Algorithms for multiquadratic number fields

D. J. Bernstein

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

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

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

Algorithms for multiquadratic number fields

D. J. Bernstein

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

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

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

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. "Short generators without quantum computers: the case of multiquadratics." Eurocrypt 2017.

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

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

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

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

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

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

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?

ms for
adratic number fields

ernstein

---

uch, Daniel J. Bernstein,
e Valence, Tanja Lange,
e van Vredendaal.
generators without
n computers: the case of
adratics." Eurocrypt 2017.

nd software:
//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
Peikert–
algebraic
(includin
that we
brought
other pr
Yet desp
significa
these pr
The bes
ideal lat
no bette
counterp
in practi

mber fields

el J. Bernstein,
Tanja Lange,
lendaal.
without
rs: the case of
Eurocrypt 2017.

e:
uad.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
Peikert–Regev: "A
algebraic and algo
(including quantur
that we employ . .
brought to bear ag
other problems on
Yet despite consid
significant progres
these problems has
The best known a
ideal lattices perfo
no better than the
counterparts, both
in practice."

s

tein,
nge,

se of
t 2017.

p.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 Lyubashev
Peikert–Regev: "All of the
algebraic and algorithmic to
(including quantum comput
that we employ . . . can also
brought to bear against SVP
other problems on ideal latti
Yet despite considerable effo
significant progress in attack
these problems has been ma
The best known algorithms
ideal lattices perform essenti
no better than their generic
counterparts, both in theory
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."

rough STOC 2009 Gentry
stem "Fully homomorphic
on using ideal lattices"
ken several years later,
asonable assumptions.

tion 1: User chooses a
h^{+}") cyclotomic field
nderlying number field.

tion 2: Attacker has a
antum computer.

er fields be attacked?
e non-quantum attacks?
bout 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 k
short ele

R: e.g.,
of a cycl

Public k

OC 2009 Gentry
ly homomorphic
deal lattices"
l years later,
assumptions.

er chooses a
otomic field
number field.

tacker has a
mputer.

e attacked?
ntum attacks?
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 Gent
short element g of

R: e.g., ring of int
of a cyclotomic fie

Public key: ideal g

Gentry
orphic
es”
er,
ns.

s a
d
eld.

a

?
cks?
tems?

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 syster
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.

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

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

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

e to 2013 Lyubashevsky–
Regev: "All of the
c and algorithmic tools
ng quantum computation)
employ . . . can also be
to bear against SVP and
oblems on ideal lattices.
ite considerable effort, no
nt progress in attacking
oblems has been made.
t known algorithms for
tices perform essentially
er than their generic
arts, both in theory and
ce."

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

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

Public key: ideal gR.

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

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

Standard
view of
i.e., all u
Log u ra
Dirichlet
Log ug =

Lyubashevsky–
All of the
rithmic tools
m computation)
. can also be
gainst SVP and
ideal lattices.
erable effort, no
s in attacking
s been made.
lgorithms for
orm essentially
eir generic
in theory and

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

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

Public key: ideal gR.

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

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

Standard algebraic
view of all generat
i.e., all ug where
Log u ranges over
Dirichlet's log-unit
$\operatorname{Log} ug = \operatorname{Log} u +$

vsky–

ols
ation)
be
P and
ices.
rt, no
king
de.
for
ially

and

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

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

Public key: ideal gR.

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

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

Standard algebraic-number-t
view of all generators of gR
i.e., all ug where $u \in R^*$:
$\operatorname{Log} u$ ranges over
Dirichlet's log-unit lattice;
$\operatorname{Log} ug = \operatorname{Log} u + \operatorname{Log} g$.

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

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

Public key: ideal gR.

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

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

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

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

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

Public key: ideal gR.

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

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

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

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

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

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

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

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

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

ey in Gentry's system:
ement g of R.

ring of integers $\mathcal{O}_K$
lotomic field K.

ey: ideal gR.

tage 1, quantum:
2016 Biasse–Song
me generator of gR.
n Eisenträger–Hallgren–
Song algorithm for R^*.

tage 2, cyclotomic:
eduction algorithm from
mpbell–Groves–Shepherd.

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

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

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

For cycl
often u
Known t
cycloton

try's system:
f R.

tegers $\mathcal{O}_K$
eld K.

gR.

uantum:
e–Song
tor of gR.
ger–Hallgren–
rithm for R^*.

yclotomic:
lgorithm from
roves–Shepherd.

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

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

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

For cyclotomic fiel
often u is a "cyclo
Known textbook b
cyclotomic units is

m:

ren–

R^*.

rom

pherd.

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

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 uni

Known textbook basis for cyclotomic units is a short b

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Now embedding easily finds g.

d algebraic-number-theory
all generators of gR,
ug where $u \in R^*$:
nges over
t's log-unit lattice;
= $\operatorname{Log} u + \operatorname{Log} g$.

ny generator ug, try to
rt $\operatorname{Log} g$ by finding lattice
og u close to $\operatorname{Log} ug$.

.g., embedding or Babai,
from basis for $\operatorname{Log} R^*$?
find short enough basis,
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 $\mathbf{Q}(\zeta)$; ring $R = \mathbf{Z}[\zeta]$.

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

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

Now embedding easily finds g.

Are you
Try to d
Ask: Do
- the gR
- Gentry
- the ori
  multili

really ma

c-number-theory
tors of gR,
$u \in R^*$:

t lattice;
Log g.

or ug, try to
y finding lattice
to Log ug.

dding or Babai,
s for Log R^*?
enough basis,
ely 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 $\mathbf{Q}(\zeta)$; ring $R = \mathbf{Z}[\zeta]$.

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

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

Now embedding easily finds g.

Are you a lattice s
Try to dismiss latt
Ask: Do attacks a
- the $gR \mapsto g$ pro
- Gentry's original
- the original Garg
  multilinear maps

really matter for u

theory
,

to
attice
g.

abai,
R*?
asis,

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

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

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

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

Now embedding easily finds g.

Are you a lattice salesman?
Try to dismiss lattice attack
Ask: Do attacks against
- the $gR \mapsto g$ problem,
- Gentry's original FHE syst
- the original Garg–Gentry–
  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 $\mathbf{Q}(\zeta)$; ring $R = \mathbf{Z}[\zeta]$.

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

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

Now embedding easily finds g.

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

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

really matter for users?

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

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

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

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

Now embedding easily finds g.

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

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

really matter for users?

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."

otomic fields,

is a “cyclotomic unit”.

textbook basis for

nic units is a short basis.

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

$\zeta)$; ring $R = \mathbf{Z}[\zeta]$.

$/(\zeta - 1)$ is a unit:

invert, or apply $\zeta \mapsto \zeta^3$

rphism to factors of $\zeta - 1$.

$/(\zeta^3 - 1)$ is a unit.

$)/(\zeta^9 - 1)$ is a unit.

a. Obtain short basis.

bedding 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 I

$I \mapsto$ sho

“Approx

$I \mapsto$ sho

lds,
otomic unit".
basis for
s a short basis.

xp($2\pi i/1024$);
$= \mathbf{Z}[\zeta]$.

s a unit:
apply $\zeta \mapsto \zeta^3$
factors of $\zeta - 1$.

is a unit.
) is a unit.
short basis.

asily 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 nonz

"Approximate Idea
$I \mapsto$ short nonzero

t".

asis.

24);

ζ^3

$\zeta - 1.$

s.

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

"Approximate Ideal-SVP":

$I \mapsto$ short nonzero vector in

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.

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."

a lattice salesman?
ismiss lattice attacks.
attacks against
$R \mapsto g$ problem,
's original FHE system,
iginal Garg–Gentry–Halevi
near maps, . . .
atter for users?

onse to the salesman:
not—but this problem
ural starting point for
other lattice problems
certainly care about.

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 Pe
limited t
"Althou
lot of st
yet foun
attackin
For com
principal
extremel
ideals. .
not so m
of cyclot
extra str
that hav

salesman?
tice attacks.
gainst
blem,
FHE system,
g–Gentry–Halevi
s, . . .
sers?

e salesman:
his problem
g point for
tice problems
care about.

al 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 s
limited to principa
“Although cycloto
lot of structure, no
yet found a way to
attacking Ideal-SV
For commonly use
principal ideals are
extremely small fra
ideals. . . . The we
not so much due t
of cyclotomics, bu
extra structure of
that have short ge

s.

em,

Halevi

n:

m

r

ms

.

"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 i

limited to principal ideals:

"Although cyclotomics have

lot of structure, nobody has

yet found a way to exploit it

attacking Ideal-SVP/BDD .

For commonly used rings,

principal ideals are an

extremely small fraction of a

ideals. . . . The weakness he

not so much due to the stru

of cyclotomics, but rather to

extra structure of principal i

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."

deal-SVP":
rtest nonzero vector in I.

imate Ideal-SVP":
rt nonzero vector in I.

s against ideal I
hort generator.

ikert says idea is "useless"
general principal ideals:
nply hadn't realized
added guarantee of a
nerator would transform
nique from useless to
ingly effective."

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

Actually
attacks
2016 Cr
Ideal-SV
$2^{N^{1/2+o(}}$
under pl
about cl
Start fro
more fea

":
ero vector in I.

al-SVP":
o vector in I.

deal I
rator.

idea is "useless"
principal ideals:
t realized
arantee of a
ould transform
n useless to
ctive."

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
attacks far beyond

2016 Cramer–Duc
Ideal-SVP attack f
$2^{N^{1/2+o(1)}}$ in deg-
under plausible ass
about class-group
Start from Biasse–
more features of c

in I.

I.

seless”

eals:

a

form

o

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–Wesolo
Ideal-SVP attack for approx $2^{N^{1/2+o(1)}}$ in deg-N cycloto
under plausible assumptions
about class-group generator
Start from Biasse–Song, use
more features of cyclotomic

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.

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?

ikert also says idea is
to principal ideals:
gh cyclotomics have a
ructure, nobody has
d a way to exploit it in
g Ideal-SVP/BDD . . .
monly used rings,
ideals are an
ly small fraction of all
. . The weakness here is
nuch due to the structure
tomics, but rather to the
ructure of principal ideals
e 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 pc

69 subm
including

says idea is
l ideals:
mics have a
obody has
o exploit it in
/P/BDD . . .
d rings,
e an
action of all
eakness here is
to the structure
t rather to the
principal ideals
enerators."

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-quantu

69 submissions (5
including 20 lattic

s

a

t in

. .

all

re is

cture

o the

deals

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 compet

69 submissions (5 withdrawn

including 20 lattice-based en

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 Φ_{1019} option, "more conservative choice of field"; NTRU-HRSS-KEM uses Φ_{701}; NTRUEncrypt uses Φ_{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 Φ_{1019} option, “more conservative choice of field”; NTRU-HRSS-KEM uses Φ_{701}; NTRUEncrypt uses Φ_{743} etc.

Can cyclotomic attacks on Gentry be extended to these systems?

, the idea produces
far beyond this case.

amer–Ducas–Wesolowski:
VP attack for approx factor
1) in deg-N cyclotomics,
ausible assumptions
ass-group generators etc.
om Biasse–Song, use
atures of cyclotomic fields.

nniques be pushed
er approx factors?
nniques be adapted
, 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 Φ_{1019} option, “more conservative choice of field”;
NTRU-HRSS-KEM uses Φ_{701};
NTRUEncrypt uses Φ_{743} etc.

Can cyclotomic attacks on Gentry be extended to these systems?

Some sy

FrodoKE
relies on
commut
the pote
due to t

produces
l this case.

as–Wesolowski:
for approx factor
V cyclotomics,
sumptions
generators etc.
-Song, use
yclotomic fields.

e pushed
factors?
e adapted
g-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 Φ_{1019} option, "more conservative choice of field";
NTRU-HRSS-KEM uses Φ_{701};
NTRUEncrypt uses Φ_{743} etc.

Can cyclotomic attacks on Gentry be extended to these systems?

Some systems avo

FrodoKEM-640, 9
relies on matrix rin
commutative rings
the potential for w
due to the extra st

T
owski:
factor
mics,

s etc.
e
fields.

NIST post-quantum competition

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

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

Can cyclotomic attacks on Gentry be extended to these systems?

Some systems avoid cycloto

FrodoKEM-640, 9616-byte k
relies on matrix rings; says t
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 Φ_{1019} option, “more conservative choice of field”; NTRU-HRSS-KEM uses Φ_{701}; NTRUEncrypt uses Φ_{743} etc.

Can cyclotomic attacks on Gentry be extended to these systems?

Some systems avoid cyclotomics.

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

NIST post-quantum competition

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

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

Can cyclotomic attacks on Gentry be extended to these systems?

Some systems avoid cyclotomics.

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

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 Φ_{1019} option, "more conservative choice of field"; NTRU-HRSS-KEM uses Φ_{701}; NTRUEncrypt uses Φ_{743} etc.

Can cyclotomic attacks on Gentry be extended to these systems?

Some systems avoid cyclotomics.

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

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.

ost-quantum competition

nissions (5 withdrawn),
g 20 lattice-based enc.

ttice-based enc systems
er-of-2 cyclotomics.
on-power-of-2 cyclotomics:
as Φ_{1019} option, "more
ntive choice of field";
HRSS-KEM uses Φ_{701};
ncrypt uses Φ_{743} etc.

lotomic attacks on Gentry
nded 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 the

Theory
are choi
"attack
$\Rightarrow$ attac
where L

m competition

withdrawn),
e-based enc.

l enc systems
clotomics.
of-2 cyclotomics:
ption, "more
e of field";
M uses Φ_{701};
s Φ_{743} etc.

tacks on Gentry
ese 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 la

Theory 1: Best ch
are choices where
"attack against cr
$\Rightarrow$ attack against
where L_F is a "lat

tition

n),
nc.

ems

omics:
ore
;
01;
c.

Gentry
ns?

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 safet

Theory 1: Best choices of fi
are choices where we know p
"attack against cryptosystem
$\Rightarrow$ attack against problem L
where L_F is a "lattice proble

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".

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?

stems avoid cyclotomics.

EM-640, 9616-byte key:
matrix rings; says that
ative rings "have
ential for weaknesses
he extra structure".

n-lite, 14720-byte key:
iddle product" to
against the weakness
fic polynomial rings".

ned NTRU Prime
1218-byte key:
a'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?

Multiqua

Assumpt
squarefr
$\prod_{j \in J} d_j$
nonempt

$K = \mathbf{Q}($
smallest
containi

K is a d
Basis: $\prod$
subset J

e.g. $\mathbf{Q}(\sqrt{}$
$\mathbf{Q} \oplus \mathbf{Q}\sqrt{}$

id cyclotomics.

616-byte key:
ngs; says that
s "have
veaknesses
tructure".

20-byte key:
uct" to
e weakness
mial rings".

U Prime
e key:
ter 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 fiel

Assumptions: $n \in$
squarefree $d_1, \ldots,$
$\prod_{j \in J} d_j$ non-squar
nonempty subset

$K = \mathbf{Q}(\sqrt{d_1}, \ldots,$
smallest subfield o
containing $\sqrt{d_1}, \ldots$

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

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

mics.

key:

hat

ey:

s

.

Two theories of lattice safety

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

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

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

What's a good test case for F?

Multiquadratic fields

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

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

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

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

Two theories of lattice safety

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

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

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

What's a good test case for F?

Multiquadratic fields

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

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

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

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

eories of lattice safety

1: Best choices of field F
ces where we know proofs
against cryptosystem C_F
k against problem L_F",
$_F$ is a "lattice problem".

flaw in theory 1: Maybe
oices make L_F weak!

2: Safety of field F is
d by extra automorphisms,
bfields, etc. Similar
n to discrete-log crypto.

a good test case for F?

Multiquadratic fields

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

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

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

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

This fiel
has 2^n a

e.g. auto
map $a +$
$a + b\sqrt{2}$
$a - b\sqrt{2}$
$a + b\sqrt{2}$
$a - b\sqrt{2}$

ttice safety

oices of field F
we know proofs
yptosystem C_F
problem L_F",
tice problem".

eory 1: Maybe
e L_F weak!

of field F is
automorphisms,
c. Similar
te-log crypto.

st case for F?

Multiquadratic fields

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

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

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

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

This field is Galois
has 2^n automorph

e.g. automorphism
map $a + b\sqrt{2} + c$
$a + b\sqrt{2} + c\sqrt{3} +$
$a - b\sqrt{2} + c\sqrt{3} -$
$a + b\sqrt{2} - c\sqrt{3} -$
$a - b\sqrt{2} - c\sqrt{3} +$

y

eld F

roofs

n C_F

$_F$",

em".

laybe

k!

is

hisms,

to.

F?

Multiquadratic fields

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

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

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

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

This field is Galois:
has 2^n automorphisms.

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

Multiquadratic fields

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

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

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

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

This field is Galois:
has 2^n automorphisms.

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

Multiquadratic fields

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

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

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

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

This field is Galois:
has 2^n automorphisms.

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

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

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

adratic fields

tions: $n \in \{0, 1, 2, \ldots\}$;
ee $d_1, \ldots, d_n \in \mathbf{Z}$;
non-square for each
ty subset $J \subseteq \{1, \ldots, n\}$.

$\sqrt{d_1}, \ldots, \sqrt{d_n}$):
subfield of $\mathbf{C}$
ng $\sqrt{d_1}, \ldots, \sqrt{d_n}$.

egree-2^n number field.
$\prod_{j \in J} d_j$ for each
$\subseteq \{1, \ldots, n\}$.

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

This field is Galois:
has 2^n automorphisms.

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

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

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

Gentry f

Use opti
PKC 20
Eurocryp

lds

$\{0, 1, 2, \ldots\}$;

$d_n \in \mathbf{Z}$;

re for each

$J \subseteq \{1, \ldots, n\}$.

$\sqrt{d_n})$:

of $\mathbf{C}$

$., \sqrt{d_n}$.

number field.

each

$, n\}$.

$\oplus \mathbf{Q}\sqrt{6}$.

This field is Galois:

has 2^n automorphisms.

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

$a + b\sqrt{2} + c\sqrt{3} + d\sqrt{6}$;

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

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

$a - b\sqrt{2} - c\sqrt{3} + d\sqrt{6}$.

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

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

$\mathbf{Q}(\sqrt{2}, \sqrt{3})$,

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

$\mathbf{Q}$.

Gentry for multiqu

Use optimizations

PKC 2010 Smart–

Eurocrypt 2011 Ge

…};

…, n}.

ld.

This field is Galois:
has 2^n automorphisms.

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

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

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

Gentry for multiquadratics

Use optimizations from
PKC 2010 Smart–Vercauter
Eurocrypt 2011 Gentry–Hale

This field is Galois:
has 2^n automorphisms.

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

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

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

Gentry for multiquadratics

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

This field is Galois:
has 2^n automorphisms.

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

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

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

Gentry for multiquadratics

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

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

This field is Galois:
has 2^n automorphisms.

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

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

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

Gentry for multiquadratics

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

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

Multiquadratics: take, e.g.,

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

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

d is Galois:

utomorphisms.

omorphisms of $\mathbf{Q}(\sqrt{2}, \sqrt{3})$

$- b\sqrt{2} + c\sqrt{3} + d\sqrt{6}$ to

$2 + c\sqrt{3} + d\sqrt{6}$;

$2 + c\sqrt{3} - d\sqrt{6}$;

$2 - c\sqrt{3} - d\sqrt{6}$;

$2 - c\sqrt{3} + d\sqrt{6}$.

$n^2/4$ subfields.

fields of $\mathbf{Q}(\sqrt{2}, \sqrt{3})$:

$\sqrt{3})$,

$\mathbf{Q}(\sqrt{3})$, $\mathbf{Q}(\sqrt{6})$,

Gentry for multiquadratics

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

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

Multiquadratics: take, e.g.,

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

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

Smart–V

Take sho

Comput

Start ov

s:

isms.

ns of $\mathbf{Q}(\sqrt{2}, \sqrt{3})$

$\sqrt{3} + d\sqrt{6}$ to

$d\sqrt{6}$;

$d\sqrt{6}$;

$d\sqrt{6}$;

$d\sqrt{6}$.

elds.

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

$\mathbf{Q}(\sqrt{6})$,

Gentry for multiquadratics

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

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

Multiquadratics: take, e.g.,

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

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

Smart–Vercauteren

Take short random

Compute q, absolu

Start over if q is n

$2, \sqrt{3})$

6 to

Gentry for multiquadratics

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

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

Multiquadratics: take, e.g.,

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

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

Smart–Vercauteren keygen:

Take short random $g \in R$.

Compute q, absolute norm o

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

Multiquadratics: take, e.g.,

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

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

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

Gentry for multiquadratics

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

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

Multiquadratics: take, e.g.,

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

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

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

Compute root r of g in $\mathbf{Z}/q$.
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 $R = \mathbf{Z}[x]/F$; not required to be ring of integers of $\mathbf{Q}[x]/F$.

Multiquadratics: take, e.g.,

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

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

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

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

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

for multiquadratics

mizations from
10 Smart–Vercauteren,
ot 2011 Gentry–Halevi.

ic irreducible polynomial.
$= \mathbf{Z}[x]/F$; not required
ng of integers of $\mathbf{Q}[x]/F$.

adratics: take, e.g.,
$- \sqrt{2} - \sqrt{3}) \cdot$
$+ \sqrt{2} - \sqrt{3}) \cdot$
$- \sqrt{2} + \sqrt{3}) \cdot$
$+ \sqrt{2} + \sqrt{3})$.
$(\sqrt{2} + \sqrt{3}) = \mathbf{Q}(\sqrt{2}, \sqrt{3})$.

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

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

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

Smart–V
Take sho
Cipherte

uadratics

from

Vercauteren,

entry–Halevi.

ple polynomial.

not required

gers of $\mathbf{Q}[x]/F$.

ake, e.g.,

3) ·

3) ·

3) ·

3).

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

Smart–Vercauteren keygen:

Take short random $g \in R$.

Compute q, absolute norm of g.

Start over if q is not prime.

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

Public key $gR = qR + (x - r)R$ is represented as (q, r).

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

Smart–Vercautere

Take short $m \in \mathbf{Z}[$

Ciphertext is $m(r)$

en,
evi.

mial.
red
x]/F.

, $\sqrt{3}$).

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

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

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

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

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

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

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

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

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

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

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 $\mathbf{Z}/q$.

Public key $gR = qR + (x - r)R$ is represented as (q, r).

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

Smart–Vercauteren encryption:

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

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

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

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

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

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

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

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

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

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

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

Vercauteren keygen:
ort random $g \in R$.
e q, absolute norm of g.
er if q is not prime.

e root r of g in $\mathbf{Z}/q$.
ey $gR = qR + (x - r)R$
ented as (q, r).

plemented multiquadratic
on of Gentry–Halevi
nic keygen speedup:
of requiring prime q,
gcd$\{b, q\} > 1$ for each
norm $a + b\sqrt{d_i}$ of g.
arefree q will work.)

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

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

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

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

Gentry s
complex
algorithr
in securi

Flaw in
for some
keygen t
in securi

n keygen:
n $g \in R$.
ute norm of g.
not prime.

f g in $\mathbf{Z}/q$.
$qR + (x - r)R$
$q, r)$.

multiquadratic
try–Halevi
speedup:
g prime q,
> 1 for each
$b\sqrt{d_i}$ of g.
will work.)

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

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

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

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

Gentry says "comp
complexity of all o
algorithms must b
in security parame

Flaw in Smart–Ver
for some choices o
keygen time is not
in security parame

of g.

.

$r)R$

dratic

,

ch

.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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.)

Vercauteren encryption:
ort $m \in \mathbf{Z}[x]/F$.
ext is $m(r) \in \mathbf{Z}/q$.

orphic operations:
ltiply ciphertexts $m(r)$
multiply messages m.

ion:
$\in \{0, 1, \ldots, q-1\}$,
e $c/g \in \mathbf{Q}[x]/F$,
o element of $\mathbf{Z}[x]/F$,
by g, subtract from c.

ion works if
efficient of $m/g \in \mathbf{Q}[x]/F$
$1/2, 1/2)$.

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.)

Why thi
Take fiel

n encryption:
$[x]/F$.
$\in \mathbf{Z}/q$.

erations:
ertexts $m(r)$
essages m.

$., q-1\}$,
$[x]/F$,
of $\mathbf{Z}[x]/F$,
tract from c.

if
$m/g \in \mathbf{Q}[x]/F$

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
Take field k of siz

on:

r)

n.

,

n c.

$\mathbf{Q}[x]/F$

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 prim

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 $p \le 2^n$, 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 $\mathbf{F}_p[x]$.

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

says "computational
ity of all of these
ms must be polynomial
ty parameter".

Smart–Vercauteren:
e choices of F,
time is not polynomial
ty parameter.

tiquadratic F, keygen is
usly slow: far too many
find prime q. (Adaptation
ry–Halevi speedup gives
olynomial 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 $p \le 2^n$, have
$\Theta(p)$ distinct linear factors h.

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

Our mul
Smart–V
adaptati

1. Gene
support
Use $R =$

putational
of these
e polynomial
ter".

rcauteren:
of F,
polynomial
ter.

c F, keygen is
far too many
q. (Adaptation
speedup gives
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.

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

Our multiquadrati
Smart–Vercauteren
adaptation of Gen

1. Generalize cryp
support n polynom
Use $R = \mathbf{Z}[\sqrt{d_1}, \ldots$

nial

al

n is
any
tation
ves
ent.)

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

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

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

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

Our multiquadratic tweaks t
Smart–Vercauteren (includin
adaptation of Gentry–Halevi

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

s happens: Fix prime p.
ld k of size p^2.

d_n are squares in k,
its completely in $k[x]$.
$\{1, 2\}$ for each
tor h of F in $\mathbf{F}_p[x]$.

c: for most $p \le 2^n$, have
stinct linear factors h.

linear factor h:
bability $\approx 1/p$,
s g in $\mathbf{F}_p[x]$,
p^2 to divide norm of g
is non-square in $\mathbf{F}_p$.

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

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

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

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

Computi

Fix posit
Assume
i.e., log

: Fix prime p.
e p^2.

ares in k,
etely in $k[x]$.
each
in $\mathbf{F}_p[x]$.

t $p \le 2^n$, have
ar factors h.

ctor h:
$1/p$,
$x]$,
e norm of g
uare in $\mathbf{F}_p$.

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

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

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

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

Computing units

Fix positive non-sq
Assume d quasipo
i.e., $\log d \in n^{O(1)}$.

e p.

x].

have

h.

g

.

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

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

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

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

Computing units

Fix positive non-square $d \in$

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

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

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

Computing units

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

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

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

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

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

Computing units

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

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

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

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

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

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

Computing units

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

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

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

tiquadratic tweaks to
/ercauteren (including
on of Gentry–Halevi):

ralize cryptosystem to
n polynomial variables.
$= \mathbf{Z}[\sqrt{d_1}, \ldots, \sqrt{d_n}]$.

outine: Construct uniform
invertible element of R/p.

se $y \in \Theta(2^n/n)$.
to be invertible mod all
$p \leq y$. Heuristically,
ance of squarefree norm.

Computing units

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

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

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

Take a r
$K = \mathbf{Q}($
Assume

The set
is the gr
of all 2^n
Analogo

Comput
all norm

c tweaks to
n (including
try–Halevi):

tosystem to
nial variables.
$\ldots, \sqrt{d_n}]$.

onstruct uniform
element of R/p.

$2^n/n)$.
ertible mod all
uristically,
uarefree norm.

Computing units

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

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

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

Take a multiquadr
$K = \mathbf{Q}(\sqrt{d_1}, \ldots,$
Assume $n > 0$ and

The set of **multiq**
is the group gener
of all $2^n - 1$ quad
Analogous to cyclo

Compute this grou
all normalized fund

o
ng
):
to
les.
niform
f R/p.
d all
orm.

Computing units

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

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

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

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

The set of **multiquadratic u**
is the group generated by ur
of all $2^n - 1$ quadratic subfi
Analogous to cyclotomic uni

Compute this group by com
all normalized fundamental u

Computing units

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

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

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

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

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

Compute this group by computing
all normalized fundamental units.

Computing units

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

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

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

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

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

Compute this group by computing
all normalized fundamental units.

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

ing units

tive non-square $d \in \mathbf{Z}$.
d quasipoly in 2^n;
$d \in n^{O(1)}$.

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

d algorithms compute
with $\varepsilon = a + b\sqrt{d}$
$(\log \varepsilon)^{1+o(1)}$; quasipoly.
ve time by instead
ting ε as product.)

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

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

Compute this group by computing all normalized fundamental units.

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 Wa
algorithr

quare $d \in \mathbf{Z}$.
ly in 2^n;

$, \pm 1, \pm\varepsilon, \pm\varepsilon^2, \ldots\}$
ng of integers of
ue $\varepsilon > 1$, the
amental unit.
g $4d$); quasipoly.

ns compute
$a + b\sqrt{d}$
$^{(1)}$; quasipoly.
instead
roduct.)

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

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

Compute this group by computing all normalized fundamental units.

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

1966 Wada: expo
algorithm for mult

Z.

$\pm\varepsilon^2, \ldots\}$
gers of
the
nit.
sipoly.

te

oly.

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

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

Compute this group by computing all normalized fundamental units.

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

1966 Wada: exponential-tim
algorithm for multiquadratic

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

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

Compute this group by computing all normalized fundamental units.

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

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

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

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

Compute this group by computing all normalized fundamental units.

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

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

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

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

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

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

Compute this group by computing all normalized fundamental units.

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

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

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

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

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

multiquadratic field

$\sqrt{d_1}, \ldots, \sqrt{d_n})$.

$n > 0$ and all $d_i > 0$.

of **multiquadratic units**
oup generated by units
-1 quadratic subfields.
us to cyclotomic units.

e this group by computing
alized fundamental units.

eyond this: compute $\mathcal{O}_K^*$.
se Eisenträger–Hallgren–
Song, but we don't want
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: $\mathbf{Q}$; $\mathbf{Q}(\sqrt{d})$.

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

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

Second s
Comput

ratic field

$\sqrt{d_n}$).

l all $d_i > 0$.

uadratic units

ated by units

ratic subfields.

otomic units.

up by computing

damental units.

s: compute $\mathcal{O}_K^*$.

äger–Hallgren–

we don't want

m 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: $\mathbf{Q}$; $\mathbf{Q}(\sqrt{d})$.

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

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

Second step:

Compute $U = \mathcal{O}_{K}^*$

0.

units

nits

elds.

its.

puting

units.

e $\mathcal{O}_K^*$.

ren–

want

ters.

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

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

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

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

Second step:

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

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

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

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

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

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

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

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

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

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

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

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

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: $\mathbf{Q}$; $\mathbf{Q}(\sqrt{d})$.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

ada: exponential-time $\mathcal{O}_K^*$
m for multiquadratics.

p: Recursively compute
ups for three proper
s $K_\sigma, K_\tau, K_{\sigma\tau}$ of K.
ses: $\mathbf{Q}$; $\mathbf{Q}(\sqrt{d})$.

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

$= \mathbf{Q}(\sqrt{2}, \sqrt{3}, \sqrt{5})$,
ate σ, τ: have
$(\sqrt{2}, \sqrt{3})$;
$(\sqrt{2}, \sqrt{5})$;
$\mathbf{Q}(\sqrt{2}, \sqrt{15})$.

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 st
identify
trying to
of produ

nential-time $\mathcal{O}_K^*$
iquadratics.

ively compute
ree proper
$K_{\sigma\tau}$ of K.
$(\sqrt{d})$.

identity
K.
$x) = x\}$.

$\sqrt{3}, \sqrt{5})$,
nave

5).

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

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

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

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

Third step:
identify $(\mathcal{O}_K^*)^2$ ins
trying to compute
of products of gen

he $\mathcal{O}_K^*$
s.

pute

Second step:

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

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

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

Proof:

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

Third step:

identify $(\mathcal{O}_K^*)^2$ inside U by
trying to compute square ro
of products of generators of

Second step:

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

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

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

Proof:

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

Third step:

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

Second step:

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

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

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

Proof:

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

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

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

Second step:

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

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

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

Proof:

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

Third step:

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

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

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

Proof:

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

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

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

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

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

Second step:

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

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

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

Proof:

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

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

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

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

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

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

step:

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

$\le \mathcal{O}^*_K$.

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

$^*_K$ then

$\mathcal{O}^*_{K_\sigma}$;

$\mathcal{O}^*_{K_\tau}$;

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

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

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 χ
with $\chi(\alpha_1),\ldots,\chi(\alpha_k) \in \{-1,1\}$.

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

Computi

Main go

where R

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

so

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

$\in U$.

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 χ with $\chi(\alpha_1), \ldots, \chi(\alpha_k) \in \{-1, 1\}$.

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

Computing genera

Main goal: Find g

where $R = \mathbf{Z}[\sqrt{d_1}$

$^*_{K_{\sigma\tau}}$).

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 χ with $\chi(\alpha_1), \ldots, \chi(\alpha_k) \in \{-1, 1\}$.

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

Computing generators

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

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

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

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

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

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

Computing generators

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

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

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

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

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

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

Computing generators

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

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.

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

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

Computing generators

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

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

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

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

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

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

Computing generators

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

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

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

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

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

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

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

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

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

Computing generators

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

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

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

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

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

ep:

$(\mathcal{O}_K^*)^2$ inside U by

o compute square roots

ucts of generators of U.

roducts.

much better using

idea from 1991 Adleman.

$_k^{e_k}$ square $\Rightarrow$

$\cdots \chi(\alpha_k)^{e_k} = 1$

quadratic character χ

$\alpha_1), \ldots, \chi(\alpha_k) \in \{-1, 1\}$.

quation, usually reducing

by 1. Use many such χ.

Computing generators

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

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

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

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

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

Unit mu

unit mul

unit mul

$\Rightarrow$ some

ide U by
square roots
erators of U.

er using
1991 Adleman.

$\Rightarrow$
${}^{e_k} = 1$
character χ
$(\alpha_k) \in \{-1, 1\}$.

sually reducing
many such χ.

Computing generators

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

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

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

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

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

Unit multiple of g
unit multiple of g
unit multiple of g
$\Rightarrow$ some ug^2 with

ots
U.

eman.

χ
$-1, 1\}$.

ucing
h χ.

Computing generators

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

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

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

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

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

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

Computing generators

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

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

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

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

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

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

Computing generators

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

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

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

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

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

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

Use quadratic characters
(with values ± 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 $R = \mathbf{Z}[\sqrt{d_1}, \ldots, \sqrt{d_n}]$.

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

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

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

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

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

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

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

Computing generators

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

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

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

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

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

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

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

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

All of this takes quasipoly time.

ing generators

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

: Reuse the equation
$(g)g\tau(g)/\sigma(g\sigma(\tau(g)))$.
oot of g^2 is $\pm g$.

compute $g\sigma(g)$?

mpute relative norm
gR from K to K_σ.
deal generated by $g\sigma(g)$.

ely compute a generator
deal: probably not $g\sigma(g)$.
$g\sigma(g)$ 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 ± 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.

Computi

Assume
(More w
to $< n^2$;

tors

given gR,

$, \ldots, \sqrt{d_n}]$.

he equation

$/\sigma(g\sigma(\tau(g)))$.

is $\pm g$.

$g\sigma(g)$?

tive norm

K to K_σ.

rated by $g\sigma(g)$.

ute a generator

ably not $g\sigma(g)$.

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

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 ± 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 g

Assume $d_1, \ldots, d_n$

(More work seems

to $<n^2$; see paper

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

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

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

All of this takes quasipoly time.

Computing short generators

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

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

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

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

All of this takes quasipoly time.

Computing short generators

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

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

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

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

All of this takes quasipoly time.

Computing short generators

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

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 ± 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.

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

ltiple of $g\sigma(g)$,
ltiple of $g\tau(g)$,
ltiple of $g\sigma(\tau(g))$
ug^2 with $u \in \mathcal{O}_K^*$.

dratic characters
lues ± 1 on g)
fy $v \in \mathcal{O}_K^*$
t vug^2 is a square.

mpute square root:
it multiple of g,
e g' with $g'\mathcal{O}_K = g\mathcal{O}_K$.

is 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 $2^n \operatorname{Log} g$
on MQ unit basis are
almost certainly in $(-0.1, 0.1)$.

u^{2^n} is ar
$\operatorname{Log} u^{2^n}$
closest v

$\sigma(g)$,
$\tau(g)$,
$\sigma(\tau(g))$
$u \in \mathcal{O}_K^*$.

racters
n g)

K
a square.

are root:
e of g,
$g'\mathcal{O}_K = g\mathcal{O}_K$.

uasipoly 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 $2^n \operatorname{Log} g$
on MQ unit basis are
almost certainly in $(-0.1, 0.1)$.

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

$g\mathcal{O}_K$.

me.

Computing short generators

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

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

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

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

Computing short generators

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

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

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

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

Computing short generators

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

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

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

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

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

Computing short generators

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

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

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

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

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

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

Computing short generators

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

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

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

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

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

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

Computing short generators

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

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

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

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

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

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

Computing short generators

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

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

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

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

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

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

ing short generators

$d_1, \ldots, d_n \geq 2^{1.03n}$.
ork seems to push bound
see paper and software.)

ltiquadratic (MQ) units.
units.
ne generator ug.

c: For most $d_1, \ldots, d_n$,
ators $\log \varepsilon$
er than $2^{0.51n}$;
cients of $2^n \operatorname{Log} g$
unit basis are
certainly in $(-0.1, 0.1)$.

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

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

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

Slightly

Find MQ

but skip

generators

$\geq 2^{1.03n}$.

to push bound

and software.)

ic (MQ) units.

tor ug.

st $d_1, \ldots, d_n$,

$51n$;

$2^n \operatorname{Log} g$

are

$(-0.1, 0.1)$.

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

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

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

Slightly simpler:

Find MQ units,
but skip finding al

ound

are.)

nits.

d_n,

1).

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

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

Use quadratic character: g^{2^n}.
Square root: $\pm g^{2^{n-1}}$.
Use quadratic character: $g^{2^{n-1}}$.
Square root: $\pm g^{2^{n-2}}$.
$\vdots$
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^{2^n} is an MQ unit.
$\operatorname{Log} u^{2^n} = 2^n \operatorname{Log} u$ is
closest vector to $2^n \operatorname{Log} ug$.

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

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

Slightly simpler:

Find MQ units,
but skip finding all units.

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

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

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

Slightly simpler:

Find MQ units,
but skip finding all units.

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

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

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

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

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

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

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

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