# Classical and Quantum Cryptanalysis for Euclidean Lattices and Subset Sums

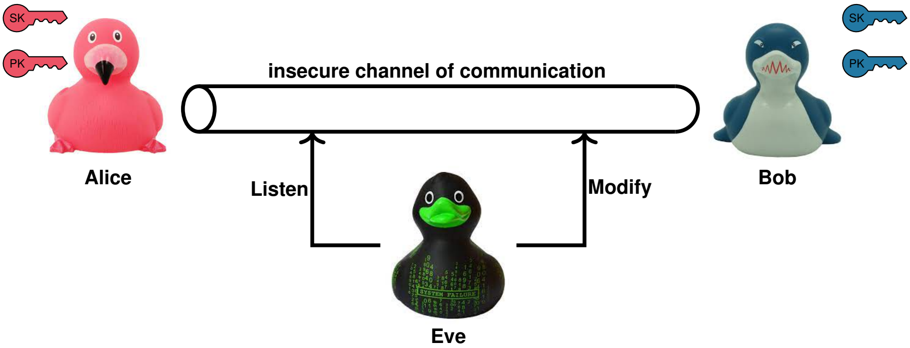
Yixin Shen

11 May, 2021

Université de Paris

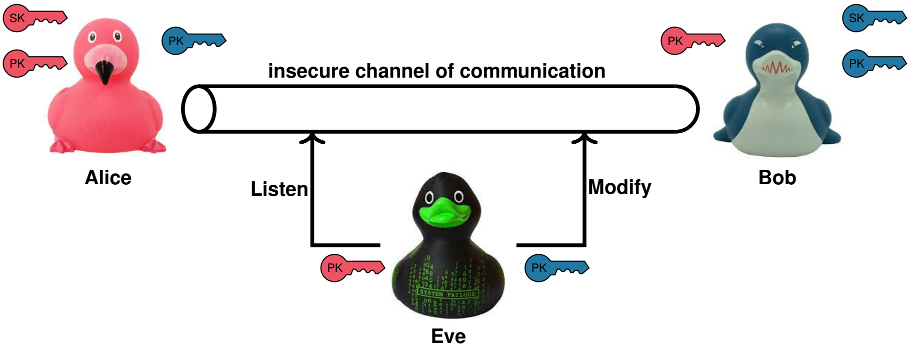
INSTITUT DE RECHERCHE EN INFORMATIQUE FONDAMENTALE

# Hard Problems in Public Key Cryptography



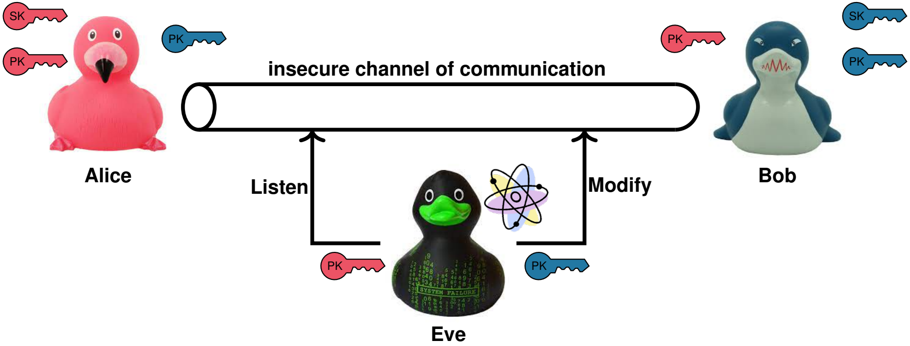

factorisation, discrete logarithm

# Hard Problems in Public Key Cryptography




factorisation, discrete logarithm

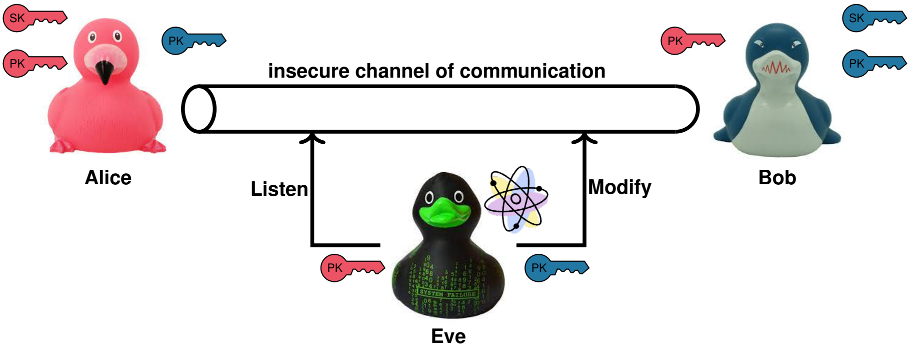
# Hard Problems in Public Key Cryptography




~~factorisation, discrete logarithm~~

**lattices**, codes, ...

# Hard Problems in Public Key Cryptography




~~factorisation, discrete logarithm~~

**lattices**, codes, ...

NIST round 3 candidates:

- encryption: 3 out of 4 candidates based on lattices
- signatures: 2 out of 3 candidates based on lattices

# Publications and Outline

- **Yoshinori Aono, Phong Q. Nguyen, and Y. S.**
  **Quantum lattice enumeration and tweaking discrete pruning.**
  **In ASIACRYPT 2018.**

- **Divesh Aggarwal, Yanlin Chen, Rajendra Kumar, and Y. S.**
  **Improved (provable) algorithms for the shortest vector problem via bounded distance decoding.**
  **In STACS 2021.**

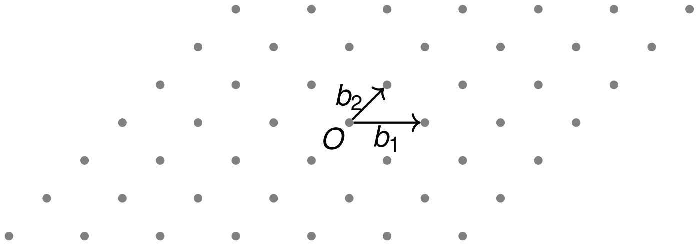
- Xavier Bonnetain, Rémi Bricout, André Schrottenloher, and Y. S.
  Improved classical and quantum algorithms for subset-sum.
  In ASIACRYPT 2020.

- Andris Ambainis, Kaspars Balodis, Janis Iraids, Kamil Khadiev, Vladislavs Klevickis, Krisjanis Prusis, Y. S, Juris Smotrovs, and Jevgenijs Vihrovs.
  Quantum lower and upper bounds for 2D-grid and Dyck language.
  In MFCS 2020.

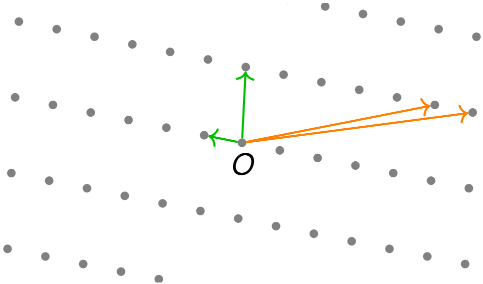
# What is a (Euclidean) lattice?

**Definition**

$\mathcal{L}(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n) = \left\{\sum_{i=1}^{n} x_i \boldsymbol{b}_i : x_i \in \mathbb{Z}\right\}$ where $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n$ is a basis of $\mathbb{R}^n$.

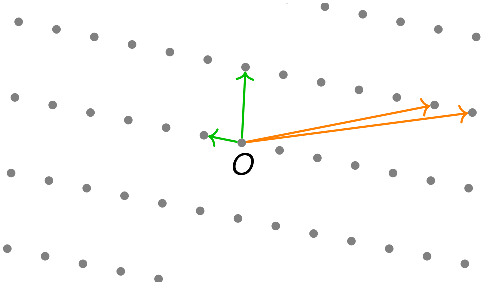


# Lattice-based cryptography: fundamental idea




- good basis: private information, makes problem easy
- bad basis: public information, makes problem hard

# Lattice-based cryptography: fundamental idea

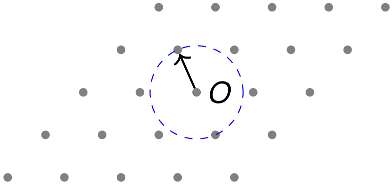



- good basis: private information, makes problem easy
- bad basis: public information, makes problem hard

Basis reduction: transform a bad basis into a good one
Main tool: BKZ algorithm and its variants

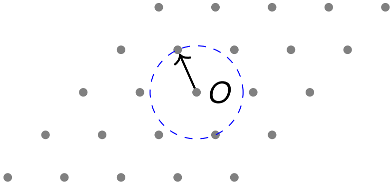
Requires to solve the (approx-)SVP problem in smaller dimensions.

# The Shortest Vector Problem

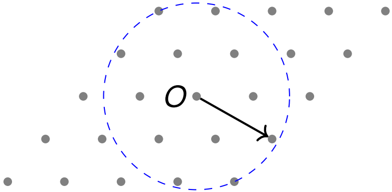



Shortest Vector Problem (SVP):
Given a basis for the lattice $\mathcal{L}$, find a shortest nonzero lattice vector.
$\lambda_1(\mathcal{L})$ = length of such a vector.

# The Shortest Vector Problem




Shortest Vector Problem (SVP):
Given a basis for the lattice $\mathcal{L}$, find a shortest nonzero lattice vector.
$\lambda_1(\mathcal{L})$ = length of such a vector.




$\gamma$-approx-SVP ($\gamma > 1$):
Given a basis of $\mathcal{L}$, find a nonzero lattice vector of length at most $\gamma \cdot \lambda_1(\mathcal{L})$.
$\gamma$ is approximation factor.

# The Shortest Vector Problem

Depending on the dimension $n$:

- NP-Hardness (randomized reduction)

Approx factor:

- $O(1)$

# The Shortest Vector Problem

Depending on the dimension $n$:

- NP-Hardness (randomized reduction)
- NP $\cap$ co-NP

Approx factor:

- $O(1)$
- $\sqrt{n}$

# The Shortest Vector Problem

Depending on the dimension $n$:

- NP-Hardness (randomized reduction)
- NP $\cap$ co-NP
- Subexponential-time algorithms

Approx factor:

- $O(1)$
- $\sqrt{n}$
- $2^{\sqrt{n}}$

# The Shortest Vector Problem

Depending on the dimension $n$:

- NP-Hardness (randomized reduction)
- NP $\cap$ co-NP
- Subexponential-time algorithms
- Poly-time algorithms

Approx factor:

- $O(1)$
- $\sqrt{n}$
- $2^{\sqrt{n}}$
- $2^{\frac{n \log \log n}{\log n}}$

# The Shortest Vector Problem

Depending on the dimension $n$:

- NP-Hardness (randomized reduction)
- NP $\cap$ co-NP
- Subexponential-time algorithms
- Poly-time algorithms

Approx factor:

- $O(1)$
- $\sqrt{n}$
- $2^{\sqrt{n}}$
- $2^{\frac{n \log \log n}{\log n}}$

Main approaches for SVP:

- Enumeration: $2^{O(n \log(n))}$ time and $\mathrm{poly}(n)$ space
- Sieving: $2^{O(n)}$ time and $2^{O(n)}$ space

# The Shortest Vector Problem

Depending on the dimension $n$:

- NP-Hardness (randomized reduction)
- NP $\cap$ co-NP
- Subexponential-time algorithms
- Poly-time algorithms

Approx factor:

- $O(1)$
- $\sqrt{n}$
- $2^{\sqrt{n}}$
- $2^{\frac{n \log \log n}{\log n}}$

Main approaches for SVP:

- Enumeration: $2^{O(n \log(n))}$ time and $\mathrm{poly}(n)$ space
- Sieving: $2^{O(n)}$ time and $2^{O(n)}$ space

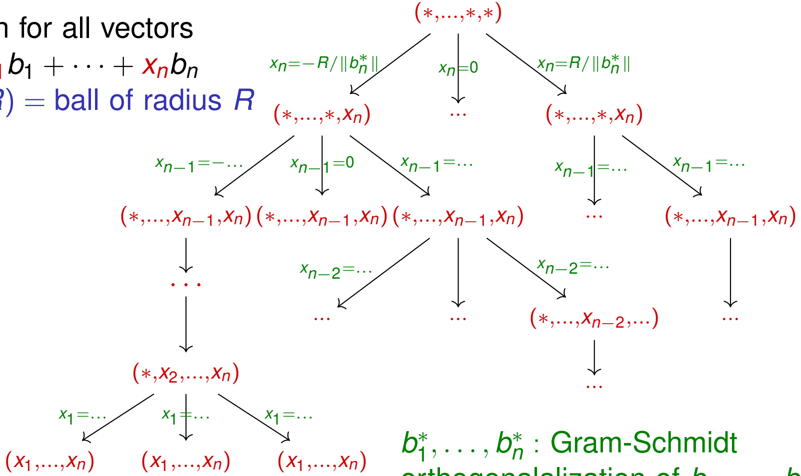
BKZ with block size $k$ solves $O(k^{n/k})$-approx-SVP using a SVP oracle in dimension $k$:

- Enumeration: time $2^{O(k \log(k))}\, \mathrm{poly}(n)$ and space $\mathrm{poly}(n)$
- Sieving: time $2^{O(k)}\, \mathrm{poly}(n)$ time and space $2^{O(k)}\, \mathrm{poly}(n)$

# Enumeration

# Enumeration Algorithm

Search for all vectors
$X = x_1 b_1 + \cdots + x_n b_n$
in $B(R)$ = ball of radius $R$

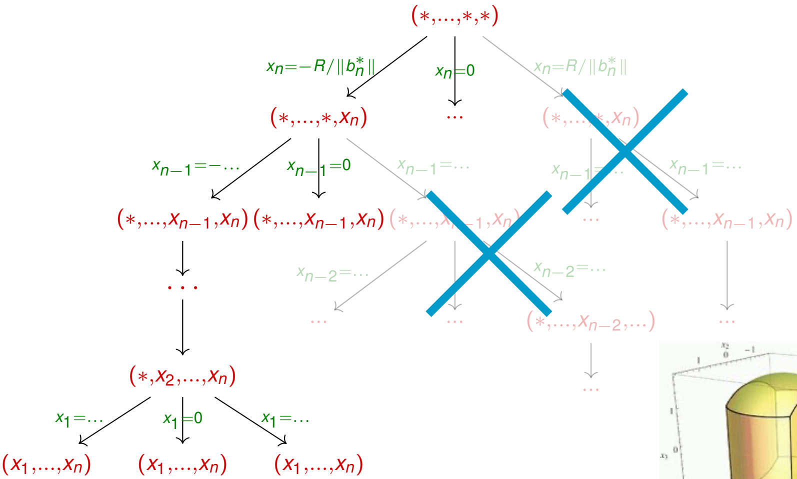


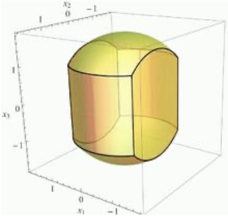

$b_1^*, \ldots, b_n^*$ : Gram-Schmidt orthogonalalization of $b_1, \ldots, b_n$

Given $x_n, \ldots, x_{i+1}$, $\|\pi_i(x)\| \leqslant R$,
$\Rightarrow$ the integer $x_i$ belongs to an interval of small length

$\pi_i$: orthogonal projection on $\mathrm{span}(b_1, \ldots, b_{i-1})^{\perp}$

# Cylinder Pruning [GNR10]



Each level remplace $\|\pi_i(x)\| \leqslant R$ by $\|\pi_i(x)\| \leqslant R_i R$ where $0 < R_i \leqslant 1$

# Quantum Speed-up for Enumeration

Quantum backtracking [Montanaro15]

- blackbox access to a tree with marked nodes:
  - can only query the local structure of the tree
- tree of size $T$, depth $n$, constant max degree

$\Rightarrow O^*(\sqrt{T})$ queries to find a marked node

# Quantum Speed-up for Enumeration

Quantum backtracking [Montanaro15]

- blackbox access to a tree with marked nodes:
  - can only query the local structure of the tree
- tree of size $T$, depth $n$, constant max degree

$\Rightarrow O^*(\sqrt{T})$ queries to find a marked node

Application to the previous enumeration algorithm (**Quantum Lattice Enumeration**):

Remark: LLL-reduced basis $\rightsquigarrow$ max degree can be $2^{\Omega(n)}$

Algorithm: transform the tree into a binary one + dichotomy

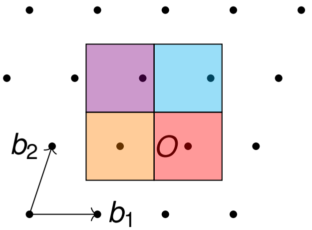
$\Rightarrow O^*(\sqrt{T})$ time to find one vector in $L \cap S(R)$
$\Rightarrow O^*(\#(L \cap S(R)) \cdot \sqrt{T})$ time to find all vectors.

# Discrete Pruning [AN17]

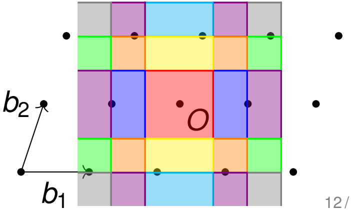
Step 0: partition space into cells

- 1 cell $\leftrightarrow$ 1 lattice vector
- cell $C(\boldsymbol{t})$ identified by tag $\boldsymbol{t} \in \mathbb{Z}^n$

- Babai's partition:




- natural partition:



# Discrete Pruning [AN17]

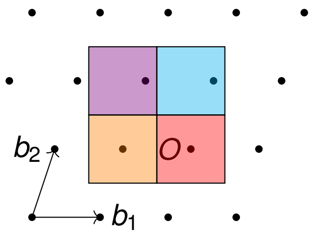
Step 0: partition space into cells

- 1 cell $\leftrightarrow$ 1 lattice vector
- cell $C(\boldsymbol{t})$ identified by tag $\boldsymbol{t} \in \mathbb{Z}^n$

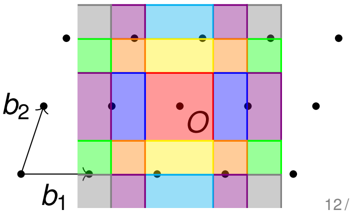
Step 1: find the pruning set

- Find $\approx M$ best cells minimizing $g(\boldsymbol{t}) = \sum_{i=1}^{n} \|b_i^*\|^2 \left(\frac{t_i^2}{4} + \frac{t_i}{4} + \frac{1}{12}\right)$
  Roughly, the smaller $g(\boldsymbol{t})$, the smaller the vector inside $C(\boldsymbol{t})$
- Equivalent to finding $R$ such that #solutions of $g(\boldsymbol{t}) \leqslant R$ is $\approx M$

- Babai's partition:




- natural partition:



# Discrete Pruning [AN17]

Step 0: partition space into cells

- 1 cell $\leftrightarrow$ 1 lattice vector
- cell $C(\boldsymbol{t})$ identified by tag $\boldsymbol{t} \in \mathbb{Z}^n$

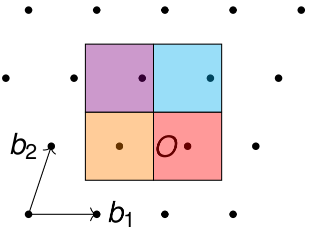
Step 1: find the pruning set

- Find $\approx M$ best cells minimizing $g(\boldsymbol{t}) = \sum_{i=1}^{n} \|b_i^*\|^2 \left(\frac{t_i^2}{4} + \frac{t_i}{4} + \frac{1}{12}\right)$ Roughly, the smaller $g(\boldsymbol{t})$, the smaller the vector inside $C(\boldsymbol{t})$
- Equivalent to finding $R$ such that #solutions of $g(\boldsymbol{t}) \leqslant R$ is $\approx M$

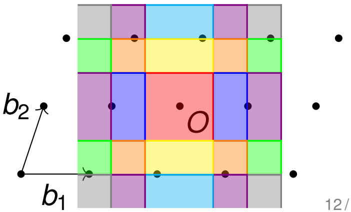
Step 2: find the shortest vector

- consider enumeration tree above obtained by backtracking

- Babai's partition:




- natural partition:



# Quantum speed-up for discrete pruning

Step 1: find $R$ such that #solutions of $g(\boldsymbol{t}) \leqslant R$ is $\approx M$ by **dichotomy**

- Quantum tree size estimation [AK17] to estimate #nodes
- Tweak cost function:

$$g(\boldsymbol{t}) = \sum_{i=1}^{n} \|b_i^*\|^2 \left(\frac{t_i^2}{4} + \frac{t_i}{4} + \frac{1}{12}\right) \quad \rightsquigarrow \quad \sum_{i=1}^{n} \|b_i^*\|^2 \left(t_i^2 + t_i\right)$$

  Linear relation between #nodes and #solutions

# Quantum speed-up for discrete pruning

Step 1: find $R$ such that #solutions of $g(\boldsymbol{t}) \leqslant R$ is $\approx M$ by **dichotomy**

- Quantum tree size estimation [AK17] to estimate #nodes
- Tweak cost function:

$$g(\boldsymbol{t}) = \sum_{i=1}^{n} \|b_i^*\|^2 \left(\frac{t_i^2}{4} + \frac{t_i}{4} + \frac{1}{12}\right) \quad \rightsquigarrow \quad \sum_{i=1}^{n} \|b_i^*\|^2 \left(t_i^2 + t_i\right)$$

  Linear relation between #nodes and #solutions

Step 2: find shortest among cells satisfying $g(\boldsymbol{t}) \leqslant R$

- dichotomy on length + mark nodes
- **Quantum backtracking** + binary tree transformation

# Quantum speed-up for discrete pruning

Step 1: find $R$ such that #solutions of $g(\boldsymbol{t}) \leqslant R$ is $\approx M$ by **dichotomy**

- Quantum tree size estimation [AK17] to estimate #nodes
- Tweak cost function:

$$g(\boldsymbol{t}) = \sum_{i=1}^{n} \|b_i^*\|^2 \left(\frac{t_i^2}{4} + \frac{t_i}{4} + \frac{1}{12}\right) \quad \rightsquigarrow \quad \sum_{i=1}^{n} \|b_i^*\|^2 \left(t_i^2 + t_i\right)$$

Linear relation between #nodes and #solutions

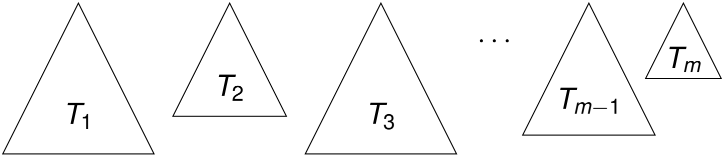
Step 2: find shortest among cells satisfying $g(\boldsymbol{t}) \leqslant R$

- dichotomy on length + mark nodes
- **Quantum backtracking** + binary tree transformation

Asymptotic quadratic improvement over classical algorithm.

# Extreme Pruning

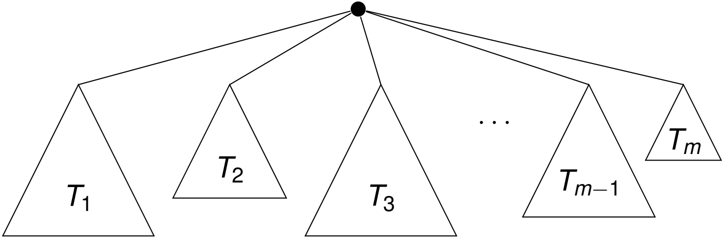
Repeat pruning with many basis:




- classical: $\sum T_i$
- naive quantum: $\sum \sqrt{T_i}$

# Extreme Pruning

Repeat pruning with many basis:




- classical: $\sum T_i$
- naive quantum: $\sum \sqrt{T_i}$
- this thesis: $\sqrt{\sum T_i}$ can be much better than the naive quantum depending on the distribution of $T_i$.

# Summary on quantum enumeration

Quantum Quadratic Speed-up for Enumeration Algorithms:

- applies to cylinder, discrete and extreme pruning
- no known quadratic speedup for sieving

# Summary on quantum enumeration

Quantum Quadratic Speed-up for Enumeration Algorithms:

- applies to cylinder, discrete and extreme pruning
- no known quadratic speedup for sieving

$\rightsquigarrow$ crossover point between sieving and enumeration is different in the quantum setting

# Summary on quantum enumeration

Quantum Quadratic Speed-up for Enumeration Algorithms:

- applies to cylinder, discrete and extreme pruning
- no known quadratic speedup for sieving

$\rightsquigarrow$ crossover point between sieving and enumeration is different in the quantum setting

[ABLR20] estimates the new crossover point for BKZ using enumeration/sieving achieving the same quality (RHF) at $k = 547$.

# Summary on quantum enumeration

Quantum Quadratic Speed-up for Enumeration Algorithms:

- applies to cylinder, discrete and extreme pruning
- no known quadratic speedup for sieving

$\rightsquigarrow$ crossover point between sieving and enumeration is different in the quantum setting

[ABLR20] estimates the new crossover point for BKZ using enumeration/sieving achieving the same quality (RHF) at $k = 547$.

Open problems:

- Study of the polynomial factors for sieving and enumeration are needed for a better comparison
- More studies on discrete pruning are needed

# Sieving

- Heuristic algorithms: fastest in practice
- Provable algorithms: important for theory $\rightarrow$ this thesis

# Results in the Classical Setting

Provable algorithms for SVP:

| Time Complexity | Space Complexity | Reference |
|---|---|---|
| $n^{\frac{n}{2e}+o(n)}$ | poly$(n)$ | [Kan87,HS07] |
| $2^{n+o(n)}$ | $2^{n+o(n)}$ | [ADRS15] |
| $2^{2.05n+o(n)}$ | $2^{0.5n+o(n)}$ | [CCL18] |
| $2^{1.7397n+o(n)}$ | $2^{0.5n+o(n)}$ | This thesis |

# Results in the Classical Setting

Provable algorithms for SVP:

| Time Complexity | Space Complexity | Reference |
|---|---|---|
| $n^{\frac{n}{2e}+o(n)}$ | $\mathrm{poly}(n)$ | [Kan87,HS07] |
| $2^{n+o(n)}$ | $2^{n+o(n)}$ | [ADRS15] |
| $2^{2.05n+o(n)}$ | $2^{0.5n+o(n)}$ | [CCL18] |
| $2^{1.7397n+o(n)}$ | $2^{0.5n+o(n)}$ | This thesis |

This thesis: first provable smooth time/space trade-off for SVP

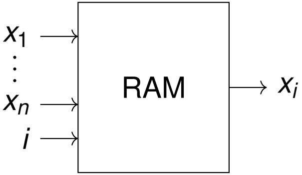
time $q^{13n+o(n)}$ space $\mathrm{poly}(n) \cdot q^{\frac{16n}{q^2}}$ $q \in [4, \sqrt{n}]$

- $q = \sqrt{n}$: time $n^{O(n)}$ and space $\mathrm{poly}(n)$, not as good as [Kan87].
- $q = 4$: time $2^{O(n)}$ and space $2^{O(n)}$, not as good as [ADRS15].

# Interlude: quantum memory models

classical access | quantum access

classical data




standard

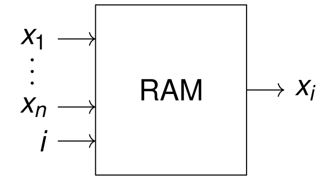
quantum data

Assumption: $O(1)$ time cost

# Interlude: quantum memory models

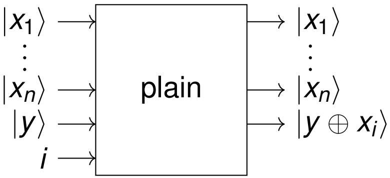
classical access | quantum access

classical data




standard

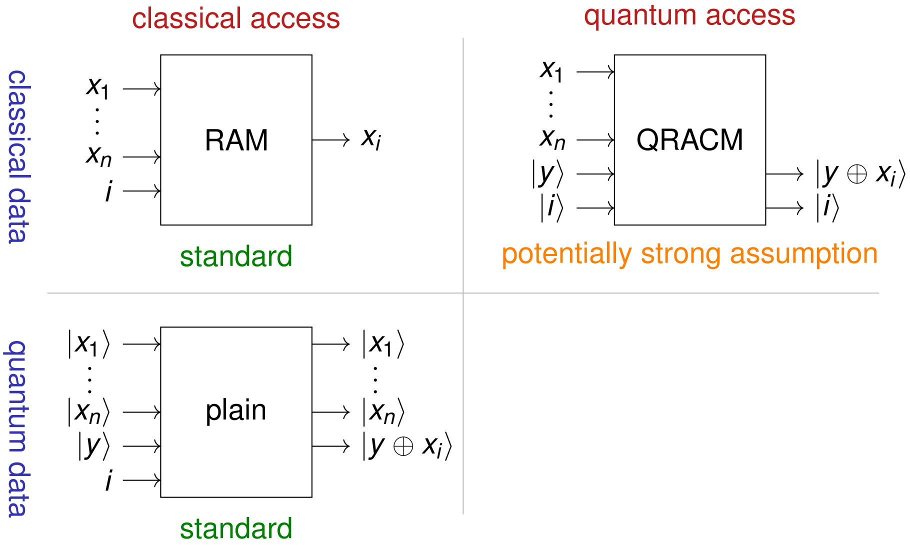
quantum data




standard

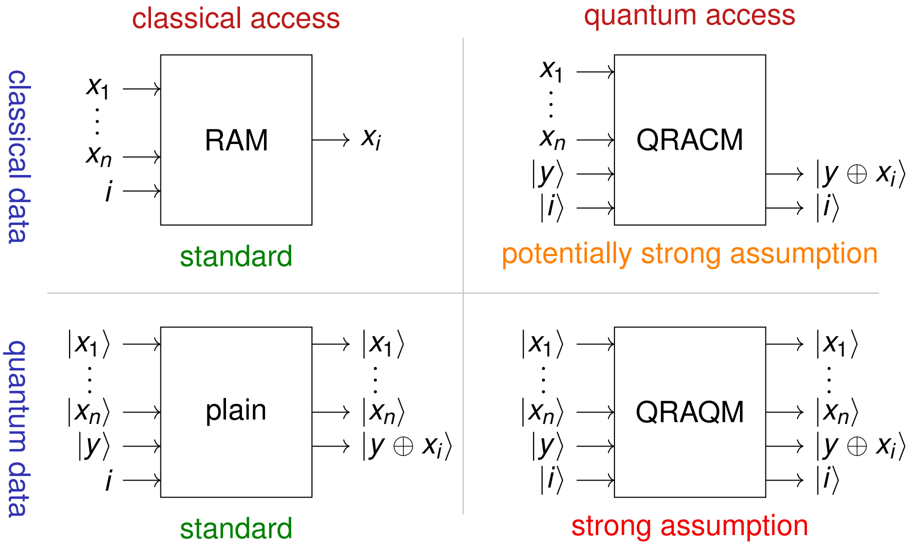
Assumption: $O(1)$ time cost

# Interlude: quantum memory models




Assumption: $O(1)$ time cost

# Interlude: quantum memory models




Assumption: $O(1)$ time cost

# Results in the Quantum Setting

Provable quantum algorithms for SVP:

| Time Complexity | Space Complexity | | | Reference |
|---|---|---|---|---|
| | Classical | Qubits | Model | |
| $2^{1.799n+o(n)}$ | $2^{1.286n+o(n)}$ | $\mathrm{poly}(n)$ | QRACM | [LMP15] |
| $2^{1.2553n+o(n)}$ | $2^{0.5n+o(n)}$ | $\mathrm{poly}(n)$ | plain | [CCL18] |
| $2^{0.9535n+o(n)}$ | $2^{0.5n+o(n)}$ | $\mathrm{poly}(n)$ | plain | This thesis |

Remark on quantum heuristic algorithms:

- better complexity: $2^{0.265n+o(n)}$ [Laarhoven15]
- requires QRACM (strong assumption)

# Sieving Algorithms

Original idea [AKS01]:

- Reduce basis
- Generate random vectors
- Repeat many times:
  - Sieve vectors

# Sieving Algorithms

Original idea [AKS01]:

- Reduce basis
- Generate random vectors
- Repeat many times:
  - Sieve vectors

Sieve:

**Input:** many vectors of length $\leqslant \ell$
**Output:** many vectors of length $\leqslant \frac{\ell}{2}$

Combine pairs of vectors to produce shorter vectors

# Sieving Algorithms

Original idea [AKS01]:

- Reduce basis
- Generate random vectors
- Repeat many times:
  - Sieve vectors

Sieve:

**Input:** many vectors of length $\leqslant \ell$
**Output:** many vectors of length $\leqslant \frac{\ell}{2}$

Combine pairs of vectors to produce shorter vectors

Idea: LLL reduced $\rightsquigarrow$ length $\ell \leqslant 2^{O(n)}\lambda_1$, sieve $O(n)$, solve SVP

# Sieving Algorithms

Original idea [AKS01]:

- Reduce basis
- Generate random vectors
- Repeat many times:
  - Sieve vectors

Sieve:

**Input:** many vectors of length $\leqslant \ell$
**Output:** many vectors of length $\leqslant \frac{\ell}{2}$

Combine pairs of vectors to produce shorter vectors

Idea: LLL reduced $\rightsquigarrow$ length $\ell \leqslant 2^{O(n)}\lambda_1$, sieve $O(n)$, solve SVP

Many heuristic variants: local sensitive hash, tuple sieve, ...
All control the length of the vectors.

# Sieving Algorithms

Original idea [AKS01]:

- Reduce basis
- Generate random vectors
- Repeat many times:
  - Sieve vectors

Sieve:
**Input:** many vectors of length $\leqslant \ell$
**Output:** many vectors of length $\leqslant \frac{\ell}{2}$

Combine pairs of vectors to produce shorter vectors

Idea: LLL reduced $\rightsquigarrow$ length $\ell \leqslant 2^{O(n)}\lambda_1$, sieve $O(n)$, solve SVP

Many heuristic variants: local sensitive hash, tuple sieve, ...
All control the length of the vectors.

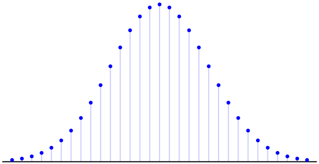
[ADRS15]'s new idea: control distribution instead of length of vectors

# Discrete Gaussian Sampling

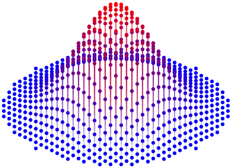
$$\rho_s(\boldsymbol{x}) = \exp\left(-\pi \frac{\|\boldsymbol{x}\|^2}{s^2}\right), \qquad D_{L,s}(\boldsymbol{x}) = \frac{\rho_s(\boldsymbol{x})}{\rho_s(L)}, \qquad \boldsymbol{x} \in \mathbb{R}^n, s > 0.$$

**Definition (Discrete Gaussian Distribution)**

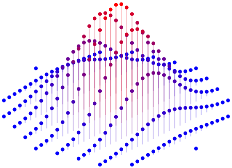
On lattice $L$ with parameter $s$: probability of $\boldsymbol{x} \in L$ is $D_{L,s}(\boldsymbol{x})$.



$L = \mathbb{Z}, s = 7$



$L = \mathbb{Z}^2, s = 7$



$L = \mathbb{Z} \times 4\mathbb{Z}, s = 7$

### Discrete Gaussian Sampling (DGS)

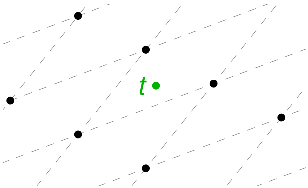
- **input:** $L$ and $s$
- **output:** random $\boldsymbol{x} \in L$ according to $D_{L,s}$.

# DGS, BDD and SVP

$$\text{SVP} \xrightarrow{[\text{ADRS15}]} \text{DGS}$$

# DGS, BDD and SVP

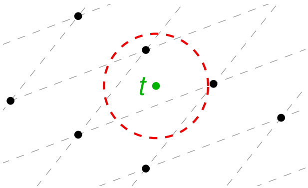
$$\begin{tikzcd} \text{SVP} \arrow[r, "\text{[CCL18]}"'] \arrow[rr, bend left, "\text{[ADRS15]}"] & \text{BDD} \arrow[r, "\text{[DSR14]}"'] & \text{DGS} \end{tikzcd}$$




**Bounded Distance Decoding ($\alpha-$BDD):**
Given a lattice $\mathcal{L}$ and a target vector $t \in \mathbb{R}^n$

# DGS, BDD and SVP

$$\begin{tikzcd}
\boxed{\text{SVP}} \arrow[r, "{[\text{CCL18}]}"'] \arrow[rr, bend left, "{[\text{ADRS15}]}"] & \boxed{\text{BDD}} \arrow[r, "{[\text{DSR14}]}"'] & \boxed{\text{DGS}}
\end{tikzcd}$$

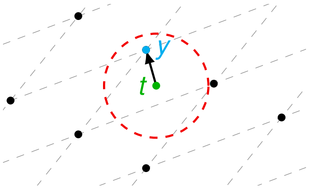



Bounded Distance Decoding ($\alpha-$BDD):
Given a lattice $\mathcal{L}$ and a target vector $t \in \mathbb{R}^n$ with distance to lattice $\leq \alpha \cdot \lambda_1(\mathcal{L})$

The two reductions use completely different DGS parameter regimes!

# DGS, BDD and SVP

$$\begin{tikzcd} \boxed{\text{SVP}} \arrow[r, "\text{[CCL18]}"'] \arrow[rr, bend left, "\text{[ADRS15]}"] & \boxed{\text{BDD}} \arrow[r, "\text{[DSR14]}"'] & \boxed{\text{DGS}} \end{tikzcd}$$




**Bounded Distance Decoding ($\alpha-$BDD):**
Given a lattice $\mathcal{L}$ and a target vector $t \in \mathbb{R}^n$ with distance to lattice $\leq \alpha \cdot \lambda_1(\mathcal{L})$ , find the closest vector $y \in \mathcal{L}$.

- $\alpha$ is decoding distance/radius
- $\alpha < \frac{1}{2}$ for unique solution

The two reductions use completely different DGS parameter regimes!

# Hardness of Discrete Gaussian Sampling

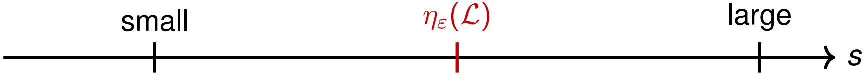
Parameter $s$ (width/standard deviation) of $D_{\mathcal{L},s}$:


- hard to sample
- SVP

- easy to sample

# Hardness of Discrete Gaussian Sampling

Parameter $s$ (width/standard deviation) of $D_{\mathcal{L},s}$:



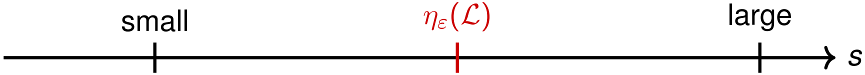

- hard to sample
- SVP

smoothing parameter

- easy to sample

# Hardness of Discrete Gaussian Sampling

Parameter $s$ (width/standard deviation) of $D_{\mathcal{L},s}$:




- hard to sample
- SVP

smoothing parameter

- easy to sample

- Open problem: $2^{O(n)}$ time, $2^{o(n)}$ space algorithm for $s = \eta_\varepsilon(\mathcal{L})$
- No known time/space trade-off for $s \ll \eta_\varepsilon(\mathcal{L})$

# Hardness of Discrete Gaussian Sampling

Parameter $s$ (width/standard deviation) of $D_{\mathcal{L},s}$:

small — $\eta_\varepsilon(\mathcal{L})$ — large $\rightarrow s$

- hard to sample
- SVP

smoothing parameter

- easy to sample

- Open problem: $2^{O(n)}$ time, $2^{o(n)}$ space algorithm for $s = \eta_\varepsilon(\mathcal{L})$
- No known time/space trade-off for $s \ll \eta_\varepsilon(\mathcal{L})$

SVP $\xrightarrow{\text{[CCL18]}}$ BDD $\xrightarrow{\text{[DSR14]}}$ DGS

new provable time/space trade-off

$\rightsquigarrow$ first provable time/space trade-off for SVP

# DGS time/space trade-off

Idea: **if** $X_1, \ldots, X_k \sim D_{\mathcal{L},s}$ and $\sum_i X_i \in q\,\mathcal{L}$ **then** $(\sum_i X_i)/q \approx D_{\mathcal{L},s\sqrt{k}/q}$

$\rightsquigarrow$ progress when $k < q^2$, repeat many times to reach $\eta_\varepsilon(\mathcal{L})$

# DGS time/space trade-off

Idea: **if** $X_1, \ldots, X_k \sim D_{\mathcal{L},s}$ and $\sum_i X_i \in q\,\mathcal{L}$ **then** $(\sum_i X_i)/q \approx D_{\mathcal{L},s\sqrt{k}/q}$
$\rightsquigarrow$ progress when $k < q^2$, repeat many times to reach $\eta_\varepsilon(\mathcal{L})$

Algorithm: given a list of $N$ vectors in $\mathcal{L}$, find $k = q^2 - 1$ of them such that their sum $\in q\,\mathcal{L}$, then repeat ($q$ is a parameter)

# DGS time/space trade-off

Idea: **if** $X_1, \ldots, X_k \sim D_{\mathcal{L},s}$ and $\sum_i X_i \in q\,\mathcal{L}$ **then** $(\sum_i X_i)/q \approx D_{\mathcal{L},s\sqrt{k}/q}$
$\rightsquigarrow$ progress when $k < q^2$, repeat many times to reach $\eta_\varepsilon(\mathcal{L})$

Algorithm: given a list of $N$ vectors in $\mathcal{L}$, find $k = q^2 - 1$ of them such that their sum $\in q\,\mathcal{L}$, then repeat ($q$ is a parameter)

- Space: need $N \gtrsim q^{n/q^2}$ to be successful — decrease with $q$
- Time: $q^n$ to produce one vector — increase with $q$

# DGS time/space trade-off

Idea: **if** $X_1, \ldots, X_k \sim D_{\mathcal{L},s}$ and $\sum_i X_i \in q\,\mathcal{L}$ **then** $(\sum_i X_i)/q \approx D_{\mathcal{L},s\sqrt{k}/q}$
$\rightsquigarrow$ progress when $k < q^2$, repeat many times to reach $\eta_\varepsilon(\mathcal{L})$

Algorithm: given a list of $N$ vectors in $\mathcal{L}$, find $k = q^2 - 1$ of them such that their sum $\in q\,\mathcal{L}$, then repeat ($q$ is a parameter)

- Space: need $N \gtrsim q^{n/q^2}$ to be successful — decrease with $q$
- Time: $q^n$ to produce one vector — increase with $q$

Difficulties:

- independence of samples
- errors in distributions

**Theorem (Simplified)**

*For $q \in [4, \sqrt{n}]$, there is an algorithm that produces $q^{16n/q^2}$ vectors from $D_{\mathcal{L},s}$ with $s \geqslant \eta_\varepsilon(\mathcal{L})$ in time $q^{13n}$ and space $q^{16n/q^2}$.*

# SVP to BDD reduction [CCL18]

**Lemma (CCL18, simplified)**

*Given a $\alpha$-BDD oracle and p an integer, one can enumerate all lattice points in a ball of radius $p\alpha\lambda_1$ using $p^n$ queries to the oracle.*

# SVP to BDD reduction [CCL18]

**Lemma (CCL18, simplified)**

*Given a $\alpha$-BDD oracle and p an integer, one can enumerate all lattice points in a ball of radius $p\alpha\lambda_1$ using $p^n$ queries to the oracle.*

Solve SVP by using a $\alpha$-BDD oracle:

- Set $p = \lceil \frac{1}{\alpha} \rceil$.
- Enumerate all points in a ball of radius $> \lambda_1$.

# SVP to BDD reduction [CCL18]

**Lemma (CCL18, simplified)**

*Given a $\alpha$-BDD oracle and p an integer, one can enumerate all lattice points in a ball of radius $p\alpha\lambda_1$ using $p^n$ queries to the oracle.*

Solve SVP by using a $\alpha$-BDD oracle:

- Set $p = \lceil \frac{1}{\alpha} \rceil$.
- Enumerate all points in a ball of radius $> \lambda_1$.

The reduction is space efficient

But $\alpha < \frac{1}{2} \implies p \geq 3 \implies$ at least $3^n$ queries

# Quantum SVP

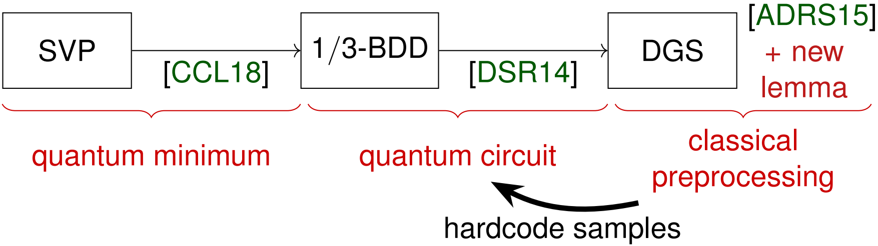
Classical SVP to BDD: do $3^n$ queries to $1/3$-BDD and keep minimum

$$\boxed{\text{SVP}} \xrightarrow{\text{[CCL18]}} \boxed{1/3\text{-BDD}} \xrightarrow{\text{[DSR14]}} \boxed{\text{DGS}}$$

[ADRS15]
+ new lemma

# Quantum SVP

Classical SVP to BDD: do $3^n$ queries to $1/3$-BDD and keep minimum

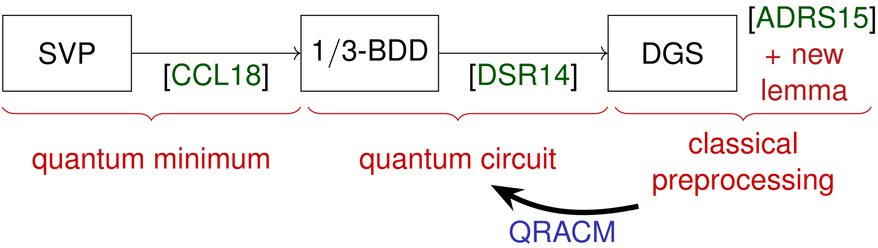



**Theorem**

*There is a quantum algorithm that solves SVP in time $2^{0.9529n+o(n)}$, classical space $2^{0.5n+o(n)}$ and $\mathrm{poly}(n)$ qubits.*

# Quantum SVP

Classical SVP to BDD: do $3^n$ queries to $1/3$-BDD and keep minimum




## Theorem

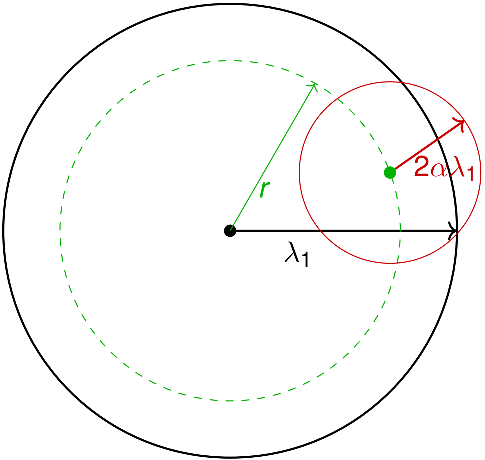
*There is a quantum algorithm that solves SVP in time $2^{0.9529n+o(n)}$, classical space $2^{0.5n+o(n)}$ and $\mathrm{poly}(n)$ qubits.*

Future work: use QRACM to speed-up the query time of the $1/3$-BDD.

$\rightsquigarrow$ time $2^{0.869n+o(n)}$ ?

# Faster SVP to BDD reduction

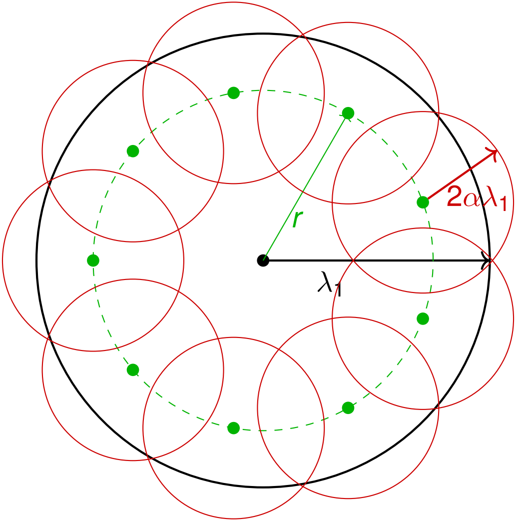
Cover the sphere of radius $\lambda_1(\mathcal{L})$ by balls of radius $2\alpha\lambda_1(\mathcal{L})$:




Use $2^n$ $\alpha-$BDD queries to enumerate points in balls of radius $2\alpha\lambda_1$

# Faster SVP to BDD reduction

Cover the sphere of radius $\lambda_1(\mathcal{L})$ by balls of radius $2\alpha\lambda_1(\mathcal{L})$:

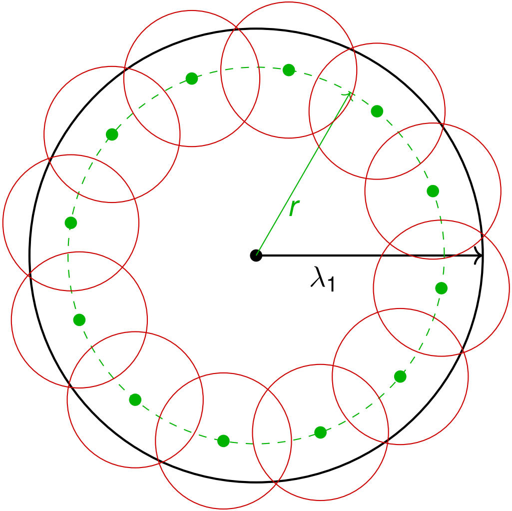



Use $2^n$ $\alpha-$BDD queries to enumerate points in balls of radius $2\alpha\lambda_1$

Each ball covers a spherical cap.

# Faster SVP to BDD reduction

Cover the sphere of radius $\lambda_1(\mathcal{L})$ by balls of radius $2\alpha\lambda_1(\mathcal{L})$:




Use $2^n$ $\alpha-$BDD queries to enumerate points in balls of radius $2\alpha\lambda_1$

Each ball covers a spherical cap.

Smaller $\alpha$:

- More balls
- Less expensive BDD

$\rightsquigarrow$ Trade-off

# Improved classical SVP

Improved SVP to BDD: do $2^n$ queries to 0.4103-BDD

$$
\begin{tikzcd}
\boxed{\text{SVP}} \arrow[r, "\text{new reduction}"'] & \boxed{\text{0.4103-BDD}} \arrow[r, "\text{[DSR14]}"'] & \boxed{\text{DGS}}
\end{tikzcd}
$$

[ADRS15] + new lemma

**Theorem**

*There is a classical algorithm that solves SVP in time $2^{1.7397n+o(n)}$, classical space $2^{0.5n+o(n)}$.*

# Improved classical SVP

Improved SVP to BDD: do $2^n$ queries to 0.4103-BDD

$$\underbrace{\boxed{\text{SVP}} \xrightarrow[\text{new reduction}]{}}_{\text{quantum minimum}} \underbrace{\boxed{0.4103\text{-BDD}} \xrightarrow[\text{[DSR14]}]{}}_{\text{quantum circuit}} \underbrace{\boxed{\text{DGS}}\ \ \begin{array}{c}\text{[ADRS15]}\\ \text{+ new}\\ \text{lemma}\end{array}}_{\text{classical}}$$

**Theorem**

*There is a classical algorithm that solves SVP in time $2^{1.7397n+o(n)}$, classical space $2^{0.5n+o(n)}$.*

**Theorem**

*There is a **quantum** algorithm that solves SVP in time $2^{1.051n+o(n)}$, classical space $2^{0.5n+o(n)}$ and $\mathrm{poly}(n)$ qubits.*

Not as good as our previous $2^{0.9529n+o(n)}$ algorithm but the story does not stop here...

# SVP and Generalized Kissing Number

Number of lattice points in a ball of radius $r$ is $\leqslant c^{n+o(n)} r^n$

$$\beta(\mathcal{L}) = \text{smallest } c \text{ that works for all } r$$

- Upper bound: $\beta(\mathcal{L}) \leqslant 2^{0.401}$ [KL78]
- Conjectured to be $\beta(\mathcal{L}) \approx 1$ for most lattices

# SVP and Generalized Kissing Number

Number of lattice points in a ball of radius $r$ is $\leqslant c^{n+o(n)} r^n$

$$\beta(\mathcal{L}) = \text{smallest } c \text{ that works for all } r$$

- Upper bound: $\beta(\mathcal{L}) \leqslant 2^{0.401}$ [KL78]
- Conjectured to be $\beta(\mathcal{L}) \approx 1$ for most lattices

$$\boxed{\alpha\text{−BDD}} \xrightarrow[\text{reduce}]{} \boxed{\mathrm{DGS}_{\eta_\varepsilon}}$$

Best known relations between $\alpha$ and $\varepsilon$ depends on $\beta(\mathcal{L})$:

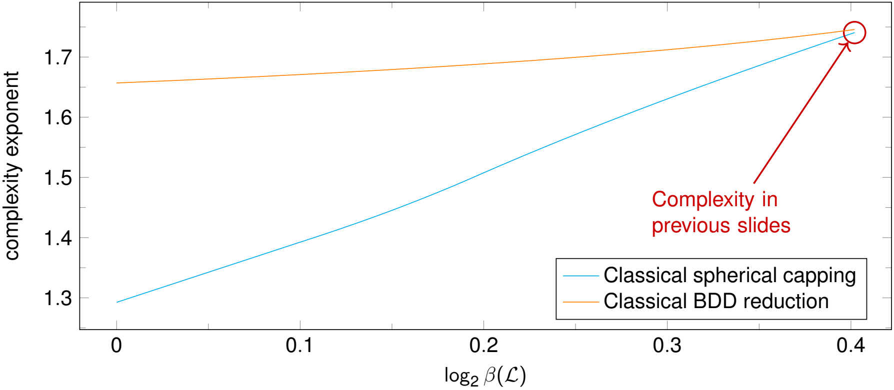
small $\beta(\mathcal{L})$ $\rightsquigarrow$ bigger $\alpha$ for fixed $\varepsilon$ $\rightsquigarrow$ less expensive BDD

# SVP and Generalized Kissing Number

Number of lattice points in a ball of radius $r$ is $\leqslant c^{n+o(n)} r^n$

$$\beta(\mathcal{L}) = \text{smallest } c \text{ that works for all } r$$

- Upper bound: $\beta(\mathcal{L}) \leqslant 2^{0.401}$ [KL78]
- Conjectured to be $\beta(\mathcal{L}) \approx 1$ for most lattices

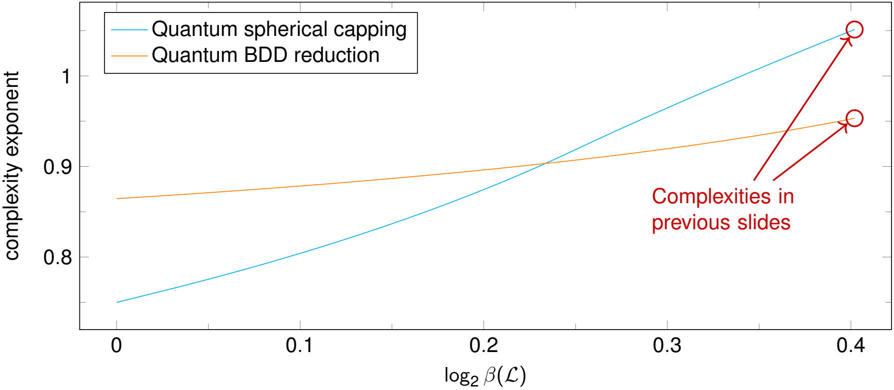


# SVP and Generalized Kissing Number

Number of lattice points in a ball of radius $r$ is $\leqslant c^{n+o(n)} r^n$

$$\beta(\mathcal{L}) = \text{smallest } c \text{ that works for all } r$$

- Upper bound: $\beta(\mathcal{L}) \leqslant 2^{0.401}$ [KL78]
- Conjectured to be $\beta(\mathcal{L}) \approx 1$ for most lattices



# Summary on sieving

Provable SVP:

- classical: time $2^{1.7397n+o(n)}$, space $2^{0.5n+o(n)}$
- quantum: $2^{0.9529n+o(n)}$, space $2^{0.5n+o(n)}$ and $\mathrm{poly}(n)$ qubits
- first time/space tradeoff: time $q^{13n}$, space $q^{16n/q^2}$ for $q \in [4, \sqrt{n}]$
- studied dependency on $\beta(\mathcal{L})$, generalized kissing number

# Summary on sieving

Provable SVP:

- classical: time $2^{1.7397n+o(n)}$, space $2^{0.5n+o(n)}$
- quantum: $2^{0.9529n+o(n)}$, space $2^{0.5n+o(n)}$ and $\mathrm{poly}(n)$ qubits
- first time/space tradeoff: time $q^{13n}$, space $q^{16n/q^2}$ for $q \in [4, \sqrt{n}]$
- studied dependency on $\beta(\mathcal{L})$, generalized kissing number

Open problems:

- Show that random lattices satisfy $\beta(\mathcal{L}) \approx 1$?
- Fill the gap between provable and heuristic algorithms for sieving?
- Exploit the subexponential space regime in our trade-off for SVP?
- $2^{O(n)}$ time, $2^{o(n)}$ space algorithm for DGS at smoothing parameter?

# Subset-Sum

# The Subset-Sum Problem

## Problem

*Given:* $\mathbf{a} = (a_1, \ldots, a_{\mathbf{n}})$ *a vector of integers, and a target* $S$*, find* $I \subseteq \{1, \ldots, n\}$ *such that* $\sum_{i \in I} a_i = S$

- The decision version is NP-complete

# The Subset-Sum Problem

**Problem**

*Given:* $\mathbf{a} = (a_1, \ldots, a_{\mathbf{n}})$ *a vector of integers, and a target* $S$*, find* $I \subseteq \{1, \ldots, n\}$ *such that* $\sum_{i \in I} a_i = S \mod 2^\ell$

- The decision version is NP-complete

# The Subset-Sum Problem

**Problem**

*Given:* $\mathbf{a} = (a_1, \ldots, a_{\mathbf{n}})$ *a vector of integers, and a target* $S$*, find* $I \subseteq \{1, \ldots, n\}$ *such that* $\sum_{i \in I} a_i = S \mod 2^\ell$ ***where a and S are chosen uniformly at random****.*

- The decision version is NP-complete
- Both cases $\ell \gg \mathbf{n}$ and $\ell \ll \mathbf{n}$ are solvable efficiently
- The case $\ell \simeq \mathbf{n}$ is hard

# The Subset-Sum Problem

**Problem**

*Given:* $\mathbf{a} = (a_1, \ldots, a_{\mathbf{n}})$ *a vector of integers, and a target* $S$*, find* $I \subseteq \{1, \ldots, n\}$ *such that* $\sum_{i \in I} a_i = S \mod 2^\ell$ ***where a and S are chosen uniformly at random.***

For $\ell = n$ (hard case):

- Classical and quantum algorithms run in time $\widetilde{\mathcal{O}}(2^{\gamma n})$
- Used as a hard problem for post-quantum cryptography [Lyu10]
- Similar techniques also apply to other problems (syndrome decoding problem) [KT17]
- Solving subset-sums is also useful in quantum hidden shift algorithms [Bon19]

# Result in the Classical Setting

The time is $\widetilde{\mathcal{O}}(2^{\gamma n})$.

| $\gamma$ | Ref. |
|---|---|
| 0.5 | [HS74] |
| 0.5 | [SS81] |
| 0.3370 | [HGJ10] |
| 0.2909 | [BCJ11] |
| 0.283 | This thesis |

# Results in the Quantum Setting

The time is $\widetilde{\mathcal{O}}\left(2^{\gamma n}\right)$.

| $\gamma$ | Ref. | Model |
|---|---|---|
| 0.3334 | [BHT98] | QRACM |
| 0.3 | [BJLM13] | QRAQM |
| 0.241 | [BJLM13] | QRAQM + conj. |
| 0.2356 | This thesis | QRACM |
| 0.226 | [HM18] | QRAQM + conj. |
| | | |
| | | |

# Results in the Quantum Setting

The time is $\widetilde{\mathcal{O}}\left(2^{\gamma n}\right)$.

| $\gamma$ | Ref. | Model |
|---|---|---|
| 0.3334 | [BHT98] | QRACM |
| 0.3 | [BJLM13] | QRAQM |
| 0.241 | [BJLM13] | QRAQM ~~+ conj.~~ |
| 0.2356 | This thesis | QRACM |
| 0.226 | [HM18] | QRAQM ~~+ conj.~~ |
| 0.2182 | This thesis | QRAQM |
| 0.2156 | This thesis | QRAQM + conj. |

# Results in the Quantum Setting

The time is $\widetilde{\mathcal{O}}(2^{\gamma n})$.

| $\gamma$ | Ref. | Model |
|---|---|---|
| 0.3334 | [BHT98] | QRACM |
| 0.3 | [BJLM13] | QRAQM |
| 0.241 | [BJLM13] | QRAQM ~~+ conj.~~ |
| 0.2356 | This thesis | QRACM |
| 0.226 | [HM18] | QRAQM ~~+ conj.~~ |
| 0.2182 | This thesis | QRAQM |
| 0.2156 | This thesis | QRAQM + conj. |

Open problems:

- Remove conjecture
- How far can we push the representation method?

# Conclusion and Future Work

- quantum quadratic speedup of enumeration
- provable time/space trade-off for SVP
- improved algorithms for provable sieving
- improved algorithms for subset-sum

Open problems:

- Can we show that random lattices satisfy $\beta(\mathcal{L}) \approx 1$ ?
- Fill the gap between provable and heuristic algorithms for sieving
- Exploit the subexponential space regime in our trade-off for SVP?
- $2^{O(n)}$ time $2^{o(n)}$ space algorithm for DGS at smoothing parameter
- Study polynomial factors for sieving and enumeration
- More studies on discrete pruning are needed
- Remove conjecture in subset-sum quantum walk