

Sieving with Streamed Memory Access

Ziyu Zhao, Jintai Ding, Bo-Yin Yang

Sept 2025, NIST

Background

Five PQC algorithms selected for standardization by NIST:

- **CRYSTALS-KYBER** (FIPS203, module LWE)
- **CRYSTALS-DILITHIUM** (FIPS204, module LWE)
- **SPHINCS+** (FIPS205, hash-based)
- **FALCON** (NTRU lattice)
- **HQC** (code-based)

The claim: lattice-based schemes are the most attractive candidates for their compactness, efficiency, and provable security.

Background

Provably secure?

- ▶ **Provable security** - what is it?
- ▶ Strict requirement on the parameters
- ▶ Tightness of the reduction
- ▶ Practical Security – Empirical and Theoretical support?
LLL and Root Hermite Factor?

Background

Most cryptographic hard problems have been stable in the last decade (if not totally broken):

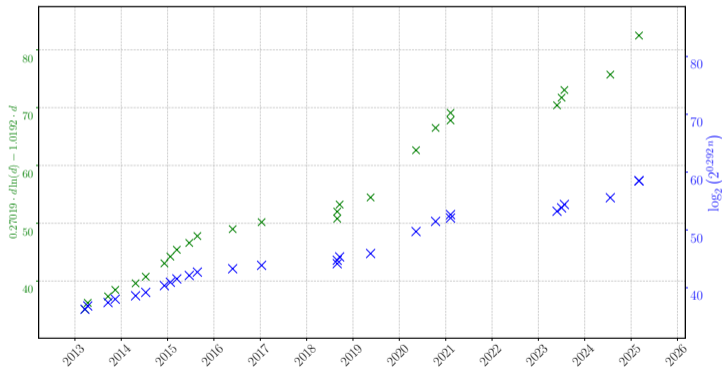
- ▶ **Factoring** - GNFS invented in 1990~1993, with RSA768 factored in 2009 and RSA832 in 2019
- ▶ **Multivariate** - Basic versions of F4/F5 and XL completed in the early 2000s
- ▶ **Decoding** - New record in 2023 with 2008's software¹

Lattice-based cryptography is very much a different story!

¹<https://isd.mceliece.org/1347.html>

Practical Lattice Attacks

Progress on the shortest vector problem:



¹Enumeration complexity uses the fitting in [APS15], ignoring all constants

Practical Lattice Attacks

In [LP10], it was estimated that LWE with parameters $(n = 256, q = 4093, \sigma = 8.35)$ took (at that time) “approximately 2^{120} seconds, which translates on our machine to about 2^{150} operations”.

Fact

BKZ-330 is sufficient for the attack, requiring about 2^{118} gates today.

- ▶ What happened?

Practical Lattice Attacks

Sieving has finally won over enumeration, becoming the benchmark for assessing SVP hardness.

[NV08]	$2^{0.415n+o(n)}$
[BGJ14]	$2^{0.377n+o(n)}$
[HK17]	$2^{0.349n+o(n)}$
[Laa15a]	$2^{0.337n+o(n)}$
[BGJ15]	$2^{0.311n+o(n)}$
[LdW15]	$2^{0.297n+o(n)}$
[BDGL16]	$2^{0.292n+o(n)}$

Impossible to attain an exponent less than $\frac{1}{2} \log_2 \left(\frac{3}{2} \right) \approx 0.292$ by better NNS strategy.²

²Kirshanova, E., Laarhoven, T.: Lower bounds on lattice sieving and information set decoding.

Practical Lattice Attacks

Fact

Improvements in the $o(n)$ term contribute to the main part of last decade's practical progress in SVP.

- ▶ XOR-POPCNT trick (simhash), [Duc18], [FBB⁺15], [Cha02]
- ▶ progressive sieving, [Duc18], [LM18]
- ▶ preprocessing (SubSieve, G6K), [Duc18], [ADH⁺19]
- ▶ Dimensions for Free, [Duc18], [ADH⁺19]

Memory Issue

“our best estimate for the gate cost of attacking Kyber512 using known techniques is about 2^{147} (or 2^{145} ...”

“Combining the above estimates of **the cost of memory access**...the realistic cost of attacking Kyber512 is the equivalent of about 2^{160} bit operations/ gates”³

³<https://csrc.nist.gov/csrc/media/Projects/post-quantum-cryptography/documents/faq/Kyber-512-FAQ.pdf>

Memory Issue

The main obstacle for solving larger SVP instances is the random memory access cost:

- ▶ Sieving database for SVP200 will require about 10 terabytes of memory.
- ▶ The limited host-device bandwidth significantly hampers the BDGL sieve's practical performance.

The issue is becoming more severe as the sieving dimension increases.

Memory Issue

Random access to a (N bits) 3D massive storage array costs $\mathcal{O}(N^{1/3})$.
A list of energy consumption for different operations:⁴

Operation	Energy (pJ)
8b Add	0.03
16b FB Add	0.4
32b FB Add	0.9
8b Mul	0.2
16b FB Mult	1.1
32b FB Mult	3.7
32b SRAM Read (8KB)	5
32b DRAM Read	640

⁴Computer Architecture: A Quantitative Approach, 6th Edition, P29, Energy numbers are from Mark Horowitz
Computing's Energy problem (and what we can do about it). ISSCC 2014

The BGJ Sieve

This work suggests the inherent structure of BGJ sieve is significantly more memory-efficient than BDGL sieve.

- ▶ $2^{0.2075n+o(n)}$ streamed main memory accesses is enough
- ▶ time complexity of a modified BGJ sieve is very close to BDGL sieve, faster for all practical dimensions.
- ▶ supported by implementation. Code available at <https://github.com/zhaoziyu0008/BGJ-Sieve-AMX>.

The BGJ Sieve

Table: Comparison with Previous GPU Records

Dim	Walltime	Platform	FLOP
179	11.2 <i>d</i>	112 cores, no GPU*	$2^{66.0} \approx 2^{13.6} \cdot (3/2)^{179/2}$ int8 op
183	30 <i>d</i>	112 cores, no GPU*	$2^{67.4} \approx 2^{13.9} \cdot (3/2)^{183/2}$ int8 op
180	51.6 <i>d</i>	4 × Nvidia RTX 2080ti	$2^{69.9} \approx 2^{17.3} \cdot (3/2)^{180/2}$ fp16 op [†]
186	50.3 <i>d</i>	4 × Nvidia A100	$2^{71.4} \approx 2^{17.0} \cdot (3/2)^{186/2}$ fp16 op

[†] See Table 1 in [DSvW21] for more details.

Recall

Sieving, a natural generalization of Gauss's algorithm

- ▶ Maintain a list of lattice vectors
- ▶ Find “Reducing-pairs” which generate new short vectors
- ▶ Replace the longest vectors with the short ones

Initialize? Arbitrarily sample $N = 3.0 \cdot (4/3)^{n/2}$ vectors
Stop? When the list saturates the ball of radius $\sqrt{4/3} \cdot \text{gh}(\mathcal{L})$

Recall

The cost of sieving is dominated by the search for reducing-pairs.

Accelerating the Nearest Neighbor Search (NNS) on high-dimensional sphere by locality-sensitive filter:

- ▶ Put the list vectors into buckets
- ▶ Search for reducing pairs in each bucket (naively)
- ▶ Reducing-pairs are more likely to be in the same bucket

Recall

Definition (Spherical cap shaped filters)

A vector \mathbf{v} can pass the filter $\mathcal{F}_{\mathbf{c},\alpha}$ with center \mathbf{c} and radius α if and only if $|\langle \mathbf{v}, \mathbf{c} \rangle| \geq \alpha \|\mathbf{v}\| \|\mathbf{c}\|$.

- BDGL sieve uses single layer of spherical cap shaped filters
- Asymptotically $\alpha \rightarrow 0.5$, $2^{0.292n+o(n)}$ buckets, each of size $2^{o(n)}$
- Bucket center \mathbf{c} comes from random product code for quick bucketing⁵
- The slowdown caused by non-uniformity of \mathbf{c} is subexponential⁶

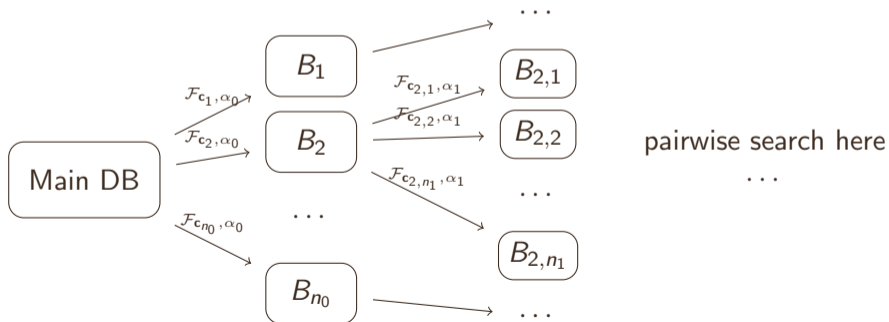
⁵[BDGL16] Anja Becker, Léo Ducas, Nicolas Gama, and Thijs Laarhoven, New directions in nearest neighbor searching with applications to lattice sieving.

⁶[Duc22] Ducas, L.: Estimating the hidden overheads in the bdgl lattice sieving algorithm.

Recall

BGJ15

Generates progressively smaller buckets by applying a series of random filters to the main database.



Memory Access Cost

Observation

The bucket size decreases by several orders of magnitude after each filter, allowing the sub-buckets to be stored in a much smaller, and therefore faster, storage device.

- ▶ No communication between these sub-buckets is necessary.
- ▶ Data movement is streamed, except for the filtering stage.

Memory Access Cost

Supported by implementation, Sieving dimension = 140

Table: Profiling Data of bgj3-amx

Step	Filter-0	Filter-1	Filter-2	Reducing
Speed (TOPS)	11.81	11.10	39.19	116.4
Bucket size	278.8GB	3.386GB	80.75MB	556.7KB
Data in	RAM	RAM	L3-Cache	L2-Cache
Total Time	544.7s	451.4s	762.4s	3397s

Memory Access Cost

BGJ sieve vs BDGL sieve?

- BDGL uses a single filter layer to generate $2^{\mathcal{O}(n)}$ buckets: randomly write to exponentially large space.
- If BGJ uses $\mathcal{O}(\log(n))$ successive filter layers, the number of subbuckets for each bucket can be $2^{\mathcal{O}(n/\log(n))}$.
- Practical value of subbuckets: $\sim 2^7$ for sieving 140.

Memory Access Cost

- ▶ The computations required are superlinear (with an exponent range from $0.292/0.2075 \approx 1.41$ to 2) in the size of the subbucket.
- ▶ Streamed memory movement of N bits costs $\mathcal{O}(N^{4/3})$.
- ▶ As long as the subbucket size is larger than some constant, the memory access overhead, caused by the streamed memory access is negligible.

Memory Access Cost

The last several layers of buckets?

For current computational architectures, setting the bucket size appropriately larger ($100 \sim 1000$) is sufficient.

For serious attacks using ASICs?

More

An implementation of BGJ sieve, with the sieving database on SSD and computation on multi-GPU, was done several months ago.

- Progressive sieve spends about 6 minutes and 8 hours on dimensions 140 and 160
- Set new lattice record — 200 dimensional Darmstadt SVP challenge

Fact

The community can now solve about 1,000,000 × harder SVP instances than ten years ago!

Thank you

Questions?