Classical and quantum 3 and 4-sieves to solve SVP with low memory

Johanna Loyer

Joint work with André Chailloux
Inria Paris

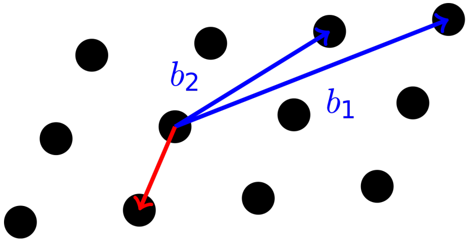
Lattice and SVP

Lattice

Given a basis $B = (\vec{b_1}, ..., \vec{b_d})$, the lattice $\mathcal{L}$ generated by B is the set of all integer linear combinations of its basis vectors: $\mathcal{L}(B) = \left\{ \sum_{i=1}^{d} z_i \vec{b_i},\ z_i \in \mathbb{Z} \right\}$.

Shortest Vector Problem (SVP)

Given a lattice $\mathcal{L}$, find the shortest non-zero vector $\vec{v} \in \mathcal{L}$.



Motivation to solve SVP

Cryptography

- NP-hard problem, hard in average, believed to be quantum-resistant.
- Problems derived from SVP: LWE, SIS, NTRU...
- Cryptosystems based on them: Kyber, Dilithium, Falcon (NIST standardization), FHE

Motivation to solve SVP

Cryptography

- NP-hard problem, hard in average, believed to be quantum-resistant.
- Problems derived from SVP: LWE, SIS, NTRU...
- Cryptosystems based on them: Kyber, Dilithium, Falcon (NIST standardization), FHE

Cryptanalysis

- Broken if we can find a reduced basis of the lattice.
- BKZ algorithm returns a reduced basis using an SVP-solver.

$\Rightarrow$ The security of these cryptosystems directly relies on the complexity of solving SVP.

Overview

1. Lattice sieving
   Configuration problem
2. Filtering
   New Random Product Code for filtering
3. Framework to solve SVP
4. Trade-offs for classic and quantum k-sieves

Sieving

Sieving

Heuristic: Lattice vectors act as random vectors.

- Implies that vectors of norm at most R are w.h.p. of norm very close to R.
- Validated by experiments [NV08] for long vectors.

Sieving

Heuristic: Lattice vectors act as random vectors.

- Implies that vectors of norm at most R are w.h.p. of norm very close to R.
- Validated by experiments [NV08] for long vectors.

Sieving step

Input: list L of N lattice vectors of norm at most R ; $\gamma < 1$.
Output: list L_{out} of N lattice vectors of norm at most $\gamma R < R$.

Sieving

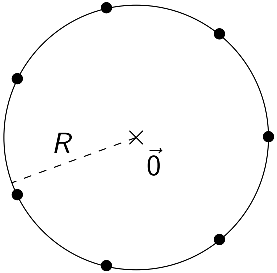
Heuristic: Lattice vectors act as random vectors.

- Implies that vectors of norm at most R are w.h.p. of norm very close to R.
- Validated by experiments [NV08] for long vectors.

Sieving step

Input: list L of N lattice vectors of norm at most R ; $\gamma < 1$.
Output: list L_{out} of N lattice vectors of norm at most $\gamma R < R$.

Initialization:
Generate N lattice vectors
of norm $\leq R$
(Klein's algorithm)



Sieving

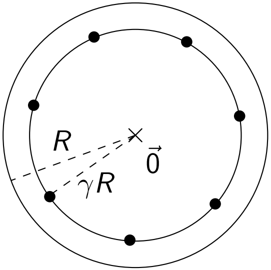
Heuristic: Lattice vectors act as random vectors.

- Implies that vectors of norm at most R are w.h.p. of norm very close to R.
- Validated by experiments [NV08] for long vectors.

Sieving step

Input: list L of N lattice vectors of norm at most R ; $\gamma < 1$.
Output: list L_{out} of N lattice vectors of norm at most $\gamma R < R$.

After 1 iteration:
vectors of norm at most γR



Sieving

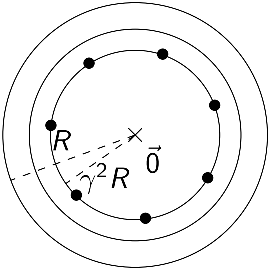
Heuristic: Lattice vectors act as random vectors.

- Implies that vectors of norm at most R are w.h.p. of norm very close to R.
- Validated by experiments [NV08] for long vectors.

Sieving step

Input: list L of N lattice vectors of norm at most R ; $\gamma < 1$.
Output: list L_{out} of N lattice vectors of norm at most $\gamma R < R$.

After 2 iterations:
vectors of norm at most $\gamma^2 R$



Sieving

Heuristic: Lattice vectors act as random vectors.

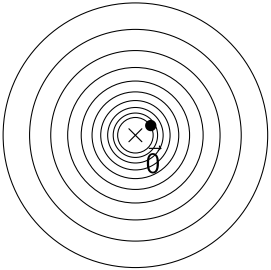
- Implies that vectors of norm at most R are w.h.p. of norm very close to R.
- Validated by experiments [NV08] for long vectors.

Sieving step

Input: list L of N lattice vectors of norm at most R ; $\gamma < 1$.
Output: list L_{out} of N lattice vectors of norm at most $\gamma R < R$.

After poly(d) iterations:
norm at most $\gamma^{\text{poly}(d)} R$.

Short vector found!

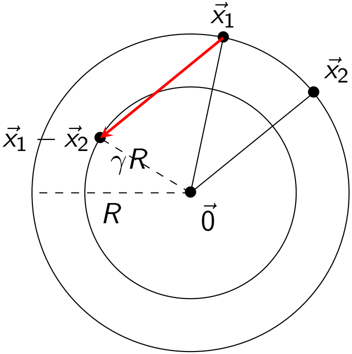


Sieving step

Nguyen-Vidick sieve [NV08] (2-sieve)

for $(\vec{\mathbf{x}}_1, \vec{\mathbf{x}}_2) \in L \times L$:
  if $\|\vec{\mathbf{x}}_1 - \vec{\mathbf{x}}_2\| \leqslant \gamma R$:
    add $\vec{\mathbf{x}}_1 - \vec{\mathbf{x}}_2$ to L_{out}

Sphere of dimension d and radius R:

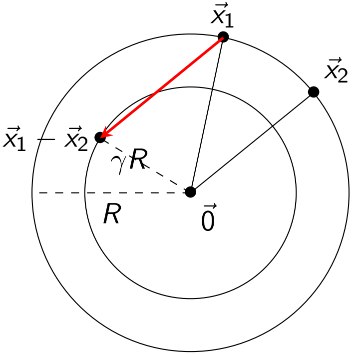


Sieving step

Nguyen-Vidick sieve [NV08] (2-sieve)

for $(\vec{\mathbf{x}}_1, \vec{\mathbf{x}}_2) \in L \times L$:
    if $\|\vec{\mathbf{x}}_1 - \vec{\mathbf{x}}_2\| \leqslant \gamma R$:
        add $\vec{\mathbf{x}}_1 - \vec{\mathbf{x}}_2$ to L_{out}

Sphere of dimension d and radius R:



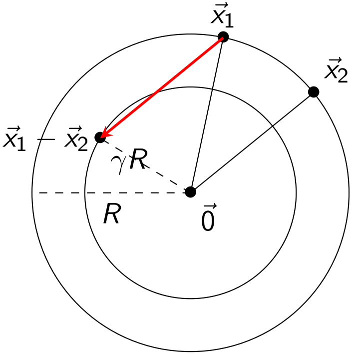
If $\vec{\mathbf{x}}_1, \vec{\mathbf{x}}_2 \in \mathcal{L}$ then $\vec{\mathbf{x}}_1 - \vec{\mathbf{x}}_2 \in \mathcal{L}$.

Sieving step

Nguyen-Vidick sieve [NV08] (2-sieve)

for $(\vec{\mathbf{x}}_1, \vec{\mathbf{x}}_2) \in L \times L$:
    if $\|\vec{\mathbf{x}}_1 - \vec{\mathbf{x}}_2\| \leqslant \gamma R$:
        add $\vec{\mathbf{x}}_1 - \vec{\mathbf{x}}_2$ to L_{out}

Sphere of dimension d and radius R:



If $\vec{\mathbf{x}}_1, \vec{\mathbf{x}}_2 \in \mathcal{L}$ then $\vec{\mathbf{x}}_1 - \vec{\mathbf{x}}_2 \in \mathcal{L}$.

Condition of reduction:
For $\gamma = 1$, $\|\vec{\mathbf{x}}_1\| = \|\vec{\mathbf{x}}_2\| = R$,

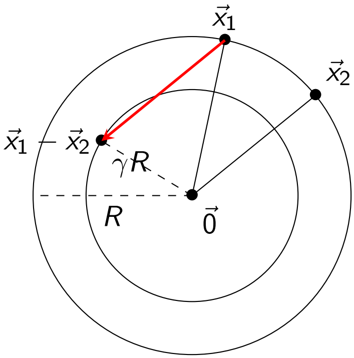
$$\|\vec{\mathbf{x}}_1 - \vec{\mathbf{x}}_2\| \leqslant \gamma R$$
$$\Leftrightarrow \text{Angle}(\vec{\mathbf{x}}_1, \vec{\mathbf{x}}_2) \leqslant \frac{\pi}{3}$$

Sieving step

Nguyen-Vidick sieve [NV08] (2-sieve)

for $(\vec{\mathbf{x}}_1, \vec{\mathbf{x}}_2) \in L \times L$:
  if $\|\vec{\mathbf{x}}_1 - \vec{\mathbf{x}}_2\| \leqslant \gamma R$:
    add $\vec{\mathbf{x}}_1 - \vec{\mathbf{x}}_2$ to L_{out}

Sphere of dimension d
and radius R:



If $\vec{\mathbf{x}}_1, \vec{\mathbf{x}}_2 \in \mathcal{L}$ then $\vec{\mathbf{x}}_1 - \vec{\mathbf{x}}_2 \in \mathcal{L}$.

Condition of reduction:
For $\gamma = 1$, $\|\vec{\mathbf{x}}_1\| = \|\vec{\mathbf{x}}_2\| = R$,

$$\|\vec{\mathbf{x}}_1 - \vec{\mathbf{x}}_2\| \leqslant \gamma R$$
$$\Leftrightarrow \text{Angle}(\vec{\mathbf{x}}_1, \vec{\mathbf{x}}_2) \leqslant \frac{\pi}{3}$$
$$\Leftrightarrow \frac{1}{R^2}\langle \vec{\mathbf{x}}_1 | \vec{\mathbf{x}}_2 \rangle \geq \frac{1}{2}.$$

Sieving step

3-sieve

for $(\vec{\mathbf{x}}_1, \vec{\mathbf{x}}_2, \vec{\mathbf{x}}_3) \in L^3$:
  if $\|\vec{\mathbf{x}}_1 + \vec{\mathbf{x}}_2 + \vec{\mathbf{x}}_3\| \leqslant \gamma R$:
    add $\vec{\mathbf{x}}_1 + \vec{\mathbf{x}}_2 + \vec{\mathbf{x}}_3$ to L_{out}

4-sieve

for $(\vec{\mathbf{x}}_1, \vec{\mathbf{x}}_2, \vec{\mathbf{x}}_3, \vec{\mathbf{x}}_4) \in L^4$
  if $\|\vec{\mathbf{x}}_1 + \vec{\mathbf{x}}_2 + \vec{\mathbf{x}}_3 + \vec{\mathbf{x}}_4\| \leqslant \gamma R$:
    add $\vec{\mathbf{x}}_1 + \vec{\mathbf{x}}_2 + \vec{\mathbf{x}}_3 + \vec{\mathbf{x}}_4$ to L_{out}

k-sieve

for $(\vec{\mathbf{x}}_1, ..., \vec{\mathbf{x}}_k) \in L^k$
  if $\|\vec{\mathbf{x}}_1 + ... + \vec{\mathbf{x}}_k\| \leqslant \gamma R$:
    add $\vec{\mathbf{x}}_1 + ... + \vec{\mathbf{x}}_k$ to L_{out}

Minimal size of the list L

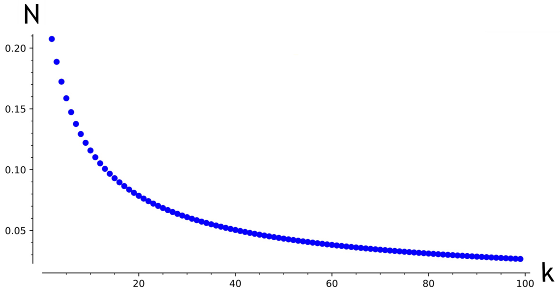
Sieving step

Input: List of N lattice vectors
Output: List of N reduced lattice vectors

$\Rightarrow$ We need that there exists N reduced vectors calculable from the N input vectors.

Notation: $2^{xd+o(d)}$

k	Memory N	Time (naive) N^k
2	0.208	0.415
3	0.189	0.566
4	0.173	0.690
5	0.159	0.794
6	0.147	0.884

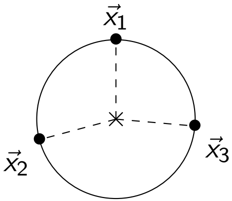


Reduction to the configuration problem

Configurations

Configuration

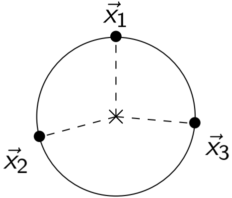
k-tuple $(\vec{\mathbf{x}}_1, \ldots, \vec{\mathbf{x}}_k)$ satisfies configuration $C = (C_{ij})_{i,j} \in \mathbb{R}^{k \times k}$ iff. $\langle \vec{\mathbf{x}}_i | \vec{\mathbf{x}}_j \rangle \leq C_{ij}$ (with $C_{ij} \leq 0$).



Configurations

Configuration

k-tuple $(\vec{\mathbf{x}}_1, ..., \vec{\mathbf{x}}_k)$ satisfies configuration $C = (C_{ij})_{i,j} \in \mathbb{R}^{k\times k}$ iff. $\langle \vec{\mathbf{x}}_i | \vec{\mathbf{x}}_j \rangle \leq C_{ij}$ (with $C_{ij} \leq 0$).



Valid configuration C: $(\vec{\mathbf{x}}_1, ..., \vec{\mathbf{x}}_k)$ satisfies $C \Rightarrow \|\vec{\mathbf{x}}_1 + ... + \vec{\mathbf{x}}_k\| \leq \gamma R$

Configurations

Configuration problem

Input: List L, a valid configuration C
Output: Tuples $(\vec{\mathbf{x}}_1, \ldots, \vec{\mathbf{x}}_k)$ for $\vec{\mathbf{x}}_i \in L$ satisfying configuration C

Configurations

Configuration problem

Input: List L, a valid configuration C
Output: Tuples $(\vec{\mathbf{x}}_1, ..., \vec{\mathbf{x}}_k)$ for $\vec{\mathbf{x}}_i \in L$ satisfying configuration C

$\Rightarrow$

***k*-sieve problem**

Input: List L
Output: Vectors $\sum_{i=1}^{k} \vec{\mathbf{x}}_i$ for $\vec{\mathbf{x}}_i \in L$ of norm $\leq \gamma R$.

Configurations

Configuration problem

Input: Lists $L_1, \ldots, L_k$, a valid configuration C
Output: Tuples $(\vec{\mathbf{x}}_1, \ldots, \vec{\mathbf{x}}_k) \in L_1 \times \ldots \times L_k$ satisfying configuration C

$\Rightarrow$

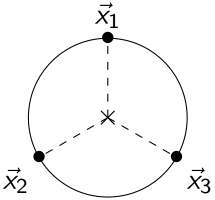
k-sieve problem

Input: List L
Output: Vectors $\sum_{i=1}^{k} \vec{\mathbf{x}}_i$ for $\vec{\mathbf{x}}_i \in L$ of norm $\leq \gamma R$.

Configurations

Balanced configuration

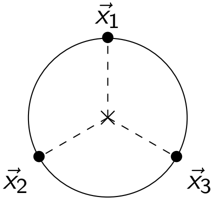
- Fix $C_{ij} = -1/k$ for $i \neq j$
- The most common configuration for reducing k-tuples
  $\Rightarrow$ Minimizes the memory $|L|$.



Configurations

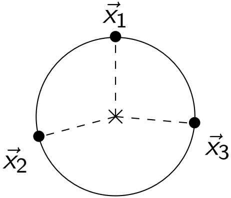
Balanced configuration

- Fix $C_{ij} = -1/k$ for $i \neq j$
- The most common configuration for reducing k-tuples
  $\Rightarrow$ Minimizes the memory $|L|$.



Any configuration

- Only constraint: $\|\vec{\mathbf{x}}_1 + ... + \vec{\mathbf{x}}_k\| \leq \gamma R$
- Rarer configurations $\Rightarrow$ Require longer list, but the tuples can be easier to find.



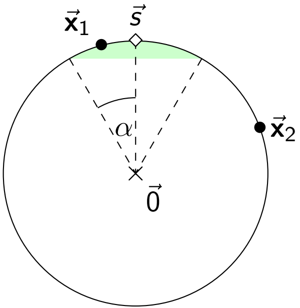
Locality Sensitive Filtering (LSF)

Filtering

Locality Sentitive Filter

A **filter** $f_{\vec{s},\alpha}$ of center $\vec{s} \in \mathbb{R}^d$ and angle $\alpha \in [0, \pi/2]$ maps a vector $\vec{x}$ to a boolean value:

- 1 if Angle$(\vec{x}, \vec{s}) \leqslant \alpha$,
- 0 else.

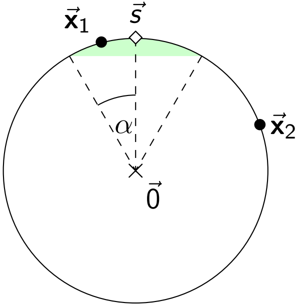


Filtering

Locality Sentitive Filter

A **filter** $f_{\vec{s},\alpha}$ of center $\vec{s} \in \mathbb{R}^d$ and angle $\alpha \in [0, \pi/2]$ maps a vector $\vec{x}$ to a boolean value:

- 1 if $\text{Angle}(\vec{x}, \vec{s}) \leqslant \alpha$,
- 0 else.



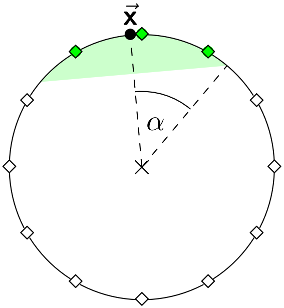
Each filter is associated with a set that we can fill with vectors.

Filtering - Random Product Code

Random Product Code (RPC) of parameters $[d, m, B]$

$$\mathfrak{C} = Q \cdot (\mathfrak{C}_1 \times \cdots \times \mathfrak{C}_m) \subset \mathbb{R}^d$$

- $\mathfrak{C}_1, \ldots, \mathfrak{C}_m$: sets of B vectors in $\mathbb{R}^{d/m}$ sampled unif. & indep. random of norm $\sqrt{1/m}$
- Q uniformly random rotation over $\mathbb{R}^d$



Codewords $\diamond$

- Uniformly distributed over the sphere
- Each codeword = center of one filter
- Decode $\vec{\mathbf{x}}$ in efficient time (subexp. or poly)

Filtering - Random Product Code

$$\mathfrak{C} = Q \cdot (\mathfrak{C}_1 \times \cdots \times \mathfrak{C}_m)$$

List Decoding Algorithm for RPC [BDGL16]

Input: Random Product Code $\mathfrak{C}$, vector $\vec{\mathbf{x}}$, angle α

Output: List of all the filters $\mathbf{F} \in \mathfrak{C}$ of angle at most α with $\vec{\mathbf{x}}$.

Filtering - Random Product Code

$$\mathfrak{C} = Q \cdot (\mathfrak{C}_1 \times \cdots \times \mathfrak{C}_m)$$

List Decoding Algorithm for RPC [BDGL16]

Input: Random Product Code $\mathfrak{C}$, vector $\vec{\mathbf{x}}$, angle α

Output: List of all the filters $\mathbf{F} \in \mathfrak{C}$ of angle at most α with $\vec{\mathbf{x}}$.

1. Apply Q^{-1} on $\vec{\mathbf{x}}$

Filtering - Random Product Code

$$\mathfrak{C} = Q \cdot (\mathfrak{C}_1 \times \cdots \times \mathfrak{C}_m)$$

List Decoding Algorithm for RPC [BDGL16]

Input: Random Product Code $\mathfrak{C}$, vector $\vec{\mathbf{x}}$, angle α

Output: List of all the filters $\mathbf{F} \in \mathfrak{C}$ of angle at most α with $\vec{\mathbf{x}}$.

1. Apply Q^{-1} on $\vec{\mathbf{x}}$
2. Identify with a tuple of m vectors: $Q^{-1}(\vec{\mathbf{x}}) := (\vec{\mathbf{x}}_1 \| ... \| \vec{\mathbf{x}}_m)$

Filtering - Random Product Code

$$\mathfrak{C} = Q \cdot (\mathfrak{C}_1 \times \cdots \times \mathfrak{C}_m)$$

List Decoding Algorithm for RPC [BDGL16]

Input: Random Product Code $\mathfrak{C}$, vector $\vec{\mathbf{x}}$, angle α

Output: List of all the filters $\mathbf{F} \in \mathfrak{C}$ of angle at most α with $\vec{\mathbf{x}}$.

1. Apply Q^{-1} on $\vec{\mathbf{x}}$
2. Identify with a tuple of m vectors: $Q^{-1}(\vec{\mathbf{x}}) := (\vec{\mathbf{x}}_1 \| \ldots \| \vec{\mathbf{x}}_m)$
3. Decode each $\vec{\mathbf{x}}_i$ with subcode $\mathfrak{C}_i$ (brute force)

Filtering - Random Product Code

$$\mathfrak{C} = Q \cdot (\mathfrak{C}_1 \times \cdots \times \mathfrak{C}_m)$$

List Decoding Algorithm for RPC [BDGL16]

Input: Random Product Code $\mathfrak{C}$, vector $\vec{\mathbf{x}}$, angle α

Output: List of all the filters $\mathbf{F} \in \mathfrak{C}$ of angle at most α with $\vec{\mathbf{x}}$.

1. Apply Q^{-1} on $\vec{\mathbf{x}}$
2. Identify with a tuple of m vectors: $Q^{-1}(\vec{\mathbf{x}}) := (\vec{\mathbf{x}}_1 \| ... \| \vec{\mathbf{x}}_m)$
3. Decode each $\vec{\mathbf{x}}_i$ with subcode $\mathfrak{C}_i$ (brute force)
4. Assemble the obtained codewords of $\mathfrak{C}_1, ..., \mathfrak{C}_m$

Filtering - Random Product Code

$$\mathfrak{C} = Q \cdot (\mathfrak{C}_1 \times \cdots \times \mathfrak{C}_m)$$

List Decoding Algorithm for RPC [BDGL16]

Input: Random Product Code $\mathfrak{C}$, vector $\vec{\mathbf{x}}$, angle α

Output: List of all the filters $\mathbf{F} \in \mathfrak{C}$ of angle at most α with $\vec{\mathbf{x}}$.

1. Apply Q^{-1} on $\vec{\mathbf{x}}$
2. Identify with a tuple of m vectors: $Q^{-1}(\vec{\mathbf{x}}) := (\vec{\mathbf{x}}_1 \| ... \| \vec{\mathbf{x}}_m)$
3. Decode each $\vec{\mathbf{x}}_i$ with subcode $\mathfrak{C}_i$ (brute force)
4. Assemble the obtained codewords of $\mathfrak{C}_1, ..., \mathfrak{C}_m$
5. Apply rotation Q to recover $\mathfrak{C}$'s codewords = nearest filters of $\vec{\mathbf{x}}$

Filtering - Solving SVP

2-sieve

For each vector: search a reducing vector within the whole list L.

2-sieve with filtering

1. Generate the filters $\triangleright$ Sample a RPC
2. Add each vector to its filters of angle at most α. $\triangleright$ List decoding algorithm
3. For each vector : search a reducing vector within its filters.

Filtering - Solving SVP

2-sieve

For each vector: search a reducing vector within the whole list L.

2-sieve with filtering

1. Generate the filters ▷ Sample a RPC
2. Add each vector to its filters of angle at most α. ▷ List decoding algorithm
3. For each vector : search a reducing vector within its filters.
   - Classically or by Grover's search

Time complexity (for minimal memory $N = 2^{0.208d+o(d)}$):

Classical 2-sieve: $2^{0.415d+o(d)}$

With filtering: $2^{0.292d+o(d)}$

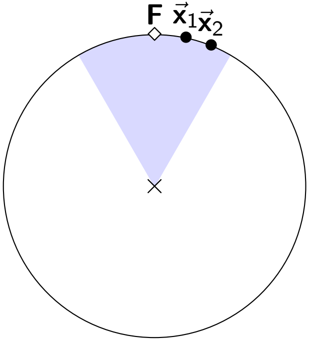
Quantum 2-sieve: $2^{0.312d+o(d)}$

With filtering: $2^{0.265d+o(d)}$

New code for filtering

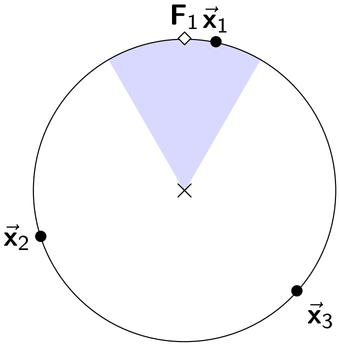
Filtering strategy for the 2-sieve

Constraint: $\langle \vec{\mathbf{x}}_1 | \vec{\mathbf{x}}_2 \rangle \geq \frac{1}{2}$



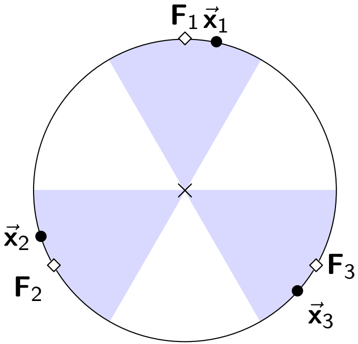
Filtering strategy for the k-sieve

Constraints: $\langle \vec{\mathbf{x}}_i | \vec{\mathbf{x}}_j \rangle \leq C_{ij}$



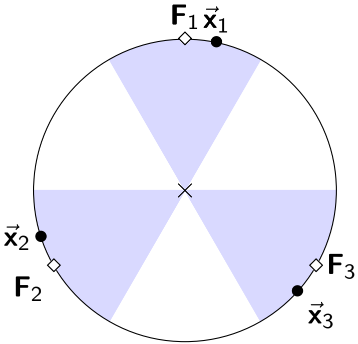
Filtering strategy for the k-sieve

Constraints: $\langle \vec{\mathbf{x}}_i | \vec{\mathbf{x}}_j \rangle \leq C_{ij}$



Filtering strategy for the k-sieve

Constraints: $\langle \vec{\mathbf{x}}_i | \vec{\mathbf{x}}_j \rangle \leq C_{ij}$



k-Random Product Code

A k-RPC $\mathfrak{C}$ is a code such that

$$\forall \mathbf{F}_1 \in \mathfrak{C}, \exists \mathbf{F}_2, \ldots, \mathbf{F}_k \in \mathfrak{C} \text{ st. } \sum_{i=1}^{k} \mathbf{F}_i = \vec{0}.$$

Framework for the k-sieve

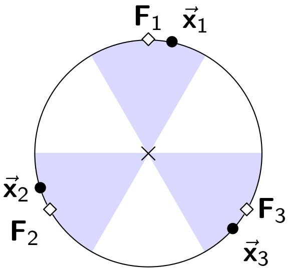
Framework

k-sieve framework to solve SVP

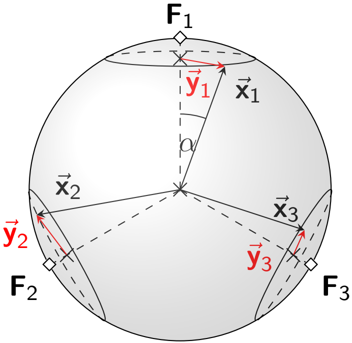
Input: list L of N lattice vectors, parameters k, angle α, configuration C

Output: list L_{out} of N reduced lattice vectors

1. Generate the tuple-filters. **Prefilter** L: for each $\vec{\mathbf{x}} \in L$, add $\vec{\mathbf{x}}$ to its nearest (unique) filter.
2. For each tuple-filter: **Find all solutions** satisfying C within the tuple-filter.
3. Repeat 1. and 2. until $|L_{out}| = N$.



Residual vectors



Search for a tuple $(\vec{\mathbf{x}}_1, \ldots, \vec{\mathbf{x}}_k)$ satisfying configuration C	$\Leftrightarrow$	Search for their residual vectors $(\vec{\mathbf{y}}_1, \ldots, \vec{\mathbf{y}}_k)$ satisfying configuration $C'_{C,\alpha}$

Framework

k-sieve framework to solve SVP

Input: list L of N lattice vectors, parameters k, angle α, configuration C
Output: list L_{out} of N reduced lattice vectors

1. Generate the filters. **Prefilter** L: for each $\vec{\mathbf{x}} \in L$, add $\vec{\mathbf{x}}$ to its nearest (unique) filter.
2. For each tuple-filter: **Find all solutions** satisfying C within the tuple-filter.
3. Repeat 1. and 2. until $|L_{out}| = N$.

Framework

k-sieve framework to solve SVP

Input: list L of N lattice vectors, parameters k, angle α, configuration C
Output: list L_{out} of N reduced lattice vectors

1. Generate the filters. **Prefilter** L: for each $\vec{\mathbf{x}} \in L$, add $\vec{\mathbf{x}}$ to its nearest (unique) filter.
2. For each tuple-filter: **Find all solutions** satisfying C within the tuple-filter.
3. Repeat 1. and 2. until $|L_{out}| = N$.

Subroutine **Find All Solutions** within a tuple-filter

Input: lists $L_1, \dots, L_k$ of residual vectors, configuration C'.
Output: the list of all tuples $(\vec{\mathbf{y}}_1, \dots, \vec{\mathbf{y}}_k) \in L_1 \times \dots \times L_k$ that satisfy C'.

Framework

k-sieve framework to solve SVP

Input: list L of N lattice vectors, parameters k, angle α, configuration C
Output: list L_{out} of N reduced lattice vectors

1. Generate the filters. **Prefilter** L: for each $\vec{\mathbf{x}} \in L$, add $\vec{\mathbf{x}}$ to its nearest (unique) filter.
2. For each tuple-filter: **Find all solutions** satisfying C within the tuple-filter.
3. Repeat 1. and 2. until $|L_{out}| = N$.

Subroutine Find All Solutions within a tuple-filter

Input: lists $L_1, \ldots, L_k$ of residual vectors, configuration C'.
Output: the list of all tuples $(\vec{\mathbf{y}}_1, \ldots, \vec{\mathbf{y}}_k) \in L_1 \times \ldots \times L_k$ that satisfy C'.

$$T(k\text{-sieve}) := \left(|L|_C + NbFilters_\alpha \cdot T(\mathbf{FAS}_{C'_{C,\alpha}})\right) \cdot NbRep_{C,\alpha}$$

Subroutine "Find All Solutions"

For $k = 2$:

- 2-sieve via quantum random walks [CL21, BCSS23]

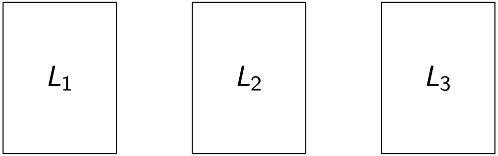
$k = 3$:

- Classic 3-sieve
- Quantum 3-sieve

$k = 4$:

- Classic 4-sieve
- Quantum 4-sieve

Classic 3-sieve — Subroutine



Configuration problem

Input: Lists L_1, L_2, L_3, configuration C'
Output: All the tuples $(\vec{\mathbf{y}}_1, \vec{\mathbf{y}}_2, \vec{\mathbf{y}}_3) \in L_1 \times L_2 \times L_3$ satisfying configuration C'

$(\vec{\mathbf{y}}_1, \vec{\mathbf{y}}_2, \vec{\mathbf{y}}_3)$ satisfies C'

$$\Leftrightarrow \begin{cases} \langle \vec{\mathbf{y}}_1 | \vec{\mathbf{y}}_2 \rangle & \leq C'_{12} \\ \langle \vec{\mathbf{y}}_1 | \vec{\mathbf{y}}_3 \rangle & \leq C'_{13} \\ \langle \vec{\mathbf{y}}_2 | \vec{\mathbf{y}}_3 \rangle & \leq C'_{23} \end{cases}$$

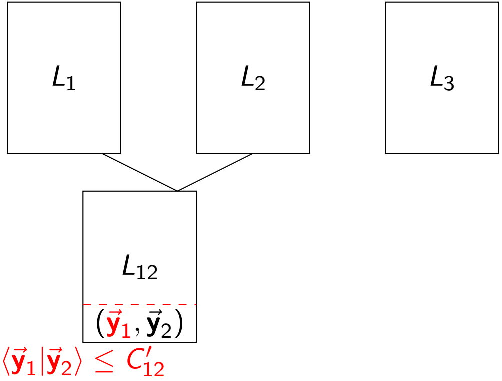
Classic 3-sieve — Subroutine

Configuration problem

Input: Lists L_1, L_2, L_3, configuration C'
Output: All the tuples $(\vec{\mathbf{y}}_1, \vec{\mathbf{y}}_2, \vec{\mathbf{y}}_3) \in L_1 \times L_2 \times L_3$ satisfying configuration C'

$(\vec{\mathbf{y}}_1, \vec{\mathbf{y}}_2, \vec{\mathbf{y}}_3)$ satisfies C'

$$\Leftrightarrow \begin{cases} \langle \vec{\mathbf{y}}_1 | \vec{\mathbf{y}}_2 \rangle & \leq C'_{12} \\ \langle \vec{\mathbf{y}}_1 | \vec{\mathbf{y}}_3 \rangle & \leq C'_{13} \\ \langle \vec{\mathbf{y}}_2 | \vec{\mathbf{y}}_3 \rangle & \leq C'_{23} \end{cases}$$



Classic 3-sieve — Subroutine

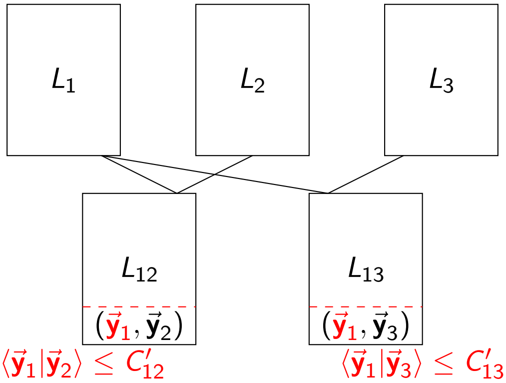
Configuration problem

Input: Lists L_1, L_2, L_3, configuration C'

Output: All the tuples $(\vec{\mathbf{y}}_1, \vec{\mathbf{y}}_2, \vec{\mathbf{y}}_3) \in L_1 \times L_2 \times L_3$ satisfying configuration C'

$(\vec{\mathbf{y}}_1, \vec{\mathbf{y}}_2, \vec{\mathbf{y}}_3)$ satisfies C'

$$\Leftrightarrow \begin{cases} \langle \vec{\mathbf{y}}_1 | \vec{\mathbf{y}}_2 \rangle & \leq C'_{12} \\ \langle \vec{\mathbf{y}}_1 | \vec{\mathbf{y}}_3 \rangle & \leq C'_{13} \\ \langle \vec{\mathbf{y}}_2 | \vec{\mathbf{y}}_3 \rangle & \leq C'_{23} \end{cases}$$



Classic 3-sieve — Subroutine

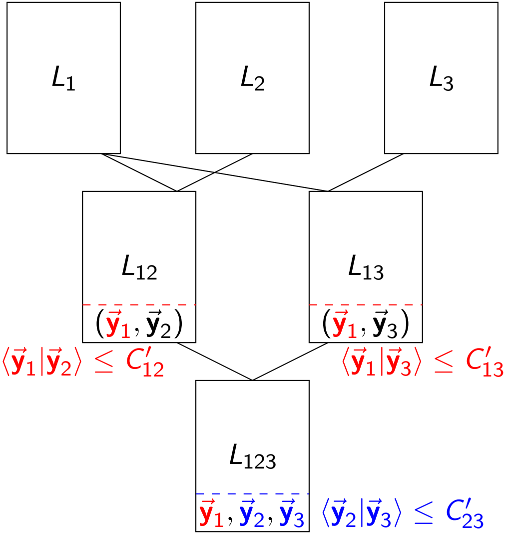
Configuration problem

Input: Lists L_1, L_2, L_3, configuration C'

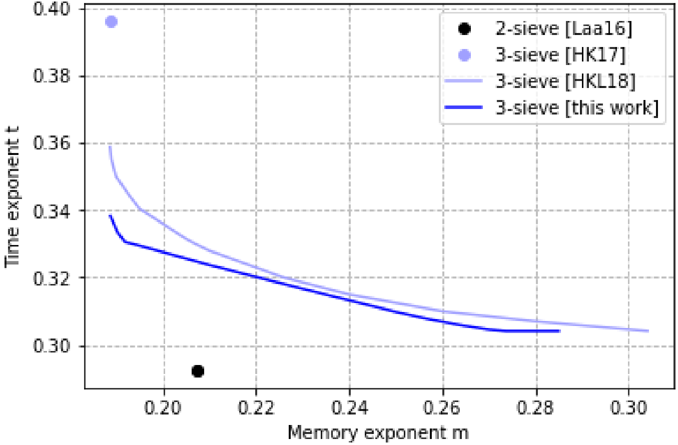
Output: All the tuples $(\vec{\mathbf{y}}_1, \vec{\mathbf{y}}_2, \vec{\mathbf{y}}_3) \in L_1 \times L_2 \times L_3$ satisfying configuration C'

$(\vec{\mathbf{y}}_1, \vec{\mathbf{y}}_2, \vec{\mathbf{y}}_3)$ satisfies C'

$$\Leftrightarrow \begin{cases} \langle \vec{\mathbf{y}}_1 | \vec{\mathbf{y}}_2 \rangle & \leq C'_{12} \\ \langle \vec{\mathbf{y}}_1 | \vec{\mathbf{y}}_3 \rangle & \leq C'_{13} \\ \langle \vec{\mathbf{y}}_2 | \vec{\mathbf{y}}_3 \rangle & \leq C'_{23} \end{cases}$$



Classic 3-sieve

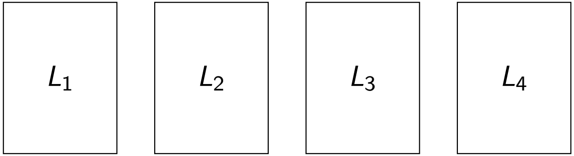


Classic 4-sieve — Subroutine

Configuration problem

Input: Lists L_1, L_2, L_3, L_4, configuration C'

Output: All the tuples $(\vec{\mathbf{y}}_1, \vec{\mathbf{y}}_2, \vec{\mathbf{y}}_3, \vec{\mathbf{y}}_4) \in L_1 \times L_2 \times L_3 \times L_4$ satisfying configuration C'



$(\vec{\mathbf{y}}_1, \vec{\mathbf{y}}_2, \vec{\mathbf{y}}_3, \vec{\mathbf{y}}_4)$ satisfies C'

$$\Leftrightarrow \begin{cases} \langle \vec{\mathbf{y}}_1 | \vec{\mathbf{y}}_2 \rangle & \leq C'_{12} \\ \langle \vec{\mathbf{y}}_1 | \vec{\mathbf{y}}_3 \rangle & \leq C'_{13} \\ \langle \vec{\mathbf{y}}_1 | \vec{\mathbf{y}}_4 \rangle & \leq C'_{14} \\ \langle \vec{\mathbf{y}}_2 | \vec{\mathbf{y}}_3 \rangle & \leq C'_{23} \\ \langle \vec{\mathbf{y}}_2 | \vec{\mathbf{y}}_4 \rangle & \leq C'_{24} \\ \langle \vec{\mathbf{y}}_3 | \vec{\mathbf{y}}_4 \rangle & \leq C'_{34} \end{cases}$$

Classic 4-sieve — Subroutine

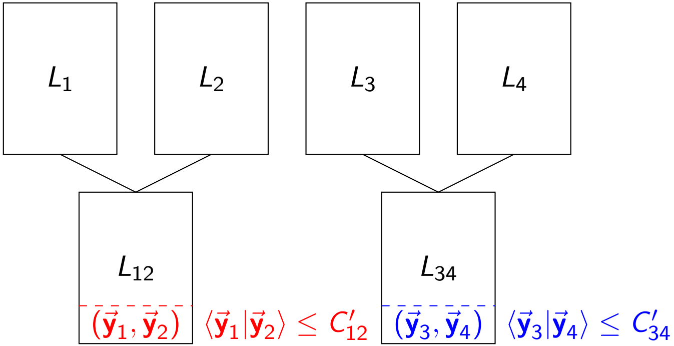
Configuration problem

Input: Lists L_1, L_2, L_3, L_4, configuration C'

Output: All the tuples $(\vec{\mathbf{y}}_1, \vec{\mathbf{y}}_2, \vec{\mathbf{y}}_3, \vec{\mathbf{y}}_4) \in L_1 \times L_2 \times L_3 \times L_4$ satisfying configuration C'

$(\vec{\mathbf{y}}_1, \vec{\mathbf{y}}_2, \vec{\mathbf{y}}_3, \vec{\mathbf{y}}_4)$ satisfies C'

$$\Leftrightarrow \begin{cases} \langle \vec{\mathbf{y}}_1 | \vec{\mathbf{y}}_2 \rangle & \leq C'_{12} \\ \langle \vec{\mathbf{y}}_1 | \vec{\mathbf{y}}_3 \rangle & \leq C'_{13} \\ \langle \vec{\mathbf{y}}_1 | \vec{\mathbf{y}}_4 \rangle & \leq C'_{14} \\ \langle \vec{\mathbf{y}}_2 | \vec{\mathbf{y}}_3 \rangle & \leq C'_{23} \\ \langle \vec{\mathbf{y}}_2 | \vec{\mathbf{y}}_4 \rangle & \leq C'_{24} \\ \langle \vec{\mathbf{y}}_3 | \vec{\mathbf{y}}_4 \rangle & \leq C'_{34} \end{cases}$$



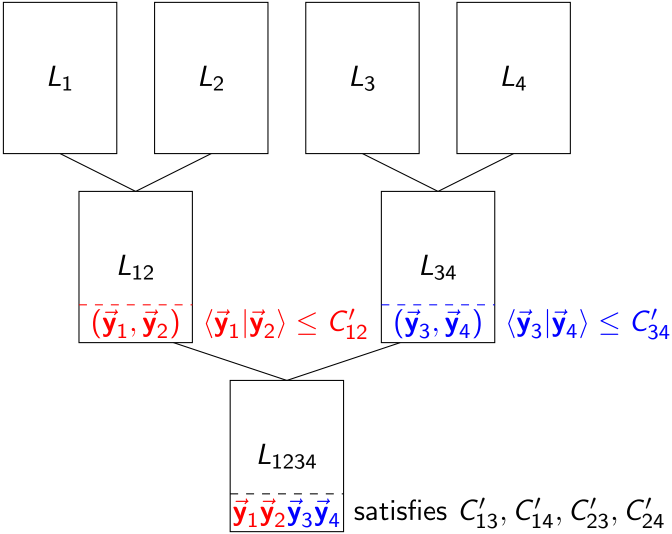
Classic 4-sieve — Subroutine

Configuration problem

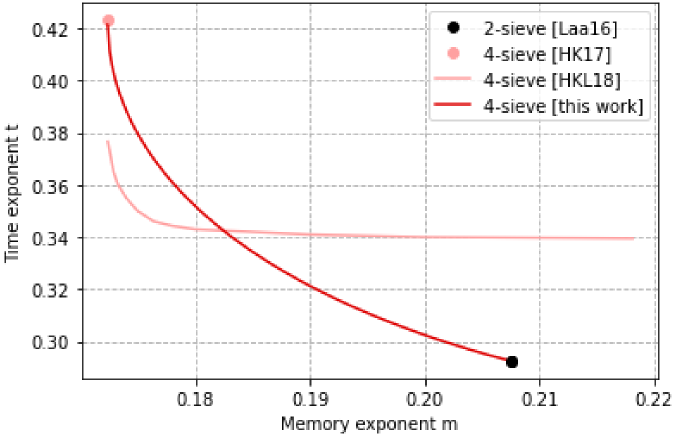
Input: Lists L_1, L_2, L_3, L_4, configuration C'
Output: All the tuples $(\vec{\mathbf{y}}_1, \vec{\mathbf{y}}_2, \vec{\mathbf{y}}_3, \vec{\mathbf{y}}_4) \in L_1 \times L_2 \times L_3 \times L_4$ satisfying configuration C'

$(\vec{\mathbf{y}}_1, \vec{\mathbf{y}}_2, \vec{\mathbf{y}}_3, \vec{\mathbf{y}}_4)$ satisfies C'

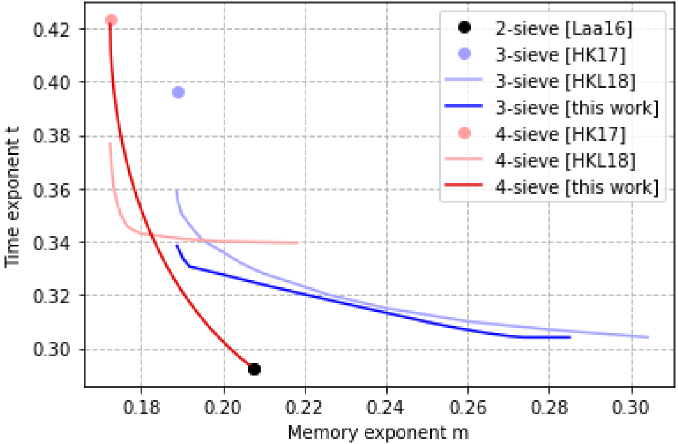
$$\Leftrightarrow \begin{cases} \langle \vec{\mathbf{y}}_1 | \vec{\mathbf{y}}_2 \rangle & \leq C'_{12} \\ \langle \vec{\mathbf{y}}_1 | \vec{\mathbf{y}}_3 \rangle & \leq C'_{13} \\ \langle \vec{\mathbf{y}}_1 | \vec{\mathbf{y}}_4 \rangle & \leq C'_{14} \\ \langle \vec{\mathbf{y}}_2 | \vec{\mathbf{y}}_3 \rangle & \leq C'_{23} \\ \langle \vec{\mathbf{y}}_2 | \vec{\mathbf{y}}_4 \rangle & \leq C'_{24} \\ \langle \vec{\mathbf{y}}_3 | \vec{\mathbf{y}}_4 \rangle & \leq C'_{34} \end{cases}$$



Classic 4-sieve



Classic k-sieves



Quantum 3-sieve — Subroutine

$$|\psi_{L_1}\rangle \qquad |\psi_{L_2}\rangle \qquad |\psi_{L_3}\rangle$$

Quantum 3-sieve — Subroutine

$$|\psi_{L_1}\rangle \qquad |\psi_{L_2}\rangle \qquad |\psi_{L_3}\rangle$$

$$\|$$

$$\frac{1}{\sqrt{|L_1|}} \sum_{\vec{\mathbf{y}}_1 \in L_1} |\mathrm{i}_{\vec{\mathbf{y}}_1}\rangle |\vec{\mathbf{y}}_1\rangle$$

Quantum 3-sieve — Subroutine

$$|\psi_{L_1}\rangle \qquad |\psi_{L_2}\rangle \qquad |\psi_{L_3}\rangle$$

$$\| \qquad \Big\downarrow \text{Grover}$$

$$\frac{1}{\sqrt{|L_1|}} \sum_{\vec{\mathbf{y}}_1 \in L_1} |\mathsf{i}_{\vec{\mathbf{y}}_1}\rangle |\vec{\mathbf{y}}_1\rangle \qquad |\psi_{L_2(\vec{\mathbf{y}}_1)}\rangle$$

$$\langle \vec{\mathbf{y}}_1 | \vec{\mathbf{y}}_2 \rangle \leq C'_{12}$$

Quantum 3-sieve — Subroutine

$$
\begin{array}{ccc}
|\psi_{L_1}\rangle & |\psi_{L_2}\rangle & |\psi_{L_3}\rangle \\
\| & \big\downarrow \text{Grover} & \big\downarrow \text{Grover} \\
\frac{1}{\sqrt{|L_1|}} \sum_{\vec{\mathbf{y}}_1 \in L_1} |\mathrm{i}_{\vec{\mathbf{y}}_1}\rangle |\vec{\mathbf{y}}_1\rangle & |\psi_{L_2(\vec{\mathbf{y}}_1)}\rangle & |\psi_{L_3(\vec{\mathbf{y}}_1)}\rangle
\end{array}
$$

$$\langle \vec{\mathbf{y}}_1 | \vec{\mathbf{y}}_2 \rangle \leq C'_{12}$$
$$\langle \vec{\mathbf{y}}_1 | \vec{\mathbf{y}}_3 \rangle \leq C'_{13}$$

Quantum 3-sieve — Subroutine

$$|\psi_{L_1}\rangle \qquad |\psi_{L_2}\rangle \qquad |\psi_{L_3}\rangle$$

$$\| \qquad \Big\downarrow \text{Grover} \qquad \Big\downarrow \text{Grover}$$

$$\frac{1}{\sqrt{|L_1|}} \sum_{\vec{\mathbf{y}}_1 \in L_1} |\mathrm{i}_{\vec{\mathbf{y}}_1}\rangle |\vec{\mathbf{y}}_1\rangle \qquad |\psi_{L_2(\vec{\mathbf{y}}_1)}\rangle \qquad |\psi_{L_3(\vec{\mathbf{y}}_1)}\rangle$$

$$\langle \vec{\mathbf{y}}_1 | \vec{\mathbf{y}}_2 \rangle \le C'_{12}$$
$$\langle \vec{\mathbf{y}}_1 | \vec{\mathbf{y}}_3 \rangle \le C'_{13}$$

$$\|$$

$$\frac{1}{\sqrt{|L_2(\vec{\mathbf{y}}_1)|}} \sum_{\vec{\mathbf{y}}_2 \in L_2(\vec{\mathbf{y}}_1)} |\mathrm{i}_{\vec{\mathbf{y}}_2}\rangle |\vec{\mathbf{y}}_2\rangle$$

Quantum 3-sieve — Subroutine

$$|\psi_{L_1}\rangle \qquad |\psi_{L_2}\rangle \qquad |\psi_{L_3}\rangle$$

$$\| \qquad \downarrow \text{Grover} \qquad \downarrow \text{Grover}$$

$$\frac{1}{\sqrt{|L_1|}} \sum_{\vec{\mathbf{y}}_1 \in L_1} |i_{\vec{\mathbf{y}}_1}\rangle |\vec{\mathbf{y}}_1\rangle \qquad |\psi_{L_2(\vec{\mathbf{y}}_1)}\rangle \qquad |\psi_{L_3(\vec{\mathbf{y}}_1)}\rangle \qquad \begin{matrix}\langle \vec{\mathbf{y}}_1 | \vec{\mathbf{y}}_2 \rangle \leq C'_{12} \\ \langle \vec{\mathbf{y}}_1 | \vec{\mathbf{y}}_3 \rangle \leq C'_{13}\end{matrix}$$

$$\| \qquad \downarrow \text{Grover}$$

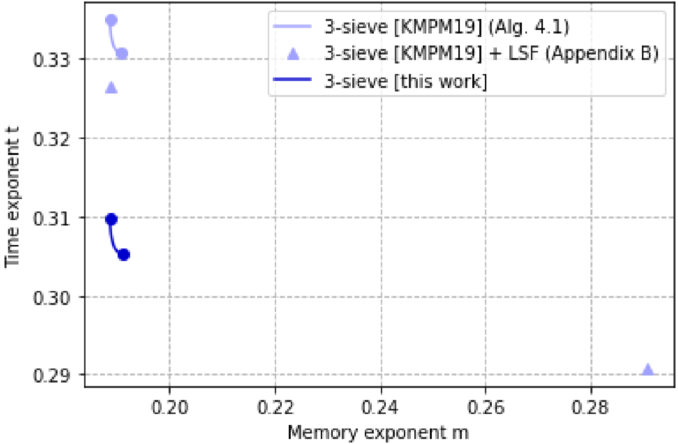
$$\frac{1}{\sqrt{|L_2(\vec{\mathbf{y}}_1)|}} \sum_{\vec{\mathbf{y}}_2 \in L_2(\vec{\mathbf{y}}_1)} |i_{\vec{\mathbf{y}}_2}\rangle |\vec{\mathbf{y}}_2\rangle \qquad |\psi_{L_3(\vec{\mathbf{y}}_1, \vec{\mathbf{y}}_2)}\rangle \qquad \langle \vec{\mathbf{y}}_2 | \vec{\mathbf{y}}_3 \rangle \leq C'_{23}$$

Quantum 3-sieve — Subroutine

$$|\psi_{L_1}\rangle|\psi_{L_2(\vec{\mathbf{y}}_1)}\rangle|\psi_{L_3(\vec{\mathbf{y}}_1,\vec{\mathbf{y}}_2)}\rangle$$

- Apply amplitude amplification
- Measure and get a reducing $(\vec{\mathbf{y}}_1, \vec{\mathbf{y}}_2, \vec{\mathbf{y}}_3)$
- Repeat to find all the solutions in $L_1 \times L_2 \times L_3$

Quantum 3-sieve

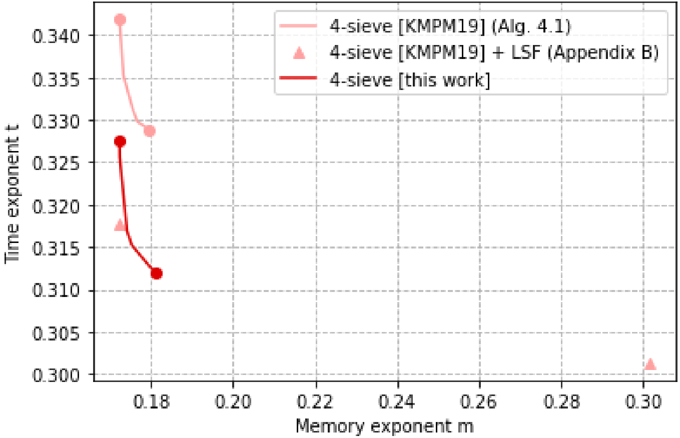


Quantum 4-sieve — Subroutine

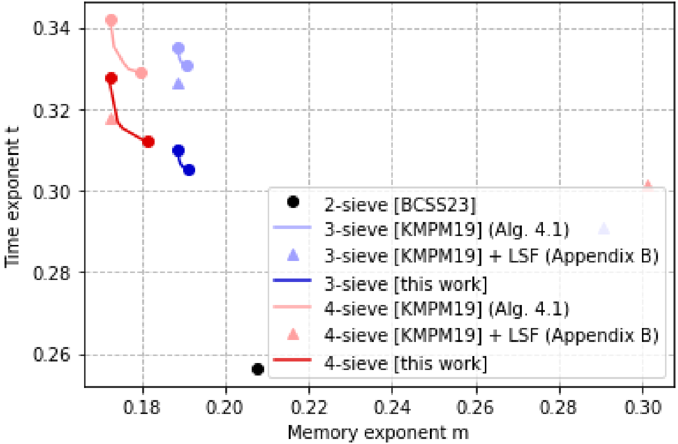
$$|\psi_{L_1}\rangle|\psi_{L_2(\vec{\mathbf{y}}_1)}\rangle|\psi_{L_3(\vec{\mathbf{y}}_1,\vec{\mathbf{y}}_2)}\rangle|\psi_{L_4(\vec{\mathbf{y}}_1,\vec{\mathbf{y}}_2)}\rangle$$

- Apply amplitude amplification
- Measure and get a reducing $(\vec{\mathbf{y}}_1, \vec{\mathbf{y}}_2, \vec{\mathbf{y}}_3, \vec{\mathbf{y}}_4)$
- Repeat to find all the solutions in $L_1 \times L_2 \times L_3 \times L_4$

Quantum 4-sieve



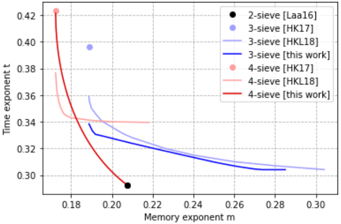
Quantum k-sieves

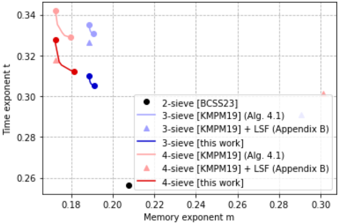


Conclusion

This work:

- Improves the 3-sieves trade-off
- New trade-offs for the 4-sieves

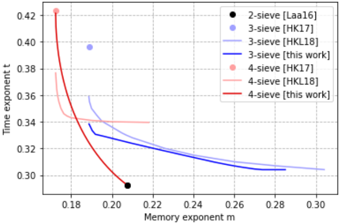


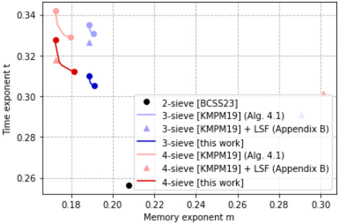


Conclusion

This work:

- Improves the 3-sieves trade-off
- New trade-offs for the 4-sieves





Further research:

- k-sieve for $k > 4$
- Mix our prefiltering with inner filtering as in [HKL18, KMPM19]
- **Classical**: Optimal merging trees
- **Quantum**: k-sieve via quantum random walks

Thank you for listening!
Any questions?

References

[NV08] P.Q. Nguyen and T. Vidick
Sieve algorithms for the shortest vector problem are practical
doi.org/10.1515/JMC.2008.009

[BDGL16] A. Becker, L. Ducas, N. Gama and T. Laarhoven
New directions in nearest neighbor searching with applications to lattice sieving
ePrint 2015/1128

[HKL18] G. Herold, E. Kirshanova and T. Laarhoven (2018)
Speed-ups and time–memory trade-offs for tuple lattice sieving
ePrint 2017/1228

[KMPM19] E. Kirshanova, E. Mårtensson, E.W. Postlethwaite and S.R. Moulik
Quantum algorithms for the approximate k-list problem and their application to lattice sieving
ePrint 2019/1016

[this work] A. Chailloux and J. Loyer
Classical and quantum 3 and 4-sieves to solve SVP with low memory
ePrint 2023/200

k-Random Product Code (Technical details)

Sample a RPC $\mathfrak{C} = Q \cdot (\mathfrak{C}_1 \times \cdots \times \mathfrak{C}_m)$

k-Random Product Code (Technical details)

Sample a RPC $\mathfrak{C} = Q \cdot (\mathfrak{C}_1 \times \cdots \times \mathfrak{C}_m)$

For each $F_1 \in \mathfrak{C}$:

$k = 3$

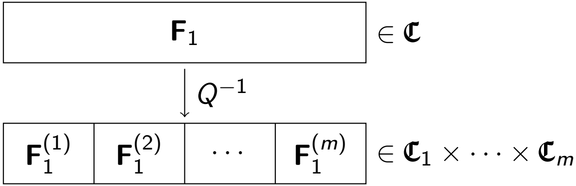
$\mathbf{F}_1$	$\in \mathfrak{C}$

k-Random Product Code (Technical details)

Sample a RPC $\mathfrak{C} = Q \cdot (\mathfrak{C}_1 \times \cdots \times \mathfrak{C}_m)$

For each $F_1 \in \mathfrak{C}$:

$k = 3$



k-Random Product Code (Technical details)

Sample a RPC $\mathfrak{C} = Q \cdot (\mathfrak{C}_1 \times \cdots \times \mathfrak{C}_m)$

For each $F_1 \in \mathfrak{C}$:

$k = 3$

$\mathbf{F}_1$

$\in \mathfrak{C}$

$\downarrow Q^{-1}$

$\mathbf{F}_1^{(1)}$	$\mathbf{F}_1^{(2)}$	$\cdots$	$\mathbf{F}_1^{(m)}$

$\in \mathfrak{C}_1 \times \cdots \times \mathfrak{C}_m$

Choose a random

$\mathbf{F}_2^{(1)}$	$\mathbf{F}_2^{(2)}$	$\cdots$	$\mathbf{F}_2^{(m)}$

such that for $i \in [m], \langle \mathbf{F}_1^{(i)} | \mathbf{F}_2^{(i)} \rangle = -\frac{1}{2m}$.

k-Random Product Code (Technical details)

Sample a RPC $\mathfrak{C} = Q \cdot (\mathfrak{C}_1 \times \cdots \times \mathfrak{C}_m)$

For each $F_1 \in \mathfrak{C}$:

$k = 3$

$\mathbf{F}_1$	$\in \mathfrak{C}$

$\downarrow Q^{-1}$

$\mathbf{F}_1^{(1)}$	$\mathbf{F}_1^{(2)}$	$\cdots$	$\mathbf{F}_1^{(m)}$	$\in \mathfrak{C}_1 \times \cdots \times \mathfrak{C}_m$

Choose a random	$\mathbf{F}_2^{(1)}$	$\mathbf{F}_2^{(2)}$	$\cdots$	$\mathbf{F}_2^{(m)}$	such that for $i \in [m]$, $\langle \mathbf{F}_1^{(i)} \vert \mathbf{F}_2^{(i)} \rangle = -\frac{1}{2m}$.

$\downarrow Q$

$\mathbf{F}_2$	with $\langle \mathbf{F}_1 \vert \mathbf{F}_2 \rangle = -\frac{1}{2}$.

Compute $\mathbf{F}_3 = -\mathbf{F}_1 - \mathbf{F}_2$.

k-Random Product Code (Technical details)

Sample a RPC $\mathfrak{C} = Q \cdot (\mathfrak{C}_1 \times \cdots \times \mathfrak{C}_m)$

For each $F_1 \in \mathfrak{C}$:

$\forall k$

$$\boxed{\mathbf{F}_1} \in \mathfrak{C}$$

$$\downarrow Q^{-1}$$

$$\boxed{\mathbf{F}_1^{(1)}}\boxed{\mathbf{F}_1^{(2)}}\boxed{\cdots}\boxed{\mathbf{F}_1^{(m)}} \in \mathfrak{C}_1 \times \cdots \times \mathfrak{C}_m$$

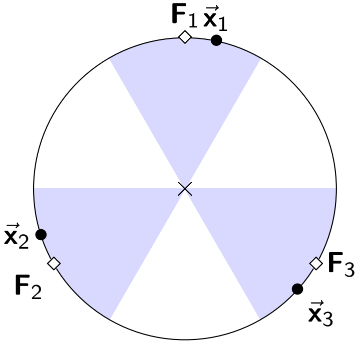
For $j = 2 \ldots k-1$, choose random $\boxed{\mathbf{F}_j^{(1)}}\boxed{\mathbf{F}_j^{(2)}}\boxed{\cdots}\boxed{\mathbf{F}_j^{(m)}}$ st. for $i \in [m], j' < j, \langle \mathbf{F}_{j'}^{(i)} | \mathbf{F}_j^{(i)} \rangle = -\frac{1}{(k-1)m}$.

$$\downarrow Q$$

$$\boxed{\mathbf{F}_j}$$ with $\langle \mathbf{F}_1 | \mathbf{F}_2 \rangle = -\frac{1}{k-1}$.

Compute $\mathbf{A}_k = -\sum_{j=1}^{k-1} \mathbf{F}_j$.

Filtering strategy for the k-sieve



Tuple-filter
$(\mathbf{F}_1, \mathbf{F}_2, \mathbf{F}_3)$

k-Random Product Code

A k-RPC $\mathfrak{C}$ is a code such that

$$\forall \mathbf{F}_1 \in \mathfrak{C}, \exists \mathbf{F}_2, ..., \mathbf{F}_k \in \mathfrak{C} \text{ st. } \sum_{i=1}^{k} \mathbf{F}_i = \vec{0}.$$

With an **efficient** decoding algorithm