Sieving with Streamed Memory Access

Ziyu Zhao, Jintai Ding, Bo-yin Yang

Academia Sinica, Dec. 28, 2024

Background

Selected Algorithms 2022:

- **CRYSTALS-KYBER** (module LWE)
- **CRYSTALS-DILITHIUM** (module LWE)
- **FALCON** (NTRU lattice)
- **SPHINCS+** (non-lattice)

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

Background – Provable Security

Provablely secure?

- **Provable security** of the lattice schemes has very strict requirement on the parameters!
- None of the lattice schemes' paramters satisfy the provable secure requirement!
- Then we need to play the games as usual: the pratical security. (But still claim provable secure?)
- So how hard are the lattice attacks? Do we have both theoretical and pratical support?

Background – How hard is the lattice reduction problem?

- In general, SVP is NP-hard.
- Approximate SVP (LWE) with polynomial factors?
- Then again, let us look at practical security in terms of concrete attacks?

Background – How hard is the lattice reduction problem?

- The first break through – LLL
- Approximation Factor: 2^n, running time polynomial!
- the length of the output of the shortest vector $= (RHF)^n \lambda_1$
  $\lambda_1 =$ the length of the SV.
- Root Hermite Factor: Theory: $\mathbf{1.075}$; Practice $\mathbf{1.02}$! Why?
  Sand pile model but ......
- BKZ: call SVP subroutine; Theory? ....
  Practical SVP?

Background – SVP Enumeration

- Given a basis B of a lattice, determine a region R, such that the SV is in R. Enumerate all the points in R
- How many lattice vectors needed in R: $2^{O(n \log n)}$, space small
  Pruning ....
  The best player for a while! (Till 2018?)

Background – Sieving – the Usurper

- Sample $2^{\alpha n}$ points, whose length is less or equal to r_0. Cover the samples with spheres of radius $r_1 < r_0$ centered at samples.
- We repeatedly use a "Sieving" procedure to shrink the size of the spheres to find the SV.

Background

"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" [1]

Is it really true?

[1] https://csrc.nist.gov/csrc/media/Projects/post-quantum-cryptography/documents/faq/Kyber-512-FAQ.pdf

Background: our work

Provable security means breaking these lattice-based schemes is not easier than solving some module/ring-LWE problems.

Hardness estimation of structured lattice problems:

- Expoliting the algebraic structure, e.g. there exists[2]quantum polynomial time algorithm for $2^{\tilde{\mathcal{O}}(\sqrt{n})}$-ideal SVP.
- General Lattice reduction algorithms, BKZ with sieving-based SVP oracle.

[2] [CDW21] Ronald Cramer, Léo Ducas, Benjamin Wesolowski. Mildly Short Vectors in Cyclotomic Ideal Lattices in Quantum Polynomial Time.

Background

We focus on the complexity of SVP oracles in BKZ. Progress on SVP in last decade[3]:

- 2013, SVP130 by Kenji Kashiwabara and Masaharu Fukase (RSR)
- 2015, SVP140 by Kenji Kashiwabara and Tadanori Teruya (RSR with small cluster)
- 2017, SVP150 by Kenji Kashiwabara and Tadanori Teruya (RSR with small clusters)
- 2018, SVP155 by M. Albrecht, L. Ducas, G. Herold, E. Kirshanova, E. Postlethwaite, M. Stevens, P. Karpman (bgj1 sieve)
- 2020, SVP170 by L. Ducas, M. Stevens, W. van Woerden (HK17[4]-sieve with GPU)
- 2021, SVP180 by L. Ducas, M. Stevens, W. van Woerden (HK17-sieve with GPU)

[3] see https://www.latticechallenge.org/svp-challenge/halloffame.php

[4] Herold, G., Kirshanova, E.: Improved Algorithms for the Approximate k-list Problem in Euclidean Norm.

Background

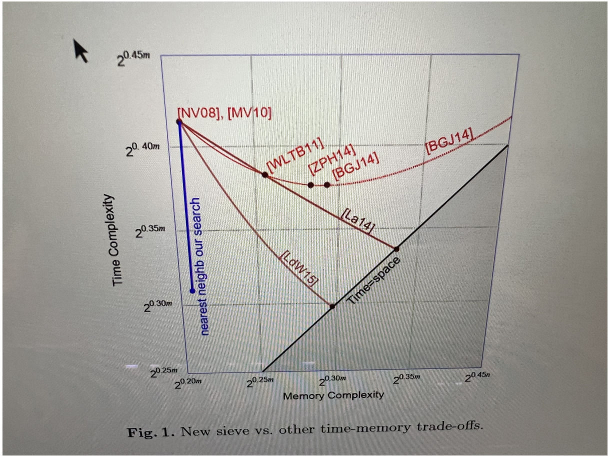
Sieving has finally won over enumeration, becoming the benchmark for assessing SVP hardness.

Gauss sieve / NV sieve	$2^{0.415n+o(n)}$
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.[5]

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

Background



$2^{0.45m}$
[NV08], [MV10]
$2^{0.40m}$
[WLTB11]
[ZPH14]
[BGJ14]
[BGJ14]
$2^{0.35m}$
[La14]
Time Complexity
nearest neighb our search
[LdW15]
$2^{0.30m}$
Time-space
$2^{0.25m}$
$2^{0.20m}$
$2^{0.25m}$
$2^{0.30m}$
$2^{0.35m}$
$2^{0.45n}$
Memory Complexity
Fig. 1. New sieve vs. other time-memory trade-offs.

Background

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

- XOR-POPCNT trick (simhash), Duc18[6], FBB$^+$15[7], Cha02[8]
- progressive sieving, Duc18, LM18[9]
- preprocessing (SubSieve, G6K) Duc18, ADH$^+$19[10]
- Dimensions for Free, Duc18, ADH$^+$19

---

[6] [Duc18] Léo Ducas, Shortest vector from lattice sieving: A few dimensions for free.

[7] [FBB$^+$15] Robert Fitzpatrick, Christian H. Bischof, Johannes Buchmann, Özgür Dagdelen, Florian Göpfert, Artur Mariano, and Bo-Yin Yang, Tuning GaussSieve for speed.

[8] [Cha02] Moses Charikar, Similarity estimation techniques from rounding algorithms.

[9] [LM18] Thijs Laarhoven and Artur Mariano, Progressive lattice sieving

[10] [ADH$^+$19] Albrecht, M.R., Ducas, L., Herold, G., Kirshanova, E., Postlethwaite, E.W., Stevens, M.: The general sieve kernel and new records in lattice reduction.

Background

"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" [11]

[11] https://csrc.nist.gov/csrc/media/Projects/post-quantum-cryptography/documents/faq/Kyber-512-FAQ.pdf

Background

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

- 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.

Background

Randomly access to a (N bits) 3D massive storage array costs $\mathcal{O}(N^{1/3})$.
A list of energy consumption for different operations:[12]

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

[12] Computer 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

Background

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 modifed BGJ sieve is very close to BDGL sieve, faster for all practical dimensions.
- supported by implementation.

Background

Table: Comparison with Previous GPU Records

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

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

Recall

$\mathcal{L}$ is an n-dimensional lattice with basis $\mathbf{b}_1, \cdots, \mathbf{b}_n$.
The Gram-Schmidt orthogonalization: $\mathbf{b}_1^*, \cdots, \mathbf{b}_n^*$.
The dual basis: $\mathbf{b}_1^\vee, \cdots, \mathbf{b}_n^\vee$.
Gaussian heuristic: $\mathrm{gh}(\mathcal{L})$

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 $\sqrt{4/3} \cdot \mathrm{gh}(\mathcal{L})$
Reducing-triples?	

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 buckets (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 $|\langle \mathbf{v}, \mathbf{c} \rangle| \geqslant \alpha \|\mathbf{v}\| \|\mathbf{c}\|$.

- BDGL sieve uses single layer of spherical cap shaped filters
- Asymptotically $\alpha \to 0.5$, $2^{0.292n+o(n)}$ buckets, each of size $2^{o(n)}$
- Bucket center $\mathbf{c}$ comes from random product code for quick bucketing[13]
- The slow down caused by non-uniformness of $\mathbf{c}$ is subexponential[14]

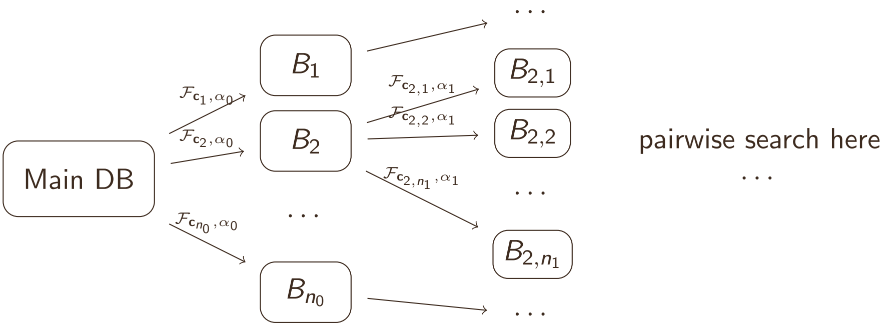
[13] [BDGL16] Anja Becker, Léo Ducas, Nicolas Gama, and Thijs Laarhoven, New directions in nearest neighbor searching with applications to lattice sieving.

[14] [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.



Time Complexity

Here, we replace the original filters with spherical cap-shaped filters.

- These filters have been shown to be optimal in terms of time complexity
- bgj1 sieve in [ADH+19] has proven efficient in practice.

Time Complexity

Let P_f be the probability that a vector will pass a random filter from $\mathcal{F}$, and P_p is the probability that a pair of vectors, which form an angle of $\pi/3$, are both accepted by the same random filter.

Theorem (Complexity of AllPairSearch-BGJ15[15])

Suppose L is a list of N uniformly random vectors in the sphere of dimension n, ρ is the exponent such that $P_f^{\rho} = P_p$, then the time complexity is $\tilde{\mathcal{O}}(N^{\rho})$.

[15][BGJ15] Becker, A., Gama, N., Joux, A.: Speeding-up lattice sieving without increasing the memory, using sub-quadratic nearest neighbor search.

Time Complexity

spherical caps $\mathcal{C}_{\mathbf{c},\alpha} = \{\mathbf{x} \in \mathbb{R}^n | \|\mathbf{x}\|^2 = 1, \langle \mathbf{x}, \mathbf{c} \rangle \geqslant \alpha \|\mathbf{c}\|\}$.
wedges (i.e. intersections of spherical caps) $\mathcal{W}_{\mathbf{c}_1,\alpha_1,\mathbf{c}_2,\alpha_2} = \mathcal{C}_{\mathbf{c}_1,\alpha_1} \cap \mathcal{C}_{\mathbf{c}_2,\alpha_2}$.

Lemma (Volume of spherical caps and wedges[16])

Let μ be the canonical Lebesgue measure, $\mathcal{S}^{n-1}$ be the unit sphere in $\mathbb{R}^n$, then for any $\alpha \in (0,1)$ we have

$$\frac{\mu(\mathcal{C}_{\mathbf{c},\alpha})}{\mu(\mathcal{S}^{n-1})} = \mathrm{poly}(n) \cdot \left(\sqrt{1-\alpha^2}\right)^n.$$

Furthermore, if the angle between $\mathbf{c}_1$ and $\mathbf{c}_2$ is θ, then

$$\frac{\mu(\mathcal{W}_{\mathbf{c}_1,\alpha,\mathbf{c}_2,\alpha})}{\mu(\mathcal{S}^{n-1})} = \mathrm{poly}(n) \cdot \left(\sqrt{1-\frac{2\alpha^2}{1+\cos\theta}}\right)^n.$$

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

Time Complexity

$P_f = \text{poly}(n) \cdot (1-\alpha^2)^{n/2}$ and $P_p = \text{poly}(n) \cdot (1-\frac{4}{3}\alpha^2)^{n/2}$. This implies that asymptotically

$$\rho \approx \ln(1-\frac{4}{3}\alpha^2)/\ln(1-\alpha^2)$$

- For original BGJ sieve, $\rho = 1.5$
- Achieves the asymptotically optimal time complexity of $\tilde{\mathcal{O}}(N^{4/3})$, when the dataset is sparse ($N = 2^{o(n)}$)

Time Complexity

- It's not guaranteed that filters optimal in the sparse regime will remain optimal in the dense regime ($N = 2^{\mathcal{O}(n)}$).
- Cross-polytope hashing is known to be optimal in the sparse case[17], but it leads to a suboptimal time complexity of $2^{0.297n+o(n)}$ when applied to lattice sieving[18].
- $2^{(0.297-0.292)\cdot(380-140)} = 2^{1.2}$, does it really matters?

[17] [AIL+15] Andoni, A., Indyk, P., Laarhoven, T., Razenshteyn, I., Schmidt, L.: Practical and optimal lsh for angular distance.

[18] [LdW15] Laarhoven, T., Weger, B.: Faster sieving for shortest lattice vectors using spherical locality-sensitive hashing

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

- 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

For last several layer of buckets?

Not observed in today's computational architectures, but it may matter for serious attacks using ASICs.

How to check for duplicates before a vector is inserted into the database without a UidHashTable?
Sort the list vectors and the newly found short vectors together, and then remove the duplicates and the longest ones.

$$\mathcal{O}(2^{0.2075n+o(n)}\log(2^{0.2075n+o(n)})) = 2^{0.2075n+o(n)}$$

Implementation Details

Fact

Lattice reduction and sieving are both numerically unstable.

The lattice basis $\mathbf{b}_1, \cdots, \mathbf{b}_n$ and the Gram-Schmidt orthogonalization $\mathbf{b}_1^*, \cdots, \mathbf{b}_n^*$ are represented by quad_float, with precision of 106 bits.

- 53 bits "double" may fail.
- We have vectorized the quad_float operations.

Implementation Details

The sieving vectors are rounded to 8 bit integers.

- save half of the bandwidth and computation resources.
- reduce the main database size by 40%.
- 4 bit is not enough.

Scale the entries by $254.0 \cdot (\sup_{1 \leqslant i \leqslant n} \|\mathbf{b}_i^*\|)^{-1}$ then round: After size reduction, overflow may not happen.

Implementation Details

Each vector, before inserted into the database, should carefully checked and normalized.
Normalization means: recover the integer coefficients wrt the local projected basis, and then compute fp32 vectors and finally round.
Dual basis of a well-reduced basis can be quite long.

Fact

Even if the sieving vectors are represented by "double", the normalization is still necessary.

Implementation Details

Choice of filtering batchsize and filter radius?

Example

For bgj1 the asymptotically optimal choice ($\alpha_0 = 0.366$) can be far from the practical optimum ($\alpha_0 = 0.315 \sim 0.325$)

Table: Chosen Filter Radius in bgj1, bgj3, bgj3, and bgj3-amx

Algorithm	α_0	α_1	α_2
bgj1	0.325	-	-
bgj2	0.257	0.280	-
bgj3	0.200	0.210	0.280
bgj3-amx	0.210	0.215	0.285

Implementation Details

XOR-POPCNT trick (simhash) is disabled in our implementation.

- the theoretical throughput for a dot product is less than 4 clock cycles
- simhash check makes the memory access pattern unpredictable
- additional 32 bytes memory per vector

Implementation Details

How do we compute dot products?

- "avx2" implementation: 8 dot products in parallel, first computed by vpdpbusd on ymm registers, then horizontally add the 32-bit results by vphaddd instruction.
- "amx" implementation: compute 256 dot products by only 3 tdpbssd instructions, transpose one of the 16 by 64 int8_t matrices before loaded into tmm registers by vpunpckldq, vpunpckhdq, and vshufi64x2.
- do not collect reducing-triples to avoid additional comparisons.

Refined Security Analysis

Does ... takes less than 2^{143} gates?
Does ... easier than breaking AES128?
Uncertainty exists regarding:

- the exact time complexity
- the memory access cost for the final bucket layers

Fact

The community can now solve about 1000000x harder SVP instances than ten years ago, with most of the progress attributable to the $o(n)$ term.

Thank you

Questions?