On the practicality of quantum sieving algorithms for the shortest vector problem

João F. Doriguello, George Giapitzakis, Alessandro Luongo, Aditya Morolia[1]

May 2, 2026

[1]Center for Quantum Technologies, Singapore

How Peter Shor's Algorithm Dooms RSA Encryption to Failure

In 1994, Peter Shor created an algorithm for a theorical computer that solved a nearly impossible problem. Now that technology is catching up, Shor's algorithm guarantees the end to RSA Encryption.

Updated: Mar 12, 2025 06:40 AM EST

BLOG FEATURED CONTENT QUANTUM COMPUTING

Quantum computing and the future of cryptography: Understanding the imminent threat

Post-Quantum

Shor's Algorithm: A Quantum Threat to Modern Cryptography

Marin Ivezic · October 18, 2017 · 48 minutes read

NEWS | PHYSICS

'Surprising and super cool.' Quantum algorithm offers faster way to hack internet encryption

Scheme to factor giant numbers could be more efficient than 30-year-old Shor's algorithm

19 SEP 2023 · 3:25 PM ET · BY ANNA KRAMER

.0 ▲2.18% Ethereum ETH $2229.24 ▲1.71% XRP XRP $1.34 ▲1.18% Solan

HOME / CRYPTO / MARKETS

Google warns quantum attack could crack Bitcoin in 9 minutes

Google's new whitepaper says it could take only minutes for a quantum system to crack Bitcoin.

ANAND SINHA • 1 HOUR AGO

REVIEWED BY MEHAB QURESHI

Post-quantum cryptography

- New cryptographic schemes (allegedly) resilient against quantum computers

Post-quantum cryptography

- New cryptographic schemes (allegedly) resilient against quantum computers
- Several candidates: lattice-based, multivariate, hash-based, code-based, isogeny-based, etc.

Post-quantum cryptography

- New cryptographic schemes (allegedly) resilient against quantum computers
- Several candidates: lattice-based, multivariate, hash-based, code-based, isogeny-based, etc.

Post-quantum lattice-based cryptography

History of Selected Algorithms Updates

Selected Algorithms: Key-Encapsulation Mechansims

Algorithm	Algorithm Information	Submitters	Comments
CRYSTALS-KYBER (2022) FIPS 203 **PQC License Summary & Excerpts**	Zip File (7MB) IP Statements Website	Peter Schwabe Roberto Avanzi Joppe Bos Leo Ducas Eike Kiltz Tancrede Lepoint Vadim Lyubashevsky John M. Schanck Gregor Seiler Damien Stehle Jintai Ding	Submit Comment View Comments

Selected Algorithms: Digital Signature Algorithms

Algorithm	Algorithm Information	Submitters	Comments
CRYSTALS-DILITHIUM (2022) FIPS 204	Zip File (11MB) IP Statements Website	Vadim Lyubashevsky Leo Ducas Eike Kiltz Tancrede Lepoint Peter Schwabe Gregor Seiler Damien Stehle Shi Bai	Submit Comment View Comments

Lattice

Definition

A **lattice** in d dimensions is a discrete subgroup of $\mathbb{R}^d$.

Lattice

Definition

A **lattice** in d dimensions is a discrete subgroup of $\mathbb{R}^d$.

$$B := (b_1, b_2, \ldots b_n), \quad b_i \in \mathbb{R}^d$$

$$\mathcal{L}(B) := \left\{ \sum_i z_i b_i : z_i \in \mathbb{Z} \right\}$$

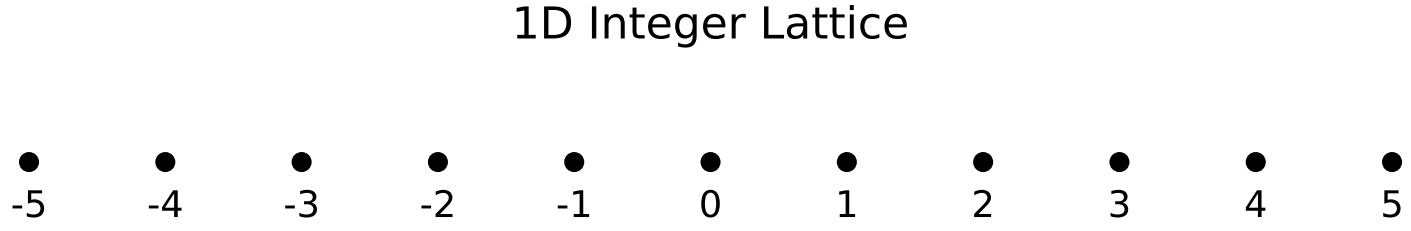
Lattice

Definition

A **lattice** in d dimensions is a discrete subgroup of $\mathbb{R}^d$.

$$B := (b_1, b_2, \ldots b_n), \quad b_i \in \mathbb{R}^d$$

$$\mathcal{L}(B) := \left\{ \sum_i z_i b_i : z_i \in \mathbb{Z} \right\}$$



Lattice

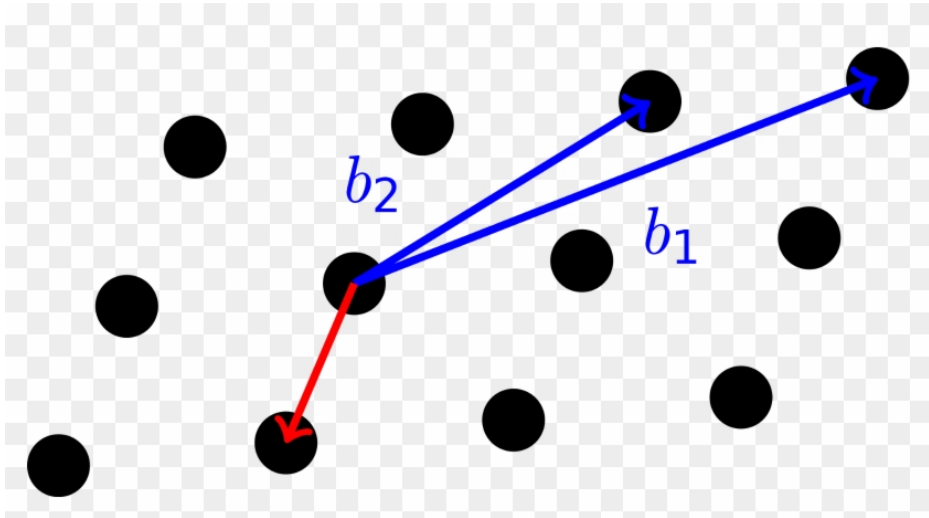
Shortest Vector Problem (SVP)

Shortest Vector Problem (SVP)

- **Input:** A matrix $B \in \mathbb{Q}^{d \times n}$.

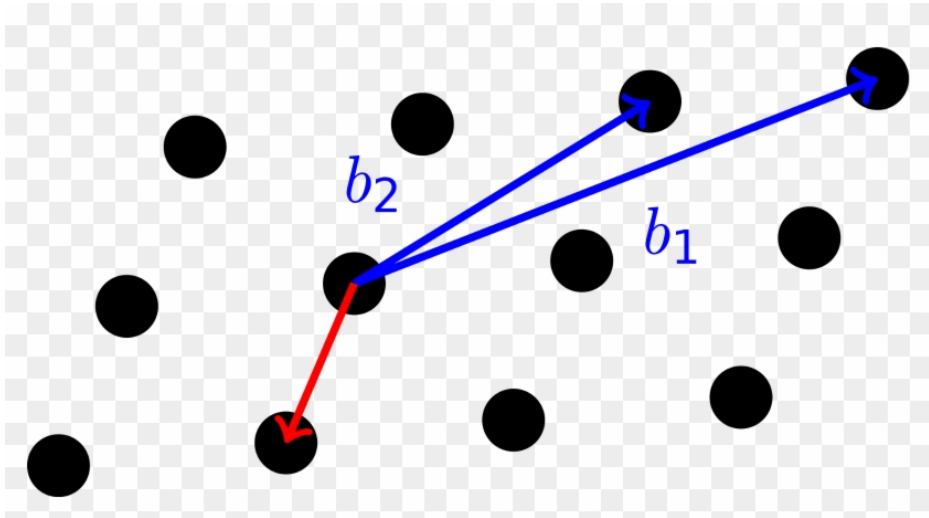
Shortest Vector Problem (SVP)

- **Input:** A matrix $B \in \mathbb{Q}^{d \times n}$.
- **Output:** The shortest *non-zero* lattice vector in $\mathcal{L}(B)$.
- Shortest in the ℓ_2 norm.



Shortest Vector Problem (SVP)

- **Input:** A matrix $B \in \mathbb{Q}^{d \times n}$.
- **Output:** The shortest *non-zero* lattice vector in $\mathcal{L}(B)$.
- Shortest in the ℓ_2 norm.



- Believed to be hard for quantum computers

The Question

Are quantum speedups for SVP cryptographically relevant?

The Question

Are quantum speedups for SVP cryptographically relevant?

- Best algorithms to attack lattice crypto try to find a "good basis" for the lattice. 😈

The Question

Are quantum speedups for SVP cryptographically relevant?

- Best algorithms to attack lattice crypto try to find a "good basis" for the lattice. 😈
- Use approximate or exact SVP algorithm with the BKZ basis reduction algorithm.

The Question

Are quantum speedups for SVP cryptographically relevant?

- Best algorithms to attack lattice crypto try to find a "good basis" for the lattice. 😈
- Use approximate or exact SVP algorithm with the BKZ basis reduction algorithm.
- Asymptotic quantum speedups are known using Grover's search.

The Question

Are quantum speedups for SVP cryptographically relevant?

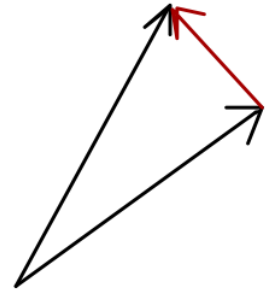
- Best algorithms to attack lattice crypto try to find a "good basis" for the lattice. 😈
- Use approximate or exact SVP algorithm with the BKZ basis reduction algorithm.
- Asymptotic quantum speedups are known using Grover's search.
- What happens when you actually instantiate the Grover oracle, account for the QRAM and error correction?

The Algorithm: Lattice sieving. Fastest in practice. Poorly understood theoretically. Let's pretend we are cryptographers 😏

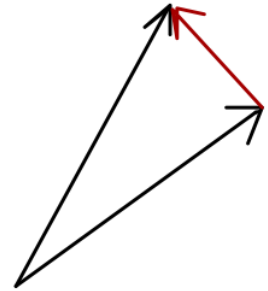
HALL OF FAME

Position	Dimension	Euclidean Norm	Seed	Contestant	Solution	Algorithm	Subm. Date	Approx. Factor
1	210	3808	0	Jintai Ding, Ziyu Zhao	vec	Sieving	2026-01-1	1.04497
2	200	3723	0	Ziyu Zhao, Jintai Ding	vec	Sieving	2025-03-4	1.04865
3	190	3613	0	Yao Sun and Shuai Chang	vec	Sieving	2024-07-21	1.04237
4	188	3582	0	Leizhang Wang, Lin Wang, Baocang Wang, Geng Wang, Huiwen Jia	vec	Sieving	2025-03-12	1.03823
5	187	3582	0	Shi Bai, Yuanmi Chen, Jiangxia Ge, Yituo He, Zhuo Huang, Juanru Li, Xiangxue Li, Chao Sun, Yu Yu	vec	Sieving	2026-01-20	1.04132

Lattice Sieving: [AKS01, NV08, MV10]

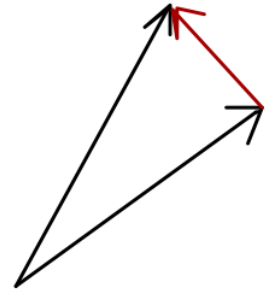


Lattice Sieving: [AKS01, NV08, MV10]



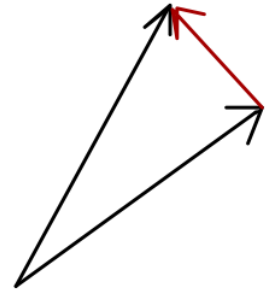
1. Sample a bunch of long lattice vectors.

Lattice Sieving: [AKS01, NV08, MV10]



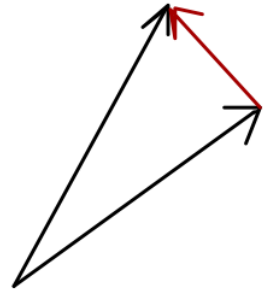
1. Sample a bunch of long lattice vectors.
2. Find $\|\vec{v} - \vec{w}\| < \max(\|\vec{v}\|, \|\vec{w}\|)$

Lattice Sieving: [AKS01, NV08, MV10]



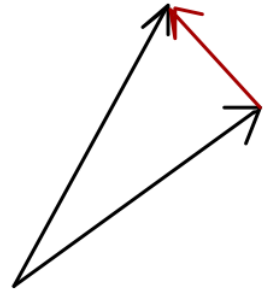
1. Sample a bunch of long lattice vectors.
2. Find $\|\vec{v} - \vec{w}\| < \max(\|\vec{v}\|, \|\vec{w}\|)$
3. Keep difference in new list, throw the parents, goto step 2.

Lattice Sieving: [AKS01, NV08, MV10]



1. Sample a bunch of long lattice vectors.
2. Find $\|\vec{v} - \vec{w}\| < \max(\|\vec{v}\|, \|\vec{w}\|)$
3. Keep difference in new list, throw the parents, goto step 2.
4. If initial list was large enough ($2^{\Theta(n)}$), after $\mathsf{poly}(n)$ steps, list has shortest vector w.h.p.

Lattice Sieving: [AKS01, NV08, MV10]

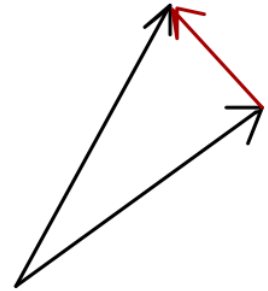


1. Sample a bunch of long lattice vectors.
2. Find $\|\vec{v} - \vec{w}\| < \max(\|\vec{v}\|, \|\vec{w}\|)$
3. Keep difference in new list, throw the parents, goto step 2.
4. If initial list was large enough ($2^{\Theta(n)}$), after $\mathsf{poly}(n)$ steps, list has shortest vector w.h.p.

Remarks

- Can do step 1 in $\mathsf{poly}(n)$ time (for each vector)

Lattice Sieving: [AKS01, NV08, MV10]

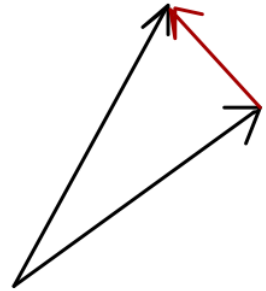


1. Sample a bunch of long lattice vectors.
2. Find $\|\vec{v} - \vec{w}\| < \max(\|\vec{v}\|, \|\vec{w}\|)$
3. Keep difference in new list, throw the parents, goto step 2.
4. If initial list was large enough ($2^{\Theta(n)}$), after $\mathsf{poly}(n)$ steps, list has shortest vector w.h.p.

Remarks

- Can do step 1 in $\mathsf{poly}(n)$ time (for each vector)
- Step 2 is rate limiting.

Lattice Sieving: [AKS01, NV08, MV10]

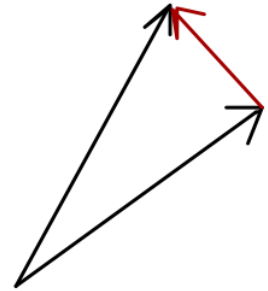


1. Sample a bunch of long lattice vectors.
2. Find $\|\vec{v} - \vec{w}\| < \max(\|\vec{v}\|, \|\vec{w}\|)$
3. Keep difference in new list, throw the parents, goto step 2.
4. If initial list was large enough ($2^{\Theta(n)}$), after $\mathsf{poly}(n)$ steps, list has shortest vector w.h.p.

Remarks

- Can do step 1 in $\mathsf{poly}(n)$ time (for each vector)
- Step 2 is rate limiting.
- Progress: $\|\vec{v} - \vec{w}\|^2 \leq \max(\|\vec{v}\|^2, \|\vec{w}\|^2)(1 - 2\cos\theta)$

Lattice Sieving: [AKS01, NV08, MV10]

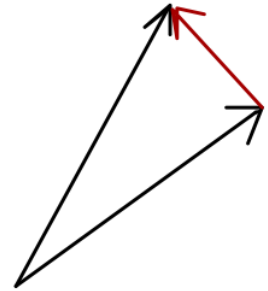


1. Sample a bunch of long lattice vectors.
2. Find $\|\vec{v} - \vec{w}\| < \max(\|\vec{v}\|, \|\vec{w}\|)$
3. Keep difference in new list, throw the parents, goto step 2.
4. If initial list was large enough ($2^{\Theta(n)}$), after $\mathsf{poly}(n)$ steps, list has shortest vector w.h.p.

Remarks

- Can do step 1 in $\mathsf{poly}(n)$ time (for each vector)
- Step 2 is rate limiting.
- Progress: $\|\vec{v} - \vec{w}\|^2 \leq \max(\|\vec{v}\|^2, \|\vec{w}\|^2)(1 - 2\cos\theta)$
- $\cos\theta > \frac{1}{2}$ enough; implied by $\theta < \frac{\pi}{3}$.

Lattice Sieving: [AKS01, NV08, MV10]



1. Sample a bunch of long lattice vectors.
2. Find $\|\vec{v} - \vec{w}\| < \max(\|\vec{v}\|, \|\vec{w}\|)$
3. Keep difference in new list, throw the parents, goto step 2.
4. If initial list was large enough ($2^{\Theta(n)}$), after $\mathsf{poly}(n)$ steps, list has shortest vector w.h.p.

Remarks

- Can do step 1 in $\mathsf{poly}(n)$ time (for each vector)
- Step 2 is rate limiting.
- Progress: $\|\vec{v} - \vec{w}\|^2 \leq \max(\|\vec{v}\|^2, \|\vec{w}\|^2)(1 - 2\cos\theta)$
- $\cos\theta > \frac{1}{2}$ enough; implied by $\theta < \frac{\pi}{3}$.

Given a really long list of vectors, find me pairs of close vectors.

Given a really long list of vectors L, find me pairs of close vectors.

Bruteforce search. GaussSieve, $2^{0.415n+o(n)}$

Given a really long list of vectors L, find me pairs of close vectors.

Bruteforce search. GaussSieve, $2^{0.415n+o(n)}$

Nearest neighbor search. Given $\vec{v}$, set L, find me vector closest to $\vec{v}$ in L.

Given a really long list of vectors L, find me pairs of close vectors.

Bruteforce search. GaussSieve, $2^{0.415n+o(n)}$

Nearest neighbor search. Given $\vec{v}$, set L, find me vector closest to $\vec{v}$ in L.

- Can use locality sensitive hashing (LSH); time $|L|^{\rho+o(n)}$, $\rho < 1$.

Given a really long list of vectors L, find me pairs of close vectors.

Bruteforce search. GaussSieve, $2^{0.415n+o(n)}$

Nearest neighbor search. Given $\vec{v}$, set L, find me vector closest to $\vec{v}$ in L.

- Can use locality sensitive hashing (LSH); time $|L|^{\rho+o(n)}$, $\rho < 1$.
- Locality sensitive filtering [BDGL16], time $2^{0.292n+o(n)}$.

Given a really long list of vectors L, find me pairs of close vectors.

Bruteforce search. GaussSieve, $2^{0.415n+o(n)}$

Nearest neighbor search. Given $\vec{v}$, set L, find me vector closest to $\vec{v}$ in L.

- Can use locality sensitive hashing (LSH); time $|L|^{\rho+o(n)}$, $\rho < 1$.
- Locality sensitive filtering [BDGL16], time $2^{0.292n+o(n)}$.
- Grover's search + LSF [LMP15], time $2^{0.265n+o(n)}$.

Given a really long list of vectors L, find me pairs of close vectors.

Bruteforce search. GaussSieve, $2^{0.415n+o(n)}$

Nearest neighbor search. Given $\vec{v}$, set L, find me vector closest to $\vec{v}$ in L.

- Can use locality sensitive hashing (LSH); time $|L|^{\rho+o(n)}$, $\rho < 1$.
- Locality sensitive filtering [BDGL16], time $2^{0.292n+o(n)}$.
- Grover's search + LSF [LMP15], time $2^{0.265n+o(n)}$.

Instantiate the oracle? Beyond asymptotics? What about dimension, say, 400? Probably more important for quantum computers?? Tech not so mature yet.

Quantum speedups for SVP? – Prior Work

- [AGPS20] initiated such a study.

Quantum speedups for SVP? – Prior Work

- [AGPS20] initiated such a study.
- Quantum circuit to implement nearest neighbor search using a "population count" filter.

Quantum speedups for SVP? – Prior Work

- [AGPS20] initiated such a study.
- Quantum circuit to implement nearest neighbor search using a "population count" filter.
- Grover's algorithm to search the filtered list.

Quantum speedups for SVP? – Prior Work

- [AGPS20] initiated such a study.
- Quantum circuit to implement nearest neighbor search using a "population count" filter.
- Grover's algorithm to search the filtered list.
- Simple QRAM, error correction model; not enough error correction.

Quantum speedups for SVP? – Prior Work

- [AGPS20] initiated such a study.
- Quantum circuit to implement nearest neighbor search using a "population count" filter.
- Grover's algorithm to search the filtered list.
- Simple QRAM, error correction model; not enough error correction.
- Minimal speedup in cryptographically relevant dimensions.

Quantum speedups for SVP? – Our Work

- Substantially improved analysis with very optimistic but realistic parameters.

Quantum speedups for SVP? – Our Work

- Substantially improved analysis with very optimistic but realistic parameters.
- Better circuits for quantum arithmetic like multiplication, QRAM.

Quantum speedups for SVP? – Our Work

- Substantially improved analysis with very optimistic but realistic parameters.
- Better circuits for quantum arithmetic like multiplication, QRAM.
- Better error correction model; 3 level magic state distillation.

Quantum speedups for SVP? – Our Work

- Substantially improved analysis with very optimistic but realistic parameters.
- Better circuits for quantum arithmetic like multiplication, QRAM.
- Better error correction model; 3 level magic state distillation.
- Concrete physical architectures: nearest neighbor connectivity (Google's Sycamore) and active volume architecture [LN22] with concrete error correction model.

Quantum speedups for SVP? – Our Work

- Substantially improved analysis with very optimistic but realistic parameters.
- Better circuits for quantum arithmetic like multiplication, QRAM.
- Better error correction model; 3 level magic state distillation.
- Concrete physical architectures: nearest neighbor connectivity (Google's Sycamore) and active volume architecture [LN22] with concrete error correction model.
- Result: Little to no quantum speedup in dimension 400 for GaussSieve.

But what is Quantum?

- Quantum computers consume “Magic States” to apply some gates (Toffoli gates). Usually the most expensive.

But what is Quantum?

- Quantum computers consume "Magic States" to apply some gates (Toffoli gates). Usually the most expensive.
- Reaction time: duration required to perform a layer of measurements, process the outcomes through a classical decoder (using algorithms such as minimum-weight perfect matching or union-find), and transmit new instructions back to the quantum hardware.

But what is Quantum?

- Quantum computers consume "Magic States" to apply some gates (Toffoli gates). Usually the most expensive.
- Reaction time: duration required to perform a layer of measurements, process the outcomes through a classical decoder (using algorithms such as minimum-weight perfect matching or union-find), and transmit new instructions back to the quantum hardware.
- Reaction depth: Number of such reactive measurements.
- Active volume: Counts the cost of expensive operations such as Toffoli gates and magic state distillation.

Fixed Point Arithmetic

- Two's-complement representation with $\kappa = 32$ (qu)bits.
- Overflows are ignored.
- Grover oracle decomposed into addition, comparison, and multiplication operations.

Fixed Point Arithmetic

- Two's-complement representation with $\kappa = 32$ (qu)bits.
- Overflows are ignored.
- Grover oracle decomposed into addition, comparison, and multiplication operations.
- Addition: Gidney's out-of-place quantum adder (Quantum'18).

Fixed Point Arithmetic

- Two's-complement representation with $\kappa = 32$ (qu)bits.
- Overflows are ignored.
- Grover oracle decomposed into addition, comparison, and multiplication operations.
- Addition: Gidney's out-of-place quantum adder (Quantum'18).
- Comparison = addition.

Fixed Point Arithmetic

- Two's-complement representation with $\kappa = 32$ (qu)bits.
- Overflows are ignored.
- Grover oracle decomposed into addition, comparison, and multiplication operations.
- Addition: Gidney's out-of-place quantum adder (Quantum'18).
- Comparison = addition.
- Multiplication: novel quantum circuit based on schoolbook multiplication with lower Toffoli-count (similar to recent work by Daniel Litinski arXiv:2410.00899).

Circuit / Resource	Toffoli-count	Toffoli-width	Reaction depth	Qubit-width	Active volume
Adder/Comparator	$\kappa - 1$	1	$2(\kappa - 1)$	3κ	$(\kappa - 1)(39 + C_{\lvert CCZ\rangle}) + 7$
Controlled adder	$2\kappa - 1$	κ	2κ	$4\kappa + 1$	$(\kappa - 1)(51 + C_{\lvert CCZ\rangle}) + 19$
Multiplier	$\kappa^2 - \kappa + 1$	$0.5\kappa^2 + 0.5\kappa$	$2\kappa \log_2 \kappa - 2\kappa - 2\log_2 \kappa + 4$	$2\kappa^2 + \kappa$	$28\kappa^2 - 42\kappa + 28$ $+(\kappa^2 - \kappa + 1)\, C_{\lvert CCZ\rangle}$
Multiplier (hybrid)	$0.5\kappa^2 - 1.5\kappa + 1$	0.5κ	$2\kappa \log_2 \kappa - 2\kappa - 2\log_2 \kappa + 2$	$1.5\kappa^2 + 0.5\kappa$	$20.25\kappa^2 - 48.75\kappa + 32$ $+(0.5\kappa^2 - 1.5\kappa + 1)\, C_{\lvert CCZ\rangle}$

Grover's Search

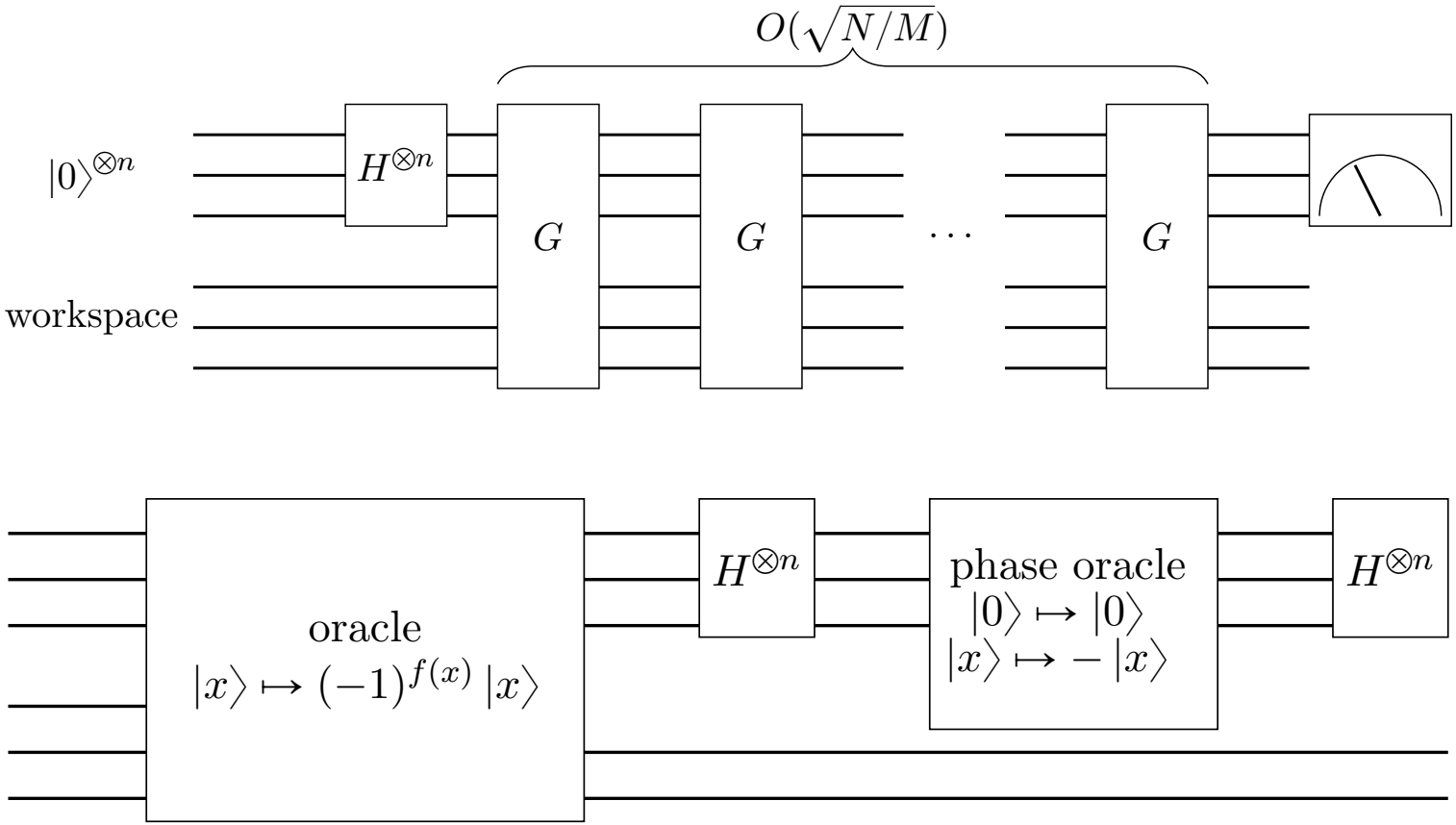


Figure: Circuit for Grover's search algorithm (top) and the Grover oracle G (bottom).

Non-Asymptotic Grover's Search

- Grover's search requires $\left\lfloor \frac{\pi}{4}\sqrt{N/M} \right\rfloor$ iterations to find one out of M solutions in a list of N elements if M is known beforehand.

Non-Asymptotic Grover's Search

- Grover's search requires $\left\lfloor \frac{\pi}{4}\sqrt{N/M} \right\rfloor$ iterations to find one out of M solutions in a list of N elements if M is known beforehand.
- We do not assume to know the number of solutions M to any search!

Non-Asymptotic Grover's Search

- Grover's search requires $\left\lfloor \frac{\pi}{4}\sqrt{N/M} \right\rfloor$ iterations to find one out of M solutions in a list of N elements if M is known beforehand.
- We do not assume to know the number of solutions M to any search!
- Exponential Grover's search is required [BBHT98].

Non-Asymptotic Grover's Search

- Grover's search requires $\left\lfloor \frac{\pi}{4}\sqrt{N/M} \right\rfloor$ iterations to find one out of M solutions in a list of N elements if M is known beforehand.
- We do not assume to know the number of solutions M to any search!
- Exponential Grover's search is required [BBHT98].
- Precise resource estimate of exponential Grover's search was done by Cade, Folkertsma, Niesen, and Weggemans (Quantum'23).

Inside the Oracle

For GaussSieve we need two Grover oracles $|x\rangle \mapsto (-1)^{f(x)} |x\rangle$:

1. Find $i \in L$ such that $\|\vec{v} \pm \vec{w}_i\| < \|\vec{v}\| \iff \vec{w}_i \cdot (\vec{w}_i \pm 2\vec{v}) < 0$.
2. Find $i \in L$ such that $\|\vec{v} \pm \vec{w}_i\| < \|\vec{w}_i\| \iff |\vec{v} \cdot \vec{w}_i| \geq \|\vec{v}\|^2 / 2$.

Inside the Oracle

For GaussSieve we need two Grover oracles $|x\rangle \mapsto (-1)^{f(x)} |x\rangle$:

1. Find $i \in L$ such that $\|\vec{v} \pm \vec{w}_i\| < \|\vec{v}\| \iff \vec{w}_i \cdot (\vec{w}_i \pm 2\vec{v}) < 0$.
2. Find $i \in L$ such that $\|\vec{v} \pm \vec{w}_i\| < \|\vec{w}_i\| \iff |\vec{v} \cdot \vec{w}_i| \geq \|\vec{v}\|^2 / 2$.

Example for $f_{\text{gauss}} : L \to \{0, 1\}$:

$$f_{\text{gauss}}(i) = 1 \iff \vec{w}_i \cdot (\vec{w}_i - 2\vec{v}) < 0.$$

Inside the Oracle

For GaussSieve we need two Grover oracles $|x\rangle \mapsto (-1)^{f(x)} |x\rangle$:

1. Find $i \in L$ such that $\|\vec{v} \pm \vec{w}_i\| < \|\vec{v}\| \iff \vec{w}_i \cdot (\vec{w}_i \pm 2\vec{v}) < 0$.
2. Find $i \in L$ such that $\|\vec{v} \pm \vec{w}_i\| < \|\vec{w}_i\| \iff |\vec{v} \cdot \vec{w}_i| \geq \|\vec{v}\|^2 / 2$.

Example for $f_{\text{gauss}} : L \to \{0, 1\}$:

$$f_{\text{gauss}}(i) = 1 \iff \vec{w}_i \cdot (\vec{w}_i - 2\vec{v}) < 0.$$

Fix $\vec{v} \in L$. The Grover oracle $|i\rangle \mapsto (-1)^{f_{\text{gauss}}(i)} |i\rangle$ is constructed as:

$$\begin{aligned}
|i\rangle |0\rangle^{\otimes \kappa(1+3D)} &\xrightarrow{\text{QRAM}} |i\rangle |\vec{w}_i\rangle |0\rangle^{\otimes \kappa(1+2D)} \\
&\xrightarrow{D \text{ adders}} |i\rangle |\vec{w}_i\rangle |\vec{w}_i - 2\vec{v}\rangle |0\rangle^{\otimes \kappa(1+D)} \\
&\xrightarrow{D \text{ multipliers}} |i\rangle |\vec{w}_i\rangle |\vec{w}_i - 2\vec{v}\rangle \left(\bigotimes_{j=1}^{D} |(\vec{w}_i)_j (\vec{w}_i - 2\vec{v})_j\rangle \right) |0\rangle^{\otimes \kappa} \\
&\xrightarrow{D-1 \text{ adders}} |i\rangle |\vec{w}_i\rangle |\vec{w}_i - 2\vec{v}\rangle \left(\bigotimes_{j=1}^{D} |(\vec{w}_i)_j (\vec{w}_i - 2\vec{v})_j\rangle \right) |\vec{w}_i^{\top} (\vec{w}_i - 2\vec{v})\rangle .
\end{aligned}$$

Sieve/Operations	QRAM	Adders	Multipliers	Extra CNOTs
`NVSieve`	1	$2D$	D	0
`NVSieve` + LSH/LSF	1	$2D$	D	0
`GaussSieve`	1 1	$4D-2$ $D+1$	$2D$ D^*	$2D\kappa+4$ 4
`GaussSieve` + LSH/LSF	1 1	$4D-2$ $D+1$	$2D$ D^*	$2D\kappa+4$ 4

Quantum RAM 1

- Use QRAM to quantumly access the list of vectors within the sieving algorithms.

Quantum RAM 1

- Use QRAM to quantumly access the list of vectors within the sieving algorithms.
- QRAM: a device that stores classical, indexed data $\{x_i \in \{0,1\}^\kappa : i \in [2^n]\}$

Quantum RAM 1

- Use QRAM to quantumly access the list of vectors within the sieving algorithms.
- QRAM: a device that stores classical, indexed data $\{x_i \in \{0,1\}^\kappa : i \in [2^n]\}$
- Allows oracle queries

$$\text{QRAM} : |i\rangle |0\rangle^{\otimes \kappa} |\overline{0}\rangle \mapsto |i\rangle |x_i\rangle |\text{garbage}_i\rangle , \quad \forall i \in [2^n].$$

Quantum RAM 2

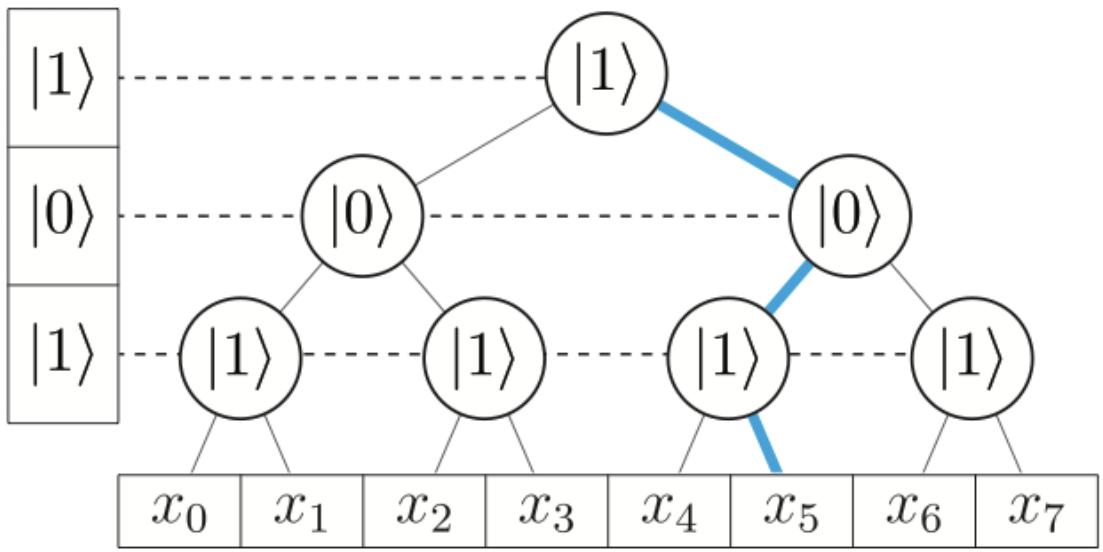
- We use the bucket-brigade circuit from Arunachalam et al. [AGCMS'15]:

Quantum RAM 2

- We use the bucket-brigade circuit from Arunachalam et al. [AGCMS'15]: Simple, noise resilient, fairly low Toffoli counts.
- QRAM is an external device that accesses a classical memory tape:

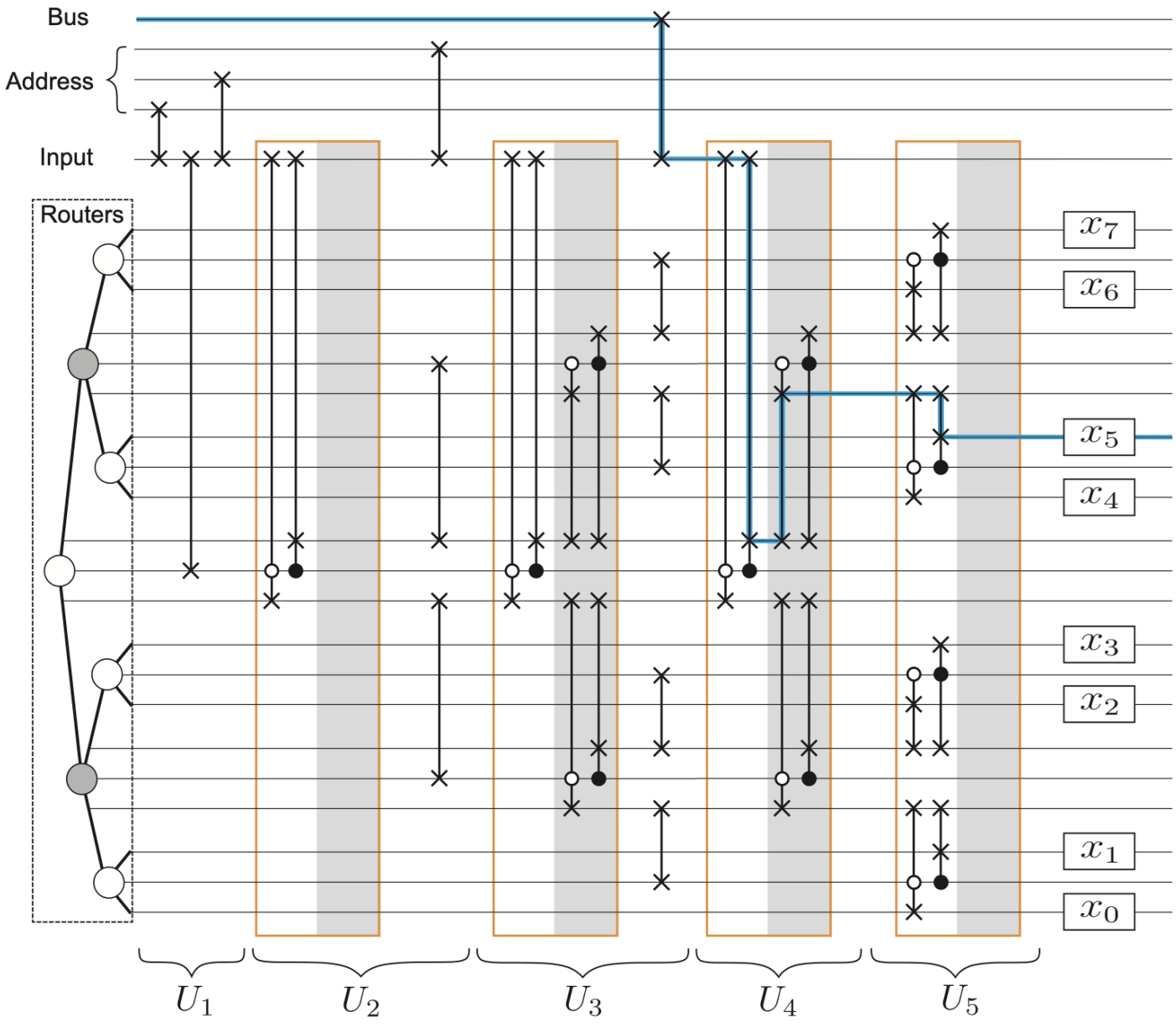
Quantum RAM 2

- We use the bucket-brigade circuit from Arunachalam et al. [AGCMS'15]: Simple, noise resilient, fairly low Toffoli counts.
- QRAM is an external device that accesses a classical memory tape: Content is stored statically and can be classically rewritten without affecting the QRAM device.

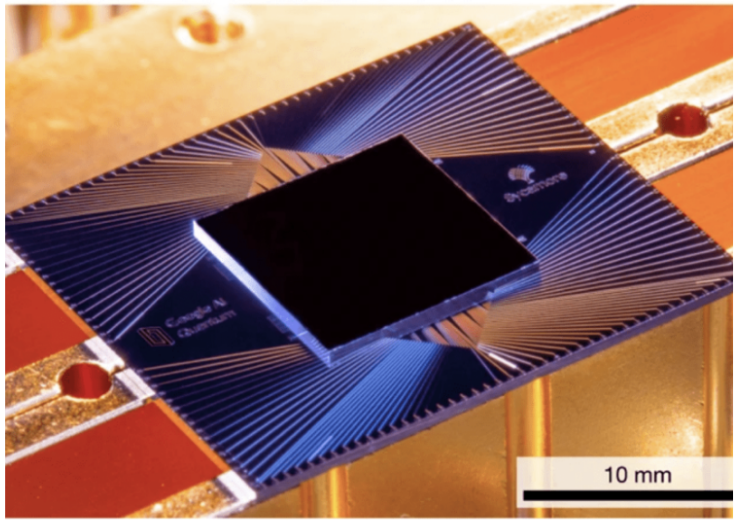


Pictures from Hann et al., PRX Quantum (2021)

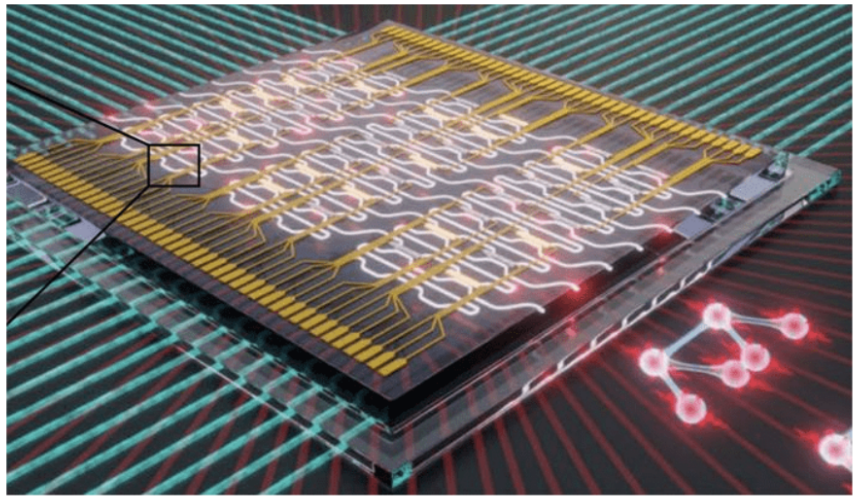
Quantum RAM 3



Quantum Error Correction 1



Baseline architecture



Active-volume architecture

Quantum Error Correction 2

- We take into account environmental noise and employ quantum error correction to protect the quantum circuits against it.

Quantum Error Correction 2

- We take into account environmental noise and employ quantum error correction to protect the quantum circuits against it.
- We employ:
  - Patch-based surface-code encodings for logical qubits [HFDM12];

Quantum Error Correction 2

- We take into account environmental noise and employ quantum error correction to protect the quantum circuits against it.
- We employ:
  - Patch-based surface-code encodings for logical qubits [HFDM12];
  - Magic distillation protocols from Litinski [Lit19,LN22]: two 15-to-1 punctured Reed-Muller code followed by a 8-to-CCZ distillation protocol [GF19].

Quantum Error Correction 2

- We take into account environmental noise and employ quantum error correction to protect the quantum circuits against it.
- We employ:
  - Patch-based surface-code encodings for logical qubits [HFDM12];
  - Magic distillation protocols from Litinski [Lit19,LN22]: two 15-to-1 punctured Reed-Muller code followed by a 8-to-CCZ distillation protocol [GF19].
- We assume:
  - Incoherent circuit-level noise for physical qubits with error 10^{-5};

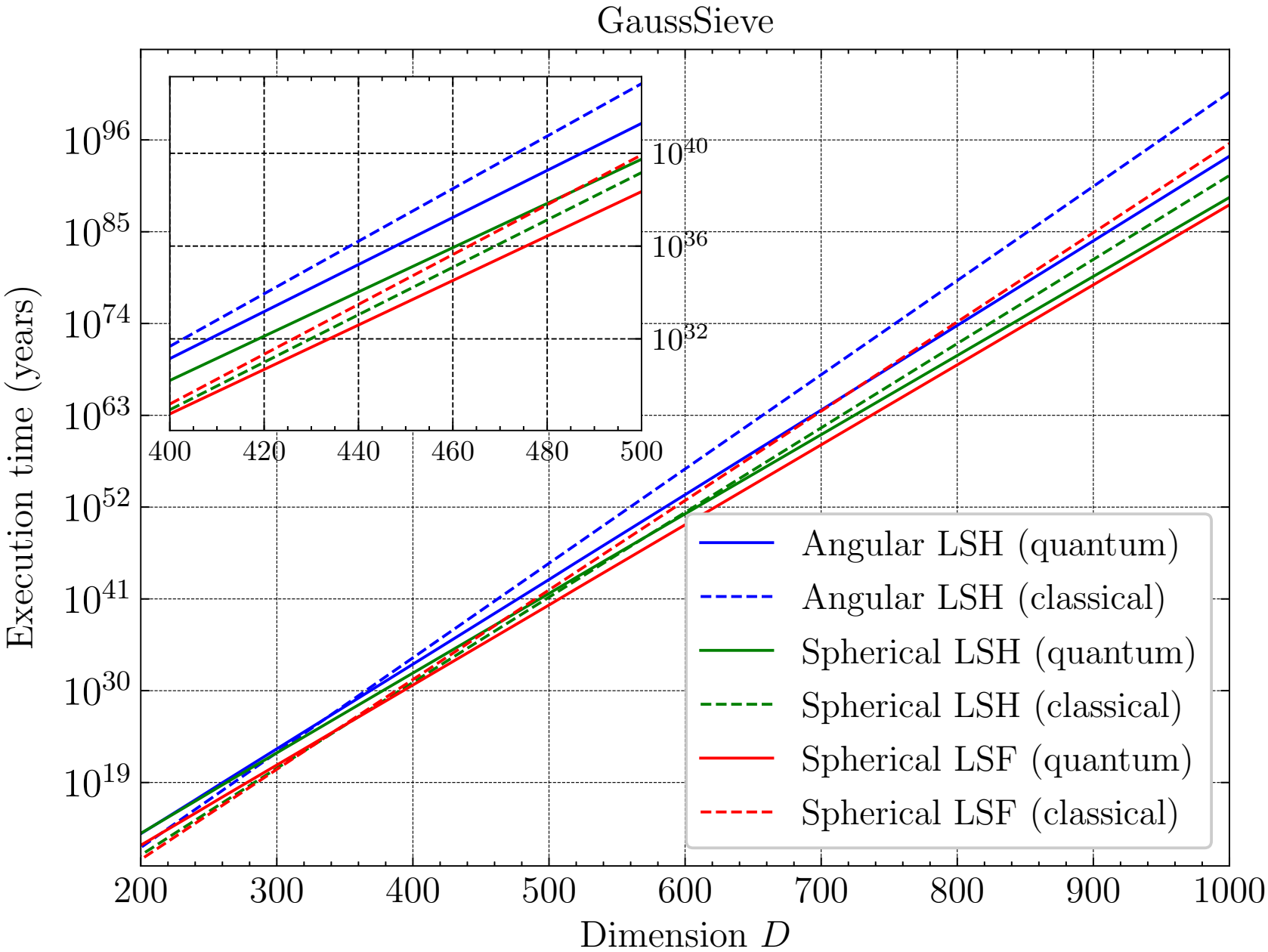
Quantum Error Correction 2

- We take into account environmental noise and employ quantum error correction to protect the quantum circuits against it.
- We employ:
  - Patch-based surface-code encodings for logical qubits [HFDM12];
  - Magic distillation protocols from Litinski [Lit19,LN22]: two 15-to-1 punctured Reed-Muller code followed by a 8-to-CCZ distillation protocol [GF19].
- We assume:
  - Incoherent circuit-level noise for physical qubits with error 10^{-5};
  - Code cycles of 100 ns (time to measure all surface-code check operators);

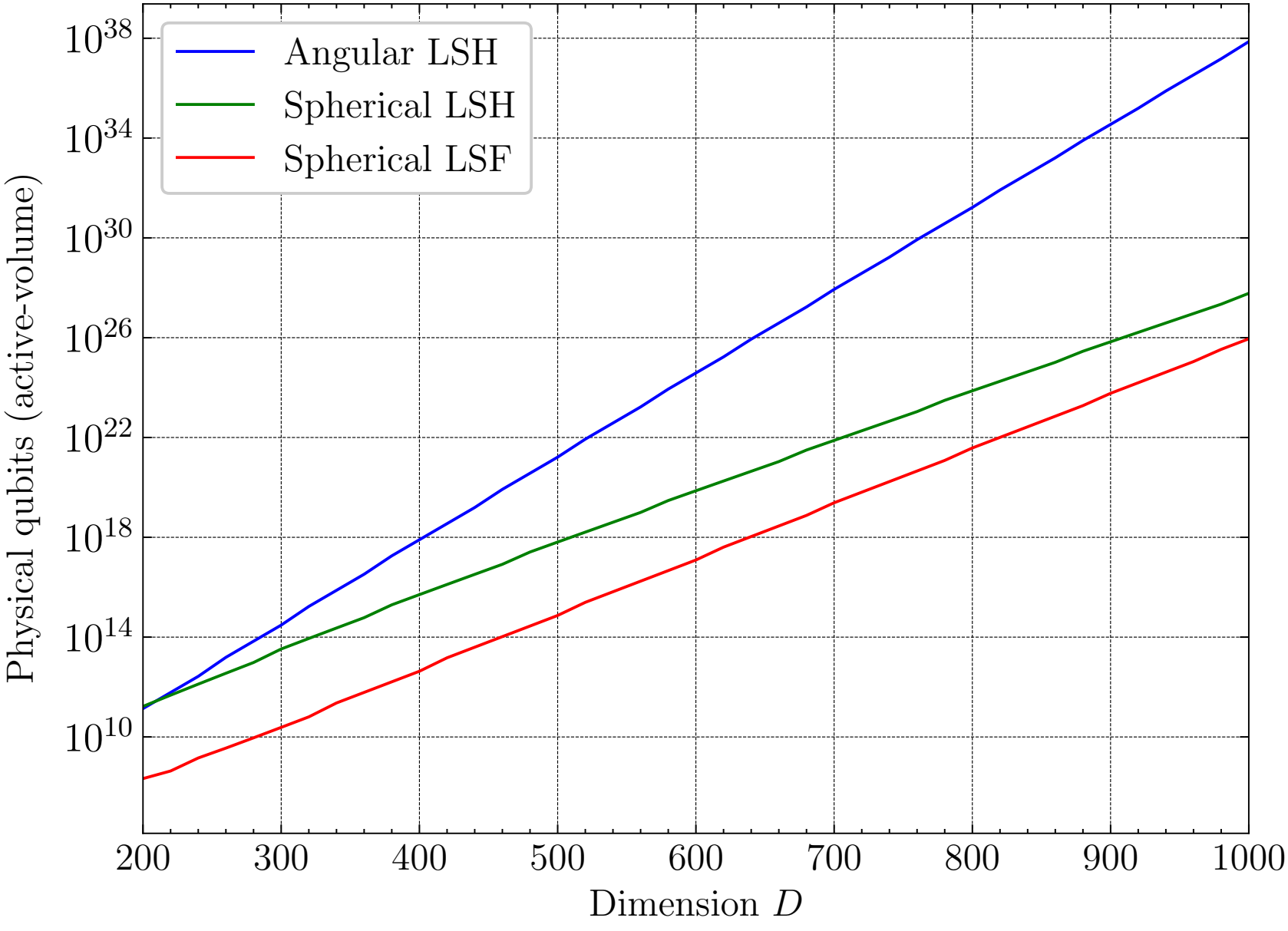
Quantum Error Correction 2

- We take into account environmental noise and employ quantum error correction to protect the quantum circuits against it.
- We employ:
  - Patch-based surface-code encodings for logical qubits [HFDM12];
  - Magic distillation protocols from Litinski [Lit19,LN22]: two 15-to-1 punctured Reed-Muller code followed by a 8-to-CCZ distillation protocol [GF19].
- We assume:
  - Incoherent circuit-level noise for physical qubits with error 10^{-5};
  - Code cycles of 100 ns (time to measure all surface-code check operators);
  - Reaction time of 1 μs (time to perform a layer of measurements, decode the outcome, and use the information to send new instructions).

Results 1


GaussSieve
Execution time (years)
10^{96}
10^{85}
10^{74}
10^{63}
10^{52}
10^{41}
10^{30}
10^{19}
10^{40}
10^{36}
10^{32}
400
420
440
460
480
500
200
300
400
500
600
700
800
900
1000
Dimension D
Angular LSH (quantum)
Angular LSH (classical)
Spherical LSH (quantum)
Spherical LSH (classical)
Spherical LSF (quantum)
Spherical LSF (classical)

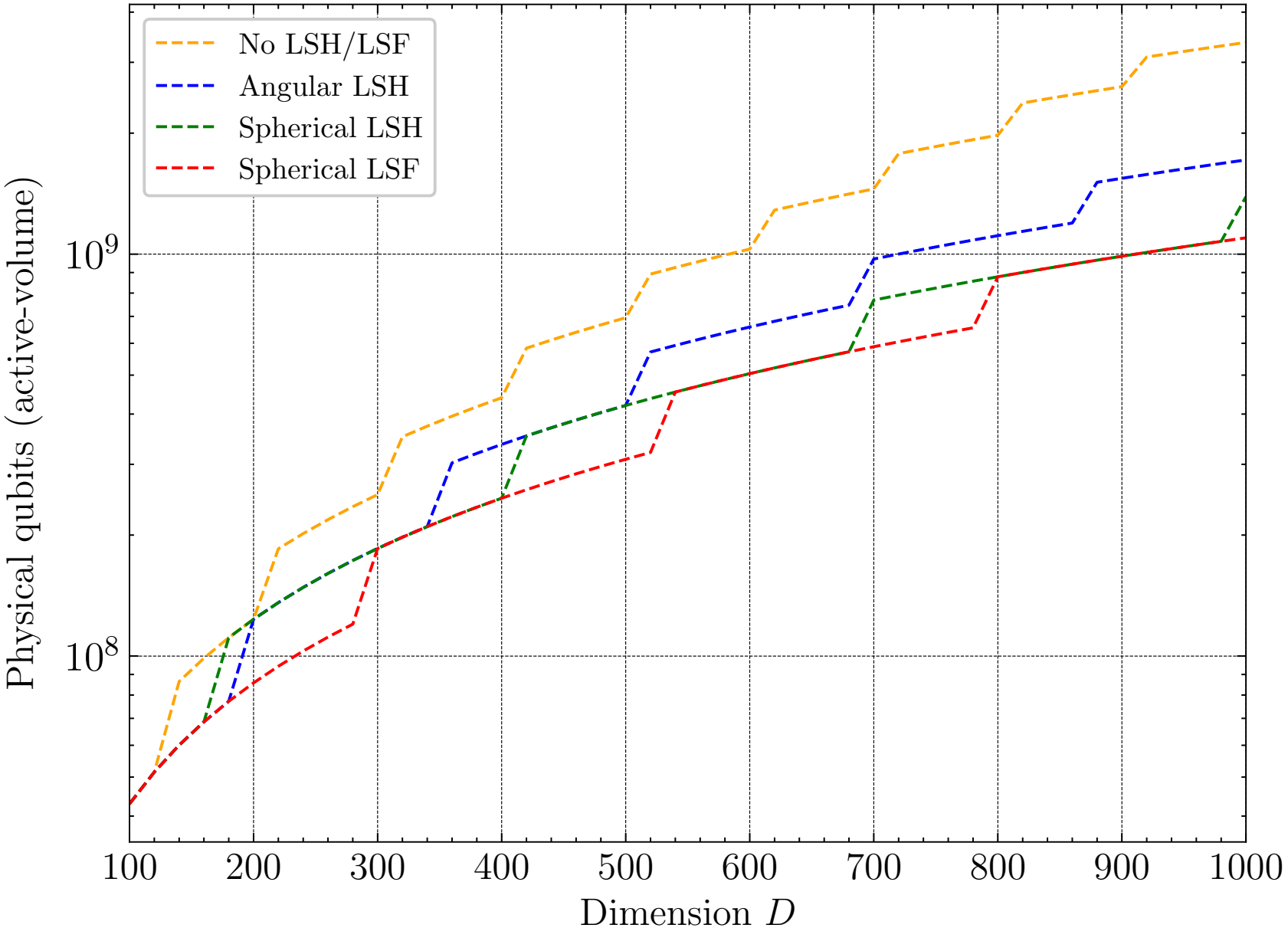
Results 2


Angular LSH
Spherical LSH
Spherical LSF
Physical qubits (active-volume)
10^{38}
10^{34}
10^{30}
10^{26}
10^{22}
10^{18}
10^{14}
10^{10}
200
300
400
500
600
700
800
900
1000
Dimension D

Results: Takeaways

- At dimension $D = 400$ (first NIST Security level), the best quantum sieve requires $\approx 10^{13}$ physical qubits to solve SVP in $\approx 10^{31}$ years.
- A 6-GHz-clock-rate single-core classical computer requires roughly the same amount of time.
- At dimension $D = 400$, the circuit time comes mostly from the quantum arithmetic circuits.
- At dimension $D = 1000$ (third NIST Security level), Grover's search provides a speedup by roughly five orders of magnitude.
- 99.9999% of physical qubits are used by the QRAM.

Results: 4


No LSH/LSF
Angular LSH
Spherical LSH
Spherical LSF
Physical qubits (active-volume)
10^9
10^8
100
200
300
400
500
600
700
800
900
1000
Dimension D

Future Directions

- Classically easy to use the notion of 'bits of security' to measure security. Quantum speedups are deceptive, composing them is tricky! Especially with QRAM. Can use tools like Google's Qualtran.

Future Directions

- Classically easy to use the notion of 'bits of security' to measure security. Quantum speedups are deceptive, composing them is tricky! Especially with QRAM. Can use tools like Google's Qualtran.
- (Classical) algorithms poorly understood. Where are the mathematical guarantees? 🙄

Future Directions: 2

- Currently, there is little to no quantum speedup in the dimensions of cryptographic interest.
- Several possible improvements:
  - New magic state distillation protocols (Gidney, Shutty, Jones, Magic State Cultivation, QIP'25);
  - New QRAM circuits (Mukhopadhyay, Scientific Reports'25);
  - Better integration between QRAM and quantum error correction (Dalzell et al., arXiv'25);
  - QRAM calls parallelisation;
  - Consider quantum-random-walk-based sieving algorithms (requires non-asymptotic quantum random walk analysis akin to Cade et al.).

Thank you! 😀

References I

Martin R. Albrecht, Vlad Gheorghiu, Eamonn W. Postlethwaite, and John M. Schanck.
Estimating quantum speedups for lattice sieves.
In Shiho Moriai and Huaxiong Wang, editors, *Advances in Cryptology – ASIACRYPT 2020*, pages 583–613, Cham, 2020. Springer International Publishing.

Miklós Ajtai, Ravi Kumar, and D. Sivakumar.
A sieve algorithm for the shortest lattice vector problem.
In *Proceedings of the Thirty-Third Annual ACM Symposium on Theory of Computing*, STOC '01, pages 601–610, New York, NY, USA, 2001. Association for Computing Machinery.

References II


Anja Becker, Léo Ducas, Nicolas Gama, and Thijs Laarhoven.
New directions in nearest neighbor searching with applications to lattice sieving.
In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '16, pages 10–24, USA, 2016. Society for Industrial and Applied Mathematics.


Thijs Laarhoven, Michele Mosca, and Joop Pol.
Finding shortest lattice vectors faster using quantum search.
Des. Codes Cryptography, 77(2-3):375–400, December 2015.


Daniel Litinski and Naomi Nickerson.
Active volume: An architecture for efficient fault-tolerant quantum computers with limited non-local connections, 2022.

References III


Daniele Micciancio and Panagiotis Voulgaris.
Faster exponential time algorithms for the shortest vector problem.
In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '10, pages 1468–1480, USA, 2010. Society for Industrial and Applied Mathematics.


Phong Q. Nguyen and Thomas Vidick.
Sieve algorithms for the shortest vector problem are practical.
Journal of Mathematical Cryptology, 2(2):181–207, 2008.