Quantum Cryptanalysis on Lattices and Codes

Ph.D. defense

Johanna Loyer

Public-key cryptography

Cryptographic problem

Secret key $\xrightarrow{\text{easy}}$ **Public key**

Secret key $\xleftarrow[\text{hard}]{}$ **Public key**

Factorization problem

Large primes p, q $\xrightarrow{\text{easy}}$ $p \cdot q$

Large primes p, q $\xleftarrow[\text{hard}]{}$ $p \cdot q$

Public-key cryptography

Cryptographic problem

Secret key —easy→ **Public key**

Secret key ←hard— **Public key**

Factorization problem

Large primes p, q —easy→ $p \cdot q$

Large primes p, q ←~~hard~~— $p \cdot q$

easy
for quantum computers

[Sho94] Shor. Algorithms for Quantum Computation: Discrete Logarithms and Factoring

Leads for quantum-safe cryptography

Lattices

Codes

Multivariate polynomials

Isogenies

My contributions

Lattice-based cryptography:

- [CL21] Chailloux-**Loyer**. Lattice sieving via quantum random walks. (ASIACRYPT21)
- [CL23] Chailloux-**Loyer**. Classical and Quantum 3 and 4-Sieves to Solve SVP with Low Memory. (PQCrypto23)

Code-based cryptography:

- [Loy23] **Loyer**. Quantum security analysis of Wave. (Submitted)
- [Wave] Banegas-Carrier-Chailloux-Couvreur-Debris-Gaborit-Karpman-**Loyer**--Niederhagen-Sendrier-Smith-Tillich.
  (NIST submission to the post-quantum cryptography standardization)

Outline

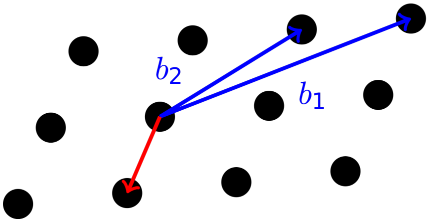
1. Lattice sieving
   - Shortest Vector Problem (SVP)
   - Sieving algorithms
   - Filtering
2. Sieving via quantum walks
   - New framework
   - Quantum walk
   - Complexity results
3. k-sieves with lower memory
4. Wave quantum security

Lattice

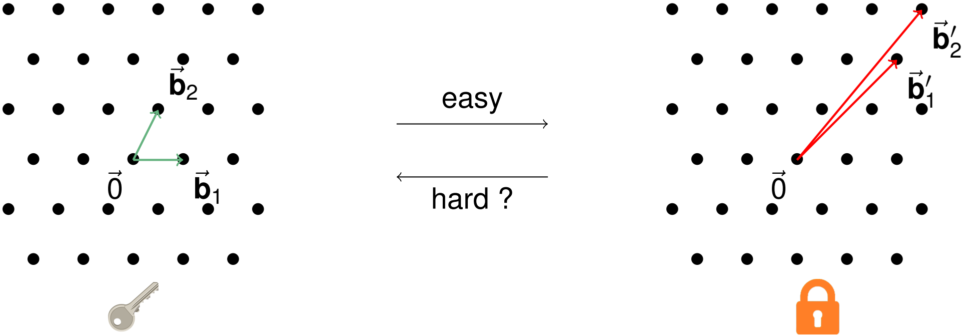
Given a basis $B = (\vec{b_1}, ..., \vec{b_d})$, the lattice $\mathcal{L}$ generated by B is the set of all integer linear combinations of its basis vectors: $\mathcal{L}(B) = \left\{ \sum_{i=1}^{d} z_i \vec{b_i},\ z_i \in \mathbb{Z} \right\}$.

Shortest Vector Problem (SVP)

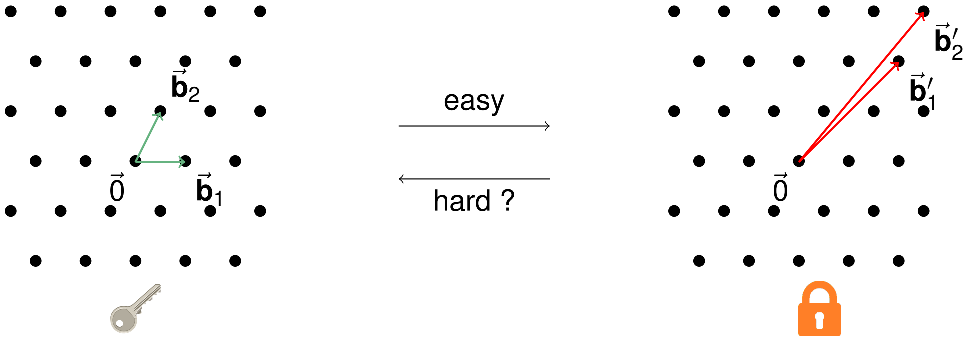
Given a lattice $\mathcal{L}$, find the shortest non-zero vector $\vec{v} \in \mathcal{L}$.



Lattice-based cryptography



Lattice-based cryptography



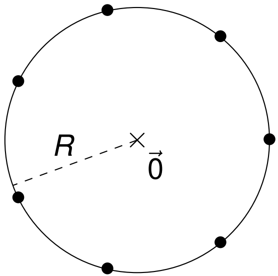
SVP → Lattice basis reduction BKZ → Lattice problems LWE, SIS, NTRU → **Break** Kyber, Dilithium, Falcon...

Outline

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 $\gamma R < R$.

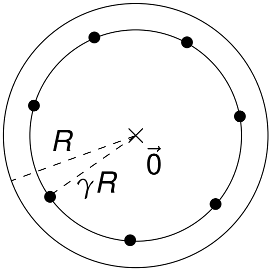
Initialization:
Generate N lattice vectors of norm $\lesssim R$ (large) by Klein's algorithm



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 $\gamma R < R$.

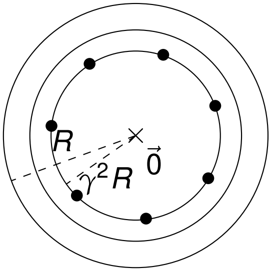
After 1 iteration:
vectors of norm at most γR



Sieving 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 $\gamma^2 R$

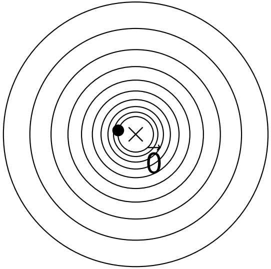


Sieving step

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 $\gamma^{\text{poly}(d)} R$.

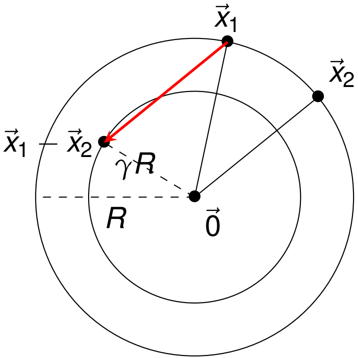
Short vector found!



Nguyen-Vidick sieving step [NV08]

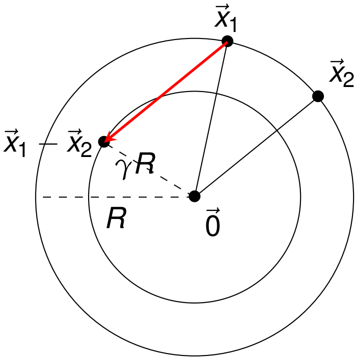
for $\vec{\boldsymbol{x}}_1, \vec{\boldsymbol{x}}_2 \in L$:

  if $\|\vec{\boldsymbol{x}}_1 - \vec{\boldsymbol{x}}_2\| \leqslant \gamma R$ **then** add $\vec{\boldsymbol{x}}_1 - \vec{\boldsymbol{x}}_2$ to L_{out}



Nguyen-Vidick sieving step [NV08]

for $\vec{x}_1, \vec{x}_2 \in L$:
  if $\|\vec{x}_1 - \vec{x}_2\| \leqslant \gamma R$ **then** add $\vec{x}_1 - \vec{x}_2$ to L_{out}



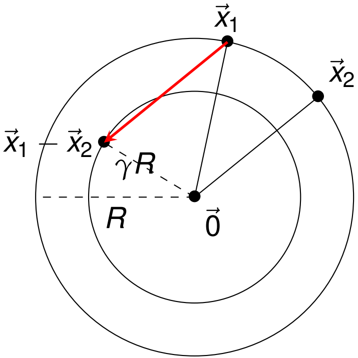
Minimal list size such that $|L| = |L_{out}| = N$:

$$\underbrace{N^2 \cdot Pr_{\vec{\mathbf{x}}_1,\vec{\mathbf{x}}_2}\left[\|\vec{\mathbf{x}}_1 - \vec{\mathbf{x}}_2\| \leq \gamma R\right]}_{\text{Number of reducing pairs}} = \underbrace{N}_{\text{Output points}}$$

$$\Rightarrow N = 2^{0.208d + o(d)}$$

Nguyen-Vidick sieving step [NV08]

for $\vec{x}_1, \vec{x}_2 \in L$:
  if $\|\vec{x}_1 - \vec{x}_2\| \leqslant \gamma R$ **then** add $\vec{x}_1 - \vec{x}_2$ to L_{out}



Minimal list size such that $|L| = |L_{out}| = N$:

$$\underbrace{N^2 \cdot Pr_{\vec{x}_1,\vec{x}_2}\left[\|\vec{x}_1 - \vec{x}_2\| \leq \gamma R\right]}_{\text{Number of reducing pairs}} = \underbrace{N}_{\text{Output points}}$$

$$\Rightarrow N = 2^{0.208d + o(d)}$$

Complexity:

- Time: $\text{poly}(d) \cdot N^2 = 2^{0.415d + o(d)}$
- Memory: $\text{poly}(d) \cdot N = 2^{0.208d + o(d)}$

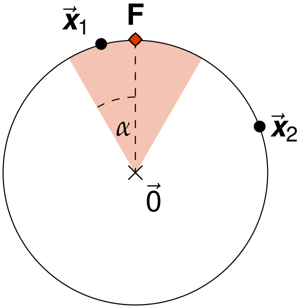
Outline

Locality Sensitive Filtering (LSF)

Main idea: Only check the near vectors ▶ Check vectors near to a same point.

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

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

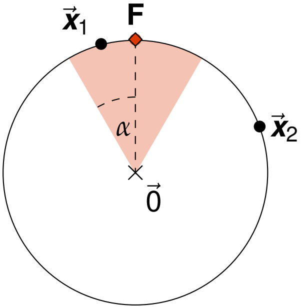


Locality Sensitive Filtering (LSF)

Main idea: Only check the near vectors ▶ Check vectors near to a same point.

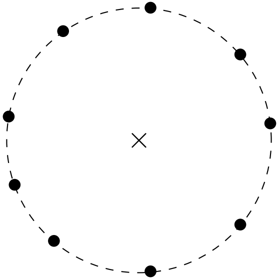
A **filter** of center **F** $\in \mathbb{R}^d$ and angle $\alpha \in [0, \frac{\pi}{2}]$ maps a vector $\vec{\mathbf{x}}$ to a boolean value:

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

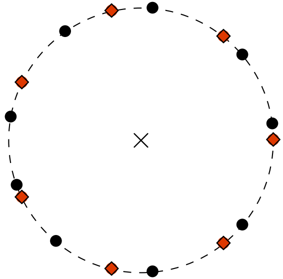


Associated with a set "bucket"

NV-sieve with filtering

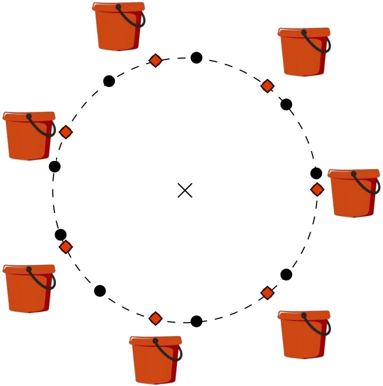


NV-sieve with filtering



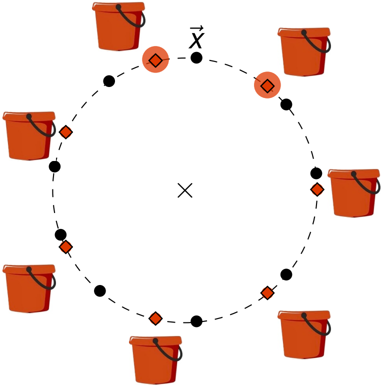
- Generate the filters ◆

NV-sieve with filtering



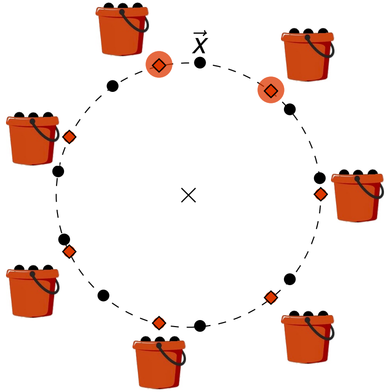
- Generate the filters ◆

NV-sieve with filtering



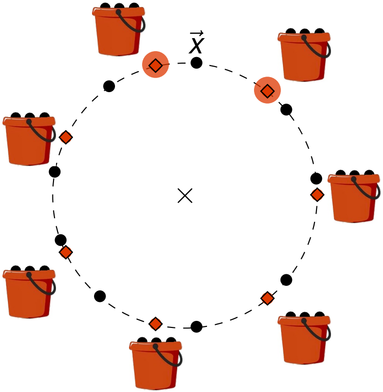
- Generate the filters ◆
- For each vector: add it to its nearest buckets.

NV-sieve with filtering



- Generate the filters ◆
- For each vector: add it to its nearest buckets.

NV-sieve with filtering



- Generate the filters ◆
- For each vector: add it to its nearest buckets.
- For each vector: search for a reducing one within its buckets.

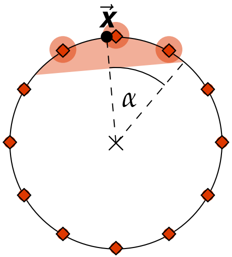
Random Product Code (RPC)

$$C = Q \cdot (C_1 \times \cdots \times C_m) \subset \mathbb{R}^d$$

- $C_1, ..., C_m$: sets of B vectors in $\mathbb{R}^{d/m}$ unif. & indep. random of norm $\sqrt{\frac{1}{m}}$
- Q uniformly random rotation over $\mathbb{R}^d$

- Points uniformly distributed over the sphere
- Efficient list decoding algorithm (subexponential or polynomial time)

1 codeword ◆ = 1 filter center



NV-sieve with filtering

- Generate the filters
- For each vector: add it to its nearest buckets
- For each vector: search for a reducing one within its buckets
  - Classically or by Grover's search

Memory complexity: $2^{0.208d+o(d)}$

Time complexity:

Classical NV-sieve: $2^{0.415d+o(d)}$
With filtering[1]: $2^{0.292d+o(d)}$

Quantum NV-sieve: $2^{0.311d+o(d)}$
With filtering[2]: $2^{\mathbf{0.265}d+o(d)}$

[1] [BDGL16] Becker-Ducas-Gama-Laarhoven. New directions in nearest neighbor searching with applications to lattice sieving.

[2] [Laa16] Laarhoven. Search problems in cryptography: from fingerprinting to lattice sieving. (PhD)

Outline

1. Lattice sieving
   - Shortest Vector Problem (SVP)
   - Sieving algorithms
   - Filtering
2. Sieving via quantum walks
   - New framework
   - Quantum walk
   - Complexity results
3. k-sieves with lower memory
4. Wave quantum security

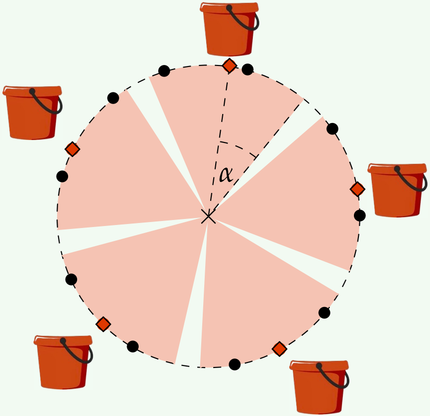
Our framework algorithm

Sieving step using quantum walks

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

Main idea: Replace Grover's search with a quantum walk.

Step 1 - Partitioning the sphere



Step 2 - Pairs finding

For each  :

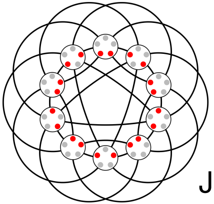
Find all the reducing pairs within  by **quantum walks**.

Quantum Walk

Input: a graph $G = (V, E)$, function $f : V \to \{0, 1\}$.
Output: a “marked" vertex $v \in V$ such that $f(v) = 1$.

Function: For vertex $v \subseteq$ , $f(v) = \begin{cases} 1 & \text{if } v \text{ contains a reducing pair,} \\ 0 & \text{otherwise.} \end{cases}$

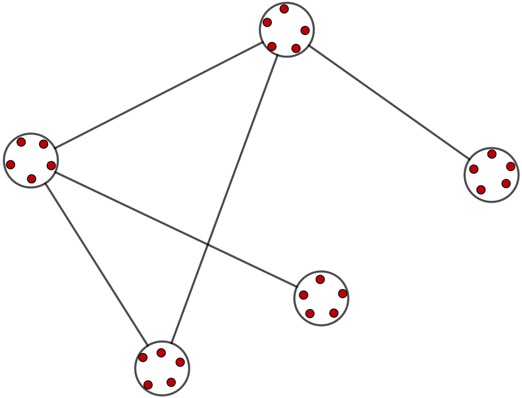
Johnson graph $J(\text{Size}$$, \text{Size}_v)$:



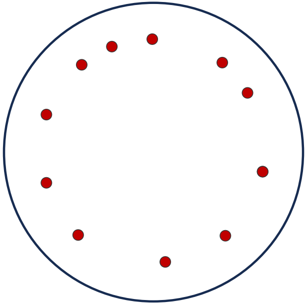
Johnson graph $J(5, 2)$

Quantum walk subroutine

Goal: Find 1 reducing pair in 🪣

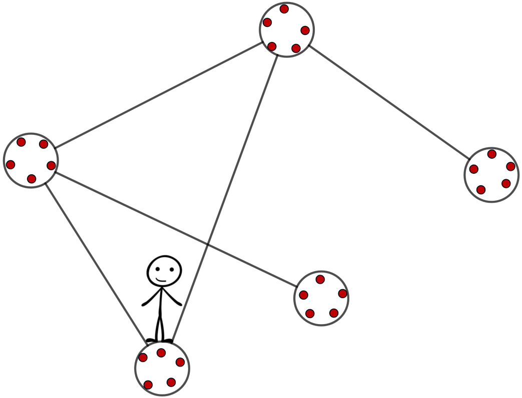


🔍 Zoom on the current vertex

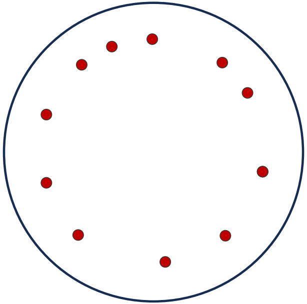


Quantum walk subroutine

Goal: Find 1 reducing pair in 

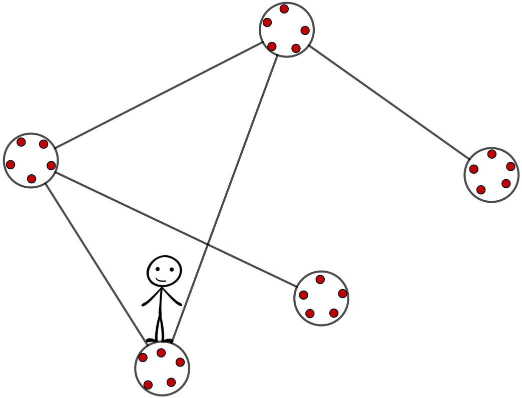


Zoom on the current vertex

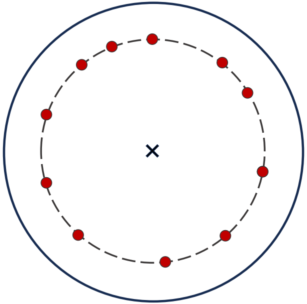


Quantum walk subroutine

Goal: Find 1 reducing pair in 🪣

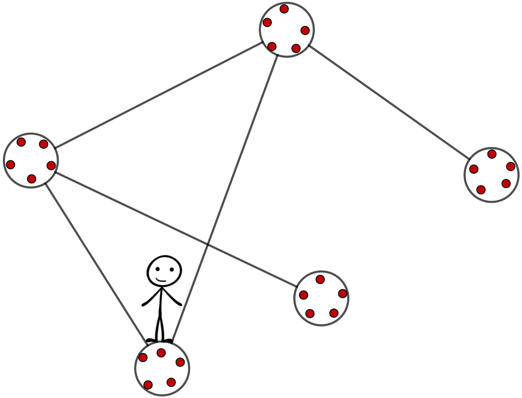


🔍 Zoom on the current vertex

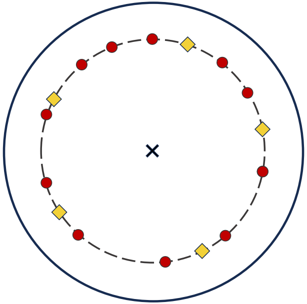


Quantum walk subroutine

Goal: Find 1 reducing pair in

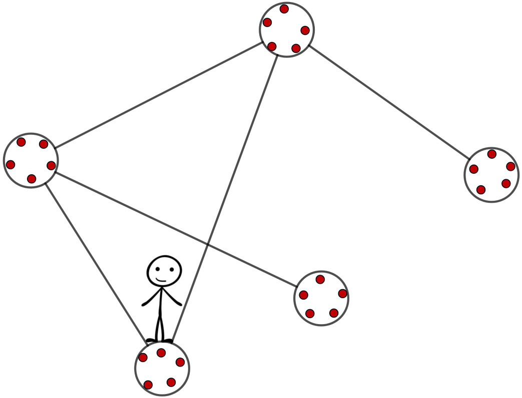


🔍 Zoom on the current vertex

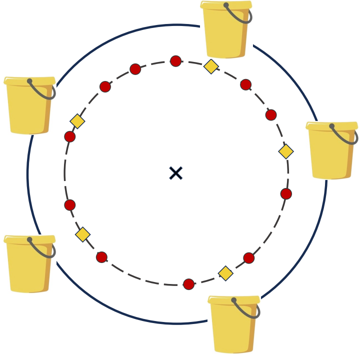


Quantum walk subroutine

Goal: Find 1 reducing pair in 🪣

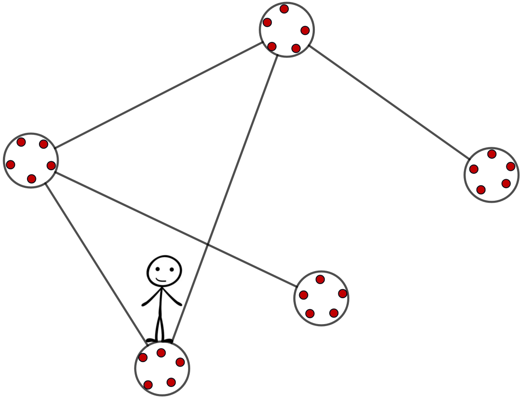


Zoom on the current vertex

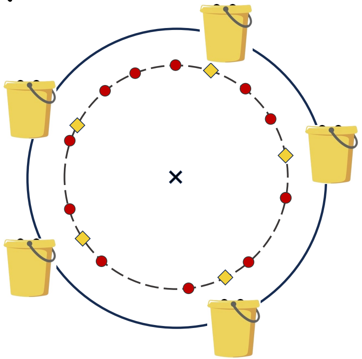


Quantum walk subroutine

Goal: Find 1 reducing pair in 

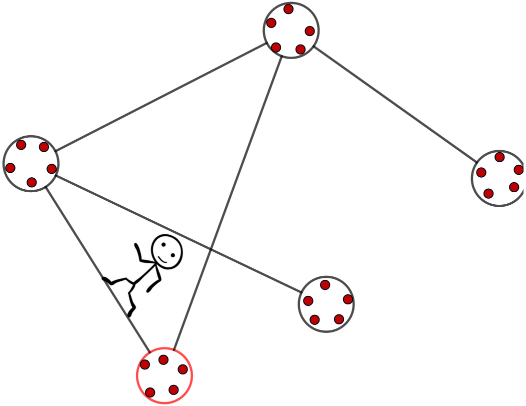


Zoom on the current vertex

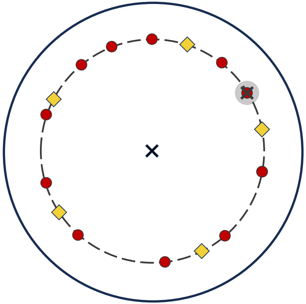


Quantum walk subroutine

Goal: Find 1 reducing pair in

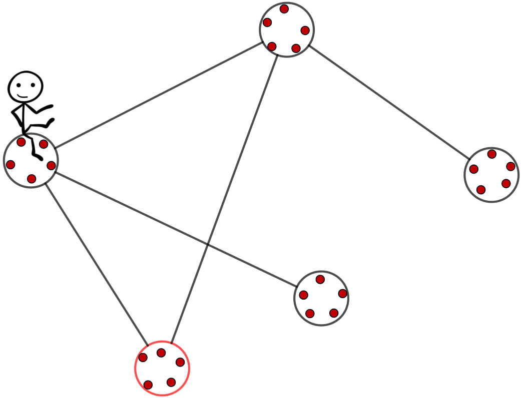


Zoom on the current vertex

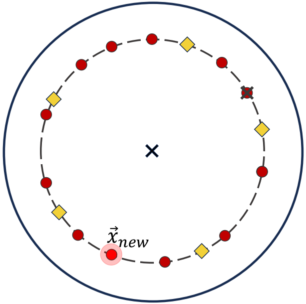


Quantum walk subroutine

Goal: Find 1 reducing pair in

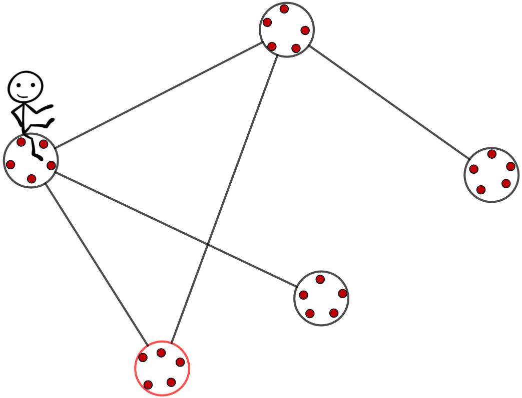


Zoom on the current vertex

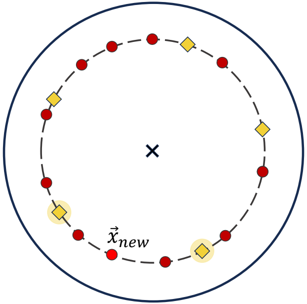


Quantum walk subroutine

Goal: Find 1 reducing pair in

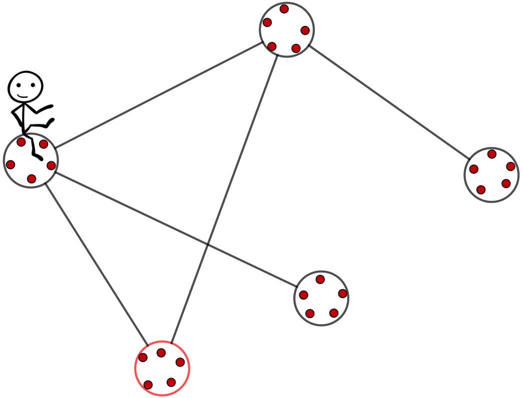


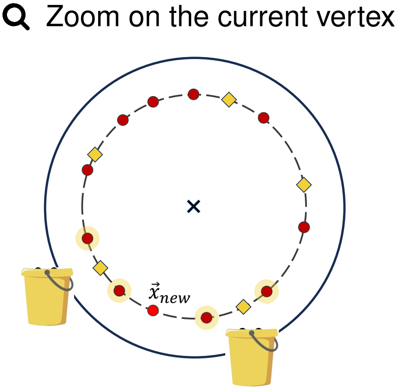
Zoom on the current vertex



Quantum walk subroutine

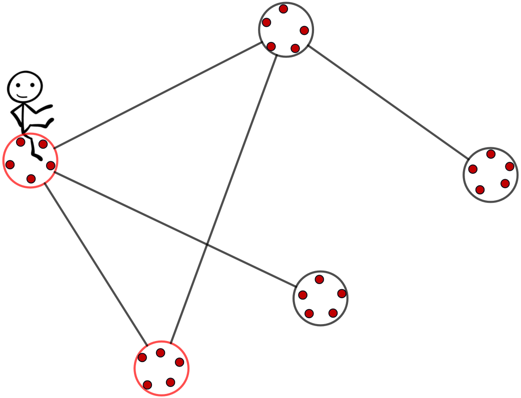
Goal: Find 1 reducing pair in



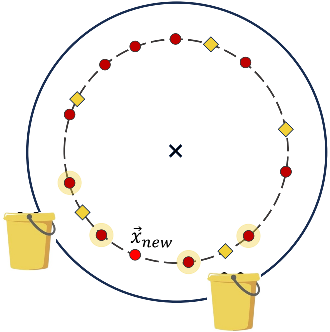


Quantum walk subroutine

Goal: Find 1 reducing pair in

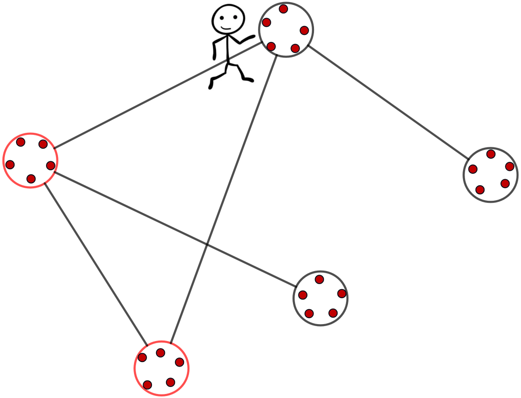


Zoom on the current vertex

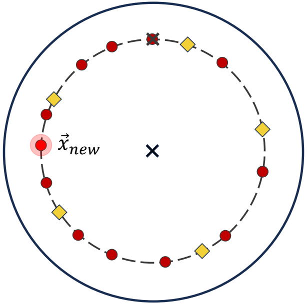


Quantum walk subroutine

Goal: Find 1 reducing pair in

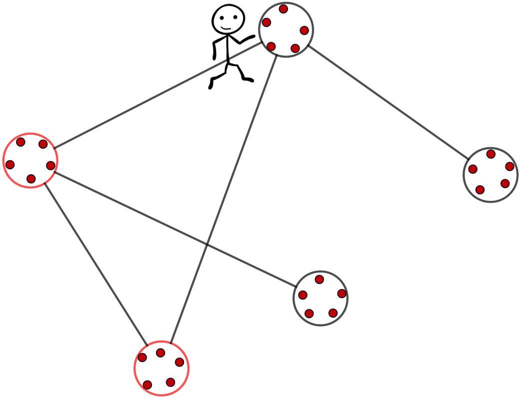


Zoom on the current vertex

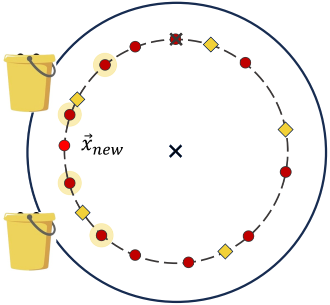


Quantum walk subroutine

Goal: Find 1 reducing pair in

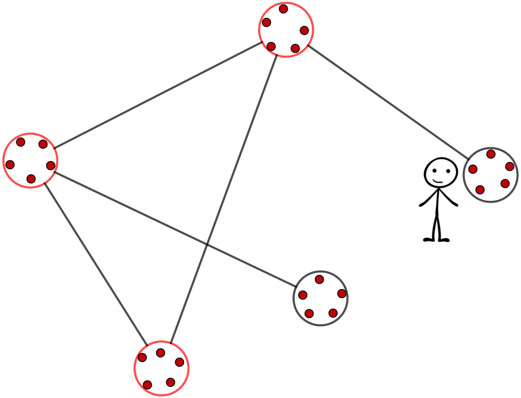


Zoom on the current vertex

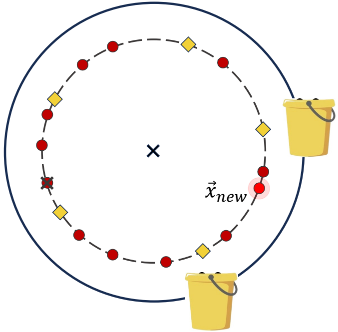


Quantum walk subroutine

Goal: Find 1 reducing pair in

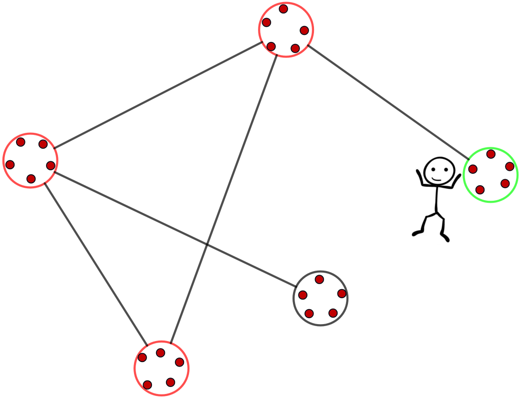


Zoom on the current vertex

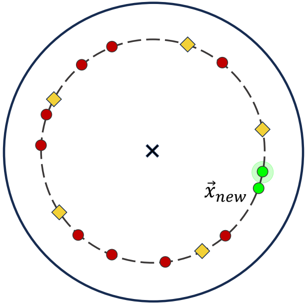


Quantum walk subroutine

Goal: Find 1 reducing pair in



Zoom on the current vertex



Classic VS Quantum walks

Classic random walk: Randomly choose 1 neighbor vertex.

Quantum walk: Quantum superposition of all the neighbor vertices.

[3] [MNRS07] Magniez-Nayak-Roland-Santha. Search via quantum walk.

Classic VS Quantum walks

Classic random walk: Randomly choose 1 neighbor vertex.

Quantum walk: Quantum superposition of all the neighbor vertices.

Time complexity[3]: $\mathcal{S} + \frac{\mathcal{U}}{\sqrt{\epsilon \cdot \delta}}$

- Setup $\mathcal{S}$: construct the 1^{st} vertex, fill 
- Update $\mathcal{U}$: update  with $\vec{\mathbf{x}}_{new}$, check , build the superposition of the neighbors

- $\epsilon \leq 1$ fraction of marked vertices
- $\delta \leq 1$ spectral gap of the graph

[3] [MNRS07] Magniez-Nayak-Roland-Santha. Search via quantum walk.

Step 1 - Partitioning the sphere

For each $\vec{\mathbf{x}} \in L$:

Add $\vec{\mathbf{x}}$ to its nearest filter's bucket 

Step 2 - Pairs finding

For each  **:**

Repeat until all the reducing pairs are found within :

Run a quantum walk (with filters ) to find a new reducing pair

Step 1 - Partitioning the sphere

For each $\vec{\mathbf{x}} \in L$:

Add $\vec{\mathbf{x}}$ to its nearest filter's bucket

Step 2 - Pairs finding

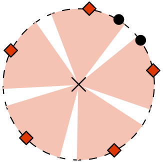
For each :

Repeat until all the reducing pairs are found within :

Run a quantum walk (with filters) to find a new reducing pair

Repeat

Repeat steps 1 and 2 until all N reduced points are found.



Complexity

Time of a sieving step: $N \cdot \left(\mathcal{S} + \frac{\mathcal{U}}{\sqrt{\epsilon\ \delta}} \right)$

Parameters:

- Size of a bucket 
- Size of a vertex 
- Size of a bucket 

Complexity

Time of a sieving step: $N \cdot \left(\mathcal{S} + \frac{\mathcal{U}}{\sqrt{\epsilon}\ \delta} \right)$

Parameters:

- Size of a bucket 
- Size of a vertex 
- Size of a bucket 

numerical optimisation →

$2^{0.08d}$

$2^{0.05d}$

$\text{poly}(d)$

Complexity

Time of a sieving step: $N \cdot \left(\mathcal{S} + \frac{\mathcal{U}}{\sqrt{\epsilon}\ \delta} \right)$

Parameters:

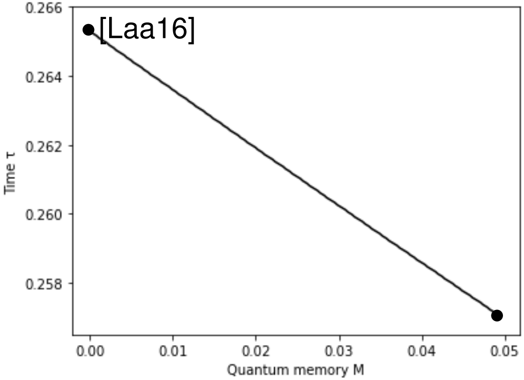
- Size of a bucket → $2^{0.08d}$
- Size of a vertex → $2^{0.05d}$
- Size of a bucket → $\text{poly}(d)$

(numerical optimisation)

Our algorithm (heuristically) solves SVP

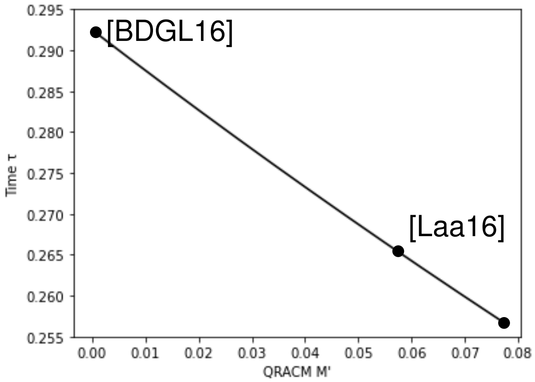
- in **time** $2^{0.257d+o(d)}$ (previous: $2^{0.265d+o(d)}$)
- with classical memory of size $2^{0.208d+o(d)}$,
- QRACM of size $2^{0.08d+o(d)}$,
- and quantum memory (QRAQM) of size $2^{0.05d+o(d)}$.

Trade-offs



Quantum memory/time trade-off.
(Exponents 2^{xd})

Trade-offs



QRACM/time trade-off.
(Exponents 2^{xd})

Trade-offs

Time	**0.2925**	0.283	0.273	**0.2653**	0.262	0.260	**0.2570**
QRACM	**0**	0.02	0.04	**0.0578**	0.065	0.070	**0.0767**
QRAQM	**0**	0	0	**0**	0.019	0.032	**0.0495**
Comment	[BDGL16] alg.			[Laa16] alg.			opt.param[4]

Time and memory exponents for our algorithm.

[4][CL21] Chailloux-Loyer. Lattice sieving via quantum random walks.

Takeaway

Conclusion

- Use quantum walks for sieving
- Generalization of the framework from [BDGL16] using two filtering layers
- New best quantum attack on lattices: $2^{0.2570d+o(d)}$ (previous: $2^{0.265d+o(d)}$)
- Go below the *conditional* lower bound[5]

[5] [KL21] Kirshanova-Laarhoven. Lower bounds on lattice sieving and information set decoding.

Outline

2-sieve [NV08]

for $(\vec{\boldsymbol{x}}_1, \vec{\boldsymbol{x}}_2) \in L^2$:
  if $\|\vec{\boldsymbol{x}}_1 - \vec{\boldsymbol{x}}_2\| \leqslant \gamma R$:
    add $\vec{\boldsymbol{x}}_1 - \vec{\boldsymbol{x}}_2$ to L_{out}

2-sieve [NV08]

for $(\vec{x}_1, \vec{x}_2) \in L^2$:
    if $\|\vec{x}_1 - \vec{x}_2\| \leqslant \gamma R$:
        add $\vec{x}_1 - \vec{x}_2$ to L_{out}

3-sieve

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

2-sieve [NV08]

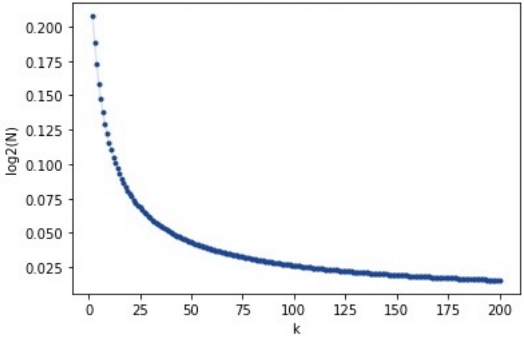
for $(\vec{x}_1, \vec{x}_2) \in L^2$:
  if $\|\vec{x}_1 - \vec{x}_2\| \leqslant \gamma R$:
    add $\vec{x}_1 - \vec{x}_2$ to L_{out}

3-sieve

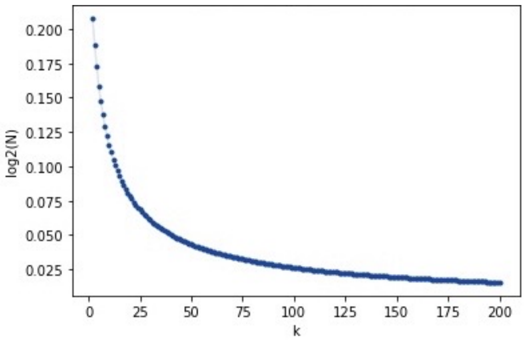
for $(\vec{x}_1, \vec{x}_2, \vec{x}_3) \in L^3$:
  if $\|\vec{x}_1 + \vec{x}_2 + \vec{x}_3\| \leqslant \gamma R$:
    add $\vec{x}_1 + \vec{x}_2 + \vec{x}_3$ to L_{out}

k-sieve

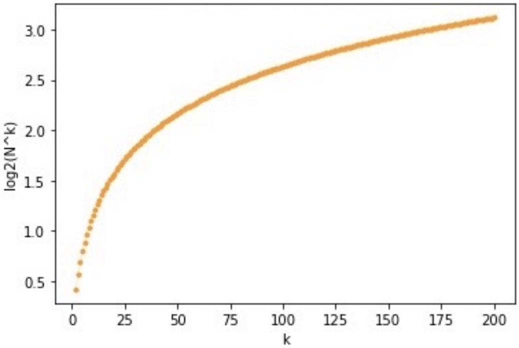
for $(\vec{x}_1, ..., \vec{x}_k) \in L^k$:
  if $\|\vec{x}_1 + ... + \vec{x}_k\| \leqslant \gamma R$:
    add $\vec{x}_1 + ... + \vec{x}_k$ to L_{out}



Minimal memory N

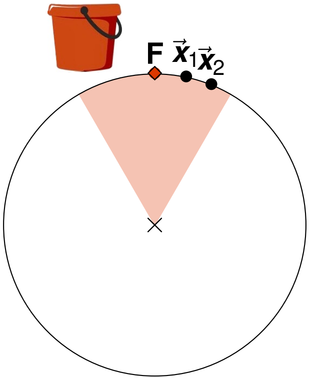


Minimal memory N

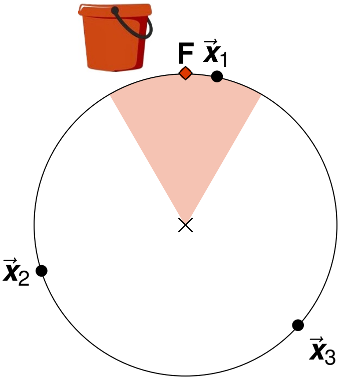


Naive time N^k

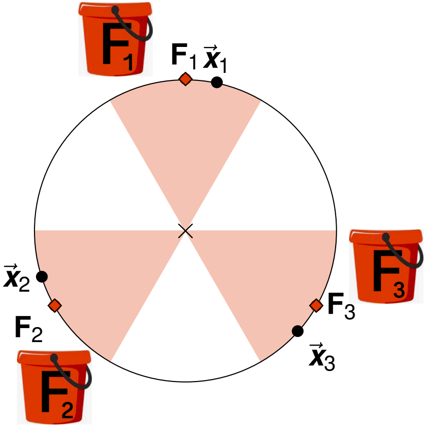
Filtering strategy for the 2-sieve



New filtering tailored for the k-sieve



New filtering tailored for the k-sieve



$$\mathbf{F}_1 + \mathbf{F}_2 + \mathbf{F}_3 = \vec{0}$$

Step 1 - Partitioning the sphere

For each $\vec{\mathbf{x}} \in L$:

Add $\vec{\mathbf{x}}$ to its nearest filter's bucket 

Step 2 - Triplets finding

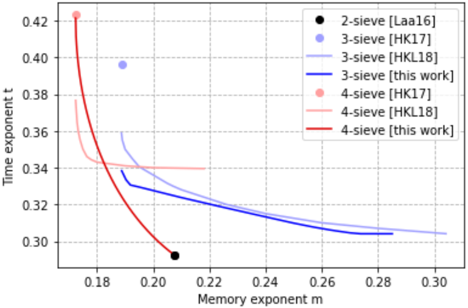
For each tuple-filter  :

Find all reducing $(\vec{\mathbf{x}}_1, \vec{\mathbf{x}}_2, \vec{\mathbf{x}}_3)$ in  $\times$  $\times$ 

Repeat

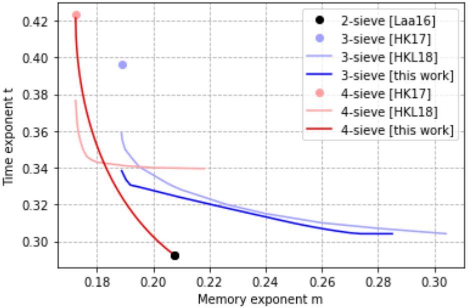
Repeat steps 1 and 2 until all N reduced points are found.

Trade-offs

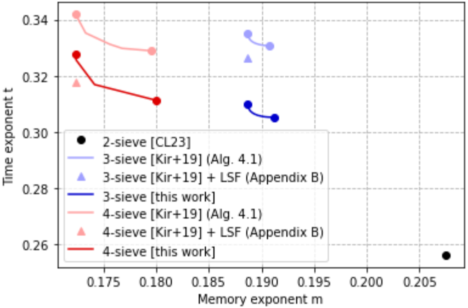


Classical k-sieves

Trade-offs



Classical k-sieves



Quantum k-sieves

Takeaway

Conclusion

- New filtering technique: k-RPC 

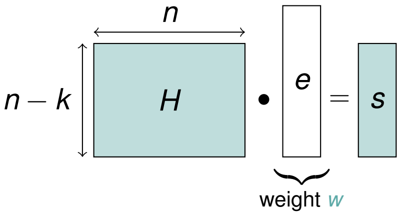
- New trade-offs, improved in some regimes
- Also go below the *conditional* lower bound[6]
- Straightforward improvements: add pairwise filtering , quantum walks...

[6] [KL21] Kirshanova-Laarhoven. Lower bounds on lattice sieving and information set decoding.

Outline

Syndrome Decoding problem

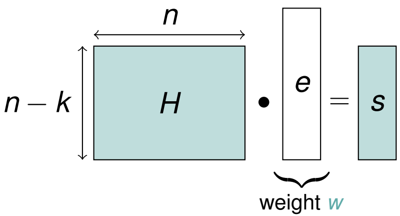
- **Public**: matrix H and vector s with elements in $\{0,1\}$, weight $w \in [\![0,n]\!]$
- **Secret**: $e \in \{0,1\}^n$ such that:



Syndrome Decoding problem

Public: matrix H and vector s with elements in $\{0,1\}$, weight $w \in [\![0,n]\!]$

Secret: $e \in \{0,1\}^n$ such that:



- WAVE digital signature:
  - H **structured** matrix $(U, U+V)$
  - **Ternary** : $\{0,1,2\}$ instead of $\{0,1\}$
  - **Large** weight w

Attacks on Wave

Key attack: Distinguish the secret key  from the uniform random

- Find $\mathbf{e} = (\mathbf{u}, \mathbf{u})$ solution to the Syndrome Decoding problem.

Attacks on Wave

Key attack: Distinguish the secret key from the uniform random

- Find $\mathbf{e} = (\mathbf{u}, \mathbf{u})$ solution to the Syndrome Decoding problem.

Forgery attack: Produce a fake signed document passing the authenticity test

- Find couple $\mathbf{s}$ and $\mathbf{e} = (\mathbf{u}, \mathbf{u})$ solution to the Syndrome Decoding problem.

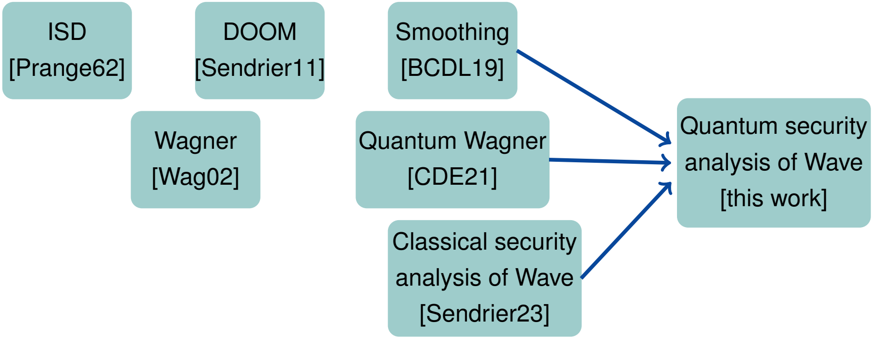
Attacks on Wave

Key attack: Distinguish the secret key from the uniform random

- Find $\mathbf{e} = (\mathbf{u}, \mathbf{u})$ solution to the Syndrome Decoding problem.

Forgery attack: Produce a fake signed document passing the authenticity test

- Find couple $\mathbf{s}$ and $\mathbf{e} = (\mathbf{u}, \mathbf{u})$ solution to the Syndrome Decoding problem.



Wave security

λ bits of security: known attacks run in time $\geq 2^{\lambda}$.

	Classical		Quantum	
NIST settings	Key attack	Forgery attack	Key attack	Forgery attack
(I)	138	129	**80**	**78**
(III)	206	194	**120**	**117**
(V)	274	258	**160**	**156**

Takeaway

Conclusion

- First quantum key attack against Wave
- Improvement of the quantum forgery attack
- NIST submission

Ongoing and future works

- Code sieving via quantum walks
  Collision finding and two filtering layers for code sieving [DEEK23]
- Optimal quantum algorithm for multiple collisions
  Extend [BCSS23] to all parameter ranges.
- 2^k-sieve with combined filtering techniques
  Trade-off from best memory to best time.

Thank you for your attention!