Quantum Lattice Enumeration in Limited Depth

Nina Bindel[1] Xavier Bonnetain[2] Marcel Tiepelt[3] **Fernando Virdia**[4]

[1] SandboxAQ, Palo Alto, CA, USA

[2] Université de Lorraine, CNRS, Inria, Nancy, France

[3] KASTEL, Karlsruhe Institute of Technology, Karlsruhe, Germany

[4] NOVA ID FCT, NOVA LINCS, Portugal

- Lattice-related hardness assumptions are some of the most popular tools when building quantum-resistant cryptographic primitives

- Lattice-related hardness assumptions are some of the most popular tools when building quantum-resistant cryptographic primitives
- The concrete hardness of the shortest vector problem (SVP) is at the core of the security estimations for lattice-based primitives

- Lattice-related hardness assumptions are some of the most popular tools when building quantum-resistant cryptographic primitives
- The concrete hardness of the shortest vector problem (SVP) is at the core of the security estimations for lattice-based primitives
- The cost of SVP solvers is often the leading term in the cost of algorithms for solving lattice problems

- There are many approaches for building an SVP solver

- There are many approaches for building an SVP solver
- So far, all cryptographically relevant solvers are classical routines

- There are many approaches for building an SVP solver
- So far, all cryptographically relevant solvers are classical routines
- At least two of these, sieving and enumeration, can be "compiled" into quantum algorithms using black-box methods [LMv13, KMPM19, ANS18, BCSS23]

- There are many approaches for building an SVP solver
- So far, all cryptographically relevant solvers are classical routines
- At least two of these, sieving and enumeration, can be “compiled” into quantum algorithms using black-box methods [LMv13, KMPM19, ANS18, BCSS23]
- While the resulting asymptotic quantum speedups are understood, there’s not a lot of work on their concrete cost; only sieving has been explored [AGPS20]

Today, I present new estimates on the concrete cost of quantum enumeration with extreme cylinder pruning (Q. Enum).

Today, I present new estimates on the concrete cost of quantum enumeration with extreme cylinder pruning (Q. Enum).

- Q. Enum algorithms were first demonstrated by Aono *et al.* [ANS18]; asymptotically, they provide a quadratic speedup

Today, I present new estimates on the concrete cost of quantum enumeration with extreme cylinder pruning (Q. Enum).

- Q. Enum algorithms were first demonstrated by Aono *et al.* [ANS18]; asymptotically, they provide a quadratic speedup
- Our work looks at the "max-depth" setting, where quantum computation is noisy, and long serial computation causes memory to "decohere" [Nat16, Pre18]

Today, I present new estimates on the concrete cost of quantum enumeration with extreme cylinder pruning (Q. Enum).

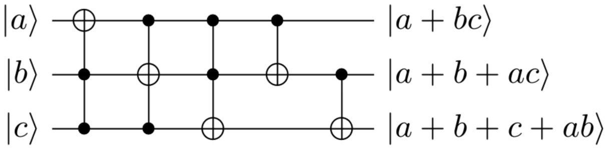
- Q. Enum algorithms were first demonstrated by Aono *et al.* [ANS18]; asymptotically, they provide a quadratic speedup
- Our work looks at the "max-depth" setting, where quantum computation is noisy, and long serial computation causes memory to "decohere" [Nat16, Pre18]
- Our results suggest that, as is the case for Grover search against block ciphers [JNRV20], quantum speedups in this setting **may** not apply

Quantum computation

To estimate the cost of quantum enumeration, we work in the "circuit model".

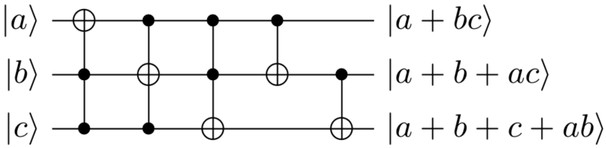
Quantum computation

To estimate the cost of quantum enumeration, we work in the "circuit model".



Quantum computation

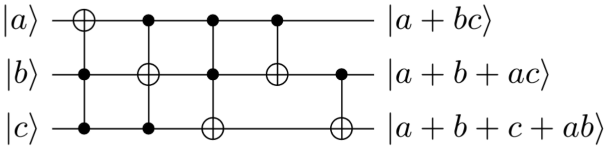
To estimate the cost of quantum enumeration, we work in the "circuit model".



- This is a quantum circuit of width 3, depth 5 and gate count 5.

Quantum computation

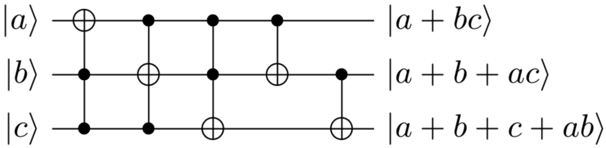
To estimate the cost of quantum enumeration, we work in the "circuit model".



- This is a quantum circuit of width 3, depth 5 and gate count 5.
- Here the wires are qubits, the nodes are gate evaluations.

Quantum computation

To estimate the cost of quantum enumeration, we work in the "circuit model".

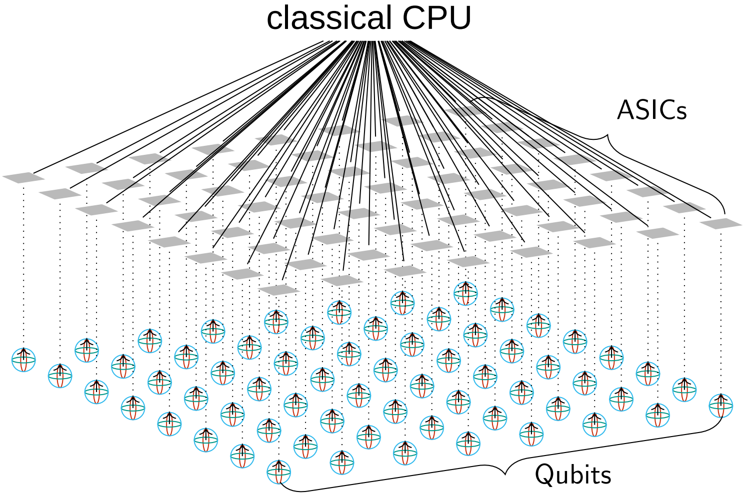


- This is a quantum circuit of width 3, depth 5 and gate count 5.
- Here the wires are qubits, the nodes are gate evaluations.
- The cost of a circuit can be expressed in terms of different metrics, e.g. by counting wires, components, depth, area. . .

[JS19] suggest that one can compare the # of quantum gates ("G metric") with classical CPU cycles.

[0]Image courtesy of Sam Jaques.

[JS19] suggest that one can compare the # of quantum gates ("G metric") with classical CPU cycles.



[0]Image courtesy of Sam Jaques.

Quantum memory

- Classical memory is easy to error-correct, quantum memory is not

Quantum memory

- Classical memory is easy to error-correct, quantum memory is not
- Currently used qubits need near-absolute-zero temperatures for data persistence; operating on them quickly leads to signal loss

Quantum memory

- Classical memory is easy to error-correct, quantum memory is not
- Currently used qubits need near-absolute-zero temperatures for data persistence; operating on them quickly leads to signal loss

New constraint: max-depth (MD)

Consider limiting the depth of quantum circuit [Nat16]:

Quantum memory

- Classical memory is easy to error-correct, quantum memory is not
- Currently used qubits need near-absolute-zero temperatures for data persistence; operating on them quickly leads to signal loss

New constraint: max-depth (MD)

Consider limiting the depth of quantum circuit [Nat16]:

- $MD = 2^{40} \approx$ "gates that presently envisioned quantum computing architectures are expected to serially perform in a year"

Quantum memory

- Classical memory is easy to error-correct, quantum memory is not
- Currently used qubits need near-absolute-zero temperatures for data persistence; operating on them quickly leads to signal loss

New constraint: max-depth (MD)

Consider limiting the depth of quantum circuit [Nat16]:

- $MD = 2^{40} \approx$ "gates that presently envisioned quantum computing architectures are expected to serially perform in a year"
- $MD = 2^{64} \approx$ "gates that current classical computing architectures can perform serially in a decade"

Quantum memory

- Classical memory is easy to error-correct, quantum memory is not
- Currently used qubits need near-absolute-zero temperatures for data persistence; operating on them quickly leads to signal loss

New constraint: max-depth (*MD*)

Consider limiting the depth of quantum circuit [Nat16]:

- $MD = 2^{40} \approx$ "gates that presently envisioned quantum computing architectures are expected to serially perform in a year"
- $MD = 2^{64} \approx$ "gates that current classical computing architectures can perform serially in a decade"
- $MD = 2^{96} \approx$ "gates that atomic scale qubits with speed of light propagation times could perform in a millennium"

Consequences of max-depth

- Consider limiting $MD \in \{2^{40}, 2^{64}, 2^{96}\}$. What happens?

Consequences of max-depth

- Consider limiting $MD \in \{2^{40}, 2^{64}, 2^{96}\}$. What happens?
- Attackers may be limited in the size of the instances of a hard problem that can be solved with a quantum circuit before decoherence

Consequences of max-depth

- Consider limiting $MD \in \{2^{40}, 2^{64}, 2^{96}\}$. What happens?
- Attackers may be limited in the size of the instances of a hard problem that can be solved with a quantum circuit before decoherence
- Multiple quantum circuits may have to be run in parallel to solve an cryptographically-sized instance, increasing the overall circuit size

Consequences of max-depth

- Consider limiting $MD \in \{2^{40}, 2^{64}, 2^{96}\}$. What happens?
- Attackers may be limited in the size of the instances of a hard problem that can be solved with a quantum circuit before decoherence
- Multiple quantum circuits may have to be run in parallel to solve an cryptographically-sized instance, increasing the overall circuit size

Example: Grover search on AES

- AES-256: $MD < 2^{k/2} = 2^{128}$, what is naively required by Grover's

Consequences of max-depth

- Consider limiting $MD \in \{2^{40}, 2^{64}, 2^{96}\}$. What happens?
- Attackers may be limited in the size of the instances of a hard problem that can be solved with a quantum circuit before decoherence
- Multiple quantum circuits may have to be run in parallel to solve an cryptographically-sized instance, increasing the overall circuit size

Example: Grover search on AES

- AES-256: $MD < 2^{k/2} = 2^{128}$, what is naively required by Grover's
- Grover search almost certainly fails if stopped early; can't read data early

Consequences of max-depth

- Consider limiting $MD \in \{2^{40}, 2^{64}, 2^{96}\}$. What happens?
- Attackers may be limited in the size of the instances of a hard problem that can be solved with a quantum circuit before decoherence
- Multiple quantum circuits may have to be run in parallel to solve an cryptographically-sized instance, increasing the overall circuit size

Example: Grover search on AES

- AES-256: $MD < 2^{k/2} = 2^{128}$, what is naively required by Grover's
- Grover search almost certainly fails if stopped early; can't read data early $\implies$ We need to account for Grover's parallelisation.

Consequences of max-depth

- Consider limiting $MD \in \{2^{40}, 2^{64}, 2^{96}\}$. What happens?
- Attackers may be limited in the size of the instances of a hard problem that can be solved with a quantum circuit before decoherence
- Multiple quantum circuits may have to be run in parallel to solve an cryptographically-sized instance, increasing the overall circuit size

Example: Grover search on AES

- AES-256: $MD < 2^{k/2} = 2^{128}$, what is naively required by Grover's
- Grover search almost certainly fails if stopped early; can't read data early $\implies$ We need to account for Grover's parallelisation.
- Grover search parallelises badly [Zal99], causing the concrete quantum advantage to strongly reduce [JNRV20].

Lattice enumeration

Lattice enumeration

- Say we are looking for a short vector $v \neq 0$ in a lattice L with basis $(b_1, \ldots, b_n)$

Lattice enumeration

- Say we are looking for a short vector $v \neq 0$ in a lattice L with basis $(b_1, \ldots, b_n)$
- Suppose we also know an upper bound R on $\|v\|$

Lattice enumeration

- Say we are looking for a short vector $v \neq 0$ in a lattice L with basis $(b_1, \ldots, b_n)$
- Suppose we also know an upper bound R on $\|v\|$
- In enumeration, we explore all (or most) vectors in L of norm $\leq R$, optionally stopping when we find the first one

Lattice enumeration

- Say we are looking for a short vector $v \neq 0$ in a lattice L with basis $(b_1, \ldots, b_n)$
- Suppose we also know an upper bound R on $\|v\|$
- In enumeration, we explore all (or most) vectors in L of norm $\leq R$, optionally stopping when we find the first one
- Conceptually, enumeration consists of depth-first search on a tree T containing short vectors as leaves

Lattice enumeration

- Say we are looking for a short vector $v \neq 0$ in a lattice L with basis $(b_1, \ldots, b_n)$
- Suppose we also know an upper bound R on $\|v\|$
- In enumeration, we explore all (or most) vectors in L of norm $\leq R$, optionally stopping when we find the first one
- Conceptually, enumeration consists of depth-first search on a tree T containing short vectors as leaves
- As used in lattice reduction, in dimension n, this requires poly(n) memory, and $\mathbb{E}[\# T] = 2^{\frac{1}{8} n \log n + o(n)}$ time on average [ABF+20]

- Given vectors $(b_1, \ldots, b_n)$, let $\pi_i(b_j)$ be the part of b_j orthogonal to $b_1, \ldots, b_{i-1}$

- Given vectors $(b_1, \ldots, b_n)$, let $\pi_i(b_j)$ be the part of b_j orthogonal to $b_1, \ldots, b_{i-1}$
- In our search for v, we start guessing possible values of $\pi_n(v)$, by choosing points in $Z_1 = \{p \in Lat(\pi_n(b_n)) \mid \|p\| \in (0, R]\}$

- Given vectors $(b_1, \ldots, b_n)$, let $\pi_i(b_j)$ be the part of b_j orthogonal to $b_1, \ldots, b_{i-1}$
- In our search for v, we start guessing possible values of $\pi_n(v)$, by choosing points in $Z_1 = \{p \in Lat(\pi_n(b_n)) \mid \|p\| \in (0, R]\}$
  - These guesses are the nodes distant 1 from the root of the enumeration tree T

- Given vectors $(b_1, \ldots, b_n)$, let $\pi_i(b_j)$ be the part of b_j orthogonal to $b_1, \ldots, b_{i-1}$
- In our search for v, we start guessing possible values of $\pi_n(v)$, by choosing points in $Z_1 = \{p \in Lat(\pi_n(b_n)) \mid \|p\| \in (0, R]\}$
  - These guesses are the nodes distant 1 from the root of the enumeration tree T
- Given a guess g for $\pi_n(v)$, we try to "extend it" into a guess for $\pi_{n-1}(v)$ by choosing points in $Z_2 = \{p \in Lat(\pi_{n-1}(b_{n-1}), \pi_{n-1}(b_n)) \mid \|p\| \in (0, R]\}$ with $\pi_n(p) = g$

- Given vectors $(b_1, \ldots, b_n)$, let $\pi_i(b_j)$ be the part of b_j orthogonal to $b_1, \ldots, b_{i-1}$
- In our search for v, we start guessing possible values of $\pi_n(v)$, by choosing points in $Z_1 = \{p \in Lat(\pi_n(b_n)) \mid \|p\| \in (0, R]\}$
  - These guesses are the nodes distant 1 from the root of the enumeration tree T
- Given a guess g for $\pi_n(v)$, we try to "extend it" into a guess for $\pi_{n-1}(v)$ by choosing points in $Z_2 = \{p \in Lat(\pi_{n-1}(b_{n-1}), \pi_{n-1}(b_n)) \mid \|p\| \in (0, R]\}$ with $\pi_n(p) = g$
  - These guesses are the nodes distant 2 from the root of the enumeration tree T

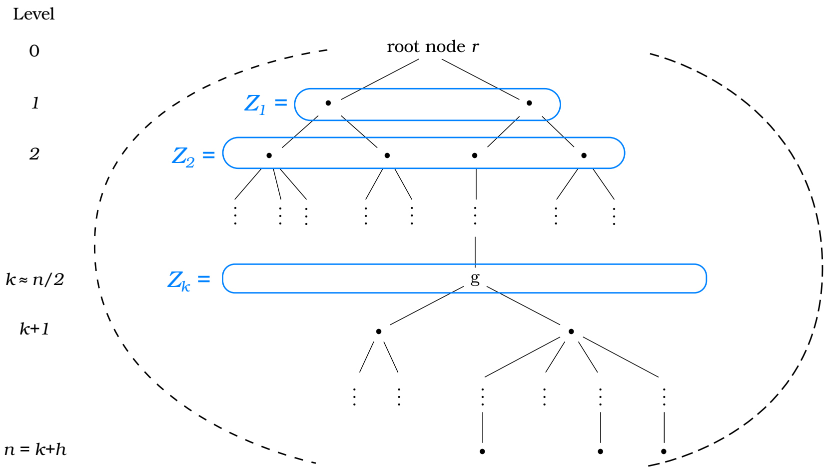
- Given vectors $(b_1, \dots, b_n)$, let $\pi_i(b_j)$ be the part of b_j orthogonal to $b_1, \dots, b_{i-1}$
- In our search for v, we start guessing possible values of $\pi_n(v)$, by choosing points in $Z_1 = \{p \in Lat(\pi_n(b_n)) \mid \|p\| \in (0, R]\}$
  - These guesses are the nodes distant 1 from the root of the enumeration tree T
- Given a guess g for $\pi_n(v)$, we try to "extend it" into a guess for $\pi_{n-1}(v)$ by choosing points in $Z_2 = \{p \in Lat(\pi_{n-1}(b_{n-1}), \pi_{n-1}(b_n)) \mid \|p\| \in (0, R]\}$ with $\pi_n(p) = g$
  - These guesses are the nodes distant 2 from the root of the enumeration tree T
- This search is done depth-first, stopping whenever we fail to extend a guess from Z_i to Z_{i+1} while maintaining norm $\leq R$;

- Given vectors $(b_1, \ldots, b_n)$, let $\pi_i(b_j)$ be the part of b_j orthogonal to $b_1, \ldots, b_{i-1}$
- In our search for v, we start guessing possible values of $\pi_n(v)$, by choosing points in $Z_1 = \{p \in Lat(\pi_n(b_n)) \mid \|p\| \in (0, R]\}$
  - These guesses are the nodes distant 1 from the root of the enumeration tree T
- Given a guess g for $\pi_n(v)$, we try to "extend it" into a guess for $\pi_{n-1}(v)$ by choosing points in $Z_2 = \{p \in Lat(\pi_{n-1}(b_{n-1}), \pi_{n-1}(b_n)) \mid \|p\| \in (0, R]\}$ with $\pi_n(p) = g$
  - These guesses are the nodes distant 2 from the root of the enumeration tree T
- This search is done depth-first, stopping whenever we fail to extend a guess from Z_i to Z_{i+1} while maintaining norm $\leq R$; we find v when it we extend a guess from Z_{n-1} to Z_n

- Given vectors $(b_1, \ldots, b_n)$, let $\pi_i(b_j)$ be the part of b_j orthogonal to $b_1, \ldots, b_{i-1}$
- In our search for v, we start guessing possible values of $\pi_n(v)$, by choosing points in $Z_1 = \{p \in Lat(\pi_n(b_n)) \mid \|p\| \in (0, R]\}$
  - These guesses are the nodes distant 1 from the root of the enumeration tree T
- Given a guess g for $\pi_n(v)$, we try to "extend it" into a guess for $\pi_{n-1}(v)$ by choosing points in $Z_2 = \{p \in Lat(\pi_{n-1}(b_{n-1}), \pi_{n-1}(b_n)) \mid \|p\| \in (0, R]\}$ with $\pi_n(p) = g$
  - These guesses are the nodes distant 2 from the root of the enumeration tree T
- This search is done depth-first, stopping whenever we fail to extend a guess from Z_i to Z_{i+1} while maintaining norm $\leq R$; we find v when it we extend a guess from Z_{n-1} to Z_n

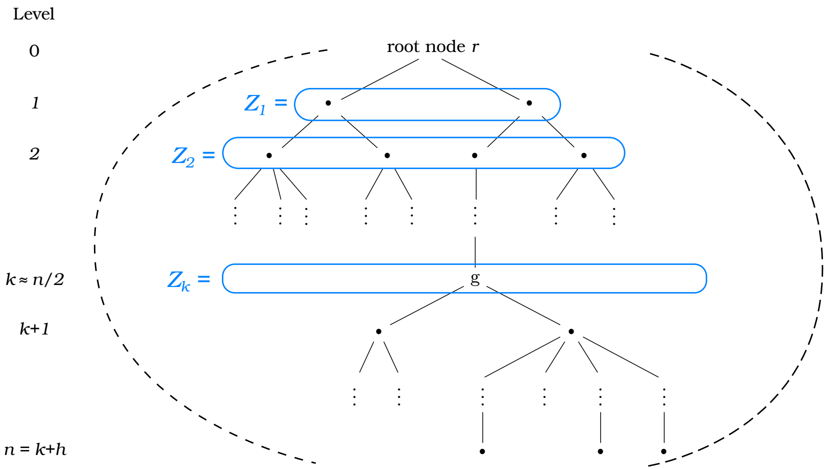
We can see this as searching for a "marked leaf" in a tree, where a leaf is marked if its norm is $\leq R$.

A look at the enumeration tree T



- Nodes located on different levels Z_k

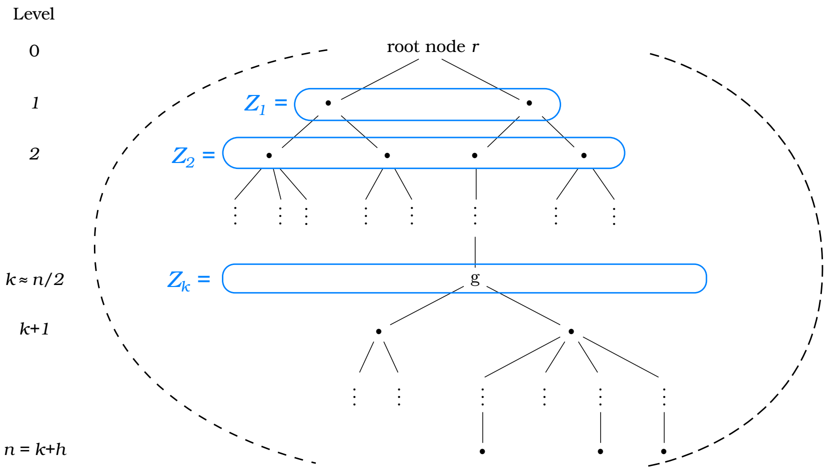
A look at the enumeration tree T



- Nodes located on different levels Z_k
- "Middle" levels super-exponentially large [GNR10]:

$$\#T \approx \#Z_{n/2}$$

A look at the enumeration tree T



- Nodes located on different levels Z_k
- "Middle" levels super-exponentially large [GNR10]:

$$\#T \approx \#Z_{n/2}$$

- The tree size can be somewhat reduced by "pruning" nodes that are unlikely to yield a marked leaf

Quantum tree search

- In 2018, Montanaro introduces two quantum tree-search algorithms, DetectMV and FindMV [Mon18]

Quantum tree search

- In 2018, Montanaro introduces two quantum tree-search algorithms, DetectMV and FindMV [Mon18]
- Given a tree T and a predicate P, DetectMV returns whether $\exists$ leaf $\in T$ such that $P(\text{leaf}) = \text{true}$ in $\tilde{O}(\sqrt{\mathcal{T} \cdot n})$ evaluations of P, where $\#T \leq \mathcal{T}$

Quantum tree search

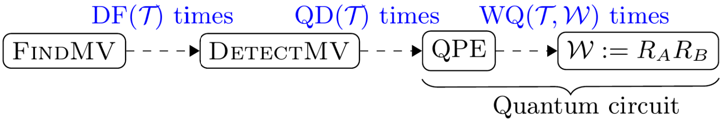
- In 2018, Montanaro introduces two quantum tree-search algorithms, DetectMV and FindMV [Mon18]
- Given a tree T and a predicate P, DetectMV returns whether $\exists$ leaf $\in T$ such that $P(\text{leaf}) = \text{true}$ in $\tilde{O}(\sqrt{\mathcal{T} \cdot n})$ evaluations of P, where $\#T \leq \mathcal{T}$
- By performing decision on every level, *DetectMV* $\mapsto$ *FindMV*, which returns such a leaf

Quantum tree search

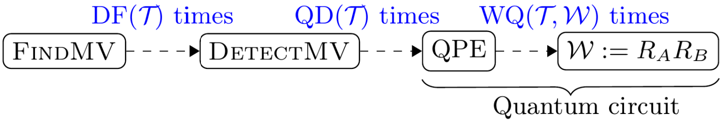
- In 2018, Montanaro introduces two quantum tree-search algorithms, DetectMV and FindMV [Mon18]
- Given a tree T and a predicate P, DetectMV returns whether $\exists$ leaf $\in T$ such that $P(\text{leaf}) = \text{true}$ in $\tilde{O}(\sqrt{\mathcal{T} \cdot n})$ evaluations of P, where $\#T \leq \mathcal{T}$
- By performing decision on every level, *DetectMV* $\mapsto$ *FindMV*, which returns such a leaf
- For trees with one (randomly distributed) marked leaf and $\#T \approx \mathcal{T}$:

  Classical average-case runtime $O(\#T) \mapsto$ quantum average case $\tilde{O}(\sqrt{\#T \cdot n})$

Montanaro's tree search

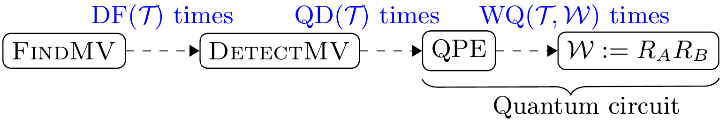


Montanaro's tree search



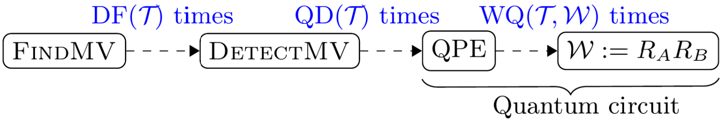
- DetectMV consists of repeating multiple Quantum Phase Estimations (QPE) of an operator W that checks predicate P;

Montanaro's tree search



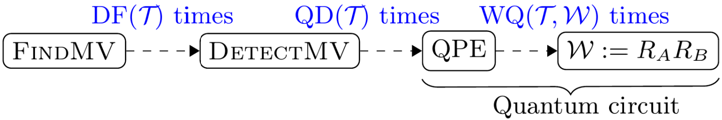
- DetectMV consists of repeating multiple Quantum Phase Estimations (QPE) of an operator W that checks predicate P; evaluating QPE(W) is the *quantum part*

Montanaro's tree search



- DetectMV consists of repeating multiple Quantum Phase Estimations (QPE) of an operator W that checks predicate P; evaluating QPE(W) is the *quantum part*
- Under conservative estimations, we serially evaluate $\sqrt{\#T \cdot n}$ times W per QPE

Montanaro's tree search



- DetectMV consists of repeating multiple Quantum Phase Estimations (QPE) of an operator W that checks predicate P; evaluating QPE(W) is the *quantum part*
- Under conservative estimations, we serially evaluate $\sqrt{\#T \cdot n}$ times W per QPE
- Our objective is to lower-bound the gate-cost of FindMV(T), while keeping the serial quantum depth within max-depht MD

To check the hypothetical depth of such a QPE we:

To check the hypothetical depth of such a QPE we:

- Chose a target scheme to attack (Kyber)

To check the hypothetical depth of such a QPE we:

- Chose a target scheme to attack (Kyber)
- Lower-bound the size of W by assuming $\text{Depth}(W) = \text{Gates}(W) = 1$

To check the hypothetical depth of such a QPE we:

- Chose a target scheme to attack (Kyber)
- Lower-bound the size of W by assuming $\text{Depth}(W) = \text{Gates}(W) = 1$
- Using the LWE estimator we find the required block size β to break Kyber using the primal attack

To check the hypothetical depth of such a QPE we:

- Chose a target scheme to attack (Kyber)
- Lower-bound the size of W by assuming $\text{Depth}(W) = \text{Gates}(W) = 1$
- Using the LWE estimator we find the required block size β to break Kyber using the primal attack
  - β is the depth n of tree

To check the hypothetical depth of such a QPE we:

- Chose a target scheme to attack (Kyber)
- Lower-bound the size of W by assuming $\text{Depth}(W) = \text{Gates}(W) = 1$
- Using the LWE estimator we find the required block size β to break Kyber using the primal attack
  - β is the depth n of tree
  - From n we obtain $\#T$ by using lower bounds for the cost of enumeration with cylinder pruning [ANSS18]

To check the hypothetical depth of such a QPE we:

- Chose a target scheme to attack (Kyber)
- Lower-bound the size of W by assuming $\text{Depth}(W) = \text{Gates}(W) = 1$
- Using the LWE estimator we find the required block size β to break Kyber using the primal attack
  - β is the depth n of tree
  - From n we obtain $\#T$ by using lower bounds for the cost of enumeration with cylinder pruning [ANSS18]
- Finally, we check if the resulting circuit depth of QPE is $\leq MD$

A back of the envelope estimation

A back of the envelope estimation

$$\mathop{\mathbb{E}}_{\substack{\text{random} \\ \text{tree } T}}[\text{Depth}(\text{QPE}(W))] \approx \mathbb{E}[\sqrt{\#T \cdot \beta}] \approx \sqrt{\mathbb{E}[\#T] \cdot \beta} \approx \begin{cases} 2^{90.3} & \text{for Kyber-512,} \\ 2^{166.2} & \text{for Kyber-768,} \\ 2^{263.7} & \text{for Kyber-1024,} \end{cases}$$

A back of the envelope estimation

$$\mathop{\mathbb{E}}_{\substack{\text{random}\\ \text{tree } T}} [\mathrm{Depth}(\mathrm{QPE}(W))] \approx \mathbb{E}[\sqrt{\# T \cdot \beta}] \approx \sqrt{\mathbb{E}[\# T] \cdot \beta} \approx \begin{cases} 2^{90.3} & \text{for Kyber-512,} \\ 2^{166.2} & \text{for Kyber-768,} \\ 2^{263.7} & \text{for Kyber-1024,} \end{cases}$$

APTN / AP

- Wait, don't drag me out of the room

A back of the envelope estimation

$$\mathop{\mathbb{E}}_{\substack{\text{random}\\ \text{tree } T}}[\mathrm{Depth}(\mathrm{QPE}(W))] \approx \mathbb{E}[\sqrt{\#T \cdot \beta}] \approx \sqrt{\mathbb{E}[\#T] \cdot \beta} \approx \begin{cases} 2^{90.3} & \text{for Kyber-512,} \\ 2^{166.2} & \text{for Kyber-768,} \\ 2^{263.7} & \text{for Kyber-1024,} \end{cases}$$

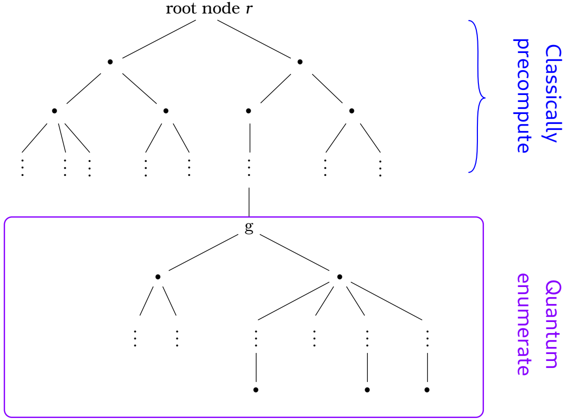

APTN / AP

- Wait, don't drag me out of the room
- I do know Jensen's inequality!
  $$\mathbb{E}[\sqrt{\#T}] \leq \sqrt{\mathbb{E}[\#T]}$$
- Just wait a handful of slides

- We plausibly don't fit within 2^{96} depth
- We need smaller trees to enumerate

- We plausibly don't fit within 2^{96} depth
- We need smaller trees to enumerate

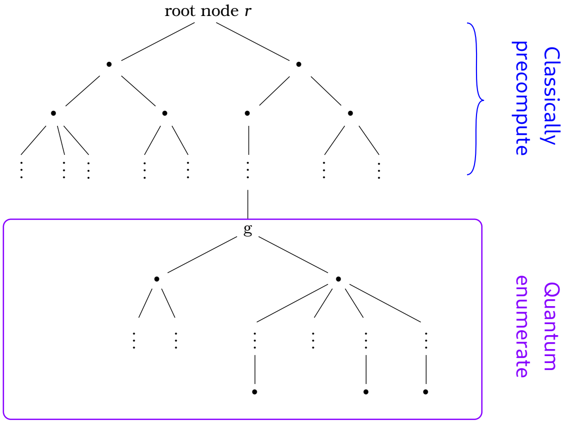
Classic trick from parallel enumeration



- We plausibly don't fit within 2^{96} depth
- We need smaller trees to enumerate

Classic trick from parallel enumeration

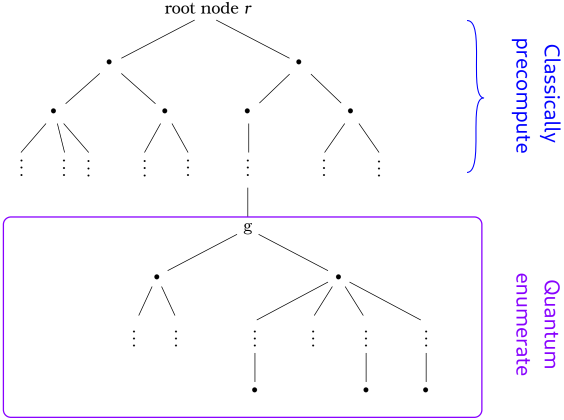
- Precompute nodes up to level $k > 1$, run FindMV on the subtrees.



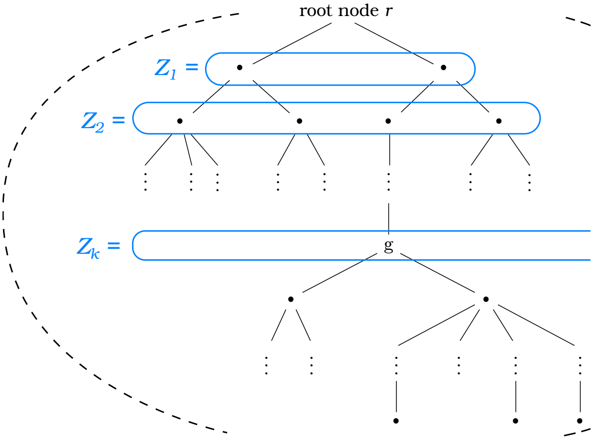
- We plausibly don't fit within 2^{96} depth
- We need smaller trees to enumerate

Classic trick from parallel enumeration

- Precompute nodes up to level $k > 1$, run FindMV on the subtrees.
- We can estimate the size of subtrees with similar techniques as for the full tree.

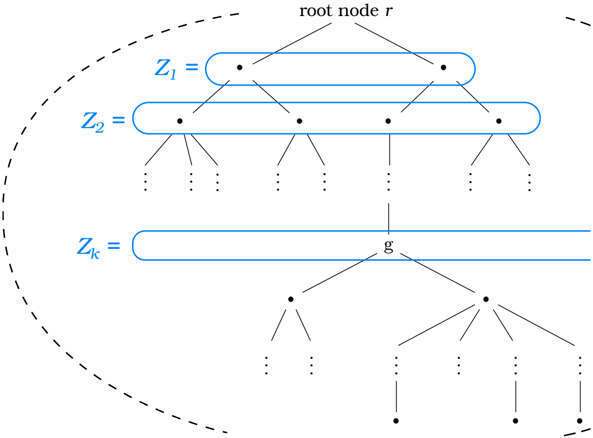


Would this work? Up to what level k do we precompute?



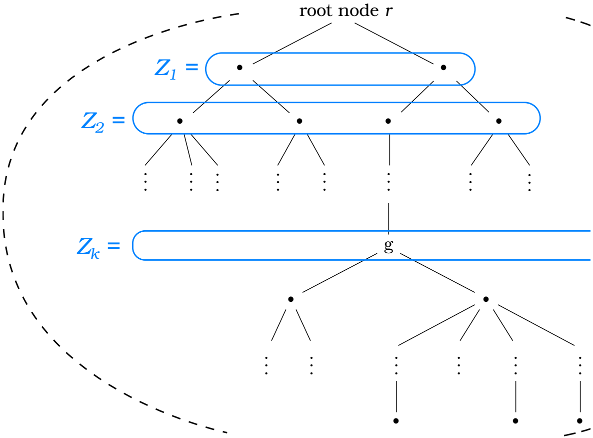
Would this work? Up to what level k do we precompute?

- $k \approx 1$: in this case most of the tree fits within the quantum enumeration subroutine $\rightarrow$ a quadratic speedup without pre-computation, but maybe not our case



Would this work? Up to what level k do we precompute?

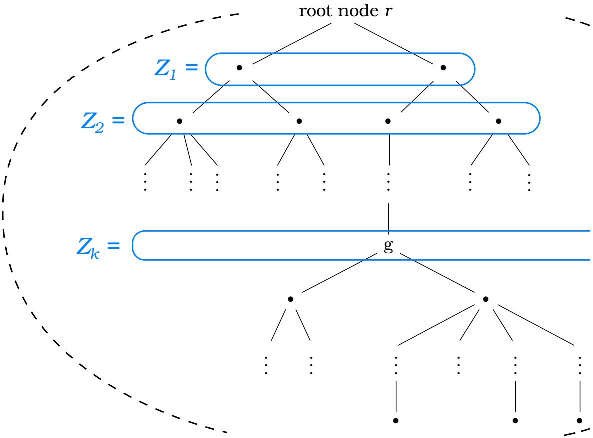
- $k \approx 1$: in this case most of the tree fits within the quantum enumeration subroutine $\rightarrow$ a quadratic speedup without pre-computation, but maybe not our case



- $k \approx n$: we run some quantum enumeration, we precomputed more than $H_{n/2}$ classically, no advantage over classical enumeration

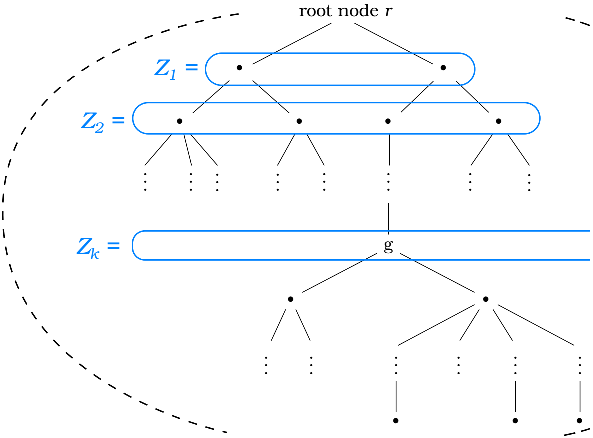
Would this work? Up to what level k do we precompute?

- $k \approx 1$: in this case most of the tree fits within the quantum enumeration subroutine $\rightarrow$ a quadratic speedup without pre-computation, but maybe not our case
- $k \approx n/2$: we run $\approx H_{n/2} := \left|Z_{n/2}\right|$ quantum enumeration calls
- $k \approx n$: we run some quantum enumeration, we precomputed more than $H_{n/2}$ classically, no advantage over classical enumeration



Would this work? Up to what level k do we precompute?

- $k \approx 1$: in this case most of the tree fits within the quantum enumeration subroutine $\to$ a quadratic speedup without pre-computation, but maybe not our case
- $k \approx n/2$: we run $\approx H_{n/2} := \left|Z_{n/2}\right|$ quantum enumeration calls $\implies$ total gate-count $\approx H_{n/2} \approx$ cost of classical enumeration
- $k \approx n$: we run some quantum enumeration, we precomputed more than $H_{n/2}$ classically, no advantage over classical enumeration



Our best chance is $k \lessapprox n/2$.

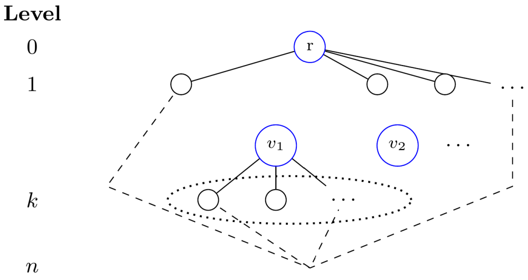
Our best chance is $k \lessapprox n/2$. However, running FindMV as many as H_k times may be too much.

Our best chance is $k \lessapprox n/2$. However, running FindMV as many as H_k times may be too much.

- Try bundling! Assume 2^y qRAM available

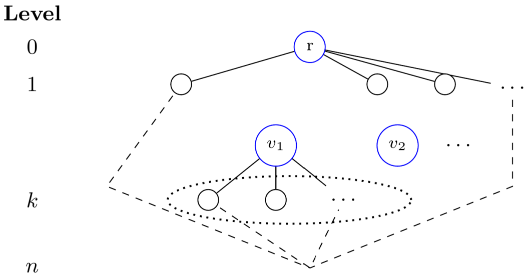
Our best chance is $k \lessapprox n/2$. However, running FindMV as many as H_k times may be too much.

- Try bundling! Assume 2^y qRAM available
- Precompute sets of 2^y elements in Z_k, collect them under a 'virtual' node v, run FindMV over the tree $T(v)$ with root v



Our best chance is $k \lessapprox n/2$. However, running FindMV as many as H_k times may be too much.

- Try bundling! Assume 2^y qRAM available
- Precompute sets of 2^y elements in Z_k, collect them under a 'virtual' node v, run FindMV over the tree $T(v)$ with root v



Disclaimer

qRAM (a.k.a. QRACM) may be quite costly to access [JR23]. Yet, many quantum-classical speedups assume it.

One last step: expected square roots

- We are trying to estimate or lower-bound $\mathbb{E}[\sqrt{\#T}]$, but the distribution of $\#T$ is unknown (Aono *et al.* [ANS18] already mention this issue)

One last step: expected square roots

- We are trying to estimate or lower-bound $\mathbb{E}[\sqrt{\#T}]$, but the distribution of $\#T$ is unknown (Aono *et al.* [ANS18] already mention this issue)
- Jensen's inequality ($\mathbb{E}[\sqrt{\#T}] \leq \sqrt{\mathbb{E}[\#T]}$) only gives us upper bounds

One last step: expected square roots

- We are trying to estimate or lower-bound $\mathbb{E}[\sqrt{\#T}]$, but the distribution of $\#T$ is unknown (Aono *et al.* [ANS18] already mention this issue)
- Jensen's inequality ($\mathbb{E}[\sqrt{\#T}] \leq \sqrt{\mathbb{E}[\#T]}$) only gives us upper bounds

Definition: Multiplicative Jensen's gap

Let X be a random variable. We say X has multiplicative Jensen's gap 2^z if

$$\sqrt{\mathbb{E}[X]} = 2^z\, \mathbb{E}[\sqrt{X}].$$

One last step: expected square roots

- We are trying to estimate or lower-bound $\mathbb{E}[\sqrt{\#T}]$, but the distribution of $\#T$ is unknown (Aono *et al.* [ANS18] already mention this issue)
- Jensen's inequality ($\mathbb{E}[\sqrt{\#T}] \leq \sqrt{\mathbb{E}[\#T]}$) only gives us upper bounds

Definition: Multiplicative Jensen's gap

Let X be a random variable. We say X has multiplicative Jensen's gap 2^z if

$$\sqrt{\mathbb{E}[X]} = 2^z\, \mathbb{E}[\sqrt{X}].$$

- Ideally, we want an upper bound to z; up to $\beta = 70$ we measure $z \approx 1$

One last step: expected square roots

- We are trying to estimate or lower-bound $\mathbb{E}[\sqrt{\#T}]$, but the distribution of $\#T$ is unknown (Aono *et al.* [ANS18] already mention this issue)
- Jensen's inequality ($\mathbb{E}[\sqrt{\#T}] \leq \sqrt{\mathbb{E}[\#T]}$) only gives us upper bounds

Definition: Multiplicative Jensen's gap

Let X be a random variable. We say X has multiplicative Jensen's gap 2^z if

$$\sqrt{\mathbb{E}[X]} = 2^z\, \mathbb{E}[\sqrt{X}].$$

- Ideally, we want an upper bound to z; up to $\beta = 70$ we measure $z \approx 1$
- Without such bounds, we can run attack cost estimates as a function of z, and see at what point the hypothetical attack becomes viable

Summarising, we obtain formulae for

- The depth of the individual QPE circuits we need to run

Summarising, we obtain formulae for

- The depth of the individual QPE circuits we need to run
- The total number of gates we evaluate

Summarising, we obtain formulae for

- The depth of the individual QPE circuits we need to run
- The total number of gates we evaluate

Quantum depth

$$\mathbb{E}\left[\mathrm{Depth}(\mathrm{QPE}(W))\right] \geq \frac{1}{2^{z}}\sqrt{\mathbb{E}\left[\# T(v)\cdot(n-k+1)\right]}\cdot \mathrm{Depth}(W), \text{ for } g \in Z_k.$$

Summarising, we obtain formulae for

- The depth of the individual QPE circuits we need to run
- The total number of gates we evaluate

Quantum depth

$$\mathbb{E}\left[\mathrm{Depth}(\mathrm{QPE}(W))\right] \geq \frac{1}{2^z}\sqrt{\mathbb{E}\left[\# T(v)\cdot(n-k+1)\right]}\cdot \mathrm{Depth}(W), \text{ for } g \in Z_k.$$

Quantum gate-cost

$$\begin{aligned}\mathop{\mathbb{E}}_{\substack{\text{random}\\ \text{tree } T}}\left[\text{Quantum Gates}\right] &\approx \frac{H_k}{2^y}\cdot \mathbb{E}\left[\mathrm{Gates}(\mathrm{FindMV}(T(g)))\right]\\ &\geq \frac{H_k}{2^y}\cdot \mathbb{E}\left[\sqrt{\# T(v)\cdot(n-k+1)}\right]\cdot \mathrm{Gates}(W)\\ &= \frac{H_k}{2^y}\cdot\frac{1}{2^z}\sqrt{\mathbb{E}\left[\# T(v)\cdot(n-k+1)\right]}\cdot \mathrm{Gates}(W)\end{aligned}$$

We can now try computing some numbers.

We can now try computing some numbers.

- We assume either $\text{Depth}(W) = \text{Gates}(W) = 1$ (in the "query-model") or an estimated lower bound based on best-known quantum arithmetic circuits (in the "circuit-model", recent work may help [BvHJ+23])

We can now try computing some numbers.

- We assume either $\mathrm{Depth}(W) = \mathrm{Gates}(W) = 1$ (in the "query-model") or an estimated lower bound based on best-known quantum arithmetic circuits (in the "circuit-model", recent work may help [BvHJ+23])
- We use the LWE-estimator to find the enumeration dimension β

We can now try computing some numbers.

- We assume either $\text{Depth}(W) = \text{Gates}(W) = 1$ (in the "query-model") or an estimated lower bound based on best-known quantum arithmetic circuits (in the "circuit-model", recent work may help [BvHJ+23])
- We use the LWE-estimator to find the enumeration dimension β
- We estimate sub-tree sizes using cylinder pruning lower-bounds [ANSS18]

We can now try computing some numbers.

- We assume either $\mathrm{Depth}(W) = \mathrm{Gates}(W) = 1$ (in the "query-model") or an estimated lower bound based on best-known quantum arithmetic circuits (in the "circuit-model", recent work may help [BvHJ+23])
- We use the LWE-estimator to find the enumeration dimension β
- We estimate sub-tree sizes using cylinder pruning lower-bounds [ANSS18]
- We estimate costs for every $k \leq n$, $y \leq 80$, $z \leq 64$

We can now try computing some numbers.

- We assume either $\text{Depth}(W) = \text{Gates}(W) = 1$ (in the "query-model") or an estimated lower bound based on best-known quantum arithmetic circuits (in the "circuit-model", recent work may help [BvHJ+23])
- We use the LWE-estimator to find the enumeration dimension β
- We estimate sub-tree sizes using cylinder pruning lower-bounds [ANSS18]
- We estimate costs for every $k \le n$, $y \le 80$, $z \le 64$
- We report z, k minimising *classical* $+$ *quantum gate-cost*

more likely to be feasible — less likely to be feasible

		$\log \mathbb{E}[\text{GCost}]$ (with $\mathcal{W}$ as in § 4.1) below...			$\log \mathbb{E}[\text{GCost}]$ (with $\mathcal{W}$ as in § 4.2) below...		
MD	Kyber	Target security	Grover on $\text{AES}_{\{128,192,256\}}$	Quasi-Sqrt $^1/_b\sqrt{\#\mathcal{T} \cdot h}$	Target security	Grover on $\text{AES}_{\{128,192,256\}}$	Quasi-Sqrt $^1/_b\sqrt{\#\mathcal{T} \cdot h}$
2^{40}	-512						
	-768						
	-1024						
2^{64}	-512						
	-768						
	-1024						
2^{96}	-512						
	-768						
	-1024						
∞	-512						
	-768						
	-1024						

more likely to be feasible → less likely to be feasible

MD	Kyber	$\log \mathbb{E}[\text{GCost}]$ (with $\mathcal{W}$ as in § 4.1) below… Target security	Grover on $\text{AES}_{\{128,192,256\}}$	Quasi-Sqrt $^1/_b\sqrt{\#\mathcal{T} \cdot h}$	$\log \mathbb{E}[\text{GCost}]$ (with $\mathcal{W}$ as in § 4.2) below… Target security	Grover on $\text{AES}_{\{128,192,256\}}$	Quasi-Sqrt $^1/_b\sqrt{\#\mathcal{T} \cdot h}$
2^{40}	-512						
	-768						
	-1024						
2^{64}	-512						
	-768						
	-1024						
2^{96}	-512						
	-768						
	-1024						
∞	-512	$z \geq 0,\ k = 0$	$z \geq 9,\ k = 0$	$z \geq 1,\ k = 0$	$z \geq 0,\ k = 0$	$z \geq 33,\ k = 0$	$z \geq 26,\ k = 0$
	-768	$z \geq 0,\ k = 0$	$z \geq 52,\ k = 0$	$z \geq 1,\ k = 0$	$z \geq 1,\ k = 0$	$z > 64$	$z \geq 27,\ k = 0$
	-1024	$z \geq 9,\ k = 0$	$z > 64$	$z \geq 1,\ k = 0$	$z \geq 35,\ k = 0$	$z > 64$	$z \geq 28,\ k = 0$

more likely to be feasible ⟶ less likely to be feasible

MD	Kyber	$\log \mathbb{E}[\text{GCost}]$ (with $\mathcal{W}$ as in § 4.1) below… Target security	Grover on $\text{AES}_{\{128,192,256\}}$	Quasi-Sqrt ${}^1/_b\sqrt{\#\mathcal{T} \cdot h}$	$\log \mathbb{E}[\text{GCost}]$ (with $\mathcal{W}$ as in § 4.2) below… Target security	Grover on $\text{AES}_{\{128,192,256\}}$	Quasi-Sqrt ${}^1/_b\sqrt{\#\mathcal{T} \cdot h}$
2^{40}	-512						
	-768						
	-1024						
2^{64}	-512						
	-768						
	-1024						
2^{96}	-512	$z \geq 0,\ k \leq 58$	$z \geq 8,\ k \leq 53$	$z \geq 1,\ k \leq 58$	$z \geq 0,\ k \leq 63$	$z \geq 33,\ k \leq 54$	$z \geq 25,\ k \leq 58$
	-768	$z \geq 23,\ k \leq 106$	$z \geq 56,\ k \leq 62$	$z \geq 36,\ k \leq 77$	$z \geq 40,\ k \leq 77$	$z > 64$	$z \geq 52,\ k \leq 77$
	-1024	$z > 64$	$z > 64$	$z > 64$	$z > 64$	$z > 64$	$z > 64$
∞	-512	$z \geq 0,\ k = 0$	$z \geq 9,\ k = 0$	$z \geq 1,\ k = 0$	$z \geq 0,\ k = 0$	$z \geq 33,\ k = 0$	$z \geq 26,\ k = 0$
	-768	$z \geq 0,\ k = 0$	$z \geq 52,\ k = 0$	$z \geq 1,\ k = 0$	$z \geq 1,\ k = 0$	$z > 64$	$z \geq 27,\ k = 0$
	-1024	$z \geq 9,\ k = 0$	$z > 64$	$z \geq 1,\ k = 0$	$z \geq 35,\ k = 0$	$z > 64$	$z \geq 28,\ k = 0$

more likely to be feasible ⟶ less likely to be feasible

MD	Kyber	$\log \mathbb{E}[\text{GCost}]$ (with $\mathcal{W}$ as in § 4.1) below… Target security	Grover on $\text{AES}_{\{128,192,256\}}$	Quasi-Sqrt $^1/_b\sqrt{\#\mathcal{T}\cdot h}$	$\log \mathbb{E}[\text{GCost}]$ (with $\mathcal{W}$ as in § 4.2) below… Target security	Grover on $\text{AES}_{\{128,192,256\}}$	Quasi-Sqrt $^1/_b\sqrt{\#\mathcal{T}\cdot h}$
2^{40}	-512						
	-768						
	-1024						
2^{64}	-512	$z \geq 0,\ k \leq 83$	$z \geq 13,\ k \leq 64$	$z \geq 14,\ k \leq 59$	$z \geq 11,\ k \leq 96$	$z \geq 29,\ k \leq 63$	$z \geq 30,\ k \leq 63$
	-768	$z \geq 39,\ k \leq 114$	$z \geq 57,\ k \leq 77$	$z \geq 52,\ k \leq 77$	$z \geq 55,\ k \leq 111$	$z > 64$	$z > 64$
	-1024	$z > 64$	$z > 64$	$z > 64$	$z > 64$	$z > 64$	$z > 64$
2^{96}	-512	$z \geq 0,\ k \leq 58$	$z \geq 8,\ k \leq 53$	$z \geq 1,\ k \leq 58$	$z \geq 0,\ k \leq 63$	$z \geq 33,\ k \leq 54$	$z \geq 25,\ k \leq 58$
	-768	$z \geq 23,\ k \leq 106$	$z \geq 56,\ k \leq 62$	$z \geq 36,\ k \leq 77$	$z \geq 40,\ k \leq 77$	$z > 64$	$z \geq 52,\ k \leq 77$
	-1024	$z > 64$	$z > 64$	$z > 64$	$z > 64$	$z > 64$	$z > 64$
∞	-512	$z \geq 0,\ k = 0$	$z \geq 9,\ k = 0$	$z \geq 1,\ k = 0$	$z \geq 0,\ k = 0$	$z \geq 33,\ k = 0$	$z \geq 26,\ k = 0$
	-768	$z \geq 0,\ k = 0$	$z \geq 52,\ k = 0$	$z \geq 1,\ k = 0$	$z \geq 1,\ k = 0$	$z > 64$	$z \geq 27,\ k = 0$
	-1024	$z \geq 9,\ k = 0$	$z > 64$	$z \geq 1,\ k = 0$	$z \geq 35,\ k = 0$	$z > 64$	$z \geq 28,\ k = 0$

more likely to be feasible → less likely to be feasible

MD	Kyber	$\log \mathbb{E}[\text{GCost}]$ (with $\mathcal{W}$ as in § 4.1) below… Target security	Grover on $\text{AES}_{\{128,192,256\}}$	Quasi-Sqrt $^1/_b\sqrt{\#\mathcal{T} \cdot h}$	$\log \mathbb{E}[\text{GCost}]$ (with $\mathcal{W}$ as in § 4.2) below… Target security	Grover on $\text{AES}_{\{128,192,256\}}$	Quasi-Sqrt $^1/_b\sqrt{\#\mathcal{T} \cdot h}$
2^{40}	-512	$z \geq 7,\ k \leq 92$	$z \geq 13,\ k \leq 83$	$z \geq 26,\ k \leq 59$	$z \geq 23,\ k \leq 96$	$z \geq 29,\ k \leq 79$	$z \geq 42,\ k \leq 63$
	-768	$z \geq 51,\ k \leq 114$	$z \geq 57,\ k \leq 106$	$z \geq 64,\ k \leq 77$	$z > 64$	$z > 64$	$z > 64$
	-1024	$z > 64$	$z > 64$	$z > 64$	$z > 64$	$z > 64$	$z > 64$
2^{64}	-512	$z \geq 0,\ k \leq 83$	$z \geq 13,\ k \leq 64$	$z \geq 14,\ k \leq 59$	$z \geq 11,\ k \leq 96$	$z \geq 29,\ k \leq 63$	$z \geq 30,\ k \leq 63$
	-768	$z \geq 39,\ k \leq 114$	$z \geq 57,\ k \leq 77$	$z \geq 52,\ k \leq 77$	$z \geq 55,\ k \leq 111$	$z > 64$	$z > 64$
	-1024	$z > 64$	$z > 64$	$z > 64$	$z > 64$	$z > 64$	$z > 64$
2^{96}	-512	$z \geq 0,\ k \leq 58$	$z \geq 8,\ k \leq 53$	$z \geq 1,\ k \leq 58$	$z \geq 0,\ k \leq 63$	$z \geq 33,\ k \leq 54$	$z \geq 25,\ k \leq 58$
	-768	$z \geq 23,\ k \leq 106$	$z \geq 56,\ k \leq 62$	$z \geq 36,\ k \leq 77$	$z \geq 40,\ k \leq 77$	$z > 64$	$z \geq 52,\ k \leq 77$
	-1024	$z > 64$	$z > 64$	$z > 64$	$z > 64$	$z > 64$	$z > 64$
∞	-512	$z \geq 0,\ k = 0$	$z \geq 9,\ k = 0$	$z \geq 1,\ k = 0$	$z \geq 0,\ k = 0$	$z \geq 33,\ k = 0$	$z \geq 26,\ k = 0$
	-768	$z \geq 0,\ k = 0$	$z \geq 52,\ k = 0$	$z \geq 1,\ k = 0$	$z \geq 1,\ k = 0$	$z > 64$	$z \geq 27,\ k = 0$
	-1024	$z \geq 9,\ k = 0$	$z > 64$	$z \geq 1,\ k = 0$	$z \geq 35,\ k = 0$	$z > 64$	$z \geq 28,\ k = 0$

- Kyber-768 and -1024 seem out of reach

- Kyber-768 and -1024 seem out of reach
- Kyber-512 within reach in the “query-model”, less clear for “circuit-model”

- Kyber-768 and -1024 seem out of reach
- Kyber-512 within reach in the “query-model”, less clear for “circuit-model”
  - However AES-128 also within reach of Grover key-search in some settings...
  - And we are being quite strict in various parts of the computation

- Kyber-768 and -1024 seem out of reach
- Kyber-512 within reach in the "query-model", less clear for "circuit-model"
  - However AES-128 also within reach of Grover key-search in some settings...
  - And we are being quite strict in various parts of the computation
- Hard to claim this attack obviously works

- Kyber-768 and -1024 seem out of reach
- Kyber-512 within reach in the “query-model”, less clear for “circuit-model”
  - However AES-128 also within reach of Grover key-search in some settings...
  - And we are being quite strict in various parts of the computation
- Hard to claim this attack obviously works

Disclaimer

Yet, we can’t fully exclude it without a clear understanding of the Jensen gap.

- Kyber-768 and -1024 seem out of reach
- Kyber-512 within reach in the “query-model”, less clear for “circuit-model”
  - However AES-128 also within reach of Grover key-search in some settings...
  - And we are being quite strict in various parts of the computation
- Hard to claim this attack obviously works

Disclaimer

Yet, we can’t fully exclude it without a clear understanding of the Jensen gap.

Can we say anything about it?

Reasons to hope Q. Enum doesn't work:

Reasons to hope Q. Enum doesn't work:

- In our numbers we observe that the cost reduces smoothly as a funciton of z
  $\implies$ approximate estimates may already help

Reasons to hope Q. Enum doesn't work:

- In our numbers we observe that the cost reduces smoothly as a funciton of z
  $\implies$ approximate estimates may already help
- Experimental evidence up to $\beta = 70$ says $z \approx 1$

Reasons to hope Q. Enum doesn't work:

- In our numbers we observe that the cost reduces smoothly as a funciton of z
  $\implies$ approximate estimates may already help
- Experimental evidence up to $\beta = 70$ says $z \approx 1$
- We can prove lower bounds on $\mathbb{E}[\sqrt{\#T}]$ based on the additive and multiplicative Jensen's gaps, implying:

$$\mathbb{E}[\sqrt{\#T}] \geq \max\left\{\sqrt{\mathbb{E}[\#T]} - \sqrt[4]{\mathbb{V}[\#T]}, \quad 2^{-\frac{1}{2\ln 2}\sqrt[4]{\mathbb{V}[\#T]}} \cdot \sqrt{\mathbb{E}[\#T]}\right\}.$$

But both depend on $\mathbb{V}[\#T]$.

Open problems

- Not much analysis on $\mathbb{V}[\# T]$

Open problems

- Not much analysis on $\mathbb{V}[\#T]$

$$\#T = \sum_{k=1}^{n} |Z_k| = \sum_{k=1}^{n} \left| \text{Ball}_k(\mathbf{0}, R) \cap Lat\Big(\pi_{n-k+1}(b_{n-k+1}), \ldots, \pi_{n-k+1}(b_n)\Big) \right|$$

Open problems

- Not much analysis on $\mathbb{V}[\#T]$

$$\#T = \sum_{k=1}^{n} |Z_k| = \sum_{k=1}^{n} \left| \mathrm{Ball}_k(\mathbf{0}, R) \cap Lat\left(\pi_{n-k+1}(b_{n-k+1}), \ldots, \pi_{n-k+1}(b_n)\right) \right|$$

$$\underset{\substack{\text{random} \\ \text{tree } T}}{\mathbb{V}}[|\mathrm{Ball}_k(\mathbf{0}, R_k) \cap \pi_{n-k+1}(\Lambda)|]? \qquad \underset{\substack{\text{random} \\ \text{tree } T}}{\mathbb{V}}[\#T]?$$

- There's some results for random real lattices [AEN], but unclear if they apply to lattices during BKZ reduction

Open problems

- Not much analysis on $\mathbb{V}[\# T]$

$$\# T = \sum_{k=1}^{n} |Z_k| = \sum_{k=1}^{n} \left| \mathsf{Ball}_k(\mathbf{0}, R) \cap Lat\Big(\pi_{n-k+1}(b_{n-k+1}), \ldots, \pi_{n-k+1}(b_n)\Big) \right|$$

$$\underset{\substack{\text{random}\\ \text{tree } T}}{\mathbb{V}}[|\mathsf{Ball}_k(\mathbf{0}, R_k) \cap \pi_{n-k+1}(\Lambda)|]? \qquad \underset{\substack{\text{random}\\ \text{tree } T}}{\mathbb{V}}[\# T]?$$

- There's some results for random real lattices [AEN], but unclear if they apply to lattices during BKZ reduction

Open problems

- We've only covered cylinder pruning. What about discrete pruning? Or ad-hoc pruning for quantum enumeration?

Open problems

- We've only covered cylinder pruning. What about discrete pruning? Or ad-hoc pruning for quantum enumeration?
- Currently searching for attack costs is an optimisation problem. Can we find a closed formula? This would allow running it as part of "estimator" scripts.

Open problems

- We've only covered cylinder pruning. What about discrete pruning? Or ad-hoc pruning for quantum enumeration?
- Currently searching for attack costs is an optimisation problem. Can we find a closed formula? This would allow running it as part of "estimator" scripts.
- There quite a few places where our analysis may not be tight, meaning actual costs are likely higher.

Conclusions

Conclusions

- Asymptotically quadratic quantum speedups on enumeration may not hold under max-depth constraints

Conclusions

- Asymptotically quadratic quantum speedups on enumeration may not hold under max-depth constraints
- Technically hard to fully exclude the viability of quantum enumeration

Conclusions

- Asymptotically quadratic quantum speedups on enumeration may not hold under max-depth constraints
- Technically hard to fully exclude the viability of quantum enumeration
- Speedups to the primal lattice attack on Kyber seem unlikely

Conclusions

- Asymptotically quadratic quantum speedups on enumeration may not hold under max-depth constraints
- Technically hard to fully exclude the viability of quantum enumeration
- Speedups to the primal lattice attack on Kyber seem unlikely

Thank you

Slides @ https://fundamental.domains

Martin R. Albrecht, Shi Bai, Pierre-Alain Fouque, Paul Kirchner, Damien Stehlé, and Weiqiang Wen.
Faster enumeration-based lattice reduction: Root hermite factor $k^{1/(2k)}$ time $k^{k/8+o(k)}$.
In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part II*, volume 12171 of *LNCS*, pages 186–212. Springer, Heidelberg, August 2020.

Yoshinori Aono, Thomas Espitau, and Phong Q. Nguyen.
Random lattices: Theory and practice.
Preprint, available at https://espitau.github.io/bin/random_lattice.pdf.

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, *ASIACRYPT 2020, Part II*, volume 12492 of *LNCS*, pages 583–613. Springer, Heidelberg, December 2020.

Yoshinori Aono, Phong Q. Nguyen, and Yixin Shen.
Quantum lattice enumeration and tweaking discrete pruning.
In Thomas Peyrin and Steven Galbraith, editors, *ASIACRYPT 2018, Part I*, volume 11272 of *LNCS*, pages 405–434. Springer, Heidelberg, December 2018.

Yoshinori Aono, Phong Q. Nguyen, Takenobu Seito, and Junji Shikata.
Lower bounds on lattice enumeration with extreme pruning.
In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 608–637. Springer, Heidelberg, August 2018.

Xavier Bonnetain, André Chailloux, André Schrottenloher, and Yixin Shen.
Finding many collisions via reusable quantum walks - application to lattice sieving.

In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology - EUROCRYPT 2023 - 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23-27, 2023, Proceedings, Part V*, volume 14008 of *Lecture Notes in Computer Science*, pages 221–251. Springer, 2023.

Shi Bai, Maya-Iggy van Hoof, Floyd B. Johnson, Tanja Lange, and Tran Ngo.
Concrete analysis of quantum lattice enumeration.
In *Advances in Cryptology - ASIACRYPT 2023*, Lecture Notes in Computer Science. Springer-Verlag, 2023.

Nicolas Gama, Phong Q. Nguyen, and Oded Regev.
Lattice enumeration using extreme pruning.
In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 257–278. Springer, Heidelberg, May / June 2010.

Samuel Jaques, Michael Naehrig, Martin Roetteler, and Fernando Virdia.
Implementing grover oracles for quantum key search on AES and LowMC.
In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part II*, volume 12106 of *LNCS*, pages 280–310. Springer, Heidelberg, May 2020.

Samuel Jaques and Arthur G. Rattew.
Qram: A survey and critique, 2023.

Samuel Jaques and John M. Schanck.
Quantum cryptanalysis in the RAM model: Claw-finding attacks on SIKE.
In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part I*, volume 11692 of *LNCS*, pages 32–61. Springer, Heidelberg, August 2019.

Elena Kirshanova, Erik Mårtensson, Eamonn W. Postlethwaite, and Subhayan Roy Moulik.
Quantum algorithms for the approximate k-list problem and their application to lattice sieving.
In Steven D. Galbraith and Shiho Moriai, editors, *ASIACRYPT 2019, Part I*, volume 11921 of *LNCS*, pages 521–551. Springer, Heidelberg, December 2019.

Thijs Laarhoven, Michele Mosca, and Joop van de Pol.
Solving the shortest vector problem in lattices faster using quantum search.
In Philippe Gaborit, editor, *Post-Quantum Cryptography - 5th International Workshop, PQCrypto 2013*, pages 83–101. Springer, Heidelberg, June 2013.

Ashley Montanaro.
Quantum-walk speedup of backtracking algorithms.
Theory Comput., 14(1):1–24, 2018.

National Institute of Standards and Technology.
Submission requirements and evaluation criteria for the Post-Quantum Cryptography standardization process.
http://csrc.nist.gov/groups/ST/post-quantum-crypto/documents/call-for-proposals-final-dec-2016.pdf, December 2016.

John Preskill.
Quantum Computing in the NISQ era and beyond.
Quantum, 2:79, August 2018.

Christof Zalka.
Grover's quantum searching algorithm is optimal.

Phys. Rev. A, 60:2746–2751, Oct 1999.