On the Security of Proofs of Sequential Work in a Post-Quantum World

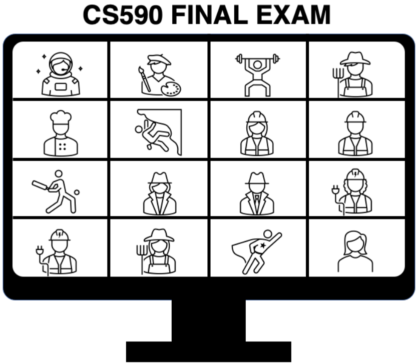
Jeremiah Blocki[1], Seunghoon Lee[1], Samson Zhou[2]

[1] Department of Computer Science, Purdue University
[2] School of Computer Science, Carnegie Mellon University

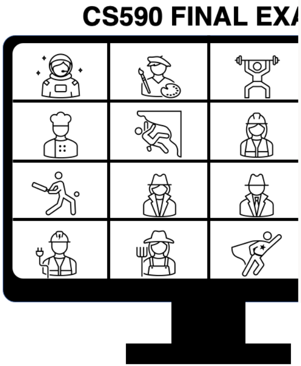
July 28, 2021

Conference on Information-Theoretic Cryptography (ITC) 2021

Motivation: Online Exams during the Pandemic



Motivation: Online Exams during the Pandemic



[CS590] 5 mins late - having internet issue

CG Cinseer Goodman
Tue 5/2/2021 9:05 PM
To: Seunghoon Lee

answer-goodman.pdf
157 KB

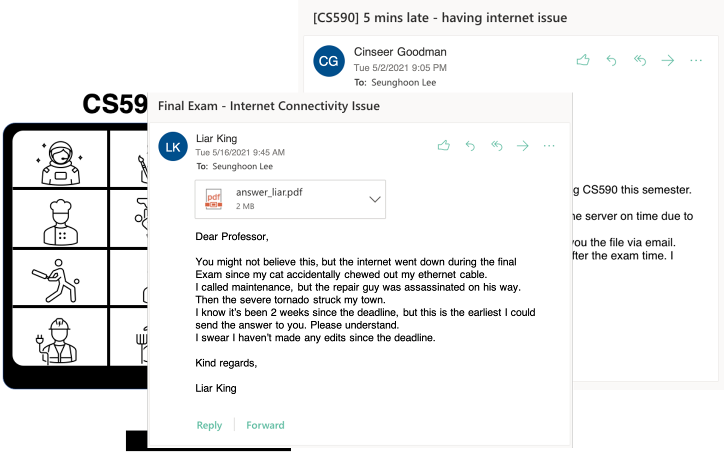
Dear Professor,

My name is Cinseer Goodman who is taking CS590 this semester.
I hope this email finds you well.
I was not able to submit the final exam to the server on time due to an unexpected internet connectivity loss.
It just went back 5 minutes later so I send you the file via email.
I promise I have not done any extra work after the exam time. I hope it works.
Thank you.

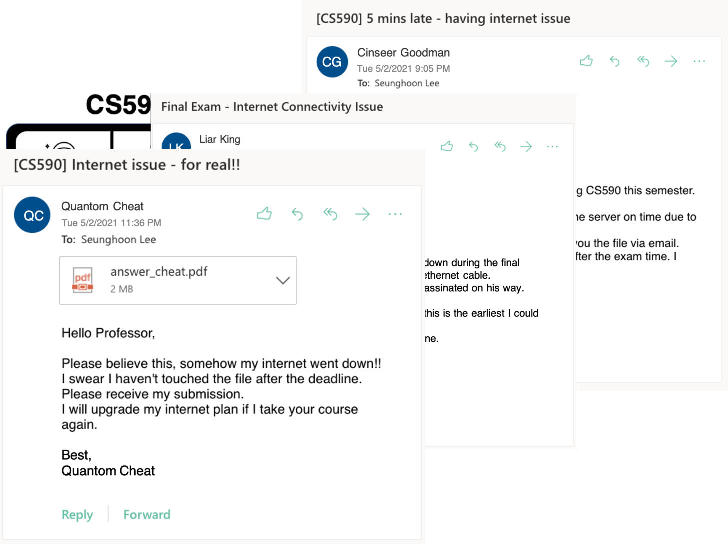
Best,
Cinseer Goodman

Reply | Forward

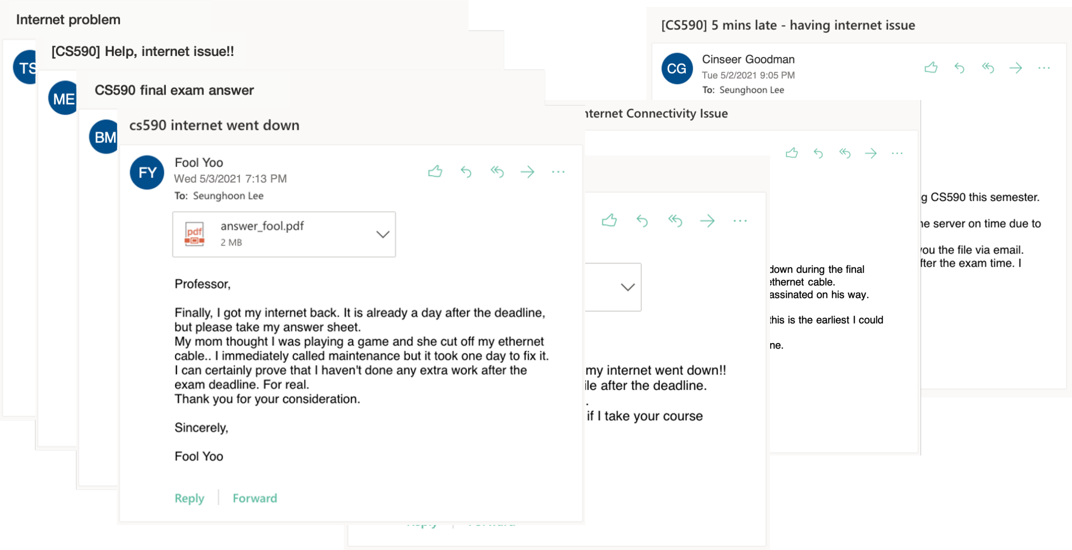
Motivation: Online Exams during the Pandemic



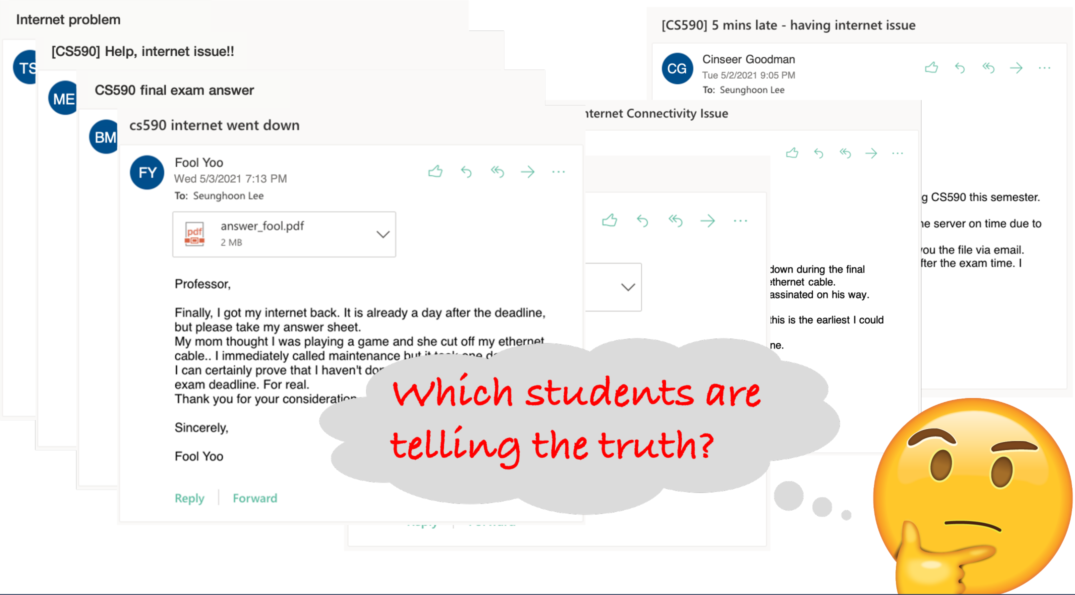
Motivation: Online Exams during the Pandemic



Motivation: Online Exams during the Pandemic



Motivation: Online Exams during the Pandemic



Solution: Proofs of Sequential Work (PoSW)

What is a Proof of Sequential Work? (Informal)

A proof that a large amount (N) of sequential work was performed after a prover committed an initial message, e.g., the solution for the final exam

Solution: Proofs of Sequential Work (PoSW)

What is a Proof of Sequential Work? (Informal)

A proof that a large amount (N) of sequential work was performed after a prover committed an initial message, e.g., the solution for the final exam

Initial approach: iterative hash chain

$$\text{☺} : \text{🗎} \mapsto \mathcal{H}(\text{🗎}) \mapsto \mathcal{H}^2(\text{🗎}) \mapsto \mathcal{H}^3(\text{🗎}) \mapsto \cdots \mapsto \mathcal{H}^{N-1}(\text{🗎}) \mapsto \mathcal{H}^{N}(\text{🗎})$$

$$\text{😜} : \text{🖹} \mapsto \mathcal{H}(\text{🖹}) \mapsto \mathcal{H}^2(\text{🖹}) \mapsto \mathcal{H}^3(\text{🖹}) \mapsto \cdots \mapsto \mathcal{H}^{N'-1}(\text{🖹}) \mapsto \mathcal{H}^{N'}(\text{🖹})$$

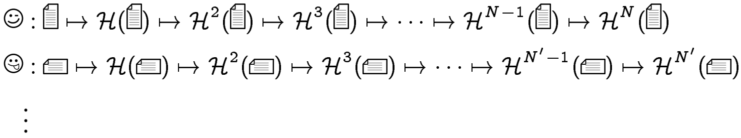
$\vdots$

Solution: Proofs of Sequential Work (PoSW)

What is a Proof of Sequential Work? (Informal)

A proof that a large amount (N) of sequential work was performed after a prover committed an initial message, e.g., the solution for the final exam

Initial approach: iterative hash chain



- Disadvantage: Instructor needs to recompute the whole thing
- Many late students? $\rightarrow$ insufficient computational resources to verify all solutions

Solution: Proofs of Sequential Work (PoSW)

What is a Proof of Sequential Work? (Informal)

A proof that a large amount (N) of sequential work was performed after a prover committed an initial message, e.g., the solution for the final exam

Initial approach: iterative hash chain

$$\text{☺} : \text{🗎} \mapsto \mathcal{H}(\text{🗎}) \mapsto \mathcal{H}^2(\text{🗎}) \mapsto \mathcal{H}^3(\text{🗎}) \mapsto \cdots \mapsto \mathcal{H}^{N-1}(\text{🗎}) \mapsto \mathcal{H}^{N}(\text{🗎})$$

$$\text{😛} : \text{🖹} \mapsto \mathcal{H}(\text{🖹}) \mapsto \mathcal{H}^2(\text{🖹}) \mapsto \mathcal{H}^3(\text{🖹}) \mapsto \cdots \mapsto \mathcal{H}^{N'-1}(\text{🖹}) \mapsto \mathcal{H}^{N'}(\text{🖹})$$

$$\vdots$$

- Disadvantage: Instructor needs to recompute the whole thing
- Many late students? $\rightarrow$ insufficient computational resources to verify all solutions

Additional requirements:

- Efficiency: instructor (verifier) should be able to quickly verify the proofs (e.g., in time $\mathsf{polylog}\ N$), and

Solution: Proofs of Sequential Work (PoSW)

What is a Proof of Sequential Work? (Informal)

A proof that a large amount (N) of sequential work was performed after a prover committed an initial message, e.g., the solution for the final exam

Initial approach: iterative hash chain

$$\smiley : \square \mapsto \mathcal{H}(\square) \mapsto \mathcal{H}^2(\square) \mapsto \mathcal{H}^3(\square) \mapsto \cdots \mapsto \mathcal{H}^{N-1}(\square) \mapsto \mathcal{H}^{N}(\square)$$

$$\smiley : \square \mapsto \mathcal{H}(\square) \mapsto \mathcal{H}^2(\square) \mapsto \mathcal{H}^3(\square) \mapsto \cdots \mapsto \mathcal{H}^{N'-1}(\square) \mapsto \mathcal{H}^{N'}(\square)$$

$$\vdots$$

- Disadvantage: Instructor needs to recompute the whole thing
- Many late students? $\rightarrow$ insufficient computational resources to verify all solutions

Additional requirements:

- Efficiency: instructor (verifier) should be able to quickly verify the proofs (e.g., in time $\mathsf{polylog}\ N$), and
- Soundness: students (prover) should *not* be able to produce a *valid* proof faster (than sequential time $\Omega(N)$, even if running in parallel).

PoSW Constructions

Mahmoody et al. [MMV13]: the first theoretical construction of a PoSW

- Verifier time $\mathtt{polylog}\ N$, and prover time $\Omega(N)$,
- Parallel cheating prover running in sequential time $< N$ cannot fool the verifier, and
- Security proof in the classical ROM.

PoSW Constructions

Mahmoody et al. [MMV13]: the first theoretical construction of a PoSW

- Verifier time $\mathtt{polylog}\ N$, and prover time $\Omega(N)$,
- Parallel cheating prover running in sequential time $< N$ cannot fool the verifier, and
- Security proof in the classical ROM.

Cohen and Pietrzak [CP18]: an improved & practical PoSW construction

- Modular security proof in the classical ROM:
  - Any parallel cheating prover (for the PoSW) must produce a long $\mathcal{H}$-sequence (whp), and
  - Any parallel cheating prover running in time $< N$ cannot produce an $\mathcal{H}$-sequence of length N (whp).

PoSW Constructions

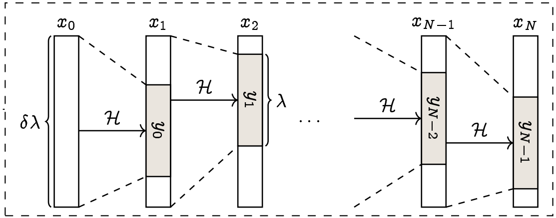
Mahmoody et al. [MMV13]: the first theoretical construction of a PoSW

- Verifier time $\mathtt{polylog}\, N$, and prover time $\Omega(N)$,
- Parallel cheating prover running in sequential time $< N$ cannot fool the verifier, and
- Security proof in the classical ROM.

Cohen and Pietrzak [CP18]: an improved & practical PoSW construction

- Modular security proof in the classical ROM:
  - Any parallel cheating prover (for the PoSW) must produce a long $\mathcal{H}$-sequence (whp), and
  - Any parallel cheating prover running in time $< N$ cannot produce an $\mathcal{H}$-sequence of length N (whp).

$x_0, x_1, \ldots, x_N \in \{0,1\}^*$ s.t.
for each $1 \leq i \leq N$, there
exist $a, b \in \{0,1\}^*$ such that
$x_i = a||\mathcal{H}(x_{i-1})||b$.



PoSW Constructions

Mahmoody et al. [MMV13]: the first theoretical construction of a PoSW

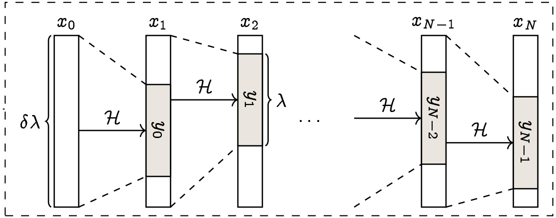
- Verifier time $\mathsf{polylog}\, N$, and prover time $\Omega(N)$,
- Parallel cheating prover running in sequential time $< N$ cannot fool the verifier, and
- Security proof in the classical ROM.

Cohen and Pietrzak [CP18]: an improved & practical PoSW construction

Post-Quantum Security?

- Modular security proof in the classical ROM:
  - Any parallel cheating prover (for the PoSW) must produce a long $\mathcal{H}$-sequence (whp), and
  - Any parallel cheating prover running in time $< N$ cannot produce an $\mathcal{H}$-sequence of length N (whp).

$x_0, x_1, \ldots, x_N \in \{0,1\}^*$ s.t. for each $1 \le i \le N$, there exist $a, b \in \{0,1\}^*$ such that $x_i = a \| \mathcal{H}(x_{i-1}) \| b$.



Post-Quantum Security of the PoSW

Cohen and Pietrzak [CP18]:

- Any parallel cheating prover (for the PoSW) must produce a long $\mathcal{H}$-sequence,
- Any parallel cheating prover running in time $< N$ cannot produce an $\mathcal{H}$-sequence of length N.

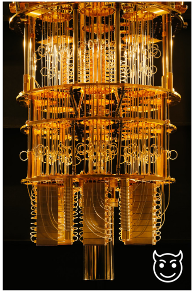
Post-Quantum Security of the PoSW

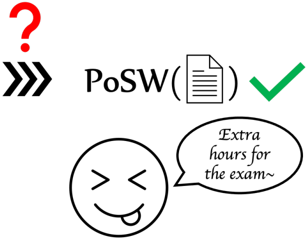
Cohen and Pietrzak [CP18]:

- Any parallel cheating prover (for the PoSW) must produce a long $\mathcal{H}$-sequence,
- Any parallel cheating prover running in time $< N$ cannot produce an $\mathcal{H}$-sequence of length N.

Key Research Questions:

- Can a sequentially time-bounded parallel *quantum* attacker produce a long $\mathcal{H}$-sequence?
- Can a sequentially time-bounded parallel *quantum* attacker produce a valid PoSW?





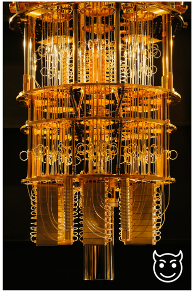
Post-Quantum Security of the PoSW

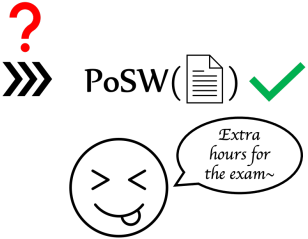
Cohen and Pietrzak [CP18]:

- Any parallel cheating prover (for the PoSW) must produce a long $\mathcal{H}$-sequence,
- Any parallel cheating prover running in time $< N$ cannot produce an $\mathcal{H}$-sequence of length N.

Key Research Questions:

- Can a sequentially time-bounded parallel *quantum* attacker produce a long $\mathcal{H}$-sequence?
- Can a sequentially time-bounded parallel *quantum* attacker produce a valid PoSW?





Short answer: NO!

Our Result. Hardness of Producing an $\mathcal{H}$-Sequence/PoSW in a Quantum Setting

Theorem (informal)

A quantum adversary making at most $q \ll 2^{\lambda/3}$ queries over $N-1$ rounds outputs an $\mathcal{H}$-sequence of length N $(x_0, \ldots, x_N$ with $|x_i| \le \delta\lambda$ where $\delta \ge 1)$ with negligible probability $\mathcal{O}\left(\frac{q^3\delta\lambda}{2^\lambda}\right)$.

Our Result. Hardness of Producing an $\mathcal{H}$-Sequence/PoSW in a Quantum Setting

Theorem (informal)

A quantum adversary making at most $q \ll 2^{\lambda/3}$ queries over $N-1$ rounds outputs an $\mathcal{H}$-sequence of length N $(x_0, \ldots, x_N$ with $|x_i| \leq \delta\lambda$ where $\delta \geq 1)$ with negligible probability $\mathcal{O}\left(\frac{q^3\delta\lambda}{2^\lambda}\right)$.

Theorem (informal)

Suppose $\mathcal{A}$ makes at most $q \ll 2^{\lambda/\log N}$ quantum queries to the random oracle $\mathcal{H}$ over at most $T = (1-\alpha)N$ rounds. Then $\mathcal{A}$ outputs a valid non-interactive PoSW with negligible probability $\mathcal{O}\left(q^2(1-\alpha)^{\frac{\lambda}{\log N}} + \frac{q^3\lambda\log N}{2^\lambda}\right)$.

Our Result. Hardness of Producing an $\mathcal{H}$-Sequence/PoSW in a Quantum Setting

Theorem (informal)

A quantum adversary making at most $q \ll 2^{\lambda/3}$ queries over $N-1$ rounds outputs an $\mathcal{H}$-sequence of length N $(x_0, \ldots, x_N$ with $|x_i| \leq \delta\lambda$ where $\delta \geq 1)$ with negligible probability $\mathcal{O}\left(\frac{q^3\delta\lambda}{2^\lambda}\right)$.

Theorem (informal)

Suppose $\mathcal{A}$ makes at most $q \ll 2^{\lambda/\log N}$ quantum queries to the random oracle $\mathcal{H}$ over at most $T = (1-\alpha)N$ rounds. Then $\mathcal{A}$ outputs a valid non-interactive PoSW with negligible probability $\mathcal{O}\left(q^2(1-\alpha)^{\frac{\lambda}{\log N}} + \frac{q^3\lambda\log N}{2^\lambda}\right)$.

- We give a direct proof for the non-interactive version of the PoSW from [CP18], and
- Our proofs are in the parallel quantum random oracle model (pqROM).

Our Result. Hardness of Producing an $\mathcal{H}$-Sequence/PoSW in a Quantum Setting

Theorem (informal)

A quantum adversary making at most $q \ll 2^{\lambda/3}$ queries over $N-1$ rounds outputs an $\mathcal{H}$-sequence of length N $(x_0, \ldots, x_N$ with $|x_i| \le \delta\lambda$ where $\delta \ge 1)$ with negligible probability $\mathcal{O}\left(\frac{q^3\delta\lambda}{2^\lambda}\right)$.

Theorem (informal)

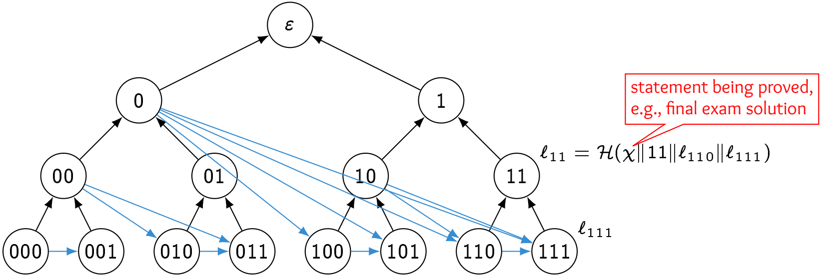
Suppose $\mathcal{A}$ makes at most $q \ll 2^{\lambda/\log N}$ quantum queries to the random oracle $\mathcal{H}$ over at most $T = (1-\alpha)N$ rounds. Then $\mathcal{A}$ outputs a valid non-interactive PoSW with negligible probability $\mathcal{O}\left(q^2(1-\alpha)^{\frac{\lambda}{\log N}} + \frac{q^3\lambda\log N}{2^\lambda}\right)$.

- We give a direct proof for the non-interactive version of the PoSW from [CP18], and
- Our proofs are in the parallel quantum random oracle model (pqROM).

Concurrent/Subsequent Work.

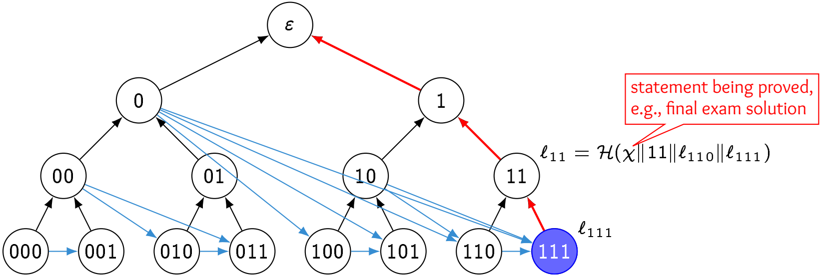
- Chung et al. [CFHL21]: also gave a comparable security bounds for the PoSW in the pqROM

The [CP18] Construction



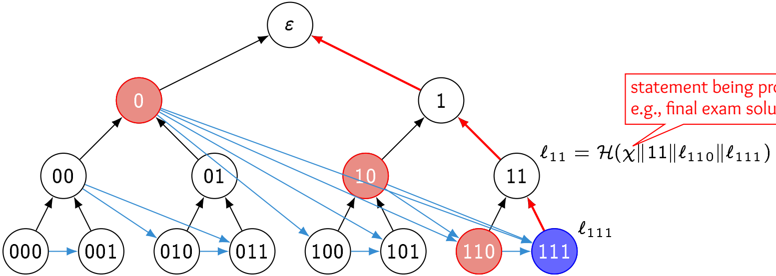
- For all leaf nodes v, add an edge (u, v) for any u that is a left sibling of a node on the path from v to the root ε

The [CP18] Construction



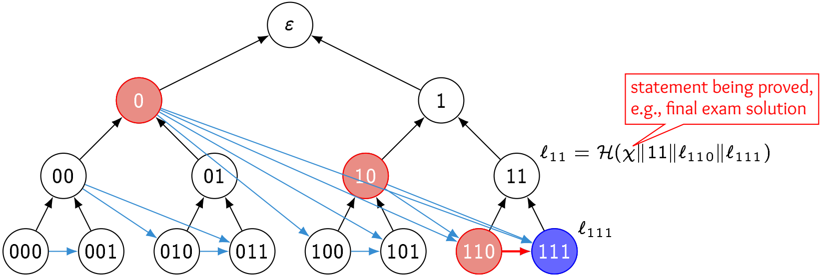
- For all leaf nodes v, add an edge (u, v) for any u that is a left sibling of a node on the path from v to the root ε

The [CP18] Construction



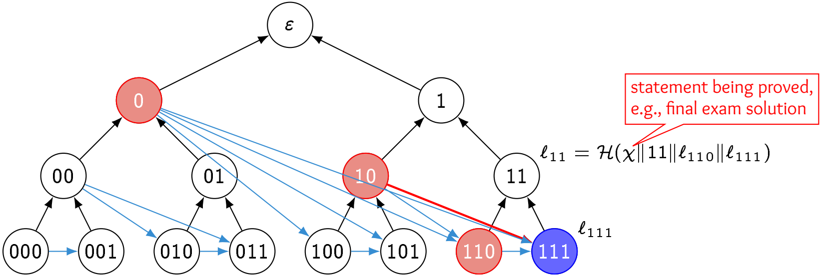
- For all leaf nodes v, add an edge (u, v) for any u that is a left sibling of a node on the path from v to the root ε

The [CP18] Construction



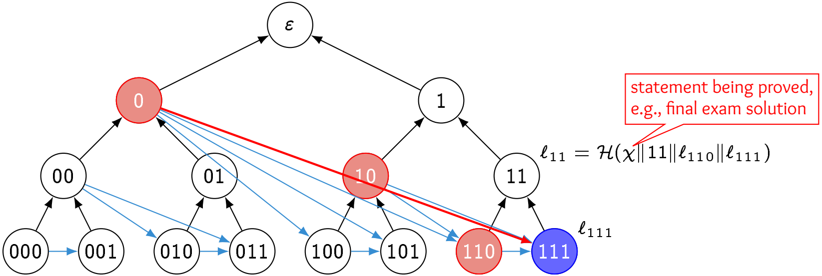
- For all leaf nodes v, add an edge (u, v) for any u that is a left sibling of a node on the path from v to the root ε

The [CP18] Construction



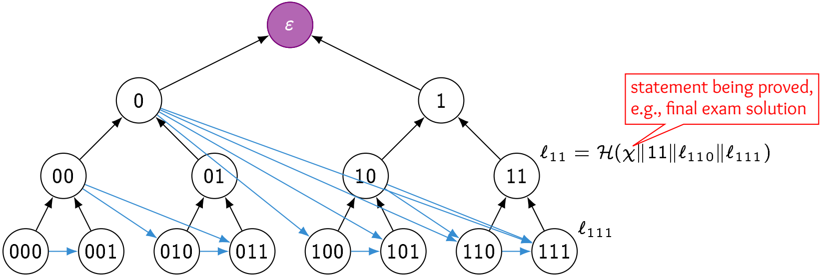
- For all leaf nodes v, add an edge (u, v) for any u that is a left sibling of a node on the path from v to the root ε

The [CP18] Construction



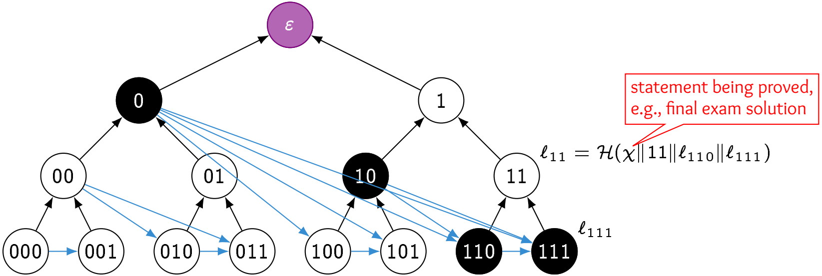
- For all leaf nodes v, add an edge (u, v) for any u that is a left sibling of a node on the path from v to the root ε

The [CP18] Construction



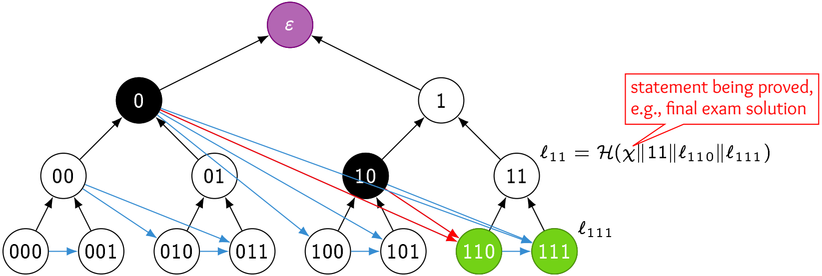
- For all leaf nodes v, add an edge (u, v) for any u that is a left sibling of a node on the path from v to the root ε
- Each node has a label, a hash of its parents
- The label of root node forms a **Merkle tree commitment** of all the other nodes
  - Verifier can audit the prover by forcing the prover to open certain labels
  - Show that they are locally consistent

The [CP18] Construction



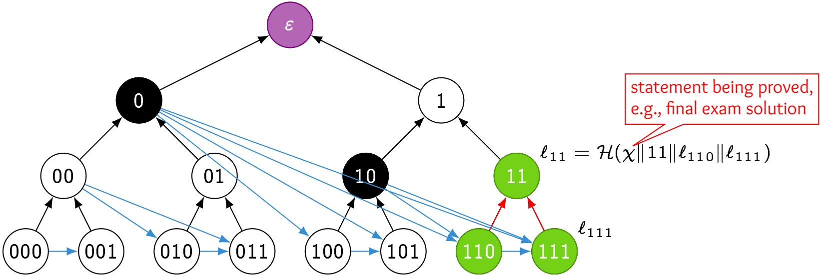
- For all leaf nodes v, add an edge (u, v) for any u that is a left sibling of a node on the path from v to the root ε
- Each node has a label, a hash of its parents
- The label of root node forms a **Merkle tree commitment** of all the other nodes
  - Verifier can audit the prover by forcing the prover to open certain labels
  - Show that they are locally consistent

The [CP18] Construction



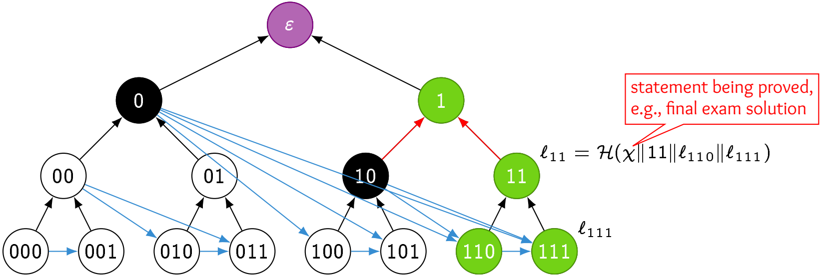
- For all leaf nodes v, add an edge (u, v) for any u that is a left sibling of a node on the path from v to the root ε
- Each node has a label, a hash of its parents
- The label of root node forms a **Merkle tree commitment** of all the other nodes
  - Verifier can audit the prover by forcing the prover to open certain labels
  - Show that they are locally consistent

The [CP18] Construction



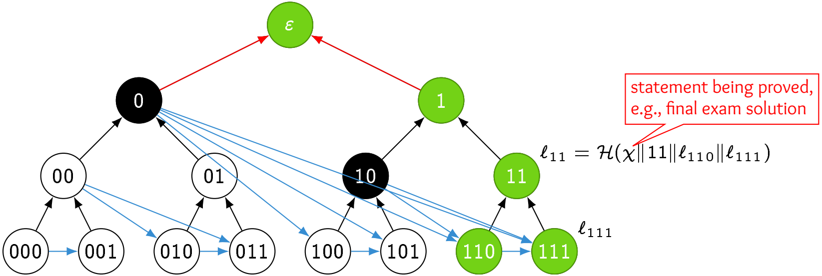
- For all leaf nodes v, add an edge (u, v) for any u that is a left sibling of a node on the path from v to the root ε
- Each node has a label, a hash of its parents
- The label of root node forms a **Merkle tree commitment** of all the other nodes
  - Verifier can audit the prover by forcing the prover to open certain labels
  - Show that they are locally consistent

The [CP18] Construction



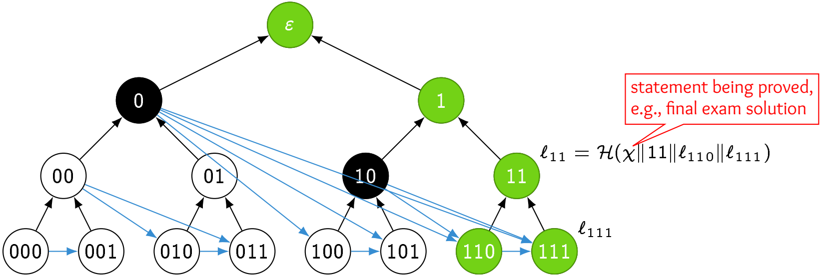
- For all leaf nodes v, add an edge (u, v) for any u that is a left sibling of a node on the path from v to the root ε
- Each node has a label, a hash of its parents
- The label of root node forms a **Merkle tree commitment** of all the other nodes
  - Verifier can audit the prover by forcing the prover to open certain labels
  - Show that they are locally consistent

The [CP18] Construction



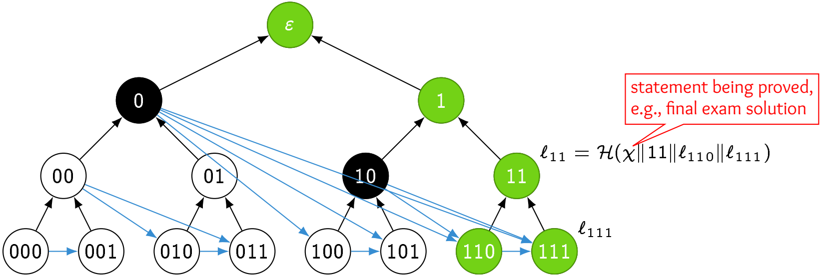
- For all leaf nodes v, add an edge (u, v) for any u that is a left sibling of a node on the path from v to the root ε
- Each node has a label, a hash of its parents
- The label of root node forms a **Merkle tree commitment** of all the other nodes
  - Verifier can audit the prover by forcing the prover to open certain labels
  - Show that they are locally consistent

The [CP18] Construction



- For all leaf nodes v, add an edge (u, v) for any u that is a left sibling of a node on the path from v to the root ε
- Each node has a label, a hash of its parents
- The label of root node forms a **Merkle tree commitment** of all the other nodes
  - Verifier can audit the prover by forcing the prover to open certain labels
  - Show that they are locally consistent
- Audit process: interactive or non-interactive (Fiat-Shamir)

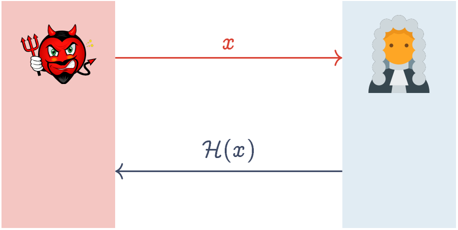
The [CP18] Construction



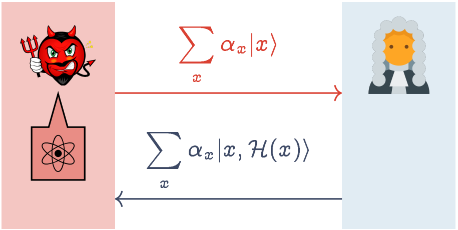
- For all leaf nodes v, add an edge (u, v) for any u that is a left sibling of a node on the path from v to the root ε
- Each node has a label, a hash of its parents
- The label of root node forms a **Merkle tree commitment** of all the other nodes
  - Verifier can audit the prover by forcing the prover to open certain labels
  - Show that they are locally consistent
- Audit process: interactive or non-interactive (Fiat-Shamir)
- Any classical ROM attacker that produces a valid PoSW in time $< N$ must produce a long $\mathcal{H}$-sequence

ROM vs qROM [BDF+11]

⟨Classical ROM⟩

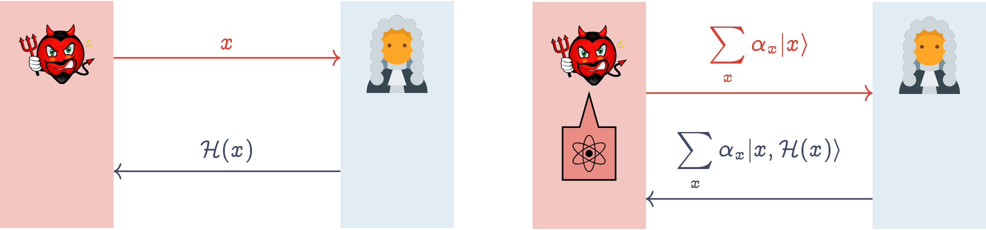


⟨Quantum ROM⟩



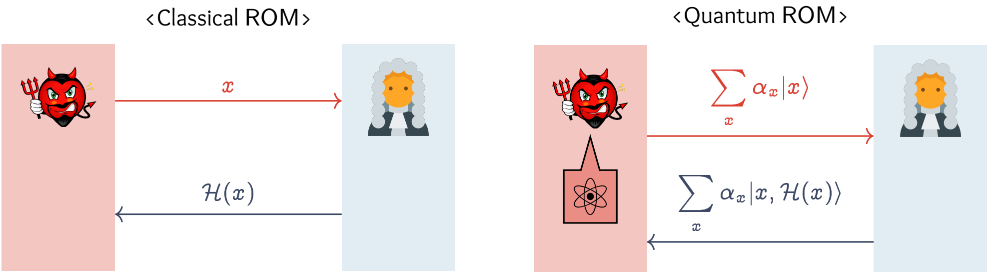
ROM vs qROM [BDF+11]

<Classical ROM> <Quantum ROM>



- Security proofs are much more challenging in the qROM
  - Programmability & Extractability (ROM: ✔, qROM: ✘)
  - Recording quantum queries?

ROM vs qROM [BDF+11]



- Security proofs are much more challenging in the qROM
  - Programmability & Extractability (ROM: ✔, qROM: ✘)
  - Recording quantum queries?
- Compressed Oracle Technique [Zha19]: change of view (compressed phase oracle (CPhsO))

$$|x, y\rangle \otimes |\mathcal{H}\rangle \mapsto |x, y \oplus \mathcal{H}(x)\rangle \otimes |\mathcal{H}\rangle$$

$$\Updownarrow$$

$$|x, y\rangle \otimes |\mathcal{H}\rangle \mapsto (-1)^{y \cdot \mathcal{H}(x)} |x, y\rangle \otimes |\mathcal{H}\rangle$$

Compressed Phase Oracle (CPhsO)

A database $\mathcal{D} := \{(x_i, y_i), i \geq 1\}$, where $\mathcal{D}(x_i) = y_i$.

How to view a random oracle?

- Classical: databases of known I/O pairs & unknown I/O pairs don't appear
- Quantum: superposition over databases (known I/O pairs + indeterminates)

Compressed Phase Oracle (CPhsO)

A database $\mathcal{D} := \{(x_i, y_i), i \geq 1\}$, where $\mathcal{D}(x_i) = y_i$.

How to view a random oracle?

- Classical: databases of known I/O pairs & unknown I/O pairs don't appear
- Quantum: superposition over databases (known I/O pairs + indeterminates)

After q queries,

The state can be viewed as

$$\sum_{x,y,z,\mathcal{D}} \alpha_{x,y,z,\mathcal{D}} |x, y, z\rangle \otimes |\mathcal{D}\rangle,$$

where

Compressed Phase Oracle (CPhsO)

A database $\mathcal{D} := \{(x_i, y_i), i \geq 1\}$, where $\mathcal{D}(x_i) = y_i$.

How to view a random oracle?

- Classical: databases of known I/O pairs & unknown I/O pairs don't appear
- Quantum: superposition over databases (known I/O pairs + indeterminates)

After q queries,

The state can be viewed as

$$\sum_{x,y,z,\mathcal{D}} \alpha_{x,y,z,\mathcal{D}} |x, y, z\rangle \otimes |\mathcal{D}\rangle,$$

where

- query registers,

Compressed Phase Oracle (CPhsO)

A database $\mathcal{D} := \{(x_i, y_i), i \geq 1\}$, where $\mathcal{D}(x_i) = y_i$.

How to view a random oracle?

- Classical: databases of known I/O pairs & unknown I/O pairs don't appear
- Quantum: superposition over databases (known I/O pairs + indeterminates)

After q queries,

The state can be viewed as

$$\sum_{x,y,z,\mathcal{D}} \alpha_{x,y,z,\mathcal{D}} | x, y, z \rangle \otimes |\mathcal{D}\rangle,$$

where

- query registers,
- auxiliary input (algorithm state), and

Compressed Phase Oracle (CPhsO)

A database $\mathcal{D} := \{(x_i, y_i), i \geq 1\}$, where $\mathcal{D}(x_i) = y_i$.

How to view a random oracle?

- Classical: databases of known I/O pairs & unknown I/O pairs don't appear
- Quantum: superposition over databases (known I/O pairs + indeterminates)

After q queries,

The state can be viewed as

$$\sum_{x,y,z,\mathcal{D}} \alpha_{x,y,z,\mathcal{D}} |x, y, z\rangle \otimes |\mathcal{D}\rangle,$$

where

- query registers,
- auxiliary input (algorithm state), and
- a compressed dataset of at most q input/output pairs.

Extending Compressed Oracle Technique to the pqROM: the oracle CPhsO^k

Extending Compressed Oracle Technique to the pqROM: the oracle CPhsO^k

Example: Single Query (simplest case)

$$|x, y, z\rangle \otimes |\mathcal{D}\rangle \xmapsto[(x,y)\notin\mathcal{D}]{\mathsf{CPhsO}} |x, y, z\rangle \otimes \sum_{w} (-1)^{y\cdot w} |\mathcal{D} \cup (x, w)\rangle.$$

- w ranges over all possible outputs of $\mathcal{H}(x)$.

Extending Compressed Oracle Technique to the pqROM: the oracle CPhsO^k

Example: Single Query (simplest case)

$$|x, y, z\rangle \otimes |\mathcal{D}\rangle \xmapsto[(x,y)\notin\mathcal{D}]{\mathsf{CPhsO}} |x, y, z\rangle \otimes \sum_{w} (-1)^{y \cdot w} |\mathcal{D} \cup (x, w)\rangle.$$

- w ranges over all possible outputs of $\mathcal{H}(x)$.

Extending Compressed Oracle Technique to the pqROM: the oracle CPhsO^k

Example: Single Query (simplest case)

$$|x, y, z\rangle \otimes |\mathcal{D}\rangle \xrightarrow[(x,y)\notin\mathcal{D}]{\text{CPhsO}} |x, y, z\rangle \otimes \sum_{w} (-1)^{y\cdot w} |\mathcal{D} \cup (x, w)\rangle.$$

- w ranges over all possible outputs of $\mathcal{H}(x)$.

Example: Parallel Query (simplest case)

$$|(x_1, y_1), \ldots, (x_k, y_k), z\rangle \otimes |\mathcal{D}\rangle$$

$$\xrightarrow{\text{CPhsO}^k} |(x_1, y_1), \ldots, (x_k, y_k), z\rangle \otimes \sum_{w_1,\ldots,w_k} (-1)^{\sum_{i=1}^{k} x_i \cdot w_i} |\mathcal{D} \cup \{(x_1, w_1), \ldots, (x_k, w_k)\}\rangle,$$

where

Extending Compressed Oracle Technique to the pqROM: the oracle CPhsO^k

Example: Single Query (simplest case)

$$|x, y, z\rangle \otimes |\mathcal{D}\rangle \xmapsto[(x,y)\notin\mathcal{D}]{\mathsf{CPhsO}} |x, y, z\rangle \otimes \sum_{w} (-1)^{y\cdot w} |\mathcal{D} \cup (x, w)\rangle.$$

- w ranges over all possible outputs of $\mathcal{H}(x)$.

Example: Parallel Query (simplest case)

$$|(x_1, y_1), \ldots, (x_k, y_k), z\rangle \otimes |\mathcal{D}\rangle$$

$$\xmapsto{\mathsf{CPhsO}^k} |\boxed{(x_1, y_1), \ldots, (x_k, y_k)}, z\rangle \otimes \sum_{w_1,\ldots,w_k} (-1)^{\sum_{i=1}^{k} x_i \cdot w_i} |\mathcal{D} \cup \{(x_1, w_1), \ldots, (x_k, w_k)\}\rangle,$$

where

- simplest case: $(x_1, y_1), \ldots, (x_k, y_k) \notin \mathcal{D}$ and all (x_i, y_i)'s are distinct, and

Extending Compressed Oracle Technique to the pqROM: the oracle CPhsO^k

Example: Single Query (simplest case)

$$|x, y, z\rangle \otimes |\mathcal{D}\rangle \xmapsto[(x,y)\notin\mathcal{D}]{\text{CPhsO}} |x, y, z\rangle \otimes \sum_{w} (-1)^{y\cdot w} |\mathcal{D} \cup (x, w)\rangle.$$

- w ranges over all possible outputs of $\mathcal{H}(x)$.

Example: Parallel Query (simplest case)

$$|(x_1, y_1), \ldots, (x_k, y_k), z\rangle \otimes |\mathcal{D}\rangle$$

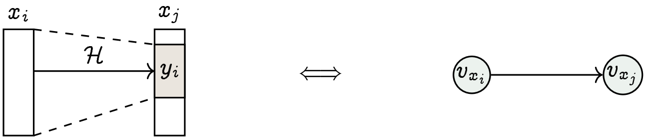
$$\xmapsto{\text{CPhsO}^k} |\boxed{(x_1, y_1), \ldots, (x_k, y_k)}, z\rangle \otimes \sum_{w_1,\ldots,w_k} (-1)^{\sum_{i=1}^{k} x_i \cdot w_i} |\mathcal{D} \cup \{(x_1, w_1), \ldots, (x_k, w_k)\}\rangle,$$

where

- simplest case: $(x_1, y_1), \ldots, (x_k, y_k) \notin \mathcal{D}$ and all (x_i, y_i)'s are distinct, and
- w_i's range over all possible outputs of $\mathcal{H}(x_i)$'s for each i.

Notations

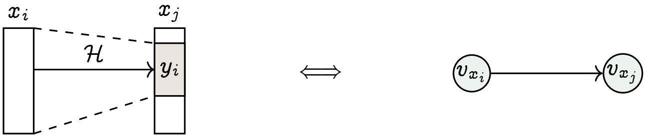
- Given a database $\mathcal{D} = \{(x_1, y_1), \ldots, (x_q, y_q)\}$, define a directed graph $G_{\mathcal{D}}$ on q nodes $(v_{x_1}, \ldots, v_{x_q})$ such that:



- $\mathsf{PATH}_s := \{\mathcal{D} : G_{\mathcal{D}}$ contains a path of length $s\}$ (set of databases), and
- $\widetilde{\mathsf{PATH}}_s := \{|(x_1, y_1), \ldots, (x_k, y_k), z\rangle \otimes |\mathcal{D}\rangle : \mathcal{D} \in \mathsf{PATH}_s\}$ (set of basis states).

Notations

- Given a database $\mathcal{D} = \{(x_1, y_1), \ldots, (x_q, y_q)\}$, define a directed graph $G_{\mathcal{D}}$ on q nodes $(v_{x_1}, \ldots, v_{x_q})$ such that:



- $\mathsf{PATH}_s := \{\mathcal{D} : G_{\mathcal{D}} \text{ contains a path of length } s\}$ (set of databases), and
- $\widetilde{\mathsf{PATH}}_s := \{|(x_1, y_1), \ldots, (x_k, y_k), z\rangle \otimes |\mathcal{D}\rangle : \mathcal{D} \in \mathsf{PATH}_s\}$ (set of basis states).

$\mathcal{D}$ contains an $\mathcal{H}$-sequence of length s $\quad\Longleftrightarrow\quad$ $\mathcal{D} \in \mathsf{PATH}_s$

Proof Ideas: Hardness of Producing an $\mathcal{H}$-sequence in a Quantum Setting

Lemma

$|\varphi\rangle$: *an initial state, and let* $|\varphi'\rangle = \mathsf{CPhsO}^k|\varphi\rangle$. *Then* $L_2(|\varphi'\rangle, \widetilde{\mathsf{PATH}}_{s+1}) - L_2(|\varphi\rangle, \widetilde{\mathsf{PATH}}_s) \leq \frac{4k\sqrt{(q+k)\delta\lambda}}{2^{\lambda/2}}$.

Interpretation/Intuition:

- $L_2(|\varphi\rangle, \widetilde{\mathsf{PATH}}_s)$: 2-norm of the projection of $|\varphi\rangle$ onto $\widetilde{\mathsf{PATH}}_s$, i.e.,

$$|\varphi\rangle = \sum_X \alpha_X |X\rangle \quad \Rightarrow \quad L_2(|\varphi\rangle, \widetilde{\mathsf{PATH}}_s) = \sqrt{\sum_{|X\rangle \in \widetilde{\mathsf{PATH}}_s} |\alpha_X|^2}.$$

- If we start with the state that is nearly orthogonal to $\widetilde{\mathsf{PATH}}_s$, then after applying the oracle CPhsO^k, the resulting state is also nearly orthogonal to $\widetilde{\mathsf{PATH}}_{s+1}$.

Proof Ideas: Hardness of Producing an $\mathcal{H}$-sequence in a Quantum Setting

Lemma

$|\varphi\rangle$: an initial state, and let $|\varphi'\rangle = \mathsf{CPhsO}^k|\varphi\rangle$. Then $L_2(|\varphi'\rangle, \widetilde{\mathsf{PATH}}_{s+1}) - L_2(|\varphi\rangle, \widetilde{\mathsf{PATH}}_s) \leq \frac{4k\sqrt{(q+k)\delta\lambda}}{2^{\lambda/2}}$.

Basic proof idea: split the states into good and bad part. (in this talk, suppose that $x_1, \ldots, x_k \notin \mathcal{D}$ and all distinct for simplicity)

$$|\varphi\rangle = |(x_1, y_1), \ldots, (x_k, y_k), z\rangle \otimes |\mathcal{D}\rangle$$

Proof Ideas: Hardness of Producing an $\mathcal{H}$-sequence in a Quantum Setting

Lemma

$|\varphi\rangle$: an initial state, and let $|\varphi'\rangle = \mathsf{CPhsO}^k|\varphi\rangle$. Then $L_2(|\varphi'\rangle, \widetilde{\mathsf{PATH}}_{s+1}) - L_2(|\varphi\rangle, \widetilde{\mathsf{PATH}}_s) \le \frac{4k\sqrt{(q+k)\delta\lambda}}{2^{\lambda/2}}$.

Basic proof idea: split the states into good and bad part. (in this talk, suppose that $x_1, \ldots, x_k \notin \mathcal{D}$ and all distinct for simplicity)

$$|\varphi\rangle = |(x_1, y_1), \ldots, (x_k, y_k), z\rangle \otimes |\mathcal{D}\rangle$$
$$\xrightarrow{\mathsf{CPhsO}^k} |\varphi'\rangle = |(x_1, y_1), \ldots, (x_k, y_k), z\rangle \otimes \sum_{w_1, \ldots, w_k} (-1)^{\sum y_i w_i} |\mathcal{D} \cup \{(x_1, w_1), \ldots, (x_k, w_k)\}\rangle$$

Proof Ideas: Hardness of Producing an $\mathcal{H}$-sequence in a Quantum Setting

Lemma

$|\varphi\rangle$: an initial state, and let $|\varphi'\rangle = \mathsf{CPhsO}^k|\varphi\rangle$. Then $L_2(|\varphi'\rangle, \widetilde{\mathsf{PATH}}_{s+1}) - L_2(|\varphi\rangle, \widetilde{\mathsf{PATH}}_s) \le \frac{4k\sqrt{(q+k)\delta\lambda}}{2^{\lambda/2}}$.

Basic proof idea: split the states into good and bad part. (in this talk, suppose that $x_1, \ldots, x_k \notin \mathcal{D}$ and all distinct for simplicity)

$$|\varphi\rangle = |(x_1, y_1), \ldots, (x_k, y_k), z\rangle \otimes |\mathcal{D}\rangle$$

$$\xmapsto{\mathsf{CPhsO}^k} |\varphi'\rangle = |(x_1, y_1), \ldots, (x_k, y_k), z\rangle \otimes \sum_{w_1, \ldots, w_k} (-1)^{\sum y_i w_i} |\mathcal{D} \cup \{(x_1, w_1), \ldots, (x_k, w_k)\}\rangle$$

$$|(x_1, y_1), \ldots, (x_k, y_k), z\rangle \otimes \sum_{\substack{w_1, \ldots, w_k \\ \in \mathsf{GOOD}}} (-1)^{\sum y_i w_i} |\mathcal{D} \cup \{(x_1, w_1), \ldots, (x_k, w_k)\}\rangle + |(x_1, y_1), \ldots, (x_k, y_k), z\rangle \otimes \sum_{\substack{w_1, \ldots, w_k \\ \in \mathsf{BAD}}} (-1)^{\sum y_i w_i} |\mathcal{D} \cup \{(x_1, w_1), \ldots, (x_k, w_k)\}\rangle$$

Proof Ideas: Hardness of Producing an $\mathcal{H}$-sequence in a Quantum Setting

Lemma

$|\varphi\rangle$: an initial state, and let $|\varphi'\rangle = \mathsf{CPhsO}^k|\varphi\rangle$. Then $L_2(|\varphi'\rangle, \widetilde{\mathsf{PATH}}_{s+1}) - L_2(|\varphi\rangle, \widetilde{\mathsf{PATH}}_s) \le \frac{4k\sqrt{(q+k)\delta\lambda}}{2^{\lambda/2}}$.

Basic proof idea: split the states into good and bad part. (in this talk, suppose that $x_1, \ldots, x_k \notin \mathcal{D}$ and all distinct for simplicity)

$$|\varphi\rangle = |(x_1, y_1), \ldots, (x_k, y_k), z\rangle \otimes |\mathcal{D}\rangle$$

$$\xmapsto{\mathsf{CPhsO}^k} |\varphi'\rangle = |(x_1, y_1), \ldots, (x_k, y_k), z\rangle \otimes \sum_{w_1, \ldots, w_k} (-1)^{\sum y_i w_i} |\mathcal{D} \cup \{(x_1, w_1), \ldots, (x_k, w_k)\}\rangle$$

$$|(x_1, y_1), \ldots, (x_k, y_k), z\rangle \otimes \sum_{\substack{w_1, \ldots, w_k \\ \in \mathsf{GOOD}}} (-1)^{\sum y_i w_i} |\mathcal{D} \cup \{(x_1, w_1), \ldots, (x_k, w_k)\}\rangle + |(x_1, y_1), \ldots, (x_k, y_k), z\rangle \otimes \sum_{\substack{w_1, \ldots, w_k \\ \in \mathsf{BAD}}} (-1)^{\sum y_i w_i} |\mathcal{D} \cup \{(x_1, w_1), \ldots, (x_k, w_k)\}\rangle$$

BAD: $\mathcal{D} \notin \mathsf{PATH}_s$ but $\mathcal{D} \cup \{(x_1, w_1), \ldots, (x_k, w_k)\} \in \mathsf{PATH}_{s+1}$

Proof Ideas: Hardness of Producing an $\mathcal{H}$-sequence in a Quantum Setting

Lemma

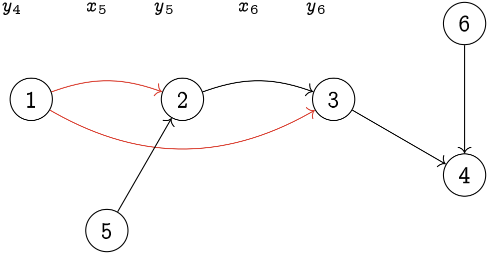
$|\varphi\rangle$: an initial state, and let $|\varphi'\rangle = \mathsf{CPhsO}^k|\varphi\rangle$. *Then* $L_2(|\varphi'\rangle, \widetilde{\mathsf{PATH}}_{s+1}) - L_2(|\varphi\rangle, \widetilde{\mathsf{PATH}}_s) \leq \frac{4k\sqrt{(q+k)\delta\lambda}}{2^{\lambda/2}}$.

Proof by example:

$$\mathcal{D} = \{(\overset{x_1}{10101}, \overset{y_1}{0001}), (\overset{x_2}{00011}, \overset{y_2}{0010}), (\overset{x_3}{00010}, \overset{y_3}{0110})$$

$$, (\underset{x_4}{01101}, \underset{y_4}{0000}), (\underset{x_5}{11110}, \underset{y_5}{0011}), (\underset{x_6}{01011}, \underset{y_6}{1101})\}$$

$G_{\mathcal{D}}$



Proof Ideas: Hardness of Producing an $\mathcal{H}$-sequence in a Quantum Setting

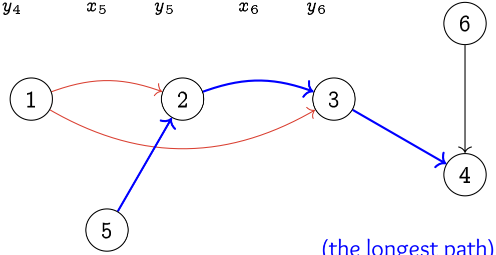
Lemma

$|\varphi\rangle$: *an initial state, and let* $|\varphi'\rangle = \mathsf{CPhsO}^k|\varphi\rangle$. *Then* $L_2(|\varphi'\rangle, \widetilde{\mathsf{PATH}}_{s+1}) - L_2(|\varphi\rangle, \widetilde{\mathsf{PATH}}_s) \le \frac{4k\sqrt{(q+k)\delta\lambda}}{2^{\lambda/2}}$.

Proof by example:

$$\mathcal{D} = \{(\overset{x_1}{10101}, \overset{y_1}{0001}), (\overset{x_2}{00011}, \overset{y_2}{0010}), (\overset{x_3}{00010}, \overset{y_3}{0110})$$
$$, (\underset{x_4}{01101}, \underset{y_4}{0000}), (\underset{x_5}{11110}, \underset{y_5}{0011}), (\underset{x_6}{01011}, \underset{y_6}{1101})\}$$

$G_{\mathcal{D}}$



(the longest path)=(5, 2, 3, 4)

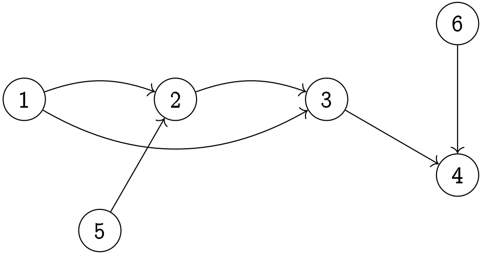
$\mathcal{D} \in \mathsf{PATH}_3$ but $\mathcal{D} \notin \mathsf{PATH}_4$

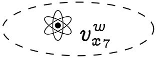
Proof Ideas: Hardness of Producing an $\mathcal{H}$-sequence in a Quantum Setting

Suppose we have one query: $x_7 = 00001$ (where $x_7 \notin \mathcal{D}$). Then the updated database is:

$$\mathcal{D}_1 = \{(10101, 0001), (00011, 0010), (00010, 0110), (01101, 0000), (11110, 0011), (01011, 1101)\} \cup \boxed{\{(00001, w)\}}$$

$G_{\mathcal{D}_1}$





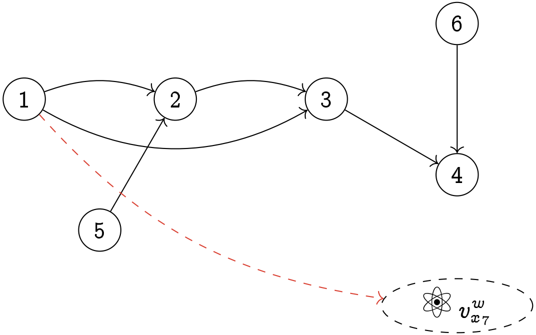
superposition over w's

Proof Ideas: Hardness of Producing an $\mathcal{H}$-sequence in a Quantum Setting

Suppose we have one query: $x_7 = 00001$ (where $x_7 \notin \mathcal{D}$). Then the updated database is:

$$\mathcal{D}_1 = \{(10101, 0001), (00011, 0010), (00010, 0110), (01101, 0000), (11110, 0011), (01011, 1101)\} \cup \{(00001, w)\}$$

$G_{\mathcal{D}_1}$



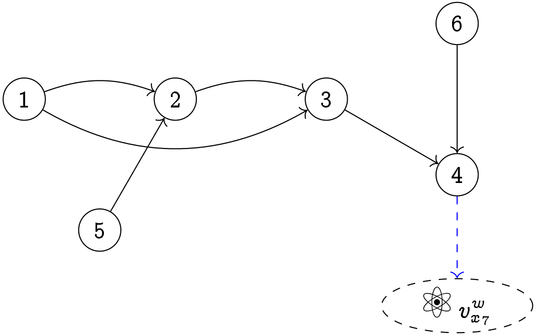
superposition over w's

Proof Ideas: Hardness of Producing an $\mathcal{H}$-sequence in a Quantum Setting

Suppose we have one query: $x_7 = 00001$ (where $x_7 \notin \mathcal{D}$). Then the updated database is:

$$\mathcal{D}_1 = \{(10101, 0001), (00011, 0010), (00010, 0110), (01101, 0000), (11110, 0011), (01011, 1101)\} \cup \{(00001, w)\}$$

$G_{\mathcal{D}_1}$



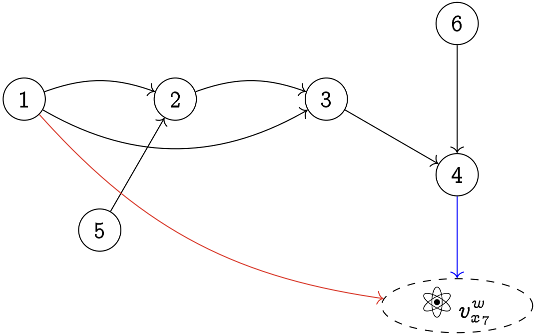
superposition over w's

Proof Ideas: Hardness of Producing an $\mathcal{H}$-sequence in a Quantum Setting

Suppose we have one query: $x_7 = 00001$ (where $x_7 \notin \mathcal{D}$). Then the updated database is:

$$\mathcal{D}_1 = \{(10101, 0001), (00011, 0010), (00010, 0110), (01101, 0000), (11110, 0011), (01011, 1101)\} \cup \{(00001, w)\}$$

$G_{\mathcal{D}_1}$



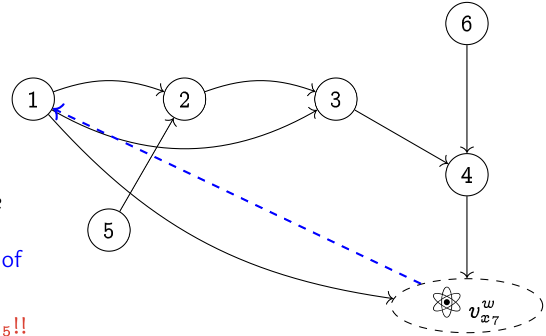
superposition over w's

Proof Ideas: Hardness of Producing an $\mathcal{H}$-sequence in a Quantum Setting

Suppose we have one query: $x_7 = 00001$ (where $x_7 \notin \mathcal{D}$). Then the updated database is:

$$\mathcal{D}_1 = \{(\boxed{10101}, 0001), (00011, 0010), (00010, 0110), (01101, 0000), (11110, 0011), (01011, 1101)\} \boxed{\cup \{(00001, w)\}}$$

$G_{\mathcal{D}_1}$



BAD if:

1. back edges from $v^w_{x_7}$ to some $i \in \{1, \dots, 6\}$.
   (e.g., $w = 1010 \Rightarrow$ substring of $x_1 = 10101$)

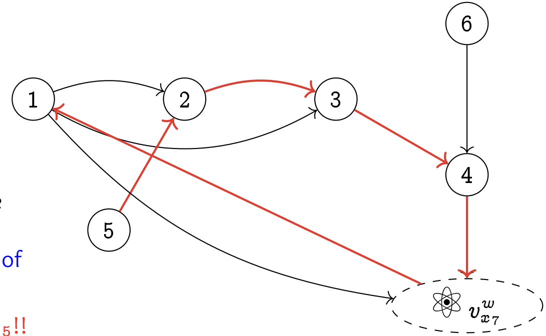
$\Rightarrow$ $\mathcal{D} \notin \mathrm{PATH}_4$ but $\mathcal{D}_1 \in \mathrm{PATH}_5$!!

Proof Ideas: Hardness of Producing an $\mathcal{H}$-sequence in a Quantum Setting

Suppose we have one query: $x_7 = 00001$ (where $x_7 \notin \mathcal{D}$). Then the updated database is:

$$\mathcal{D}_1 = \{(10101, 0001), (00011, 0010), (00010, 0110), (01101, 0000), (11110, 0011), (01011, 1101)\} \cup \{(00001, w)\}$$

$G_{\mathcal{D}_1}$



superposition over w's

BAD if:

1. back edges from $v_{x_7}^w$ to some $i \in \{1, \dots, 6\}$. (e.g., $w = 1010 \Rightarrow$ substring of $x_1 = 10101$)

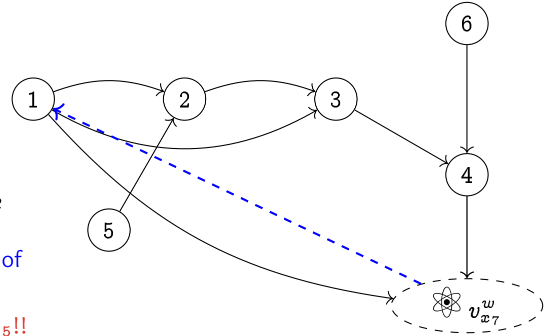
$\Rightarrow \mathcal{D} \notin \text{PATH}_4$ but $\mathcal{D}_1 \in \text{PATH}_5$!!

Proof Ideas: Hardness of Producing an $\mathcal{H}$-sequence in a Quantum Setting

Suppose we have one query: $x_7 = 00001$ (where $x_7 \notin \mathcal{D}$). Then the updated database is:

$$\mathcal{D}_1 = \{(\boxed{10101}, 0001), (00011, 0010), (00010, 0110), (01101, 0000), (11110, 0011), (01011, 1101)\} \boxed{\cup \{(00001, w)\}}$$

$G_{\mathcal{D}_1}$



superposition over w's

BAD if:

1. back edges from $v^w_{x_7}$ to some $i \in \{1, \ldots, 6\}$.
   (e.g., $w = 1010 \Rightarrow$ substring of $x_1 = 10101$)

$\Rightarrow$ $\mathcal{D} \notin \mathsf{PATH}_4$ but $\mathcal{D}_1 \in \mathsf{PATH}_5$!!

Key observation:

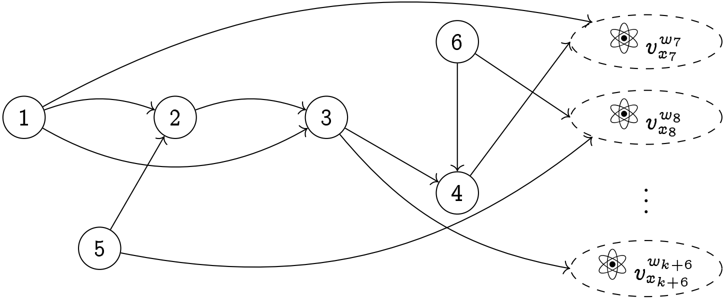
The fraction of such w's is negligible! ($\mathcal{O}(q\delta\lambda)$ out of 2^λ)

Proof Ideas: Hardness of Producing an $\mathcal{H}$-sequence in a Quantum Setting

For a parallel query: $x_7, \ldots, x_{k+6}$ (where $x_7, \ldots, x_{k+6} \notin \mathcal{D}$ and all x_i's are distinct for simplicity),

$$\mathcal{D}_k = \{(10101, 0001), (00011, 0010), (00010, 0110) \\ , (01101, 0000), (11110, 0011), (01011, 1101)\} \cup \boxed{\{(x_7, w_7), \ldots, (x_{k+6}, w_{k+6})\}}$$

$G_{\mathcal{D}_k}$

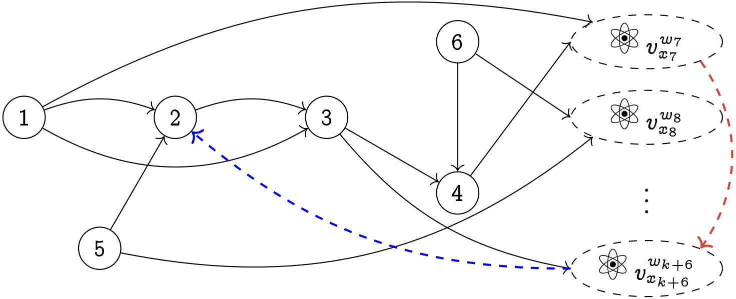


Proof Ideas: Hardness of Producing an $\mathcal{H}$-sequence in a Quantum Setting

For a parallel query: $x_7, \ldots, x_{k+6}$ (where $x_7, \ldots, x_{k+6} \notin \mathcal{D}$ and all x_i's are distinct for simplicity),

$$\mathcal{D}_k = \{(10101, 0001), (00011, 0010), (00010, 0110), (01101, 0000), (11110, 0011), (01011, 1101)\} \cup \boxed{\{(x_7, w_7), \ldots, (x_{k+6}, w_{k+6})\}}$$

$G_{\mathcal{D}_k}$



BAD if:

1. internal edges between $v_{x_i}^{w_i}$'s, and
2. back edges from $v_{x_i}^{w_i}$ to some $j \in \{1, \ldots, 6\}$.

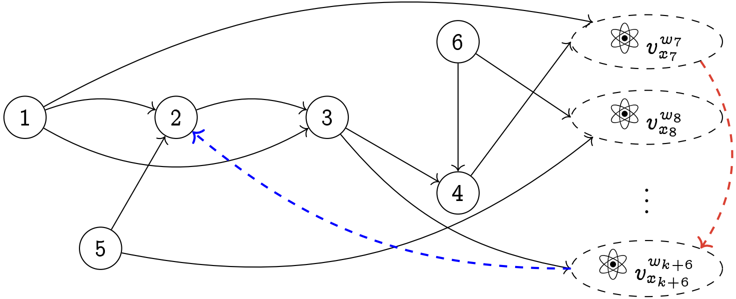
$\Rightarrow$ $\mathcal{D} \notin \text{PATH}_4$ but $\mathcal{D}_k \in \text{PATH}_5$!!

Proof Ideas: Hardness of Producing an $\mathcal{H}$-sequence in a Quantum Setting

For a parallel query: $x_7, \ldots, x_{k+6}$ (where $x_7, \ldots, x_{k+6} \notin \mathcal{D}$ and all x_i's are distinct for simplicity),

$$\mathcal{D}_k = \{(10101, 0001), (00011, 0010), (00010, 0110) \\ , (01101, 0000), (11110, 0011), (01011, 1101)\} \cup \{(x_7, w_7), \ldots, (x_{k+6}, w_{k+6})\}$$

$G_{\mathcal{D}_k}$



BAD if:

1. internal edges between $v_{x_i}^{w_i}$'s, and
2. back edges from $v_{x_i}^{w_i}$ to some $j \in \{1, \ldots, 6\}$.

$\Rightarrow$ $\mathcal{D} \notin \text{PATH}_4$ but $\mathcal{D}_k \in \text{PATH}_5$!!

Key observation: The fraction of such $w_7, \ldots, w_{k+6}$'s is negligibly small! ($(q + k)\delta\lambda$ out of 2^λ for each w_i)

Proof Ideas: Hardness of Producing an $\mathcal{H}$-sequence in a Quantum Setting

We have shown: k parallel queries in a single round,

$$L_2(|\varphi'\rangle, \widetilde{\mathsf{PATH}}_{s+1}) - L_2(|\varphi\rangle, \widetilde{\mathsf{PATH}}_s) \leq \frac{4k\sqrt{(q+k)\delta\lambda}}{2^{\lambda/2}}.$$

Proof Ideas: Hardness of Producing an $\mathcal{H}$-sequence in a Quantum Setting

We have shown: k parallel queries in a single round,

$$L_2(|\varphi'\rangle, \widetilde{\mathsf{PATH}}_{s+1}) - L_2(|\varphi\rangle, \widetilde{\mathsf{PATH}}_s) \le \frac{4k\sqrt{(q+k)\delta\lambda}}{2^{\lambda/2}}.$$

Throughout $N-1$ rounds: k_i parallel queries in each round, with $\sum_{i=1}^{N-1} k_i \le q$.

Proof Ideas: Hardness of Producing an $\mathcal{H}$-sequence in a Quantum Setting

We have shown: k parallel queries in a single round,

$$L_2(|\varphi'\rangle, \widetilde{\mathsf{PATH}}_{s+1}) - L_2(|\varphi\rangle, \widetilde{\mathsf{PATH}}_s) \leq \frac{4k\sqrt{(q+k)\delta\lambda}}{2^{\lambda/2}}.$$

Throughout $N-1$ rounds: k_i parallel queries in each round, with $\sum_{i=1}^{N-1} k_i \leq q$.

- By triangle inequality,

$$L_2(|\varphi_{N-1}\rangle, \widetilde{\mathsf{PATH}}_N) \leq \sum_{i=1}^{N-1} \frac{4k_i\sqrt{2q\delta\lambda}}{2^{\lambda/2}} = \mathcal{O}\left(\frac{\sqrt{q^3\delta\lambda}}{2^{\lambda/2}}\right).$$

$\Rightarrow$ $\mathcal{A}$ measures a database in PATH_N with probability at most $\mathcal{O}\left(\frac{q^3\delta\lambda}{2^\lambda}\right)$,

i.e., $\mathcal{A}$ can produce an $\mathcal{H}$-sequence of length N with only negligible probability $\mathcal{O}\left(\frac{q^3\delta\lambda}{2^\lambda}\right)$.

Proof Ideas: Hardness of Producing an $\mathcal{H}$-sequence in a Quantum Setting

We have shown: k parallel queries in a single round,

$$L_2(|\varphi'\rangle, \widetilde{\mathsf{PATH}}_{s+1}) - L_2(|\varphi\rangle, \widetilde{\mathsf{PATH}}_s) \leq \frac{4k\sqrt{(q+k)\delta\lambda}}{2^{\lambda/2}}.$$

Throughout $N-1$ rounds: k_i parallel queries in each round, with $\sum_{i=1}^{N-1} k_i \leq q$.

- By triangle inequality,

$$L_2(|\varphi_{N-1}\rangle, \widetilde{\mathsf{PATH}}_N) \leq \sum_{i=1}^{N-1} \frac{4k_i\sqrt{2q\delta\lambda}}{2^{\lambda/2}} = \mathcal{O}\left(\frac{\sqrt{q^3\delta\lambda}}{2^{\lambda/2}}\right).$$

$\Rightarrow$ $\mathcal{A}$ measures a database in PATH_N with probability at most $\mathcal{O}\left(\frac{q^3\delta\lambda}{2^\lambda}\right)$,

i.e., $\mathcal{A}$ can produce an $\mathcal{H}$-sequence of length N with only negligible probability $\mathcal{O}\left(\frac{q^3\delta\lambda}{2^\lambda}\right)$.

- We only considered the simplest case in this talk - see the paper for the full security proof!

Proof Ideas: Hardness of Producing an $\mathcal{H}$-sequence in a Quantum Setting

We have shown: k parallel queries in a single round,

$$L_2(|\varphi'\rangle, \widetilde{\mathsf{PATH}}_{s+1}) - L_2(|\varphi\rangle, \widetilde{\mathsf{PATH}}_s) \leq \frac{4k\sqrt{(q+k)\delta\lambda}}{2^{\lambda/2}}.$$

Throughout $N-1$ rounds: k_i parallel queries in each round, with $\sum_{i=1}^{N-1} k_i \leq q$.

- By triangle inequality,

$$L_2(|\varphi_{N-1}\rangle, \widetilde{\mathsf{PATH}}_N) \leq \sum_{i=1}^{N-1} \frac{4k_i\sqrt{2q\delta\lambda}}{2^{\lambda/2}} = \mathcal{O}\left(\frac{\sqrt{q^3\delta\lambda}}{2^{\lambda/2}}\right).$$

$\Rightarrow$ $\mathcal{A}$ measures a database in PATH_N with probability at most $\mathcal{O}\left(\frac{q^3\delta\lambda}{2^\lambda}\right)$,

i.e., $\mathcal{A}$ can produce an $\mathcal{H}$-sequence of length N with only negligible probability $\mathcal{O}\left(\frac{q^3\delta\lambda}{2^\lambda}\right)$.

- We only considered the simplest case in this talk - see the paper for the full security proof!

Security of a non-interactive PoSW: similar argument using the result above - details in the paper (`https://arxiv.org/pdf/2006.10972.pdf`)

Concluding Remarks

Takeaways.

- PoSW allows a prover to convince a resource-bounded verifier that the prover invested a substantial amount of sequential time to produce a valid proof.

Concluding Remarks

Takeaways.

- PoSW allows a prover to convince a resource-bounded verifier that the prover invested a substantial amount of sequential time to produce a valid proof.
- Any attacker in the pqROM making $q \ll 2^{\lambda/3}$ total queries in $N-1$ sequential rounds cannot find an $\mathcal{H}$-sequence of length N except with negligible probability $\mathcal{O}\left(\frac{q^3\delta\lambda}{2^\lambda}\right)$.

Concluding Remarks

Takeaways.

- PoSW allows a prover to convince a resource-bounded verifier that the prover invested a substantial amount of sequential time to produce a valid proof.
- Any attacker in the pqROM making $q \ll 2^{\lambda/3}$ total queries in $N-1$ sequential rounds cannot find an $\mathcal{H}$-sequence of length N except with negligible probability $\mathcal{O}\left(\frac{q^3\delta\lambda}{2^\lambda}\right)$.
- Any attacker in the pqROM making $q \ll 2^{\lambda/\log N}$ total queries in sequential time $T=(1-\alpha)N$ cannot produce a valid non-interactive PoSW ([CP18] construction) except with negligible probability $\mathcal{O}\left(q^2(1-\alpha)^{\frac{\lambda}{\log N}}+\frac{q^3\lambda\log N}{2^\lambda}\right)$.

Concluding Remarks

Takeaways.

- PoSW allows a prover to convince a resource-bounded verifier that the prover invested a substantial amount of sequential time to produce a valid proof.
- Any attacker in the pqROM making $q \ll 2^{\lambda/3}$ total queries in $N-1$ sequential rounds cannot find an $\mathcal{H}$-sequence of length N except with negligible probability $\mathcal{O}\left(\frac{q^3 \delta \lambda}{2^\lambda}\right)$.
- Any attacker in the pqROM making $q \ll 2^{\lambda / \log N}$ total queries in sequential time $T = (1-\alpha)N$ cannot produce a valid non-interactive PoSW ([CP18] construction) except with negligible probability $\mathcal{O}\left(q^2(1-\alpha)^{\frac{\lambda}{\log N}} + \frac{q^3 \lambda \log N}{2^\lambda}\right)$.

Open Questions.

- Can we tighten the security bound from q^3 to q^2?

Concluding Remarks

Takeaways.

- PoSW allows a prover to convince a resource-bounded verifier that the prover invested a substantial amount of sequential time to produce a valid proof.
- Any attacker in the pqROM making $q \ll 2^{\lambda/3}$ total queries in $N-1$ sequential rounds cannot find an $\mathcal{H}$-sequence of length N except with negligible probability $\mathcal{O}\left(\frac{q^3 \delta \lambda}{2^\lambda}\right)$.
- Any attacker in the pqROM making $q \ll 2^{\lambda/\log N}$ total queries in sequential time $T = (1-\alpha)N$ cannot produce a valid non-interactive PoSW ([CP18] construction) except with negligible probability $\mathcal{O}\left(q^2(1-\alpha)^{\frac{\lambda}{\log N}} + \frac{q^3 \lambda \log N}{2^\lambda}\right)$.

Open Questions.

- Can we tighten the security bound from q^3 to q^2?
- Establishing security for larger q: can we extract more than $\lambda/\log N$ challenges from a single RO output?

Concluding Remarks

Takeaways.

- PoSW allows a prover to convince a resource-bounded verifier that the prover invested a substantial amount of sequential time to produce a valid proof.
- Any attacker in the pqROM making $q \ll 2^{\lambda/3}$ total queries in $N-1$ sequential rounds cannot find an $\mathcal{H}$-sequence of length N except with negligible probability $\mathcal{O}\left(\frac{q^3\delta\lambda}{2^\lambda}\right)$.
- Any attacker in the pqROM making $q \ll 2^{\lambda/\log N}$ total queries in sequential time $T=(1-\alpha)N$ cannot produce a valid non-interactive PoSW ([CP18] construction) except with negligible probability $\mathcal{O}\left(q^2(1-\alpha)^{\frac{\lambda}{\log N}}+\frac{q^3\lambda\log N}{2^\lambda}\right)$.

Open Questions.

- Can we tighten the security bound from q^3 to q^2?
- Establishing security for larger q: can we extract more than $\lambda/\log N$ challenges from a single RO output?
- Can we prove for an interactive PoSW?

Concluding Remarks

Takeaways.

- PoSW allows a prover to convince a resource-bounded verifier that the prover invested a substantial amount of sequential time to produce a valid proof.
- Any attacker in the pqROM making $q \ll 2^{\lambda/3}$ total queries in $N-1$ sequential rounds cannot find an $\mathcal{H}$-sequence of length N except with negligible probability $\mathcal{O}\left(\frac{q^3 \delta \lambda}{2^\lambda}\right)$.
- Any attacker in the pqROM making $q \ll 2^{\lambda/\log N}$ total queries in sequential time $T = (1-\alpha)N$ cannot produce a valid non-interactive PoSW ([CP18] construction) except with negligible probability $\mathcal{O}\left(q^2(1-\alpha)^{\frac{\lambda}{\log N}} + \frac{q^3 \lambda \log N}{2^\lambda}\right)$.

Open Questions.

- Can we tighten the security bound from q^3 to q^2?
- Establishing security for larger q: can we extract more than $\lambda/\log N$ challenges from a single RO output?
- Can we prove for an interactive PoSW?
- Can we extend our security proof for other PoSW constructions?

Concluding Remarks

Takeaways.

- PoSW allows a prover to convince a resource-bounded verifier that the prover invested a substantial amount of sequential time to produce a valid proof.
- Any attacker in the pqROM making $q \ll 2^{\lambda/3}$ total queries in $N-1$ sequential rounds cannot find an $\mathcal{H}$-sequence of length N except with negligible probability $\mathcal{O}\left(\frac{q^3\delta\lambda}{2^\lambda}\right)$.
- Any attacker in the pqROM making $q \ll 2^{\lambda/\log N}$ total queries in sequential time $T=(1-\alpha)N$ cannot produce a valid non-interactive PoSW ([CP18] construction) except with negligible probability $\mathcal{O}\left(q^2(1-\alpha)^{\frac{\lambda}{\log N}} + \frac{q^3\lambda\log N}{2^\lambda}\right)$.

Open Questions.

- Can we tighten the security bound from q^3 to q^2?
- Establishing security for larger q: can we extract more than $\lambda/\log N$ challenges from a single RO output?
- Can we prove for an interactive PoSW?
- Can we extend our security proof for other PoSW constructions?
- Can techniques extend to other primitives, e.g., Proofs of Space, Memory-Hard Functions, etc.?

References I

Dan Boneh, Özgür Dagdelen, Marc Fischlin, Anja Lehmann, Christian Schaffner, and Mark Zhandry, *Random oracles in a quantum world*, ASIACRYPT 2011 (Dong Hoon Lee and Xiaoyun Wang, eds.), LNCS, vol. 7073, Springer, Heidelberg, December 2011, pp. 41–69.

Kai-Min Chung, Serge Fehr, Yu-Hsuan Huang, and Tai-Ning Liao, *On the compressed-oracle technique, and post-quantum security of proofs of sequential work*, 2021.

Bram Cohen and Krzysztof Pietrzak, *Simple proofs of sequential work*, EUROCRYPT 2018, Part II (Jesper Buus Nielsen and Vincent Rijmen, eds.), LNCS, vol. 10821, Springer, Heidelberg, April / May 2018, pp. 451–467.

Mohammad Mahmoody, Tal Moran, and Salil P. Vadhan, *Publicly verifiable proofs of sequential work*, ITCS 2013 (Robert D. Kleinberg, ed.), ACM, January 2013, pp. 373–388.

Mark Zhandry, *How to record quantum queries, and applications to quantum indifferentiability*, CRYPTO 2019, Part II (Alexandra Boldyreva and Daniele Micciancio, eds.), LNCS, vol. 11693, Springer, Heidelberg, August 2019, pp. 239–268.