Slow-boiled frogs

Daniel J. Bernstein

Background:
Lattice computations

Euclid's algorithm

Input: positive $u, v \in \mathbb{Z}$.

Goal: identify a basis
for the 1-dimensional lattice $u\mathbb{Z} + v\mathbb{Z}$.

Algorithm: Repeatedly replace (u, v)
with shorter $(u - v, v)$ or shorter $(u, v - u)$,
stopping when no improvement is possible.

Often-faster algorithm jumps through subtractions:
$(u, v) \mapsto (u \bmod v, v)$ or $(u, v) \mapsto (u, v \bmod u)$.

Reducing lattice bases in dimension 2

Lagrange and Gauss encountered
2-dimensional lattices in number theory.

Applied a simple, fast algorithm to find
shortest-length nonzero vectors in those lattices.
i.e., solved "SVP": the "shortest-vector problem".

Input: independent vectors $u, v \in \mathbb{R}^2$.

Algorithm: Repeatedly replace (u, v)
with shorter $(u \bmod v, v)$ or shorter $(u, v \bmod u)$,
with "centered" notion of mod.

Higher dimensions: existence

Hermite wrote a letter (published 1850) to Jacobi showing that any dim-n lattice L for $n \geq 1$ has a nonzero vector of length $\leq (4/3)^{(n-1)/4} (\det L)^{1/n}$. Proof generalizes Lagrange.

Higher dimensions: existence

Hermite wrote a letter (published 1850) to Jacobi showing that any dim-n lattice L for $n \geq 1$ has a nonzero vector of length $\leq (4/3)^{(n-1)/4} (\det L)^{1/n}$. Proof generalizes Lagrange.

1896 Minkowski convex-body theorem implies nonzero vector of length $\leq (2/(\operatorname{vol} B_n)^{1/n})(\det L)^{1/n}$ where B_n is the n-dimensional unit ball.
As $n \to \infty$: $2/(\operatorname{vol} B_n)^{1/n} \in (2 + o(1))(n/2\pi e)^{1/2}$.

Higher dimensions: existence

Hermite wrote a letter (published 1850) to Jacobi showing that any dim-n lattice L for $n \geq 1$ has a nonzero vector of length $\leq (4/3)^{(n-1)/4} (\det L)^{1/n}$. Proof generalizes Lagrange.

1896 Minkowski convex-body theorem implies nonzero vector of length $\leq (2/(\operatorname{vol} B_n)^{1/n})(\det L)^{1/n}$ where B_n is the n-dimensional unit ball.
As $n \to \infty$: $2/(\operatorname{vol} B_n)^{1/n} \in (2 + o(1))(n/2\pi e)^{1/2}$.

1956 Rogers: shortest nonzero length is $(1 + o(1))(n/2\pi e)^{1/2} (\det L)^{1/n}$ for most lattices.

Sufficiently fast lattice computations

Lattices show up in many more math papers. Sometimes the papers do lattice computations.

e.g. 1967 Coveyou–Macpherson "Fourier analysis of random number generators" encountered lattices with $n \leq 10$. Solved SVP by enumeration of lattice vectors after preliminary lattice-basis reduction.

Sufficiently fast lattice computations

Lattices show up in many more math papers. Sometimes the papers do lattice computations.

e.g. 1967 Coveyou–Macpherson "Fourier analysis of random number generators" encountered lattices with $n \leq 10$. Solved SVP by enumeration of lattice vectors after preliminary lattice-basis reduction.

e.g. 1982 Lenstra–Lenstra–Lovasz "Factoring polynomials with rational coefficients" included a polynomial-time algorithm for length at most $(4/3+\epsilon)^{(n-1)/4}(\det L)^{1/n}$, which is good enough for factorization (and many other applications).

Asymptotics improving

1983 Kannan, 1987 Kannan:
enumeration solves SVP-n in time $2^{O(n \log n)}$.

Asymptotics improving

1983 Kannan, 1987 Kannan:
enumeration solves SVP-n in time $2^{O(n \log n)}$.

1987 Schnorr: the BKZ-(β, n) algorithm produces length at most $(\sqrt{6}\beta)^{n/\beta}(\det L)^{1/n}$ for $2 \leq \beta \leq n$, at the expense of calling an SVP-β subroutine.

Asymptotics improving

1983 Kannan, 1987 Kannan:
enumeration solves SVP-n in time $2^{O(n \log n)}$.

1987 Schnorr: the BKZ-(β, n) algorithm produces length at most $(\sqrt{6}\beta)^{n/\beta}(\det L)^{1/n}$ for $2 \leq \beta \leq n$, at the expense of calling an SVP-β subroutine.

BKZ-(β, n) using SVP-β enumeration is poly-time if $\beta \in \Theta(\log n / \log \log n)$ as $n \to \infty$,
so poly-time for length $(1 + o(1))^n (\det L)^{1/n}$.

Important trends continuing today

Already visible in these papers from the 1980s:

- Lattice algorithms becoming faster and faster.
- Also becoming more and more complicated, with more and more avenues for speedups.
- *Analyses* becoming *much* more complicated.

Important trends continuing today

Already visible in these papers from the 1980s:

- Lattice algorithms becoming faster and faster.
- Also becoming more and more complicated, with more and more avenues for speedups.
- *Analyses* becoming *much* more complicated.

This is great for writing more papers!

See, e.g., survey of 20 advances in 2018–2021.

Important trends continuing today

Already visible in these papers from the 1980s:

- Lattice algorithms becoming faster and faster.
- Also becoming more and more complicated, with more and more avenues for speedups.
- *Analyses* becoming *much* more complicated.

This is great for writing more papers!
See, e.g., survey of 20 advances in 2018–2021.

Would need an entire talk to summarize current performance conjectures for pruned enumeration, sieving, near-neighbor sieving, dimensions for free, tuple sieving, hybrid attacks, dual attacks, etc.

Lattice-based cryptography

NP-hard PKE: the myth begins

1978 Hellman–Merkle "Hiding information and signatures in trapdoor knapsacks": "The knapsack problem is an NP-complete combinatorial problem that is strongly believed to be computationally difficult to solve in general. Specific instances of this problem that appear very difficult to solve unless one possesses 'trapdoor information' used in the design of the problem are demonstrated."

Knapsack problems

Basic knapsack problem: Find small $c_1, \ldots, c_n \in \mathbb{Z}$ (simplest version: all 0 or 1, "subset-sum problem") given $v_1, \ldots, v_n$ and $s = c_1 v_1 + \cdots + c_n v_n$.

Knapsack problems are lattice problems

Basic knapsack problem: Find small $c_1, \ldots, c_n \in \mathbb{Z}$ (simplest version: all 0 or 1, "subset-sum problem") given $v_1, \ldots, v_n$ and $s = c_1 v_1 + \cdots + c_n v_n$.

One approach to a solution:
Easy to find *big* $t_1, \ldots, t_n \in \mathbb{Z}$
with $s = t_1 v_1 + \cdots + t_n v_n$.
Note $(t_1 - c_1)v_1 + \cdots + (t_n - c_n)v_n = 0$.
Finding $c_1, \ldots, c_n$ is equivalent to finding a vector in L close to $(t_1, \ldots, t_n)$, where L is the lattice of $(a_1, \ldots, a_n)$ with $a_1 v_1 + \cdots + a_n v_n = 0$.

More lattice-based cryptosystems

1978 Shamir, 1979 Graham–Shamir, etc. proposed further "knapsack" cryptosystems.

More lattice-based cryptosystems

1978 Shamir, 1979 Graham–Shamir, etc. proposed further "knapsack" cryptosystems.

Practically all "knapsack" proposals were then broken by 1982 Shamir, 1983 Adleman, 1983 Brickell–Lagarias–Odlyzko, etc.

More lattice-based cryptosystems

1978 Shamir, 1979 Graham–Shamir, etc. proposed further "knapsack" cryptosystems.

Practically all "knapsack" proposals were then broken by 1982 Shamir, 1983 Adleman, 1983 Brickell–Lagarias–Odlyzko, etc.

"The attacks broke NP-hard problems?"
— Conjecturally no poly-time PKE is NP-hard.

NTRU

1996 Hoffstein–Pipher–Silverman preprint "NTRU: a new high speed public key cryptosystem":

Public key: $v \in (\mathbb{Z}/q)[x]/(x^n - 1)$.
Ciphertext: $av + b$ for small $a, b \in \mathbb{Z}[x]/(x^n - 1)$.

So ciphertext is small-coeff linear combination of $v, xv, \ldots, x^{n-1}v, 1, x, \ldots, x^{n-1}$.

There's also a decryption procedure using small private key (k, ℓ) where $0 = kv + \ell$.
(Require coeffs of ℓ to be in, e.g., $\{-3, 0, 3\}$; multiply ciphertext by k; "reduce" mod 3; etc.)

Not dead yet

Many subsequent variants of NTRU:
e.g., using $x^n + 1$ instead of $x^n - 1$.

Not dead yet . . . with bigger ciphertexts

Many subsequent variants of NTRU:
e.g., using $x^n + 1$ instead of $x^n - 1$.

Most obvious change: n keeps increasing to
try to compensate for advances in attacks. e.g.:

- 1996 preprint proposed $n = 83$ for security 2^{80}.
- 1998 revision proposed $n = 167$ for security 2^{78}.

Not dead yet . . . with bigger ciphertexts

Many subsequent variants of NTRU:
e.g., using $x^n + 1$ instead of $x^n - 1$.

Most obvious change: n keeps increasing to try to compensate for advances in attacks. e.g.:

- 1996 preprint proposed $n = 83$ for security 2^{80}.
- 1998 revision proposed $n = 167$ for security 2^{78}.
- 2010 Lindner–Peikert proposed $n = 256$ for security "about" 2^{150}, "at least" 2^{128}.

Not dead yet . . . with bigger ciphertexts

Many subsequent variants of NTRU:
e.g., using $x^n + 1$ instead of $x^n - 1$.

Most obvious change: n keeps increasing to try to compensate for advances in attacks. e.g.:

- 1996 preprint proposed $n = 83$ for security 2^{80}.
- 1998 revision proposed $n = 167$ for security 2^{78}.
- 2010 Lindner–Peikert proposed $n = 256$ for security "about" 2^{150}, "at least" 2^{128}.
- 2017 NTRU-HPS, 2017 Kyber, 2017 Frodo proposed $n = 509, 512, 640$ for security 2^{128}.

Not dead yet . . . with bigger ciphertexts

Many subsequent variants of NTRU:
e.g., using $x^n + 1$ instead of $x^n - 1$.

Most obvious change: n keeps increasing to try to compensate for advances in attacks. e.g.:

- 1996 preprint proposed $n = 83$ for security 2^{80}.
- 1998 revision proposed $n = 167$ for security 2^{78}.
- 2010 Lindner–Peikert proposed $n = 256$ for security "about" 2^{150}, "at least" 2^{128}.
- 2017 NTRU-HPS, 2017 Kyber, 2017 Frodo proposed $n = 509$, 512, 640 for security 2^{128}.
- 2030: ?

Why do some people claim confidence?

Myths about history: "the underlying worst-case problems—e.g., approximating short vectors in lattices—have been deeply studied by some of the great mathematicians and computer scientists going back at least to Gauss, and appear to be very hard."

Why do some people claim confidence?

Myths about history: "the underlying worst-case problems—e.g., approximating short vectors in lattices—have been deeply studied by some of the great mathematicians and computer scientists going back at least to Gauss, and appear to be very hard."

More hype: the problem of finding small a, b given $v, av + b \in (\mathbb{Z}/q)[x]/(x^n + 1)$ for n a power of 2 (or, more generally, the "Ring-LWE" problem) has "very strong hardness guarantees".

Why do some people claim confidence?

Myths about history: "the underlying worst-case problems—e.g., approximating short vectors in lattices—have been deeply studied by some of the great mathematicians and computer scientists going back at least to Gauss, and appear to be very hard."

More hype: the problem of finding small a, b given $v, av + b \in (\mathbb{Z}/q)[x]/(x^n + 1)$ for n a power of 2 (or, more generally, the "Ring-LWE" problem) has "very strong hardness guarantees".

Whenever a lattice problem is broken: Downplay that; hype a different problem that isn't broken.

Theorems begging questions

e.g. 2012 Lyubashevsky–Peikert–Regev theorem:
can convert a poly-time Ring-LWE algorithm
into a poly-time algorithm to solve
"worst-case problems on ideal lattices".

Are these ideal-lattice problems hard?

Theorems begging questions

e.g. 2012 Lyubashevsky–Peikert–Regev theorem:
can convert a poly-time Ring-LWE algorithm
into a poly-time algorithm to solve
"worst-case problems on ideal lattices".

Are these ideal-lattice problems hard?

Focus for rest of talk: given $n = 2^e$ and ideal $I \neq 0$
of $R = \mathbb{Z}[x]/(x^n + 1)$, find "short" nonzero $\alpha \in I$.
"Short": within polynomial factor of shortest.

Theorems begging questions

e.g. 2012 Lyubashevsky–Peikert–Regev theorem:
can convert a poly-time Ring-LWE algorithm
into a poly-time algorithm to solve
"worst-case problems on ideal lattices".

Are these ideal-lattice problems hard?

Focus for rest of talk: given $n = 2^e$ and ideal $I \neq 0$
of $R = \mathbb{Z}[x]/(x^n + 1)$, find "short" nonzero $\alpha \in I$.
"Short": within polynomial factor of shortest.

Sometimes cryptosystems use ideals directly.
e.g. 2009 Gentry: short secret g; public $I = gR$.

Computational algebraic number theory

Many useful algorithms

1993 Cohen list of "the main computational tasks of algebraic number theory":

- Compute integral bases, prime ideals, $\operatorname{ord}_P \alpha$.
- Compute Galois group of $K = (R - \{0\})^{-1}R$.
- Compute regulator and unit group R^*.
- Compute class number and class group.
- Recognize principal ideals; compute generators.

"In the rest of this book, we will give algorithms for these tasks . . . " (Next slide: more about speed.)

Traditional application: Diophantine equations.

S-units (aka "S-smooth numbers")

"S-units": $U_S = \{\alpha \in K^* : \operatorname{ord}_P \alpha \neq 0 \Rightarrow P \in S\}$.
"S-unit lattice": $\operatorname{Log}_S U_S$.

For typical R, conjecturally subexponential search through small elements of R finds $\operatorname{Log}_S U_S$ where y is subexp and S has all prime ideals of norm $\leq y$.

S-units (aka "S-smooth numbers")

"S-units": $U_S = \{\alpha \in K^* : \operatorname{ord}_P \alpha \neq 0 \Rightarrow P \in S\}$.
"S-unit lattice": $\operatorname{Log}_S U_S$.

For typical R, conjecturally subexponential search through small elements of R finds $\operatorname{Log}_S U_S$ where y is subexp and S has all prime ideals of norm $\leq y$.

Also gives $\operatorname{Log}_T U_T$ for any $T \subseteq S$; in particular, gives $\operatorname{Log} R^*$; also conjecturally gives class group.

S-units (aka "S-smooth numbers")

"S-units": $U_S = \{\alpha \in K^* : \operatorname{ord}_P \alpha \neq 0 \Rightarrow P \in S\}$.
"S-unit lattice": $\operatorname{Log}_S U_S$.

For typical R, conjecturally subexponential search through small elements of R finds $\operatorname{Log}_S U_S$ where y is subexp and S has all prime ideals of norm $\leq y$.

Also gives $\operatorname{Log}_T U_T$ for any $T \subseteq S$; in particular, gives $\operatorname{Log} R^*$; also conjecturally gives class group.

Given nonzero ideal I of R, search small elements $\beta \in I$. Any βI^{-1} supported on S gives class of I etc.

S-units (aka "S-smooth numbers")

"S-units": $U_S = \{\alpha \in K^* : \operatorname{ord}_P \alpha \neq 0 \Rightarrow P \in S\}$.
"S-unit lattice": $\operatorname{Log}_S U_S$.

For typical R, conjecturally subexponential search through small elements of R finds $\operatorname{Log}_S U_S$ where y is subexp and S has all prime ideals of norm $\leq y$.

Also gives $\operatorname{Log}_T U_T$ for any $T \subseteq S$; in particular, gives $\operatorname{Log} R^*$; also conjecturally gives class group.

Given nonzero ideal I of R, search small elements $\beta \in I$. Any βI^{-1} supported on S gives class of I etc.

Same idea: NFS for integer factorization.

Short generators

2014.02 Bernstein "A subfield-logarithm attack against ideal lattices":

1. To find short g given $I = gR$: find some generator gu in subexp time; reduce $\operatorname{Log} gu$ mod $\operatorname{Log} R^*$ to find $\operatorname{Log} g$. "I think that this approach to finding generators is reasonably well known among computational algebraic number theorists."

Short generators

2014.02 Bernstein "A subfield-logarithm attack against ideal lattices":

1. To find short g given $I = gR$: find some generator gu in subexp time; reduce $\operatorname{Log} gu$ mod $\operatorname{Log} R^*$ to find $\operatorname{Log} g$. "I think that this approach to finding generators is reasonably well known among computational algebraic number theorists."

2. Exploit subfields to use lower-dim lattices.

Subsequent work: e.g., 2017 Bauch–Bernstein–de Valence–Lange–van Vredendaal; 2019 Biasse–van Vredendaal; 2020 Biasse–Fieker–Hofmann–Page.

Simple reduction

One technique to reduce mod L (e.g., 2000 Cohen, pages 375–376): reduce mod each $v \in D$ and repeat, after precomputing D that generates L.

Works best when elements of D are short: e.g., all elements of L below some size bound.

Simple reduction mod cyclotomic units

One technique to reduce mod L (e.g., 2000 Cohen, pages 375–376): reduce mod each $v \in D$ and repeat, after precomputing D that generates L.

Works best when elements of D are short: e.g., all elements of L below some size bound.

2014.10 Campbell–Groves–Shepherd: for $R = \mathbb{Z}[x]/(x^n + 1)$ with n a power of 2, the usual cyclotomic units $(1 - x^3)/(1 - x)$ etc. are short. Conjecturally fast break of 2009 Gentry.

More analysis under various conjectures: 2015.04 Cramer–Ducas–Peikert–Regev, 2016 Biasse–Song.

Close principal multiples

2015.02 Bernstein: "We also know that the ideal I contains [the target short element] . . . This means that there can't be much gap between [that element] and I: the ratio is some small ideal J, and replacing I by IJ will give us exactly the desired ideal . . . [After computing] the class group, the class of I, and the classes of all small prime ideals, we don't have to waste time enumerating J for which IJ isn't principal, since the condition 'IJ is principal' linearly constrains the exponents in J's factorization."

Example of close principal multiples

2015.04 Bernstein for, e.g., $x^{256}+1$: "If we know the classes of (say) ideals $I, P_1, P_2, \ldots, P_{10}$, and we want to search through a range of 2^{182} possibilities for a principal ideal of the form $IP_1^{e_1}P_2^{e_2}\cdots P_{10}^{e_{10}}$ with small $(e_1, e_2, \ldots, e_{10})$, then we can immediately restrict attention to a linear subspace that typically includes just 1 possibility. It's conceivable that the classes will conspire against the search, but the main heuristic used in class-group computations is that small primes rarely engage in such conspiracies."

S-unit attacks

2016.08 Bernstein: "The idea of

- solving a close-vector problem in the unit lattice, to recover a short g from any unit multiple ug

generalizes straightforwardly to

- solving a close-vector problem in the S-unit lattice, to recover a short g from any S-unit multiple ug."

Quantifying close principal multiples

2016.09 Cramer–Ducas–Wesolowski, considering close principal multiples as in 2015.02 Bernstein for $R = \mathbb{Z}[x]/(x^n + 1)$ with n a power of 2, conjecturing asymptotics: quantum poly-time algorithm to find α within $\exp(n^{1/2+o(1)})$ of shortest.

Previous poly-time algorithms: $\exp(n^{1+o(1)})$.

Quantifying close principal multiples

2016.09 Cramer–Ducas–Wesolowski, considering close principal multiples as in 2015.02 Bernstein for $R = \mathbb{Z}[x]/(x^n + 1)$ with n a power of 2, conjecturing asymptotics: quantum poly-time algorithm to find α within $\exp(n^{1/2+o(1)})$ of shortest.

Previous poly-time algorithms: $\exp(n^{1+o(1)})$.

For comparison, 2015.04 Cramer–Ducas–Peikert–Regev: known algorithms "apply only to *principal* ideals"; this is a "barrier".

Quantifying S-unit attacks

2019 Pellet-Mary–Hanrot–Stehlé,
using S-unit attacks as in 2016 Bernstein,
taking small S, conjecturing asymptotics:
approx factor $\exp(n^{1/4+o(1)})$ in time $\exp(n^{1/2+o(1)})$,
after precomputation depending only on R.

BKZ etc.: $\exp(n^{1/4+o(1)})$ in time $\exp(n^{3/4+o(1)})$.

Quantifying S-unit attacks

2019 Pellet-Mary–Hanrot–Stehlé,
using S-unit attacks as in 2016 Bernstein,
taking small S, conjecturing asymptotics:
approx factor $\exp(n^{1/4+o(1)})$ in time $\exp(n^{1/2+o(1)})$,
after precomputation depending only on R.

BKZ etc.: $\exp(n^{1/4+o(1)})$ in time $\exp(n^{3/4+o(1)})$.

For comparison, 2016/2016/2018 Peikert:
approx factor $\exp(n^{1/2+o(1)})$ is a "barrier",
an "inherent barrier", and a "natural barrier"
for the number-theoretic algorithms.

Quantifying S-unit attacks, continued

2021.08.20 Bernstein talk "S-unit attacks",
reporting results from joint papers in progress:
S-unit attacks find α within poly factor of shortest,
conjecturally for $\#S, \#D,$ time $\in \exp(n^{1/2+o(1)})$.
Also, many speedups for cyclotomics,
and many successful reduction experiments.

Another writeup of the cyclotomic speedups:
2021.10 Bernard–Lesavourey–Nguyen–Roux-Langlois.

Finding cyclotomic S-units

Cyclotomic constructions used in the 2021.08 talk:

- $(1-x^3)/(1-x)$ etc., as in the 2014.10 attack.
- $1+x$.
- Jacobi$\Sigma_p(\chi_1,\chi_2) = \sum_{a\in\mathbb{F}_p^*-\{1\}} \chi_1(a)\chi_2(1-a)$.

  Fragments used implicitly in the 2016.09 attack: exponent vectors of Gauss sums.
- Search small elements of subring $\mathbb{R}\cap R$.
- Square roots: see next slide.

Of course, always apply automorphisms. See also 2022 Bernstein "Fast norm computation in smooth-degree Abelian number fields".

Square-root details

U_p (meaning U_S where $p \in P \Leftrightarrow P \in S$) for $p \in 1 + 2n\mathbb{Z}$ is almost generated by cyclotomic units, Jacobi sums, and $\mathbb{R} \cap U_p$, assuming $h^+ = 1$.

Index is a power of 2, so find the full group U_p by repeatedly adjoining square roots.

Square-root details

U_p (meaning U_S where $p \in P \Leftrightarrow P \in S$) for $p \in 1 + 2n\mathbb{Z}$ is almost generated by cyclotomic units, Jacobi sums, and $\mathbb{R} \cap U_p$, assuming $h^+ = 1$.

Index is a power of 2, so find the full group U_p by repeatedly adjoining square roots.

How to find square products of powers of current generators? Map the group in many ways to $\mathbb{F}_2$: ord_P for all P; random quadratic characters (squareness mod random odd prime ideals). Then fast linear algebra over $\mathbb{F}_2$ finds squares, as in NFS starting with 1991 Adleman.

What about larger S?

For, e.g., $U_{p,q}$: Jacobi sums give p-units and q-units, but they don't produce mixing of p with q.

Index after all of the constructions so far, including square roots, is $h^- \in \exp(n^{1+o(1)})$.

Standard cyclotomic algorithms quickly compute h^-.

What about larger S?

For, e.g., $U_{p,q}$: Jacobi sums give p-units and q-units, but they don't produce mixing of p with q.

Index after all of the constructions so far, including square roots, is $h^- \in \exp(n^{1+o(1)})$.

Standard cyclotomic algorithms quickly compute h^-.

2021 Bernard–Lesavourey–Nguyen–Roux-Langlois: adjoining h^-th roots is "hopeless in general" since h^- has huge prime factors.

What about larger S?

For, e.g., $U_{p,q}$: Jacobi sums give p-units and q-units, but they don't produce mixing of p with q.

Index after all of the constructions so far, including square roots, is $h^- \in \exp(n^{1+o(1)})$.

Standard cyclotomic algorithms quickly compute h^-.

2021 Bernard–Lesavourey–Nguyen–Roux-Langlois: adjoining h^-th roots is "hopeless in general" since h^- has huge prime factors.

Does this mean we have to fill in the remaining S-units by searching small elements of R?

Adjoining huge roots

Have group generated by $g_1, g_2, \ldots$. Goal: reduce index by huge prime r, by adjoining an rth root g'.

Adjoining huge roots

Have group generated by $g_1, g_2, \ldots$. Goal: reduce index by huge prime r, by adjoining an rth root g'.

Map the group in many ways to $\mathbb{Z}/r$:
ord_P; random order-r characters (followed by discrete logs). Fast linear algebra over $\mathbb{Z}/r$ then finds exponent vector $(e_1, e_2, \ldots)$ where $g_1^{e_1} g_2^{e_2} \cdots$ is nontrivially an rth power.

Adjoining huge roots, continued

Expect $e_1, e_2, \ldots$ to have same scale as r, so can't afford to write down the coefficients of $g_1^{e_1} g_2^{e_2} \cdots$.

Adjoining huge roots, continued

Expect $e_1, e_2, \ldots$ to have same scale as r, so can't afford to write down the coefficients of $g_1^{e_1} g_2^{e_2} \cdots$.

But the rth root g' is much smaller. Recover g' from $\operatorname{Log} g' = (e_1/r) \operatorname{Log} g_1 + (e_2/r) \operatorname{Log} g_2 + \cdots$.

Adjoining huge roots, continued

Expect $e_1, e_2, \ldots$ to have same scale as r, so can't afford to write down the coefficients of $g_1^{e_1} g_2^{e_2} \cdots$.

But the rth root g' is much smaller. Recover g' from $\operatorname{Log} g' = (e_1/r) \operatorname{Log} g_1 + (e_2/r) \operatorname{Log} g_2 + \cdots$.

Summary: thanks to cyclotomic structure, computing $U_{p,q}$ boils down to computing $\mathbb{R} \cap U_{p,q}$, which is a problem of degree only $n/2$.

Unsurprising for number theorists

Very long history of number theorists studying general number fields. Often proofs and speedups exploit subfields, automorphisms, etc. Example:

- *Cyclotomics* have short textbook generators of a finite-index subgroup of the unit group.
- For *general* number fields, best methods known are subexp, not poly. (Even with a quantum computer, how do we find *short* generators?)

Given previous literature on faster computations exploiting special fields, why would we be surprised by faster lattice attacks exploiting special fields?

The cyclotomic debate in crypto

2014.02 Bernstein: $R = \mathbb{Z}[x]/(x^p - x - 1)$ is a safer ring choice for lattice-based crypto. Prime degree, so no intermediate subfields; very large Galois group, so very far from automorphisms.

The cyclotomic debate in crypto

2014.02 Bernstein: $R = \mathbb{Z}[x]/(x^p - x - 1)$ is a safer ring choice for lattice-based crypto. Prime degree, so no intermediate subfields; very large Galois group, so very far from automorphisms.

Compare 2017 Kyber: "$\mathbb{Z}[X]/(X^n + 1)$ is one of the most widely studied, and best understood, rings (along with other cyclotomic rings) in algebraic number theory. The fact that no attacks have been found against its use for cryptosystems like Kyber makes it a much more conservative choice than some ring that is harder to analyze and may show weaknesses only after many more years of study."

Last topic: A trap for the unwary

Common practice in lattice-based cryptography: if a lattice appears, analyze it as a random lattice with the same dimension and determinant. Deceptive name: "Gaussian heuristic".

Heuristic 1: shortest nonzero length is always $(1 + o(1))(n/2\pi e)^{1/2}(\det L)^{1/n}$.

Last topic: A trap for the unwary

Common practice in lattice-based cryptography: if a lattice appears, analyze it as a random lattice with the same dimension and determinant. Deceptive name: "Gaussian heuristic".

Heuristic 1: shortest nonzero length is always $(1 + o(1))(n/2\pi e)^{1/2}(\det L)^{1/n}$.

Heuristic 2: high-probability simple reduction needs $\#D$ at least exponential in n.

Last topic: A trap for the unwary

Common practice in lattice-based cryptography: if a lattice appears, analyze it as a random lattice with the same dimension and determinant. Deceptive name: "Gaussian heuristic".

Heuristic 1: shortest nonzero length is always $(1 + o(1))(n/2\pi e)^{1/2}(\det L)^{1/n}$.

Heuristic 2: high-probability simple reduction needs $\#D$ at least exponential in n.

Why this is a trap: The heuristics are wrong for, e.g., $\mathbb{Z}^n$ and other orthogonal lattices.

How to not understand S-unit lattices

The analysis from 2019 Pellet-Mary–Hanrot–Stehlé applied these heuristics to S-unit lattices, concluding that S-unit attacks need time $\exp(n^{1+o(1)})$ for poly approx factor; "there is a pre-processing phase of exponential time" to reach factors below $\exp(n^{1/2+o(1)})$; optimal $\#S$ is $n^{1+o(1)}$; optimal $\#D$ is $\exp(n^{1+o(1)})$.

2021.08.24 Ducas–Pellet-Mary, same reasoning: for $\#S, \#D \in \exp(n^{1/2+o(1)})$, success probability of S-unit reduction "would be **ridiculously** small".

Non-randomness of S-unit lattices

2021.10 Bernstein–Lange: those heuristics say the probability converges to 0 for $\#S \to \infty$ if $\#D \in \#S^{1+o(1)}$; but it actually converges to 1.

Paper gives 5 different ways to see that S-unit lattices are much closer to orthogonal than to random; 1 more way where this is conjectured.

How this was summarized in the 2021.08.20 talk: the S-unit lattice is an "amazingly special lattice", in particular regarding its "analytic features".

Further analysis, experiments, more speedups: see `https://s-unit.attacks.cr.yp.to`.