Estimating quantum speedups for lattice sieves

Martin R. Albrecht[1], Vlad Gheorghiu[2], Eamonn W. Postlethwaite[1], John M. Schanck[2]

[1]Information Security Group, Royal Holloway, University of London,
[2]Institute for Quantum Computing, University of Waterloo, Canada

What

- a better understanding of the non asymptotic complexity of quantum lattice sieves.

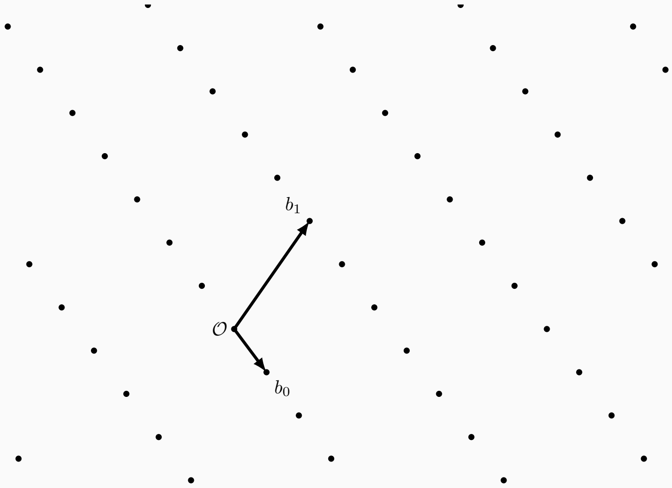
How

- by designing quantum circuits and software which optimises these circuits with respect to a number of germane cost metrics.

Why

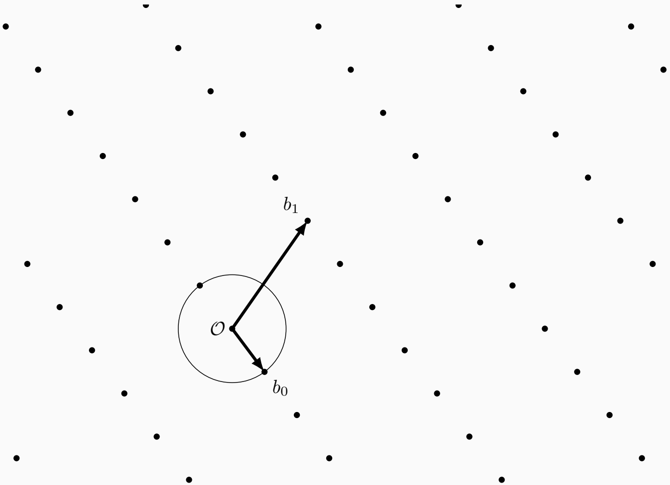
- this complexity is central to many estimates of the cost of cryptanalysis against lattice constructions.

What: lattices



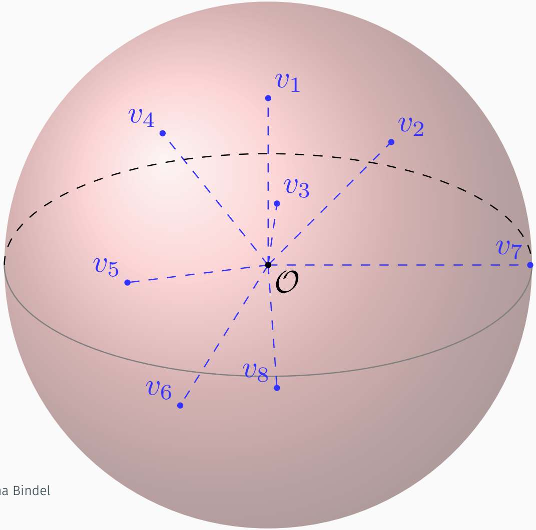
$\Lambda = \mathrm{Span}_{\mathbb{Z}}(b_0, \ldots, b_{d-1})$, $B = \{b_0, \ldots, b_{d-1}\} \subset \mathbb{R}^d$ a basis

What: lattices



SVP: find $v \in \Lambda \setminus \{0\}$ such that $\|v\|_2 \leq \|w\|_2$ for all $w \in \Lambda \setminus \{0\}$

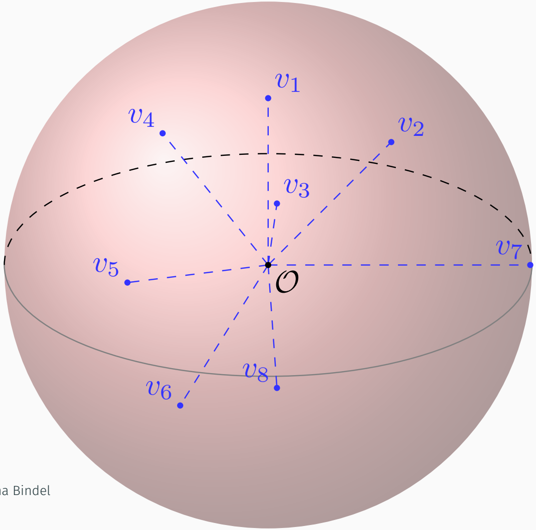
What: lattice sieves



Template Credit: Nina Bindel

heuristic says (after normalisation) v_i are i.i.d. uniform on S^{d-1}

What: lattice sieves



Template Credit: Nina Bindel

find pairs (v_i, v_j) such that $\left\|v_i - v_j\right\|_2 \leq 1 \iff \langle v_i, v_j \rangle \geq \cos(\pi/3)$.

What: different lattice sieves

Sieve (NNS subroutine)[1]	$\log_2 \mathrm{time}_C$	$\log_2 \mathrm{time}_Q$
NV style [NV08]	$0.415d$	$0.311d$
RandomBucket [BGJ15, ADH$^+$19]	$0.349d$	$0.301d$
ListDecoding [BDGL16]	$0.292d$	$0.265d$

[1] All complexities are missing $+o(d)$ terms.

What: different lattice sieves

Sieve (NNS subroutine)[1]	$\log_2 \text{time}_C$	$\log_2 \text{time}_Q$
NV style [NV08]	$0.415d$	$0.311d$
RandomBucket [BGJ15, ADH$^+$19]	$0.349d$	$0.301d$
ListDecoding [BDGL16]	$0.292d$	$0.265d$

The quantum variants of these sieves use Grover's search algorithm to instantiate the search for reducing pairs.

All require exponential space, $2^{\Theta(d)}$.

[1] All complexities are missing $+o(d)$ terms.

How: classical and quantum search

Let $[N] = \{1, \ldots, N\}$ and $f\colon [N] \to \{0,1\}$ be an unstructured predicate, with *roots*

$$\mathrm{Ker}(f) = \{x\colon f(x) = 0\}.$$

How: classical and quantum search

Let $[N] = \{1, \ldots, N\}$ and $f\colon [N] \to \{0, 1\}$ be an unstructured predicate, with *roots*

$$\mathrm{Ker}(f) = \{x\colon f(x) = 0\}.$$

We can find a root

- classically by evaluating $f(1), \ldots, f(N)$,
- quantumly by measuring $\mathbf{G}(f)^j \mathbf{D}|0\rangle$.

How: classical and quantum search

Let $[N] = \{1, \dots, N\}$ and $f\colon [N] \to \{0, 1\}$ be an unstructured predicate, with *roots*

$$\mathrm{Ker}(f) = \{x\colon f(x) = 0\}.$$

We can find a root

- classically by evaluating $f(1), \dots, f(N)$,
- quantumly by measuring $\mathbf{G}(f)^j \mathbf{D}|0\rangle$.

If $|\mathrm{Ker}(f)| \ll N$ then we expect $O(N)$ queries to f classically, and $j \in O(\sqrt{N})$ queries to $\mathbf{G}(f)$ quantumly.

How: filtered search

A potentially cheaper way is to use a filter, some predicate

$$g\colon [N] \to \{0,1\}, |\mathsf{Ker}(g) \cap \mathsf{Ker}(f)| \geq 1.$$

Then (classically) we can evaluate

$$g(1), f(1) \text{ when } g(1) = 0, \ldots, g(N), f(N) \text{ when } g(N) = 0.$$

How: filtered search

A potentially cheaper way is to use a filter, some predicate

$$g\colon [N] \to \{0,1\}, |\mathrm{Ker}(g) \cap \mathrm{Ker}(f)| \geq 1.$$

Then (classically) we can evaluate

$$g(1), f(1) \text{ when } g(1) = 0, \ldots, g(N), f(N) \text{ when } g(N) = 0.$$

What makes a good filter? Cheap to evaluate, and

$$\rho_f(g) = 1 - \frac{|\mathrm{Ker}(f) \cap \mathrm{Ker}(g)|}{|\mathrm{Ker}(g)|}, \quad \eta_f(g) = 1 - \frac{|\mathrm{Ker}(f) \cap \mathrm{Ker}(g)|}{|\mathrm{Ker}(f)|}$$

the false positive and negative rate, are both small.

How: filtered *quantum* search

Branching based on g is not possible within Grover's algorithm. We give a technical lemma that (roughly) says, let

How: filtered *quantum* search

Branching based on g is not possible within Grover's algorithm. We give a technical lemma that (roughly) says, let

- g be a filter for predicate $f : [N] \to \{0, 1\}$,
- $P, Q, \gamma \in \mathbb{R}$ such that
  - $P/\gamma \leq |\mathsf{Ker}(g)| \leq \gamma P$, and
  - $1 \leq Q \leq |\mathsf{Ker}(f) \cap \mathsf{Ker}(g)|$.

Then we can find a root of f with constant probability with a cost dominated by $\frac{\gamma}{2}\sqrt{N/Q}$ calls to $\mathsf{G}(g)$.

How: filtered *quantum* search

Branching based on g is not possible within Grover's algorithm. We give a technical lemma that (roughly) says, let

- g be a filter for predicate $f\colon [N] \to \{0,1\}$,
- $P, Q, \gamma \in \mathbb{R}$ such that
  - $P/\gamma \leq |\mathsf{Ker}(g)| \leq \gamma P$, and
  - $1 \leq Q \leq |\mathsf{Ker}(f) \cap \mathsf{Ker}(g)|$.

Then we can find a root of f with constant probability with a cost dominated by $\frac{\gamma}{2}\sqrt{N/Q}$ calls to $\mathsf{G}(g)$.

The cost of a Grover query encoding the filter, $\mathsf{G}(g)$, and not one encoding the predicate, $\mathsf{G}(f)$, is then the crucial quantity.

How: `popcount` is our filter

For lattices vectors $u, v_1, \ldots, v_N$, the reduction predicate of u is

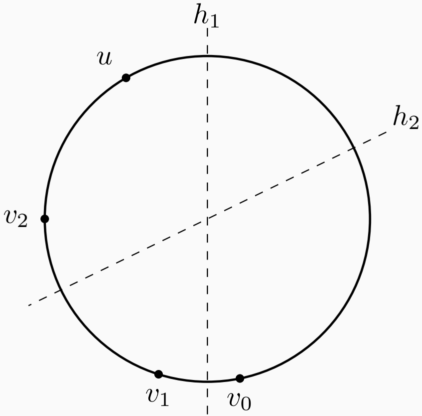
$$f_u \colon \{v_1, \ldots, v_N\} \to \{0, 1\},\ f_u(v_i) = 0 \iff \langle u, v_i \rangle \geq \cos(\pi/3).$$

How: popcount is our filter

For lattices vectors $u, v_1, \ldots, v_N$, the reduction predicate of u is

$$f_u\colon \{v_1, \ldots, v_N\} \to \{0, 1\},\ f_u(v_i) = 0 \iff \langle u, v_i \rangle \geq \cos(\pi/3).$$

For the filter g we use 'XOR and popcount' [FBB+14].



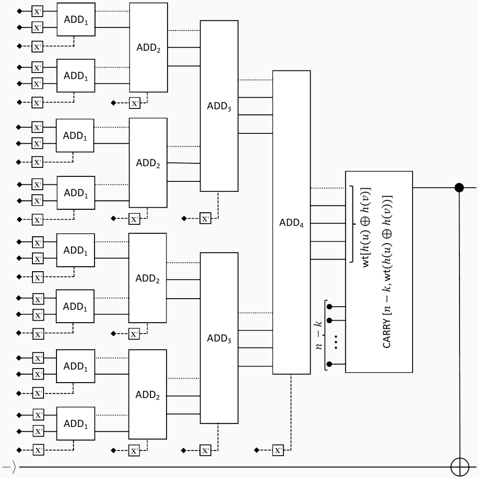
$$\texttt{popcount}_{1,2}(u, v_0) = 1$$

$$\texttt{popcount}_{1,2}(u, v_1) = 0$$

$$\texttt{popcount}_{1,2}(u, v_2) = 0$$

$$(k, n) = (1, 2)$$

How: circuits for G($\texttt{popcount}_{k,n}$)



We also analyse $\rho_{f_u}(\texttt{popcount}_{k,n})$ and $\eta_{f_u}(\texttt{popcount}_{k,n})$ as a function of k, n, d.

How: cost metrics

Following [JS19] we measure the cost of running a quantum circuit in terms of the classical control required to run it, under a number of different assumptions which imply the following cost metrics.

How: cost metrics

Following [JS19] we measure the cost of running a quantum circuit in terms of the classical control required to run it, under a number of different assumptions which imply the following cost metrics.

- gates: quantum gates cost $\Theta(1)$,

How: cost metrics

Following [JS19] we measure the cost of running a quantum circuit in terms of the classical control required to run it, under a number of different assumptions which imply the following cost metrics.

- gates: quantum gates cost $\Theta(1)$,
- depth-width: {quantum gates, identity wires} cost $\Theta(1)$,
- error: {quantum gates, identity wires} cost $\Omega(\log^2(DW))$.

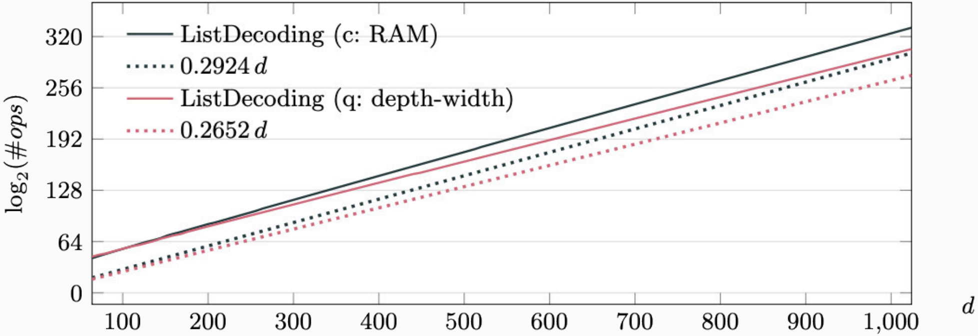
How: cost metrics

Following [JS19] we measure the cost of running a quantum circuit in terms of the classical control required to run it, under a number of different assumptions which imply the following cost metrics.

- gates: quantum gates cost $\Theta(1)$,
- depth-width: {quantum gates, identity wires} cost $\Theta(1)$,
- error: {quantum gates, identity wires} cost $\Omega(\log^2(DW))$.

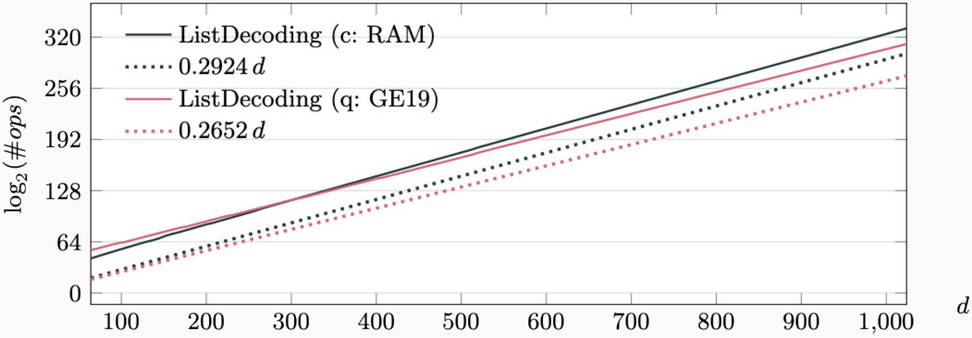
In particular we use the error correction model of Gidney–Ekerå [GE19] and the Clifford+T gate set.

Estimates: ListDecoding depth-width



ListDecodingSearch. Comparing c: (RAM) with q: (depth-width), and the leading terms of the asymptotic complexities.

Estimates: ListDecoding Gidney–Ekerå error correction



ListDecodingSearch. Comparing c: (RAM) with q: (GE19), and the leading terms of the asymptotic complexities.

Discussion I

Our estimates suggest less than advantage for quantum sieves than the asymptotic $2^{(0.292-0.265)d+o(d)}$, without entirely ruling out their relevance.

Discussion I

Our estimates suggest less than advantage for quantum sieves than the asymptotic $2^{(0.292-0.265)d+o(d)}$, without entirely ruling out their relevance.

Quantum Metric	d	$\log_2 \text{time}_C$	$\log_2 \text{time}_Q$	$\log_2$ memory
Gidney–Ekerå	312	119	119	78
Gidney–Ekerå	352	130	128	87
Gidney–Ekerå	824	270	256	187
Depth-Width	544	189	176	128
Gidney–Ekerå	544	189	182	128

All classical costs are in a simple RAM model, the above table is for ListDecoding.

Discussion II

Our analyses do not account for the cost of qRAM and RAM, required in $\mathsf{G}(g)$ and g respectively, which we assign unit cost. While both do not have unit cost in practice, qRAM seems to have a much higher cost.

We also do not capture the natural clock speed error correction implies: after each layer of quantum circuit depth non trivial classical processing must occur.

Finally, we do not apply depth constraints, the impact of which on Grover's search is more than classical search, which can be trivially parallelised.

NNS to SVP

We have examined the NNS subroutine of lattice sieve algorithms. While this is the primary subroutine, it is not the full story.

NNS to SVP

We have examined the NNS subroutine of lattice sieve algorithms. While this is the primary subroutine, it is not the full story.
Potential underestimates

- the NNS subroutine is iterated poly(d) times,
- other subroutines, e.g. used for LSF or lattice vector sampling, are not accounted for.

NNS to SVP

We have examined the NNS subroutine of lattice sieve algorithms. While this is the primary subroutine, it is not the full story.

Potential underestimates

- the NNS subroutine is iterated $\text{poly}(d)$ times,
- other subroutines, e.g. used for LSF or lattice vector sampling, are not accounted for.

Potential overestimates

- using 'dimensions for free' techniques [Duc18], NNS in dimension d solves SVP in dimension $d' > d$,
- many heuristic tricks [ADH$^+$19, FBB$^+$14] are not captured.

Thanks

All data and our software can be found at

`https://github.com/jschanck/eprint-2019-1161`

The paper can be found at

`https://eprint.iacr.org/2019/1161`

References i

- Martin R. Albrecht, Léo Ducas, Gottfried Herold, Elena Kirshanova, Eamonn W. Postlethwaite, and Marc Stevens, *The general sieve kernel and new records in lattice reduction*, EUROCRYPT, 2019.
- Anja Becker, Léo Ducas, Nicolas Gama, and Thijs Laarhoven, *New directions in nearest neighbor searching with applications to lattice sieving*, Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, 2016.
- Anja Becker, Nicolas Gama, and Antoine Joux, *Speeding-up lattice sieving without increasing the memory, using sub-quadratic nearest neighbor search*, Cryptology ePrint Archive, Report 2015/522, 2015, `https://eprint.iacr.org/2015/522`.
- Léo Ducas, *Shortest vector from lattice sieving: A few dimensions for free*, EUROCRYPT, 2018.
- Robert Fitzpatrick, Christian Bischof, Johannes Buchmann, Özgür Dagdelen, Florian Göpfert, Artur Mariano, and Bo-Yin Yang, *Tuning gausssieve for speed*, LATINCRYPT, 2014.
- Craig Gidney and Martin Ekerå, *How to factor 2048 bit rsa integers in 8 hours using 20 million noisy qubits*, 2019.

- Samuel Jaques and John M. Schanck, *Quantum cryptanalysis in the ram model: Claw-finding attacks on sike*, CRYPTO, 2019.
- Phong Q. Nguyen and Thomas Vidick, *Sieve algorithms for the shortest vector problem are practical*, Journal of Mathematical Cryptology **2** (2008), no. 2, 181–207.