Algorithms for attacking lattices

Daniel J. Bernstein

2017 Dilithium

"In this paper, we present a new digital signature scheme Dilithium, whose security is based on the hardness of finding short vectors in lattices."

"It can be shown that in the (classical) random oracle model, Dilithium is SUF-CMA secure based on the hardness of the standard MLWE and MSIS lattice problems."

"Since we are aiming for long-term security, we have analyzed the applicability of lattice attacks from a very favorable, to the attacker, viewpoint."

2022 NIST

"Enumeration algorithms . . . have run times that are super-exponential . . . Sieving algorithms . . . require an exponential amount of memory. . . . The performance of sieving algorithms has been improving [306–314], however recent results [315] indicate that improvements in locally sensitive hash techniques, which have resulted in the largest decreases in asymptotic complexity for sieving thus far, cannot be improved further. . . . understanding of the concrete security of lattice-based cryptosystems has greatly improved over the past several years"

2024 HAETAE (version 2.1)

"We introduce HAETAE, a new post-quantum digital signature scheme, whose security is based on the hardness of the module versions of the lattice problems LWE and SIS."

"Our scheme relies on the difficulty of hard lattice problems, which have been well-studied for a long time."

"For setting parameters, we estimated the costs of practical attacks, as in Dilithium, Falcon, and many other NIST-submitted schemes."

Conclusion: Lattices are secure.

Conclusion: Lattices are secure.

End of talk.

Conclusion: Lattices are secure.

End of talk?

Why do people claim SVP is strong?

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 people claim SVP is strong?

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."

Reality: Lagrange and Gauss encountered 2-dimensional lattices in number theory and applied a simple, fast SVP algorithm.

Why do people claim SVP is strong?

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."

Reality: Lagrange and Gauss encountered 2-dimensional lattices in number theory and applied a simple, fast SVP algorithm.

Basically Euclid's algorithm: Replace lattice basis u, v with shorter $u \pm v, v$ or shorter $u, v \pm u$.

Mathematicians proving existence

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

Mathematicians proving existence

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

Easy improvement, using 1896 Minkowski convex-body theorem: length at most $(2/\text{vol}\, B_n^{1/n})(\det L)^{1/n}$ where B_n is the n-dimensional unit ball. Have $2/\text{vol}\, B_n^{1/n} \in (2 + o(1))\sqrt{n/2\pi e}$ as $n \to \infty$.

Mathematicians proving existence

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

Easy improvement, using 1896 Minkowski convex-body theorem: length at most $(2/\operatorname{vol} B_n^{1/n})(\det L)^{1/n}$ where B_n is the n-dimensional unit ball. Have $2/\operatorname{vol} B_n^{1/n} \in (2+o(1))\sqrt{n/2\pi e}$ as $n \to \infty$.

Lattices show up in many math papers. Most of those papers do *not* study speed.

Sufficiently fast lattice computations

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

Sufficiently fast 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 $(2\beta)^{n/\beta}(\det L)^{1/n}$ for $2 \le \beta \le n$, 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 $(2\beta)^{n/\beta}(\det L)^{1/n}$ for $2 \le \beta \le n$, 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 $(1 + o(1))^n(\det L)^{1/n}$.

BKZ

One "tour" of BKZ-(β, n):

- Start with basis $b_1, b_2, \ldots, b_n$.
- Replace b_1 with short combination of $b_1, b_2, \ldots, b_\beta$. Can tweak to still have basis.

BKZ

One "tour" of BKZ-(β, n):

- Start with basis $b_1, b_2, \ldots, b_n$.
- Replace b_1 with short combination of $b_1, b_2, \ldots, b_\beta$. Can tweak to still have basis.
- Replace b_2 with short combination of $b_2, \ldots, b_{\beta+1}$ (projected orthogonally to b_1).

BKZ

One "tour" of BKZ-(β, n):

- Start with basis $b_1, b_2, \ldots, b_n$.
- Replace b_1 with short combination of $b_1, b_2, \ldots, b_\beta$. Can tweak to still have basis.
- Replace b_2 with short combination of $b_2, \ldots, b_{\beta+1}$ (projected orthogonally to b_1).
- Replace b_3 with short combination of $b_3, \ldots, b_{\beta+2}$ (projected orthogonally to b_1, b_2).

BKZ

One "tour" of BKZ-(β, n):

- Start with basis $b_1, b_2, \ldots, b_n$.
- Replace b_1 with short combination of $b_1, b_2, \ldots, b_\beta$. Can tweak to still have basis.
- Replace b_2 with short combination of $b_2, \ldots, b_{\beta+1}$ (projected orthogonally to b_1).
- Replace b_3 with short combination of $b_3, \ldots, b_{\beta+2}$ (projected orthogonally to b_1, b_2).
- Replace b_4, then b_5, ..., then $b_{n-\beta+1}$.

BKZ

One "tour" of BKZ-(β, n):

- Start with basis $b_1, b_2, \ldots, b_n$.
- Replace b_1 with short combination of $b_1, b_2, \ldots, b_\beta$. Can tweak to still have basis.
- Replace b_2 with short combination of $b_2, \ldots, b_{\beta+1}$ (projected orthogonally to b_1).
- Replace b_3 with short combination of $b_3, \ldots, b_{\beta+2}$ (projected orthogonally to b_1, b_2).
- Replace b_4, then b_5, ..., then $b_{n-\beta+1}$.

Continue through some number of tours.

Enumeration

Given basis $b_1, b_2, \ldots, b_n$,
search all small $(c_1, c_2, \ldots, c_n) \in \mathbb{Z}^n$
to find shortest nonzero $c_1 b_1 + c_2 b_2 + \cdots + c_n b_n$.

Enumeration

Given basis $b_1, b_2, \ldots, b_n$,
search all small $(c_1, c_2, \ldots, c_n) \in \mathbb{Z}^n$
to find shortest nonzero $c_1 b_1 + c_2 b_2 + \cdots + c_n b_n$.

Can show: If $c_1 b_1 + c_2 b_2 + \cdots + c_n b_n$
is shorter than shortest vector found so far
then $|c_j| \le H_j$ for an efficiently computable H_j.

Enumeration

Given basis $b_1, b_2, \ldots, b_n$,
search all small $(c_1, c_2, \ldots, c_n) \in \mathbb{Z}^n$
to find shortest nonzero $c_1 b_1 + c_2 b_2 + \cdots + c_n b_n$.

Can show: If $c_1 b_1 + c_2 b_2 + \cdots + c_n b_n$
is shorter than shortest vector found so far
then $|c_j| \leq H_j$ for an efficiently computable H_j.

Preliminary reduction of $b_1, b_2, \ldots, b_n$
makes H_j smaller, speeding up enumeration.

Enumeration

Given basis $b_1, b_2, \ldots, b_n$,
search all small $(c_1, c_2, \ldots, c_n) \in \mathbb{Z}^n$
to find shortest nonzero $c_1 b_1 + c_2 b_2 + \cdots + c_n b_n$.

Can show: If $c_1 b_1 + c_2 b_2 + \cdots + c_n b_n$
is shorter than shortest vector found so far
then $|c_j| \le H_j$ for an efficiently computable H_j.

Preliminary reduction of $b_1, b_2, \ldots, b_n$
makes H_j smaller, speeding up enumeration.

"Recursive preprocessing": BKZ-(β, n) calls
Enum-β, which calls BKZ-(β', β) with $\beta' < \beta$.

Important trends

Already visible in these papers, continuing today:

- Lattice algorithms becoming more complicated.
- *Analyses* becoming *much* more complicated.

Important trends

Already visible in these papers, continuing today:

- Lattice algorithms becoming more complicated.
- *Analyses* becoming *much* more complicated.

Complicated attack analyses are a security risk:

- Errors/uncertainties regarding algorithm cost.
- Difficulties optimizing parameters.
- Difficulties managing the search for speedups.

Important trends

Already visible in these papers, continuing today:

- Lattice algorithms becoming more complicated.
- *Analyses* becoming *much* more complicated.

Complicated attack analyses are a security risk:

- Errors/uncertainties regarding algorithm cost.
- Difficulties optimizing parameters.
- Difficulties managing the search for speedups.

e.g. "pruned enumeration": What happens if we require $|c_j| \leq (1/2)H_j$? No *guarantee* of success, but what's the *chance* that it works if we randomize $b_1, b_2, \ldots, b_n$? What if we modify the $1/2$?

Sieving

2001 Ajtai–Kumar–Sivakumar:
"sieving" solves SVP-n in time $2^{O(n)}$.

Basic sieving idea:

- Start with LLL-reduced basis $b_1, \ldots, b_n$.
- Write down random combinations $c_1 b_1 + \cdots + c_n b_n$.

Sieving

2001 Ajtai–Kumar–Sivakumar:
"sieving" solves SVP-n in time $2^{O(n)}$.

Basic sieving idea:

- Start with LLL-reduced basis $b_1, \ldots, b_n$.
- Write down random combinations $c_1 b_1 + \cdots + c_n b_n$.
- Given $2^{\Theta(n)}$ combinations, find $2^{\Theta(n)}$ close pairs u, v, meaning $|u - v| < 0.5 \min\{|u|, |v|\}$.

Sieving

2001 Ajtai–Kumar–Sivakumar:
"sieving" solves SVP-n in time $2^{O(n)}$.

Basic sieving idea:

- Start with LLL-reduced basis $b_1, \ldots, b_n$.
- Write down random combinations $c_1 b_1 + \cdots + c_n b_n$.
- Given $2^{\Theta(n)}$ combinations, find $2^{\Theta(n)}$ close pairs u, v, meaning $|u - v| < 0.5 \min\{|u|, |v|\}$.
- Find $2^{\Theta(n)}$ close differences of those pairs.

Sieving

2001 Ajtai–Kumar–Sivakumar:
"sieving" solves SVP-n in time $2^{O(n)}$.

Basic sieving idea:

- Start with LLL-reduced basis $b_1, \ldots, b_n$.
- Write down random combinations $c_1 b_1 + \cdots + c_n b_n$.
- Given $2^{\Theta(n)}$ combinations, find $2^{\Theta(n)}$ close pairs u, v, meaning $|u - v| < 0.5 \min\{|u|, |v|\}$.
- Find $2^{\Theta(n)}$ close differences of those pairs.
- Find $2^{\Theta(n)}$ close differences of differences.

Sieving

2001 Ajtai–Kumar–Sivakumar:
"sieving" solves SVP-n in time $2^{O(n)}$.

Basic sieving idea:

- Start with LLL-reduced basis $b_1, \ldots, b_n$.
- Write down random combinations $c_1 b_1 + \cdots + c_n b_n$.
- Given $2^{\Theta(n)}$ combinations, find $2^{\Theta(n)}$ close pairs u, v, meaning $|u - v| < 0.5 \min\{|u|, |v|\}$.
- Find $2^{\Theta(n)}$ close differences of those pairs.
- Find $2^{\Theta(n)}$ close differences of differences.
- etc.

What is the SVP exponent?

Approximate α for some algorithms believed to take time $2^{(\alpha+o(1))n}$ (without quantum computation):

- 0.415: 2008 Nguyen–Vidick.
- 0.415: 2010 Micciancio–Voulgaris.

What is the SVP exponent?

Approximate α for some algorithms believed to take time $2^{(\alpha+o(1))n}$ (without quantum computation):

- 0.415: 2008 Nguyen–Vidick.
- 0.415: 2010 Micciancio–Voulgaris.
- 0.384: 2011 Wang–Liu–Tian–Bi.

What is the SVP exponent?

Approximate α for some algorithms believed to take time $2^{(\alpha+o(1))n}$ (without quantum computation):

- 0.415: 2008 Nguyen–Vidick.
- 0.415: 2010 Micciancio–Voulgaris.
- 0.384: 2011 Wang–Liu–Tian–Bi.
- 0.378: 2013 Zhang–Pan–Hu.

What is the SVP exponent?

Approximate α for some algorithms believed to take time $2^{(\alpha+o(1))n}$ (without quantum computation):

- 0.415: 2008 Nguyen–Vidick.
- 0.415: 2010 Micciancio–Voulgaris.
- 0.384: 2011 Wang–Liu–Tian–Bi.
- 0.378: 2013 Zhang–Pan–Hu.
- 0.337: 2014 Laarhoven.

What is the SVP exponent?

Approximate α for some algorithms believed to take time $2^{(\alpha+o(1))n}$ (without quantum computation):

- 0.415: 2008 Nguyen–Vidick.
- 0.415: 2010 Micciancio–Voulgaris.
- 0.384: 2011 Wang–Liu–Tian–Bi.
- 0.378: 2013 Zhang–Pan–Hu.
- 0.337: 2014 Laarhoven.
- 0.298: 2015 Laarhoven–de Weger.
- 0.292: 2015 Becker–Ducas–Gama–Laarhoven.

What is the SVP exponent?

Approximate α for some algorithms believed to take time $2^{(\alpha+o(1))n}$ (without quantum computation):

- 0.415: 2008 Nguyen–Vidick.
- 0.415: 2010 Micciancio–Voulgaris.
- 0.384: 2011 Wang–Liu–Tian–Bi.
- 0.378: 2013 Zhang–Pan–Hu.
- 0.337: 2014 Laarhoven.
- 0.298: 2015 Laarhoven–de Weger.
- 0.292: 2015 Becker–Ducas–Gama–Laarhoven.

"Locality-sensitive hashing" of lattice vectors v gives subquadratic search for v close to u.

Many more SVP improvements

e.g. 2017 Ducas "dimensions for free":
Solve SVP-n using Sieving-$(n - d)$.

Many more SVP improvements

e.g. 2017 Ducas "dimensions for free":
Solve SVP-n using Sieving-$(n - d)$.

Basic idea:

- Project lattice orthogonally to $b_1, b_2, \ldots, b_d$.

Many more SVP improvements

e.g. 2017 Ducas "dimensions for free":
Solve SVP-n using Sieving-$(n - d)$.

Basic idea:

- Project lattice orthogonally to $b_1, b_2, \ldots, b_d$.
- Sieve. Gives $2^{\Theta(n-d)}$ short vectors in projected lattice, not just shortest nonzero vector.

Many more SVP improvements

e.g. 2017 Ducas "dimensions for free":
Solve SVP-n using Sieving-$(n - d)$.

Basic idea:

- Project lattice orthogonally to $b_1, b_2, \ldots, b_d$.
- Sieve. Gives $2^{\Theta(n-d)}$ short vectors in projected lattice, not just shortest nonzero vector.
- Hope that shortest nonzero vector in original lattice projects to one of those vectors.

Many more SVP improvements

e.g. 2017 Ducas "dimensions for free":
Solve SVP-n using Sieving-$(n - d)$.

Basic idea:

- Project lattice orthogonally to $b_1, b_2, \ldots, b_d$.
- Sieve. Gives $2^{\Theta(n-d)}$ short vectors in projected lattice, not just shortest nonzero vector.
- Hope that shortest nonzero vector in original lattice projects to one of those vectors.

High success probability with $d \in \Theta(n/\log n)$.
Maybe better to increase d, try repeatedly.

Interlude: memory-access costs

Cost metrics for algorithms

Algorithm designers typically count “operations”.

Cost metrics for algorithms

Algorithm designers typically count “operations”.

Real-world complication: the cost of a memory access `x[i]` increases rapidly with the size of `x`, even though `x[i]` is just one “operation”.

Cost metrics for algorithms

Algorithm designers typically count "operations".

Real-world complication: the cost of a memory access `x[i]` increases rapidly with the size of `x`, even though `x[i]` is just one "operation".

Sometimes algorithm designers consider this cost, often ending up with very different algorithms.

Cost metrics for algorithms

Algorithm designers typically count "operations".

Real-world complication: the cost of a memory access `x[i]` increases rapidly with the size of `x`, even though `x[i]` is just one "operation".

Sometimes algorithm designers consider this cost, often ending up with very different algorithms.

Some examples of how this complication changes cost exponents: NFS, collisions, batch NFS.

Simplifying attack analyses

For the first six years of the NIST competition, NIST consistently asked submissions to reach the security level of AES-128 as measured by "classical gates": bit operations, *not* memory-access costs.

NIST discouraged research into memory-access costs. Highlighted features of "classical gates":
(1) "accurately measured" for known attacks;
(2) does not "overestimate" real-world costs.

See, e.g., 2016 "gates"; 2019 report regarding NTRU Prime; 2020 "criteria"; 2020 report regarding NTRU; 2022.07 exclusion of NTRU-509.

A sudden complication

2022.11: NIST suddenly switched to counting costs of memory access. On this basis, announced plans to standardize Kyber-512.

A sudden complication

2022.11: NIST suddenly switched to counting costs of memory access. On this basis, announced plans to standardize Kyber-512.

2023.10: I pointed out serious mistakes in how NIST was tallying memory-access costs in known attacks.

The simplest issue: NIST's calculation "40 bits of security more than would be suggested by the RAM model" was incorrectly multiplying the following:

- a 2^{40} estimate of cost per memory access;
- an estimate for the number of *bit operations*, rather than the number of *memory accesses*.

The collapse of memory-access costs

Subsequent claims regarding the cost exponent of SVP-β including memory-access costs:

Month	Source	Claimed exponent
2023.11	Schanck	0.396β
2023.11	Schanck	
2023.12	NIST	
2024.01	Jaques	

The collapse of memory-access costs

Subsequent claims regarding the cost exponent of SVP-β including memory-access costs:

Month	Source	Claimed exponent
2023.11	Schanck	0.396β
2023.11	Schanck	0.349β
2023.12	NIST	
2024.01	Jaques	

The collapse of memory-access costs

Subsequent claims regarding the cost exponent of SVP-β including memory-access costs:

Month	Source	Claimed exponent
2023.11	Schanck	0.396β
2023.11	Schanck	0.349β
2023.12	NIST	0.349β, or 0.329β in 3 dimensions
2024.01	Jaques	

The collapse of memory-access costs

Subsequent claims regarding the cost exponent of SVP-β including memory-access costs:

Month	Source	Claimed exponent
2023.11	Schanck	0.396β
2023.11	Schanck	0.349β
2023.12	NIST	0.349β, or 0.329β in 3 dimensions
2024.01	Jaques	0.311β, or 0.292β in 3 dimensions

The collapse of memory-access costs

Subsequent claims regarding the cost exponent of SVP-β including memory-access costs:

Month	Source	Claimed exponent
2023.11	Schanck	0.396β
2023.11	Schanck	0.349β
2023.12	NIST	0.349β, or 0.329β in 3 dimensions
2024.01	Jaques	0.311β, or 0.292β in 3 dimensions

This is “well studied”?

The collapse of memory-access costs

Subsequent claims regarding the cost exponent of SVP-β including memory-access costs:

Month	Source	Claimed exponent
2023.11	Schanck	0.396β
2023.11	Schanck	0.349β
2023.12	NIST	0.349β, or 0.329β in 3 dimensions
2024.01	Jaques	0.311β, or 0.292β in 3 dimensions

This is “well studied”?

Maybe subexponential factors save the day, but the study of those is in its infancy.

Overconfidence

2023.11 Schanck comment about the 0.396: "This matters because the best exponent was 0.415 a decade ago, and the 0.349 and 0.292 estimates play into a false narrative that the cost of lattice attacks has been falling precipitously since then."

Overconfidence

2023.11 Schanck comment about the 0.396: “This matters because the best exponent was 0.415 a decade ago, and the 0.349 and 0.292 estimates play into a false narrative that the cost of lattice attacks has been falling precipitously since then.”

2024.03 Schanck “update on lattice cryptanalysis” admits the “narrative” is correct?

Overconfidence

2023.11 Schanck comment about the 0.396: "This matters because the best exponent was 0.415 a decade ago, and the 0.349 and 0.292 estimates play into a false narrative that the cost of lattice attacks has been falling precipitously since then."

2024.03 Schanck "update on lattice cryptanalysis" admits the "narrative" is correct? Disputes the speedups?

Overconfidence

2023.11 Schanck comment about the 0.396: “This matters because the best exponent was 0.415 a decade ago, and the 0.349 and 0.292 estimates play into a false narrative that the cost of lattice attacks has been falling precipitously since then.”

2024.03 Schanck “update on lattice cryptanalysis” admits the “narrative” is correct? Disputes the speedups? No: switches to talking about Kyber-768.

Overconfidence

2023.11 Schanck comment about the 0.396: "This matters because the best exponent was 0.415 a decade ago, and the 0.349 and 0.292 estimates play into a false narrative that the cost of lattice attacks has been falling precipitously since then."

2024.03 Schanck "update on lattice cryptanalysis" admits the "narrative" is correct? Disputes the speedups? No: switches to talking about Kyber-768.

2024.04: Without commenting on the collapse, NIST states that it will standardize Kyber-512.

What went wrong here?

A simple pre-quantum analogy:

- Ask people to optimize discrete logs, ignoring memory-access costs: baby-step-giant-step discrete-log algorithm.

What went wrong here?

A simple pre-quantum analogy:

- Ask people to optimize discrete logs, ignoring memory-access costs: baby-step-giant-step discrete-log algorithm.
- Suddenly start counting memory-access costs: much higher exponent for baby-step-giant-step.

What went wrong here?

A simple pre-quantum analogy:

- Ask people to optimize discrete logs, ignoring memory-access costs: baby-step-giant-step discrete-log algorithm.
- Suddenly start counting memory-access costs: much higher exponent for baby-step-giant-step.
- But then people find algorithms eliminating those costs: e.g., Pollard's rho method, or, for parallelization, van Oorschot–Wiener.

If we exclude parameter sets
that mention memory-access costs,
then lattices are safe?

Many attack avenues

Further advances against SVP will be unsurprising.

e.g. 2020 Albrecht–Bai–Fouque–Kirchner–Stehlé–Wen and 2020 Albrecht–Bai–Li–Rowell achieved better enumeration exponents; what's the impact on tuple lattice sieving (combining sieving and enumeration)?

Many attack avenues

Further advances against SVP will be unsurprising.

e.g. 2020 Albrecht–Bai–Fouque–Kirchner–Stehlé–Wen and 2020 Albrecht–Bai–Li–Rowell achieved better enumeration exponents; what's the impact on tuple lattice sieving (combining sieving and enumeration)?

But the rest of this talk will instead consider avenues for lattice attacks *beyond* SVP attacks.

Wait, what about the proofs?

Wait, what about the proofs?

The proofs don't say systems are as hard to break as SVP.

Wait, what about the proofs?

The proofs don't say systems are as hard to break as SVP. Let's look at what a proof actually says.

Wait, what about the proofs?

The proofs don't say systems are as hard to break as SVP. Let's look at what a proof actually says.

Theorem 4 from 2023 Barbosa–Barthe–Doczkal–Don–Fehr–Grégoire–Huang–Hülsing–Lee–Wu "Fixing and mechanizing the security proof of Fiat-Shamir with aborts and Dilithium" says that the "EF-CMA" advantage of a Dilithium attack $\mathcal{F}$ is at most $P_1 + P_2 + P_3 + P_4 + P_5$.

(Formula in paper is missing the second "$+$"; fixed in Springer version.)

MLWE

Define $R = \mathbb{Z}[x]/(x^n + 1)$.

First term P_1 is advantage of a specific algorithm derived from $\mathcal{F}$ in breaking the following "MLWE" problem: distinguish $As + e \in (R/q)^k$ from uniform random, given uniform random $A \in (R/q)^{k \times \ell}$, when entries of $s \in R^\ell$ and $e \in R^k$ are chosen from the uniform distribution on $\{-\eta, \ldots, -1, 0, 1, \ldots, \eta\}$.

MLWE

Define $R = \mathbb{Z}[x]/(x^n + 1)$.

First term P_1 is advantage of a specific algorithm derived from $\mathcal{F}$ in breaking the following "MLWE" problem: distinguish $As + e \in (R/q)^k$ from uniform random, given uniform random $A \in (R/q)^{k\times\ell}$, when entries of $s \in R^\ell$ and $e \in R^k$ are chosen from the uniform distribution on $\{-\eta, \ldots, -1, 0, 1, \ldots, \eta\}$.

$q = 2^{23} - 2^{13} + 1$ is the Dilithium modulus.
$n = 256$ is Dilithium's base dimension.
(k, ℓ) is, e.g., $(4, 4)$ for Dilithium-2.
η is, e.g., 2 for Dilithium-2.

MLWE is not SVP, part 1

Distinguishing $As + e$ from uniform random can be easier than finding s, e.

MLWE is not SVP, part 1

Distinguishing $As + e$ from uniform random can be easier than finding s, e.

Example of gap in the literature: reported costs of "dual attacks" are normally for finding s, e, but the same attacks are faster when used as distinguishers.

MLWE is not SVP, part 1

Distinguishing $As + e$ from uniform random can be easier than finding s, e.

Example of gap in the literature: reported costs of "dual attacks" are normally for finding s, e, but the same attacks are faster when used as distinguishers.

Does a distinguisher break Dilithium? Maybe, maybe not. It makes the theorem vacuous.

MLWE is not SVP, part 2

Finding s, e is equivalent to
finding a vector in L close to $(0, As + e)$ where
$L = \{(u, v) \in R^\ell \times R^k : v = Au \text{ in } (R/q)^k\}$.

MLWE is not SVP, part 2

Finding s, e is equivalent to finding a vector in L close to $(0, As + e)$ where $L = \{(u, v) \in R^\ell \times R^k : v = Au \text{ in } (R/q)^k\}$.

Can attack that by finding a short nonzero vector in another lattice L' built from L, s, e: an artificial "gap" lattice with an unusually short nonzero vector.

MLWE is not SVP, part 2

Finding s, e is equivalent to finding a vector in L close to $(0, As + e)$ where $L = \{(u, v) \in R^\ell \times R^k : v = Au \text{ in } (R/q)^k\}$.

Can attack that by finding a short nonzero vector in another lattice L' built from L, s, e: an artificial "gap" lattice with an unusually short nonzero vector.

Typically use BKZ-(β, n) to reduce to SVP-β, with β chosen so that the "gap" is visible in dimension β. A closer look shows that people are continuing to find new algorithms and optimizations here: see, e.g., 2024.01 Xia–Wang–Wang–Gu–Wang.

MLWE is not SVP, part 3

Each s, e coefficient is small and can be guessed.

2003 Schnorr, 2007 Howgrave-Graham, etc.:
can productively mix guessing techniques with
lattice techniques to form “hybrid” attacks.

MLWE is not SVP, part 3

Each s, e coefficient is small and can be guessed.

2003 Schnorr, 2007 Howgrave-Graham, etc.: can productively mix guessing techniques with lattice techniques to form "hybrid" attacks.

For $q \in n^{Q_0+o(1)}$: existing heuristics imply that non-hybrid "primal" attacks cost $2^{(\rho+o(1))n}$ where $z_0 = 2Q_0/(Q_0+1/2)^2$ and $\rho = z_0 \log_4(3/2)$.

2023.12 Bernstein: same heuristics imply that *simple* hybrid primal attacks cost $2^{(\rho-\rho H_0+o(1))n}$ where $H_0 = 1/(1+(\log_2(2\eta+1))/0.057981 z_0)$.

Useful subroutines for hybrid attacks

2016 Laarhoven, 2019 Doulgerakis–Laarhoven–de Weger, 2020 Ducas–Laarhoven–van Woerden: can find an element of L closest to t with time exponent $\approx$0.234, after an L-dependent t-independent precomputation with time exponent $\approx$0.292.

Useful subroutines for hybrid attacks

2016 Laarhoven, 2019 Doulgerakis–Laarhoven–de Weger, 2020 Ducas–Laarhoven–van Woerden: can find an element of L closest to t with time exponent $\approx$0.234, after an L-dependent t-independent precomputation with time exponent $\approx$0.292.

2020 Espitau–Kirchner analysis of Howgrave-Graham "nearest-colattice" algorithm: find an element of L *close* to t using a BKZ-(β, n) computation and a β-dimensional closest-vector computation. Closeness $\approx$ BKZ-(β, n) shortness. BKZ and CVP use t-independent lattices.

MLWE is not SVP, part 4

The lattices here have a special algebraic structure: they're R-modules where R is the cyclotomic ring $\mathbb{Z}[x]/(x^n + 1)$ with n a power of 2.

MLWE is not SVP, part 4

The lattices here have a special algebraic structure: they're R-modules where R is the cyclotomic ring $\mathbb{Z}[x]/(x^n + 1)$ with n a power of 2.

Example of why this is a concern: for ideals of R, "S-unit attacks" achieve approximation factor $2^{n^{1/2+o(1)}}$ in quantum poly time (assuming "$h^+ = 1$"). This line of work keeps breaking claimed "barriers".

MLWE is not SVP, part 4

The lattices here have a special algebraic structure: they're R-modules where R is the cyclotomic ring $\mathbb{Z}[x]/(x^n + 1)$ with n a power of 2.

Example of why this is a concern: for ideals of R, "S-unit attacks" achieve approximation factor $2^{n^{1/2+o(1)}}$ in quantum poly time (assuming "$h^+ = 1$"). This line of work keeps breaking claimed "barriers".

Conjecturally poly approx factor in subexponential time. Could the ideas handle more general modules?

SelfTargetMSIS attacks

Second term P_2 in Theorem 4 is advantage of a specific algorithm derived from $\mathcal{F}$ in breaking the following "SelfTargetMSIS" problem:
given uniform random (A, t)
with $A \in (R/q)^{k \times \ell}$ and $t \in (R/q)^k$,
find μ, z, c, v with $G(\mu, Az + v - ct) = c$ and all entries of z, c, v at most $\max\{2(\gamma_1 - \beta), 4\gamma_2 + 2\}$.

SelfTargetMSIS attacks

Second term P_2 in Theorem 4 is advantage of a specific algorithm derived from $\mathcal{F}$ in breaking the following "SelfTargetMSIS" problem:
given uniform random (A, t)
with $A \in (R/q)^{k\times\ell}$ and $t \in (R/q)^k$,
find μ, z, c, v with $G(\mu, Az + v - ct) = c$ and all entries of z, c, v at most $\max\{2(\gamma_1 - \beta), 4\gamma_2 + 2\}$.

The quantities $\gamma_1 - \beta, \gamma_2$ appear in signature verification. G is the Dilithium hash function producing vectors with small entries. (This is a normal hash function followed by "SampleInBall".)

Is this proof content-free?

SelfTargetMSIS *feels* like it's simply restating the problem of forging Dilithium signatures.

Dilithium verification forces $G(\mu, Az + v - ct) = c$, and forces z, c, v to have small entries.

Is this proof content-free?

SelfTargetMSIS *feels* like it's simply restating the problem of forging Dilithium signatures.

Dilithium verification forces $G(\mu, Az + v - ct) = c$, and forces z, c, v to have small entries.

One difference: Dilithium has $t = As + e$; SelfTargetMSIS has t chosen uniformly at random. Distinguishing these breaks MLWE.

Interaction ("CMA")

Another difference:
Dilithium attackers can ask for signatures;
the SelfTargetMSIS problem doesn't allow this.

Interaction ("CMA")

Another difference:
Dilithium attackers can ask for signatures;
the SelfTargetMSIS problem doesn't allow this.

The terms P_3, P_4, P_5 in Theorem 4 account for this, in terms of the number of signatures, the number of hash calls, and various quantities "p", "δ", "ϵ".

Interaction ("CMA")

Another difference:
Dilithium attackers can ask for signatures;
the SelfTargetMSIS problem doesn't allow this.

The terms P_3, P_4, P_5 in Theorem 4 account for this, in terms of the number of signatures, the number of hash calls, and various quantities "p", "δ", "ϵ".

Supposedly all of these are small enough.
Has anyone checked the calculations?

Why is SelfTargetMSIS a lattice problem?

Dilithium documentation says: “H is a cryptographic hash function whose structure is completely independent of the algebraic structure of its inputs . . . the only approach for obtaining a solution appears to be picking some w, computing $H'(\mu||\mathbf{w}) = c$, and then finding $\mathbf{z}, \mathbf{u}'$ such that $\mathbf{Az} + \mathbf{u}' = \mathbf{w} + c\mathbf{t}$”.

Why is SelfTargetMSIS a lattice problem?

Dilithium documentation says: "H is a cryptographic hash function whose structure is completely independent of the algebraic structure of its inputs . . . the only approach for obtaining a solution appears to be picking some w, computing $H'(\mu||\mathbf{w}) = c$, and then finding $\mathbf{z}, \mathbf{u}'$ such that $\mathbf{Az} + \mathbf{u}' = \mathbf{w} + c\mathbf{t}$".

i.e. pick μ, w; compute $c = G(\mu, w)$;
find short z, v such that $Az + v = w + ct$.

Multiple targets inside SelfTargetMSIS

2022.11 Wang–Xia–Shi–Wan–Zhang–Gu:
better approaches to attacking SelfTargetMSIS.

Multiple targets inside SelfTargetMSIS

2022.11 Wang–Xia–Shi–Wan–Zhang–Gu:
better approaches to attacking SelfTargetMSIS.

e.g. pick $\mu_1, w_1, \ldots, \mu_B, w_B$.
Compute $c_1 = G(\mu_1, w_1), \ldots, c_B = G(\mu_B, w_B)$.

Multiple targets inside SelfTargetMSIS

2022.11 Wang–Xia–Shi–Wan–Zhang–Gu:
better approaches to attacking SelfTargetMSIS.

e.g. pick $\mu_1, w_1, \ldots, \mu_B, w_B$.
Compute $c_1 = G(\mu_1, w_1), \ldots, c_B = G(\mu_B, w_B)$.

Goal is now to find short z, v such that
$Az + v \in \{w_1 + c_1 t, \ldots, w_B + c_B t\}$.

Multiple targets inside SelfTargetMSIS

2022.11 Wang–Xia–Shi–Wan–Zhang–Gu:
better approaches to attacking SelfTargetMSIS.

e.g. pick $\mu_1, w_1, \ldots, \mu_B, w_B$.
Compute $c_1 = G(\mu_1, w_1), \ldots, c_B = G(\mu_B, w_B)$.

Goal is now to find short z, v such that
$Az + v \in \{w_1 + c_1 t, \ldots, w_B + c_B t\}$.

Use multi-target close-vector algorithms.
Should be able to succeed with smaller β.

Proofs, revisited

"It can be shown that in the (classical) random oracle model, Dilithium is SUF-CMA secure based on the hardness of the standard MLWE and MSIS lattice problems."

Proofs, revisited

"It can be shown that in the (classical) random oracle model, Dilithium is SUF-CMA secure based on the hardness of the standard MLWE and MSIS lattice problems."

This is based on an outline of a way to convert a SelfTargetMSIS attack into a *slower* attack against MSIS. This is not evidence against the idea that SelfTargetMSIS is easier to break than MSIS!

Proofs, revisited

"It can be shown that in the (classical) random oracle model, Dilithium is SUF-CMA secure based on the hardness of the standard MLWE and MSIS lattice problems."

This is based on an outline of a way to convert a SelfTargetMSIS attack into a *slower* attack against MSIS. This is not evidence against the idea that SelfTargetMSIS is easier to break than MSIS!

Showing proofs to *cryptanalysts* is good:
proof gaps can help identify attacks.
Telling *users* about proofs is usually misleading.

NCC-Sign and HAETAE

NCC-Sign uses "SelfTargetRSIS", which is a special case of Dilithium's SelfTargetMSIS, but takes different rings: non-cyclotomic $x^n - x - 1$ with prime n, or cyclotomic $x^n - x^{n/2} + 1$ with $n = 2^a 3^b$. Assumes SelfTargetRSIS is as hard as RSIS.

HAETAE replaces Dilithium's SelfTargetMSIS with "BimodalSelfTargetMSIS", and says "we use the fact that the only known way to solve BimodalSelfTargetMSIS is to solve MSIS".

What about the multi-target attacks from 2022?

Unstable cryptanalytic picture

Some attack avenues that need further study:

- Enumeration.
- Tuple lattice sieving.
- Hybrid attacks.
- Multi-target attacks in SelfTargetMSIS.
- Dual attacks.
- BKZ.
- S-unit attacks.