 $\log \epsilon$
are larger than $2^{0.51n}$;
so coefficients of $2^n \log g$
on MQ unit basis are
almost certainly in $(-0.1, 0.1)$.

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

MQ unit lattice is orthogonal.
Round $2^n \log ug$ to find $2^n \log u$
and $2^n \log g$. 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: $\pm g^{2^n-2}$.

Square root: $\pm g$. Done!

MQ cryptosystem is broken
for all of these fields.
Computing short generators

Assume $d_1, \ldots, d_n \geq 2^{1.03 n}$.

(More work seems to push bound to $< n^{2}$; see paper and software.)

Find multiquadratic (MQ) units.

Find all units.

Find some generator $u g$.

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

$u^{2n}$ is an MQ unit.

$\log u^{2n} = 2^n \log u$ is closest vector to $2^n \log u g$.

MQ unit lattice is orthogonal.

Round $2^n \log u g$ to find $2^n \log u$ and $2^n \log g$. Deduce $\pm g^{2n}$.

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

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

Use quadratic character: $g^{2n-1}$.

Square root: $\pm g^{2n-2}$.

$\vdots$

Square root: $\pm g$. Done!

MQ cryptosystem is broken for all of these fields.

Slightly simpler:

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

\[ u^{2^n} \text{ is an MQ unit.} \]
\[ \log u^{2^n} = 2^n \log u \text{ is closest vector to } 2^n \log u^g. \]

MQ unit lattice is orthogonal.
Round $2^n \log u^g$ to find $2^n \log u$ and $2^n \log g$. 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: $\pm g^{2^n-2}$.

Square root: $\pm g$. Done!

MQ cryptosystem is broken for all of these fields.

Slightly simpler:
Find MQ units, but skip finding all units.
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 \( u^g \).

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

\( u^{2^n} \) is an MQ unit.
\( \log u^{2^n} = 2^n \log u \) is closest vector to \( 2^n \log u^g \).

MQ unit lattice is orthogonal.
Round \( 2^n \log u^g \) to find \( 2^n \log u \) and \( 2^n \log g \). 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.
\( u^{2n} \) is an MQ unit.

\( \log u^{2n} = 2^n \log u \) is closest vector to \( 2^n \log ug \).

MQ unit lattice is orthogonal.

Round \( 2^n \log ug \) to find \( 2^n \log u \) and \( 2^n \log g \). 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: \( \pm g^{2^{n-2}} \).

Square root: \( \pm g \). Done!

MQ cryptosystem is broken for all of these fields.

Slightly simpler:

Find MQ units, but skip finding all units.
$u^{2^n}$ is an MQ unit.

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

MQ unit lattice is orthogonal.

Round $2^n \Log ug$ to find $2^n \Log u$ and $2^n \Log g$. 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.

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

Log $u^{2n} = 2^n \log u$ is closest vector to $2^n \log ug$.

MQ unit lattice is orthogonal.

Round $2^n \log ug$ to find $2^n \log u$ and $2^n \log g$. Deduce $\pm g^{2^n}$.

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

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

Square root: $\pm g$. Done!

MQ cryptosystem is broken for all of these fields.

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: Log $ug^{2^{n-1}}$.
Round: Log $u$. 
\( u^{2^n} \) is an MQ unit.
\[ \log u^{2^n} = 2^n \log u \] is closest vector to \( 2^n \log u g \).

MQ unit lattice is orthogonal.
Round \( 2^n \log u g \) to find \( 2^n \log u \) and \( 2^n \log g \). 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.
Recursively find \( u g^{2^{n-1}} \) where \( u \) is an MQ unit; i.e., skip square-root computations.
Take logs: \( \log u g^{2^{n-1}} \).
Round: \( \log u \).
Deduce \( \pm g^{2^{n-1}} \).
Use quadratic character: \( g^{2^{n-1}} \).
Square root: \( \pm g^{2^{n-2}} \).
\[ \vdots \]
Square root: \( \pm g \).