Advanced Cryptography
CS 655

Week 15:

- Quantum Random Oracle Model
- Oblivious RAM

Homework 4 Released
Course Presentations: Next Week (Schedule Announced Soon)

Spring 2023

Recap

- Quantum Basics
- Grover Search
- Quantum Random Oracle Model
- Useful Results
  - $\psi = \sum_{x,y} \alpha_{x,y} |x, y\rangle$ and $\psi' = \sum_{x,y} \beta_{x,y} |x, y\rangle$ ➔ Measurement can distinguish two states with probability at most $4\sqrt{\sum_{x,y} \left|\alpha_{x,y} - \beta_{x,y}\right|^2}$
  - Upper bound Euclidean distance between final states ψ and ψ′ when we use oracles H and H’ in terms of ``query magnitude” on bad inputs x where $H(x) \neq H'(x)$.

State in QROM

- Typically we write

Output Register

$$|\psi\rangle = \sum_{x,y,z} \alpha_{x,y,z} |x, y, z\rangle$$

Auxilliary State (not associated with current RO query)

Input Register

$$\mathrm{RO}|\psi\rangle = \sum_{x,y,z} \alpha_{x,y,z} |x, RO(X) \oplus y, z\rangle$$

Query Magnitude

- Let $S \subset \{0,1\}^n$ be a set of inputs and let $\psi = \sum_{x,y} \alpha_{x,y} |x, y\rangle$ be a quantum state. Then

$$QM(\psi, S) \coloneqq \sum_{x \in S} \sum_{y,z} \alpha_{x,y,z}^2$$

- If $A^{H(.)}(w)$ generates states $\psi_{0,H,w}, \psi_{1,H,w}, \ldots, \psi_{T,H,w}$ we can write

$$QM(A, H, w, S) \coloneqq \sum_{i<T} QM\left(\psi_{i,H,w}, S\right)$$

Theorem: If $H(x) = H'(x)$ for all inputs $x \notin S$ then the Euclidean distance between the final states $\psi_{T,H,w}$ and $\psi_{T,H\prime,w}$ is at most $\mathrm{T} \cdot QM(A, H, w, S)$

Homework Hint

- **Intuition:** for a random/small sets S we expect $QM(A, H, w, S)$ to be small with high probabilty.
- If $S_1, \dots S_r \subset \{0,1\}^n$ are disjoint sets of inputs then

$$\sum_{i \le r} QM(\psi, S_i) = \sum_{i \le r} \sum_{x \in S} \sum_{y,z} \alpha_{x,y,z}^2 \le \sum_{x} \sum_{y,z} \alpha_{x,y,z}^2 = 1$$

- If $A^{H(.)}(w)$ generates states $\psi_{0,H,w}, \psi_{1,H,w}, \dots, \psi_{T,H,w}$ we can write

$$QM(A, H, w, S) := \sum_{i < T} QM\left(\psi_{i,H,w}, S\right)$$

$$\sum_{i \le r} QM(A, H, w, S_i) \le T$$

Quantum Computing: Useful Theorem

Let $\psi = \sum_{x,y} \alpha_{x,y} |x, y\rangle$ and $\psi' = \sum_{x,y} \beta_{x,y} |x, y\rangle$

Fix a quantum measurement and let $\mathfrak{D}$ and $\mathfrak{D}'$ be the distribution of outputs when we measure quantum states ψ and ψ' respectively.

Definition: Total Variation Distance

$$\mathrm{TVD}(\mathfrak{D}, \mathfrak{D}') \coloneqq \sum_{w} |\mathrm{Pr}_{\mathfrak{D}}[w] - \mathrm{Pr}_{\mathfrak{D}'}[w]|$$

Quantum Computing: Useful Theorem

Theorem: Let $\psi = \sum_{x,y} \alpha_{x,y}|x,y\rangle$ and $\psi' = \sum_{x,y} \beta_{x,y}|x,y\rangle$ be two quantum states. Fix *any* measurement and let $\mathfrak{D}$ and $\mathfrak{D}'$ be the distribution of outputs when we measure quantum states ψ and ψ′ respectively. We have

$$\mathrm{TVD}(\mathfrak{D}, \mathfrak{D}') \coloneqq \sum_{w} \left|\Pr_{\mathfrak{D}}[w] - \mathfrak{D}'_{[w]}[w]\right| \leq 4\sqrt{\sum_{x,y} \left|\alpha_{x,y} - \beta_{x,y}\right|^2}$$

Intuition: WHP we cannot distinguish between ``close” states ψ and ψ′ with any measurement

Phase Oracle vs Standard Oracle

- Typically we write

Output Register

$$|\psi\rangle = \sum_{x,y,z} \alpha_{x,y,z} |x, y, z\rangle$$

Auxilliary State (not associated with current RO query)

Input Register

$$\mathrm{StO}|\psi\rangle = \sum_{x,y,z} \alpha_{x,y,z} |x, RO(x) \oplus y, z\rangle$$

Phase Oracle

$$\mathrm{PhO}|\psi\rangle = \sum_{x,y,z} \alpha_{x,y,z} (-1)^{y \cdot RO(x)} |x, y, z\rangle$$

Equivalence: Phase/Standard Oracle

Lemma 3. *For any adversary A making queries to* StO*, let B be the adversary that is identical to A, except it performs the Hadamard transformation* $\mathsf{H}^{\otimes n}$ *to the response registers before and after each query. Then* $\Pr[A^{\mathsf{StO}}() = 1] = \Pr[B^{\mathsf{PhO}}() = 1]$

$$\mathrm{StO}|\psi\rangle = \sum_{x,y,z} \alpha_{x,y,z}|x, RO(x) \oplus y, z\rangle$$

$$\mathrm{PhO}|\psi\rangle = \sum_{x,y,z} \alpha_{x,y,z}(-1)^{y \cdot RO(x)}|x, y, z\rangle$$

Let $\mathrm{Had_{out}} = \mathrm{I}^{\otimes \mathrm{n}}\, \mathrm{H}^{\otimes \mathrm{n}}\, \mathrm{I}^{\otimes |z|}$ be a unitary transformation which applies the Hadamard transform to response registers (y) and identity transform elsewhere. Then

$$\mathrm{Had_{out}PhO(Had_{out}}|\psi\rangle) = \mathrm{StO}|\psi\rangle = \sum_{x,y,z} \alpha_{x,y,z}|x, RO(x) \oplus y, z\rangle$$

Views of the Quantum Random Oracle

We can view a function $\boldsymbol{H}: \{\mathbf{0},\mathbf{1}\}^{2\lambda} \to \{\mathbf{0},\mathbf{1}\}^{\lambda}$ as a string of length $\boldsymbol{\lambda 2^{2\lambda}}$

H(x) simply returns the $\boldsymbol{\lambda}$ bit string at locations $\mathbf{x}\boldsymbol{\lambda}, \dots, (\mathbf{x}+\mathbf{1})\boldsymbol{\lambda}-\mathbf{1}$

Now we can view the state as

$$|\psi\rangle \otimes |H\rangle = \sum_{x,y,z} \alpha_{x,y,z} |x,y,z\rangle \otimes |H\rangle$$

Standard oracle performs map

$$|x,y,z\rangle \otimes |H\rangle \to |x, y \oplus H(x), z\rangle \otimes |H\rangle$$

Algorithm can only apply unitary transforms to first part of state $|x,y,z\rangle$

Views of the Quantum Random Oracle

We can view a function $\boldsymbol{H}: \{\mathbf{0},\mathbf{1}\}^{2\lambda} \rightarrow \{\mathbf{0},\mathbf{1}\}^{\lambda}$ as a string of length $\boldsymbol{\lambda 2^{2\lambda}}$

World 1 (StO’): Pick random function H and run algorithm with initial state $|\psi_0\rangle \otimes |H\rangle$

World 2 (StO): run algorithm with initial state (uniform superposition of all oracles)

$$|\psi_0\rangle \bigotimes \left(\frac{1}{\sqrt{2^{\lambda 2^{2\lambda}}}} \sum_H |H\rangle \right)$$

Views of the Quantum Random Oracle

World 1 (StO'): Pick random function H and run algorithm with initial state $|\psi_0\rangle \otimes |H\rangle$

World 2 (StO): run algorithm with initial state (uniform superposition of all oracles)

$$|\psi_0\rangle \bigotimes \left(\frac{1}{\sqrt{2^{\lambda 2^{2\lambda}}}} \sum_H |H\rangle \right)$$

Lemma 2. StO *and* StO$'$ *are perfectly indistinguishable. That is, for any adversary* A *making oracle queries, let* $A^{\mathsf{StO}}()$ *and* $A^{\mathsf{StO}'}()$ *denote the algorithm interfacing with* StO *and* StO$'$*, respectively. Then* $\Pr[A^{\mathsf{StO}}() = 1] = \Pr[A^{\mathsf{StO}'}() = 1]$

Another View

World 2’: $|H\rangle = \{(x, \perp): x \in \{0,1\}^{2\lambda}\}$ where $\perp$ indicates that $H(x)$ is not yet assigned

Run algorithm with initial state $|\psi_0\rangle \otimes |H\rangle$

Oracle Map (Intuitions): if $H(x) = \perp$

$$|x, y, z\rangle \otimes |H\rangle \to 2^{\lambda/2} \sum_w |x, y \oplus w, z\rangle \otimes |H \cup (x, w)\rangle$$

Where $H \cup (x, w)$ replaces $(x, \perp)$ with (x, w)

Idea: measuring red state yields a list of query/output pairs!

Not Quite that Simple…

World 2': $|H\rangle = \{(x, \perp): x \in \{0,1\}^{2\lambda}\}$ where $\perp$ indicates that $H(x)$ is not yet assigned

Run algorithm with initial state $|\psi_0\rangle \otimes |H\rangle$

Oracle Map (Intuitions): if $H(x) = \perp$

$$|x, y, z\rangle \otimes |H\rangle \rightarrow 2^{\lambda/2} \sum_w |x, y \oplus w, z\rangle \otimes |H \cup (x, w)\rangle$$

Where $H \cup (x, w)$ replaces $(x, \perp)$ with (x, w)

Idea: measuring red state yields a list of query/output pairs!

Question: What if $H(x) \neq \perp$? How do we make sure oracle is unitary transformation?

Towards Unitary Transform

World 2’: $|H\rangle = \left\{(x, \perp): x \in \{0,1\}^{2\lambda}\right\}$ where $\perp$ indicates that $H(x)$ is not yet assigned

Run algorithm with initial state $|\psi_0\rangle \otimes |H\rangle$

Build oracle out of unitary transforms: $\mathrm{StdDecomp}$ and StO'

$$\mathrm{StdOracle} \coloneqq \mathrm{StdDecomp} \circ \mathrm{StO}' \circ \mathrm{StdDecomp}$$
$$\mathrm{PhsOracle} \coloneqq \mathrm{StdDecomp} \circ \mathrm{PhsO}' \circ \mathrm{StdDecomp}$$

Where

$$\mathrm{StO}'|x, y, z\rangle \otimes |H\rangle \to |x, y \oplus H(x), z\rangle \otimes |H\rangle$$
$$\mathrm{PhsO}'|x, y, z\rangle \otimes |H\rangle \to (-1)^{y \cdot H(x)}|x, y, z\rangle \otimes |H\rangle$$

Note: Define $y \oplus \perp \coloneqq y$ and $y \cdot \perp \coloneqq 0$ for completeness

Towards Unitary Transform

Build oracle out of unitary transforms: $\mathrm{StdDecomp}$ and StO'

$$\mathrm{StdDecomp}|x,y,z\rangle \otimes |H\rangle \to |x,y,z\rangle \otimes \mathrm{StdDecomp}_x|H\rangle$$

$$\mathrm{StdDecomp}_x|H\rangle = 2^{-\frac{\lambda}{2}} \sum_w |H \cup (x,w)\rangle \quad \text{if } H(x) = \perp$$

Reversibility: If $H'(x) = \perp$ then

$$\mathrm{StdDecomp}_x\left(2^{-\frac{\lambda}{2}} \sum_w |H' \cup (x,w)\rangle\right) = |H'\rangle$$

Towards Unitary Transform

Build oracle out of unitary transforms: $\mathrm{StdDecomp}$ and StO'

$$\mathrm{StdDecomp}|x,y,z\rangle \otimes |H\rangle \rightarrow |x,y,z\rangle \otimes \mathrm{StdDecomp}_x|H\rangle$$

$$\mathrm{StdDecomp}_x|H\rangle = 2^{-\frac{\lambda}{2}} \sum_w |H \cup (x,w)\rangle \quad \text{if } H(x) = \perp$$

State is untouched

If $H'(x) = \perp$ and $z \neq 0$ then

$$\mathrm{StdDecomp}_x\left(2^{-\frac{\lambda}{2}} \sum_w (-1)^{z \cdot w} |H' \cup (x,w)\rangle\right) = 2^{-\frac{\lambda}{2}} \sum_w (-1)^{z \cdot w} |H' \cup (x,w)\rangle$$

Reversibility:

$$\mathrm{StdDecomp}_x\left(2^{-\frac{\lambda}{2}} \sum_w |H' \cup (x,w)\rangle\right) = |H'\rangle$$

All pairs of the form (x,w) are removed

Compressed Oracles

World 2': $|H\rangle = \left\{(x,\perp): x \in \{0,1\}^{\lambda}\right\}$ where $\perp$ indicates that $H(x)$ is not yet assigned

Run algorithm with initial state $|\psi_0\rangle \otimes |H\rangle$

Build oracle out of unitary transforms: $\mathrm{StdDecomp}$ and $\mathrm{StO'}$

$$\mathrm{StdOracle} \coloneqq \mathrm{StdDecomp} \circ \mathrm{StO'} \circ \mathrm{StdDecomp}$$
$$\mathrm{PhsOracle} \coloneqq \mathrm{StdDecomp} \circ \mathrm{PhsO'} \circ \mathrm{StdDecomp}$$

Compressed Versions: CPhsO and CPhsO

Idea: If we know there are only T queries then $|H\rangle$ will never have more than T entries that are not $\perp$ ➔ can compress representation of $|H\rangle$

Compressed Oracles

Compressed Versions: CPhsO and CPhsO

Idea: If we know there are only T queries then $|H\rangle$ will never have more than T entries that are not ⊥ ➔ can compress representation of $|H\rangle$

Lemma 4. CStO *and* StO *are perfectly indistinguishable.* CPhsO *and* PhO *are perfectly indistinguishable. That is, for any adversary* A*, we have* $\Pr[A^{\mathsf{CStO}}() = 1] = \Pr[A^{\mathsf{StO}}() = 1]$*, and for any adversary* B*, we have* $\Pr[B^{\mathsf{CPhsO}}() = 1] = \Pr[A^{\mathsf{PhO}}() = 1]$.

A Helpful Lemma

Lemma 5. *Consider a quantum algorithm A making queries to a random oracle H and outputting tuples $(x_1, \ldots, x_k, y_1, \ldots, y_k, z)$. Let R be a collection of such tuples. Suppose with probability p, A outputs a tuple such that (1) the tuple is in R and (2) $H(x_i) = y_i$ for all i. Now consider running A with the oracle* CStO, *and suppose the database D is measured after A produces its output. Let p' be the probability that (1) the tuple is in R, and (2) $D(x_i) = y_i$ for all i (and in particular $D(x_i) \neq \bot$). Then $\sqrt{p} \leq \sqrt{p'} + \sqrt{k/2^n}$*

Example:

$$R_{collision} = \left\{(x_1, y), (x_2, y) : x_1, x_2 \in \{0,1\}^{2\lambda} \wedge y \in \{0,1\}^{\lambda}\right\}$$

p denotes probability that A outputs a hash collision (regular QROM).

p' denotes probability that we measure a database with some colliding pair (when using compressed oracle CStO)

A Helpful Lemma

Lemma 5. *Consider a quantum algorithm A making queries to a random oracle H and outputting tuples $(x_1, \ldots, x_k, y_1, \ldots, y_k, z)$. Let R be a collection of such tuples. Suppose with probability p, A outputs a tuple such that (1) the tuple is in R and (2) $H(x_i) = y_i$ for all i. Now consider running A with the oracle* CStO, *and suppose the database D is measured after A produces its output. Let p' be the probability that (1) the tuple is in R, and (2) $D(x_i) = y_i$ for all i (and in particular $D(x_i) \neq \bot$). Then $\sqrt{p} \leq \sqrt{p'} + \sqrt{k/2^n}$*

Example:

$$R_{collision} = \left\{(x_1, y), (x_2, y) : x_1, x_2 \in \{0,1\}^{2\lambda} \wedge y \in \{0,1\}^{\lambda}\right\}$$

p denotes probability that A outputs a hash collision (regular QROM).

p' denotes probability that we measure a database with some colliding pair

Typically, much easier to upper bound p' ➔ upper bound for p (quantity we want to upper bound)

Grover Search Revisited

Theorem 1. *For any adversary making q queries to* CStO *or* CPhsO *and an arbitrary number of database read queries, if the database D is measured after the q queries, the probability it contains a pair of the form $(x, 0^n)$ is at most $O(q^2/2^n)$.*

Corollary 1. *After making q quantum queries to a random oracle, the probability of finding a pre-image of 0^n is at most $O(q^2/2^n)$.*

Proof: Define $R_0 = \left\{(x,0): x \in \{0,1\}^{2\lambda}\right\}$ ➔ Let p denote probability outputs $(\mathrm{x}, \mathrm{y}=0) \in R_0$ with $\mathrm{H}(\mathrm{x}) = \mathrm{y} = 0$ i.e., found a pre-image

Theorem 1 shows that $p' = O\left(\frac{q^2}{2^\lambda}\right)$. Now applying Lemma 5 we have

$$\sqrt{p} \leq O\left(\sqrt{\frac{q^2}{2^\lambda}}\right) + 2^{-\frac{\lambda}{2}} = O\left(\frac{q}{\sqrt{2^\lambda}}\right) \Rightarrow p = O\left(\frac{q^2}{2^\lambda}\right)$$

Hash Collisions

Theorem 2. *For any adversary making q queries to* CStO *or* CPhsO *and an arbitrary number of database read queries, if the database D is measured after the q queries, the resulting database will contain a collision with probability at most $O(q^3/2^n)$*

Corollary 2. *After making q quantum queries to a random oracle, the probability of finding a collision is at most $O(q^3/2^n)$.*

Proof: $R_{collision} = \left\{(x_1, y), (x_2, y): x_1, x_2 \in \{0,1\}^{2\lambda} \wedge y \in \{0,1\}^{\lambda}\right\}$ ➔ Let p denote probability attacker outputs$(x_1, y), (x_2, y) \in R_{collision}$ with $\mathrm{H}(\mathrm{x}_1) = \mathrm{H}(\mathrm{x}_2) = \mathrm{y}$ i.e., found a collision

Theorem 1 shows that $p' = O\left(\frac{q^3}{2^\lambda}\right)$. Now applying Lemma 5 we have

$$\sqrt{p} \leq O\left(\sqrt{\frac{q^3}{2^\lambda}}\right) + O\left(\frac{1}{\sqrt{2^\lambda}}\right) = O\left(\frac{q\sqrt{q}}{\sqrt{2^\lambda}}\right) \Rightarrow p = O\left(\frac{q^3}{2^\lambda}\right)$$

Proof of Theorem 2

Theorem 2. *For any adversary making q queries to* CStO *or* CPhsO *and an arbitrary number of database read queries, if the database D is measured after the q queries, the resulting database will contain a collision with probability at most $O(q^3/2^n)$*

Proof: We first define projections P, Q, R, S such that $P + Q + R + S = I$

P projects onto random oracles that contain a collision

Consider the state

$$|\psi\rangle = \sum_{x,y,z,H} \alpha_{x,y,z,H} |x,y,z\rangle \otimes |H\rangle \Rightarrow P|\psi\rangle := \sum_{H \in Collision} \sum_{x,y,z} \alpha_{x,y,z,H} |x,y,z\rangle \otimes |H\rangle$$

Where

$$\text{Collision} = \{H : \exists x_1, x_2 \; s.t.\, H(x_1) = H(x_2) \neq \perp\}$$

Proof of Theorem 2

Theorem 2. *For any adversary making q queries to* CStO *or* CPhsO *and an arbitrary number of database read queries, if the database D is measured after the q queries, the resulting database will contain a collision with probability at most $O(q^3/2^n)$*

Proof: We first define projections P, Q, R, S such that $P + Q + R + S = I$

Q projects onto states that do contain a collision such that $y \neq 0$ and $H(x) = \perp$

Consider the state

$$|\psi\rangle = \sum_{x,y,z,H} \alpha_{x,y,z,H} |x,y,z\rangle \otimes |H\rangle \Rightarrow Q|\psi\rangle := \sum_{x,y\neq 0,z} \sum_{\substack{H \notin Collision, \\ H(x)=\perp}} \alpha_{x,y,z,H} |x,y,z\rangle \otimes |H\rangle$$

Where

$$\text{Collision} = \{H: \exists x_1, x_2 \; s.t. H(x_1) = H(x_2) \neq \perp\}$$

Proof of Theorem 2

Theorem 2. *For any adversary making q queries to* CStO *or* CPhsO *and an arbitrary number of database read queries, if the database D is measured after the q queries, the resulting database will contain a collision with probability at most $O(q^3/2^n)$*

Proof: We first define projections P, Q, R, S such that $P + Q + R + S = I$

R projects onto states that do contain a collision such that $y \neq 0$ and $H(x) \neq \perp$

Consider the state

$$|\psi\rangle = \sum_{x,y,z,H} \alpha_{x,y,z,H} |x, y, z\rangle \otimes |H\rangle \Rightarrow R|\psi\rangle \coloneqq \sum_{x,y\neq 0,z} \sum_{\substack{H \notin Collision, \\ H(x)\neq\perp}} \alpha_{x,y,z,H} |x, y, z\rangle \otimes |H\rangle$$

Where

$$\text{Collision} = \{H: \exists x_1, x_2 \; s.t. H(x_1) = H(x_2) \neq \perp\}$$

Proof of Theorem 2

Theorem 2. *For any adversary making q queries to* CStO *or* CPhsO *and an arbitrary number of database read queries, if the database D is measured after the q queries, the resulting database will contain a collision with probability at most $O(q^3/2^n)$*

Proof: We first define projections P, Q, R, S such that $P + Q + R + S = I$

S projects onto states that do contain a collision such that $y = 0$

Consider the state

$$|\psi\rangle = \sum_{x,y,z,H} \alpha_{x,y,z,H} |x, y, z\rangle \otimes |H\rangle \Rightarrow R|\psi\rangle := \sum_{x,y=0,z} \sum_{H \notin Collision,} \alpha_{x,y,z,H} |x, y, z\rangle \otimes |H\rangle$$

Where

$$\text{Collision} = \{H : \exists x_1, x_2 \ s.t. H(x_1) = H(x_2) \neq \perp\}$$

Proof of Theorem 2

Theorem 2. *For any adversary making* q *queries to* CStO *or* CPhsO *and an arbitrary number of database read queries, if the database* D *is measured after the* q *queries, the resulting database will contain a collision with probability at most* $O(q^3/2^n)$

Proof: We first define projections P, Q, R, S such that $P + Q + R + S = I$

S projects onto states that do contain a collision such that $y = 0$

We want to bound $\|\mathrm{P} \circ \mathrm{CPhsO} \circ |\psi\rangle\|$ (Euclidean norm of "bad/collision state") after each random oracle query.

$$\mathrm{CPhsO}|\psi\rangle = \mathrm{CPhsO}(P|\psi\rangle + Q|\psi\rangle + R|\psi\rangle + S|\psi\rangle)$$

Proof of Theorem 2

Theorem 2. *For any adversary making q queries to* CStO *or* CPhsO *and an arbitrary number of database read queries, if the database D is measured after the q queries, the resulting database will contain a collision with probability at most $O(q^3/2^n)$*

Proof: We first define projections P, Q, R, S such that $P + Q + R + S = I$

S projects onto states that do contain a collision such that $y = 0$

$$\begin{aligned} \|\mathrm{P} \circ \mathrm{CPhsO} \circ |\psi\rangle\| \leq & \|\mathrm{P} \circ \mathrm{CPhsO} \circ P \circ |\psi\rangle\| \\ & + \|\mathrm{P} \circ \mathrm{CPhsO} \circ Q \circ |\psi\rangle\| \\ & + \|\mathrm{R} \circ \mathrm{CPhsO} \circ R \circ |\psi\rangle\| \\ & + \|\mathrm{P} \circ \mathrm{CPhsO} \circ S \circ |\psi\rangle\| \end{aligned}$$

Proof of Theorem 2

Theorem 2. *For any adversary making q queries to* CStO *or* CPhsO *and an arbitrary number of database read queries, if the database D is measured after the q queries, the resulting database will contain a collision with probability at most $O(q^3/2^n)$*

Proof:

$$\begin{aligned}\|\mathrm{P}\circ\mathrm{CPhsO}\circ|\psi\rangle\| \le\ & \|\mathrm{P}\circ\mathrm{CPhsO}\circ P\circ|\psi\rangle\| \\ & +\|\mathrm{P}\circ\mathrm{CPhsO}\circ Q\circ|\psi\rangle\| \\ & +\|\mathrm{R}\circ\mathrm{CPhsO}\circ R\circ|\psi\rangle\| \\ & +\|\mathrm{P}\circ\mathrm{CPhsO}\circ S\circ|\psi\rangle\|\end{aligned}$$

Old Projection before RO query

Fact 1: $\|\mathrm{P}\circ\mathrm{CPhsO}\circ P\circ|\psi\rangle\| \le \|\mathrm{CPhsO}\circ P\circ|\psi\rangle\| = \|P\circ|\psi\rangle\|$

Projection can only decrease norm

CPhsO is unitary ➔ preserves norms

Proof of Theorem 2

Theorem 2. *For any adversary making q queries to* CStO *or* CPhsO *and an arbitrary number of database read queries, if the database D is measured after the q queries, the resulting database will contain a collision with probability at most $O(q^3/2^n)$*

Proof:

$$\begin{aligned} \|\mathrm{P} \circ \mathrm{CPhsO} \circ |\psi\rangle\| \le \|P \circ |\psi\rangle\| &+ \|\mathrm{P} \circ \mathrm{CPhsO} \circ Q \circ |\psi\rangle\| \\ &+ \|\mathrm{P} \circ \mathrm{CPhsO} \circ R \circ |\psi\rangle\| \\ &+ \|\mathrm{P} \circ \mathrm{CPhsO} \circ S \circ |\psi\rangle\| \end{aligned}$$

Fact 2: $\|\mathrm{P} \circ (\mathrm{CPhsO} \circ S \circ |\psi\rangle)\| \le \|\mathrm{P} \circ S \circ |\psi\rangle\| = 0$

CPhsO does not modify states in projection $S \circ |\psi\rangle$!

➔ $\mathrm{CPhsO} \circ S \circ |\psi\rangle = S \circ |\psi\rangle$

States in projection S do no have collision!

Proof of Theorem 2

Theorem 2. *For any adversary making q queries to* CStO *or* CPhsO *and an arbitrary number of database read queries, if the database D is measured after the q queries, the resulting database will contain a collision with probability at most $O(q^3/2^n)$*

Proof:

$$\begin{aligned}\|\mathrm{P}\circ \mathrm{CPhsO}\circ|\psi\rangle\| \leq \|P\circ|\psi\rangle\| + \|\mathrm{P}\circ\mathrm{CPhsO}\circ Q\circ|\psi\rangle\| \\ +\|\mathrm{P}\circ\mathrm{CPhsO}\circ R\circ|\psi\rangle\| + 0\end{aligned}$$

Fact 3: $\|\mathrm{P}\circ(\mathrm{CPhsO}\circ R\circ|\psi\rangle)\| \leq \sqrt{\frac{q}{2^\lambda}}\|R\circ|\psi\rangle\|$

Proof: Skipped (similar to Fact 4)

Proof of Theorem 2

Theorem 2. *For any adversary making q queries to* CStO *or* CPhsO *and an arbitrary number of database read queries, if the database D is measured after the q queries, the resulting database will contain a collision with probability at most $O(q^3/2^n)$*

Proof:

$$\|\mathrm{P} \circ \mathrm{CPhsO} \circ |\psi\rangle\| \leq \|P \circ |\psi\rangle\| + \sqrt{\frac{q}{2^\lambda}}\|R \circ |\psi\rangle\| + \|\mathrm{P} \circ \mathrm{CPhsO} \circ Q \circ |\psi\rangle\|$$

Fact 4: $\|\mathrm{P} \circ (\mathrm{CPhsO} \circ Q \circ |\psi\rangle)\| \leq \sqrt{\frac{q}{2^\lambda}}\|Q \circ |\psi\rangle\|$

Proof of Theorem 2

Theorem 2. *For any adversary making q queries to* CStO *or* CPhsO *and an arbitrary number of database read queries, if the database D is measured after the q queries, the resulting database will contain a collision with probability at most $O(q^3/2^n)$*

Proof:

$$\|\mathrm{P} \circ \mathrm{CPhsO} \circ |\psi\rangle\| \leq \|P \circ |\psi\rangle\| + \sqrt{\frac{q}{2^\lambda}}\|R \circ |\psi\rangle\| + \sqrt{\frac{q}{2^\lambda}}\|Q \circ |\psi\rangle\|$$

$$\leq \|P \circ |\psi\rangle\| + \sqrt{\frac{q}{2^\lambda}}$$

➔ Norm of $\|P \circ |\psi\rangle\|$ increase by at most $\sqrt{\frac{q}{2^\lambda}}$ after each query

➔Norm on bad states after q queries is at most q$\sqrt{\frac{q}{2^\lambda}} = \sqrt{\frac{q^3}{2^\lambda}}$

➔ measuring final state yields bad database (containing collision) with probability at most $\frac{q^3}{2^\lambda}$

Proof of Theorem 2

Theorem 2. *For any adversary making q queries to* CStO *or* CPhsO *and an arbitrary number of database read queries, if the database D is measured after the q queries, the resulting database will contain a collision with probability at most $O(q^3/2^n)$*

Fact 4: $\|\mathrm{P} \circ (\mathrm{CPhsO} \circ Q \circ |\psi\rangle)\| \leq \sqrt{\frac{q}{2^\lambda}} \|Q \circ |\psi\rangle\|$

Recall the projection Q

$$|\psi\rangle = \sum_{x,y,z,H} \alpha_{x,y,z,H} |x,y,z\rangle \otimes |H\rangle \Rightarrow Q|\psi\rangle := \sum_{x,y\neq 0,z} \sum_{\substack{H \notin Collision, \\ H(x)=\perp}} \alpha_{x,y,z,H} |x,y,z\rangle \otimes |H\rangle$$

Thus,

$$\mathrm{CPhsO} \circ Q \circ |\psi\rangle = \sum_{x,y\neq 0,z} \sum_{\substack{H \notin Collision, \\ H(x)=\perp}} \alpha_{x,y,z,H} |x,y,z\rangle \otimes \sum_{w} 2^{-\lambda/2} |H \cup (x,w)\rangle$$

Proof of Theorem 2

Theorem 2. *For any adversary making q queries to* CStO *or* CPhsO *and an arbitrary number of database read queries, if the database D is measured after the q queries, the resulting database will contain a collision with probability at most $O(q^3/2^n)$*

Fact 4: $\|\mathrm{P} \circ (\mathrm{CPhsO} \circ Q \circ |\psi\rangle)\| \leq \sqrt{\frac{q}{2^\lambda}} \|Q \circ |\psi\rangle\|$

Recall the projection Q

$$|\psi\rangle = \sum_{x,y,z,H} \alpha_{x,y,z,H} |x,y,z\rangle \otimes |H\rangle \Rightarrow Q|\psi\rangle := \sum_{x,y\neq 0,z} \sum_{\substack{H \notin Collision, \\ H(x)=\perp}} \alpha_{x,y,z,H} |x,y,z\rangle \otimes |H\rangle$$

Thus,

$$\mathrm{P} \circ \mathrm{CPhsO} \circ Q \circ |\psi\rangle = \sum_{x,y\neq 0,z} \sum_{\substack{H \notin Collision, \\ H(x)=\perp}} \alpha_{x,y,z,H} |x,y,z\rangle \otimes \sum_{w \in Bad(H)} 2^{-\lambda/2} |H \cup (x,w)\rangle$$

Where $Bad(H)$ is the set of ouputs already recorded in H. ➔ $|Bad(H)| \leq q$

Proof of Theorem 2

Theorem 2. *For any adversary making q queries to* CStO *or* CPhsO *and an arbitrary number of database read queries, if the database D is measured after the q queries, the resulting database will contain a collision with probability at most $O(q^3/2^n)$*

Fact 4: $\|\mathrm{P} \circ (\mathrm{CPhsO} \circ Q \circ |\psi\rangle)\| \leq \sqrt{\frac{q}{2^\lambda}} \|Q \circ |\psi\rangle\|$

Thus,

$$\mathrm{P} \circ \mathrm{CPhsO} \circ Q \circ |\psi\rangle = \sum_{x,y\neq 0,z} \sum_{\substack{H\notin Collision,\\ H(x)=\perp}} \alpha_{x,y,z,H} |x,y,z\rangle \otimes \sum_{w\in Bad(H)} 2^{-\lambda/2} |H \cup (x,w)\rangle$$

Where $Bad(H)$ is the set of outputs already recorded in H. ➔ $|Bad(H)| \leq |H| \leq q$

$$\|\mathrm{P} \circ (\mathrm{CPhsO} \circ Q \circ |\psi\rangle)\|^2 = \sum_{x,y\neq 0,z} \sum_{\substack{H\notin Collision,\\ H(x)=\perp}} \sum_{w\in Bad(H)} 2^{-\lambda} \alpha_{x,y,z,H}^2 \leq q 2^{-\lambda} \sum_{x,y\neq 0,z} \sum_{\substack{H\notin Collision,\\ H(x)=\perp}} \alpha_{x,y,z,H}^2$$

$$= q 2^{-\lambda} \|Q \circ |\psi\rangle\|^2$$

Course Evaluation

- I would appreciate if you take the time to fill out your course evaluation.
- What did you like about the course? What could be improved? Let me know!
- Your feedback is anonymous and will not impact grades.

Final Project Presentation (Next Week)

Tuesday, April 25th	Thursday, April 27th
Albert Yu (3:00-3:18 PM)	Blake and Xiuyu (3:00-3:28 PM)
Nicolas Harrell (3:18-3:36 PM)	Adithya and Jacob (3:28-3:56 PM)
Hongoa Wang (3:36-3:54 PM)	Jimmy Hwang (3:56-4:14 PM)
Zhongtang Luo (3:54-4:12 PM)	

Individuals: 14 minute presentation + 3 minute Q&A + 1 minute transition

Groups: 24 minute presentation + 3 minute Q&A + 1 minute transition
- It is expected that both team members will give part of the presentation
- You may choose how to divide the presentation

E-mail slides to Hassan at least <u>one hour before</u> class on the day of your presentation (CC me)

Format: PDF/PPT

Final Exam and Project Report

- Final Exam (Take Home):
  - Released Thursday, April 27^{th} at 5PM
  - Due: Friday, April 28^{th} at 5PM on Gradescope
  - Should take ≈ 2 hours

- Project Report
  - 8-12 pages
    - Introduce/Motivate the Problem, Define the Problem Clearly, Summarize Related Work et…
    - **Results**
      - **Note:** This can include failed approaches if you clearly describe what you tried and explain why this approach did not work.
    - **Future** Work: If you were to continue working on this problem what would you do?
  - Official Due Date: Friday, April 28^{th} @ 11:59PM
  - E-mail me a PDF and CC Hassan
  - I won't penalize late solutions submitted before Thursday, May 4^{th} at 11:59PM ☺

Side Channel: Memory Access Pattern

Program 1: secret value x

```
if (x < 5)
  z = A[0]
  A[1] = z*z
else
  z = A[100]
  A[101] = z*z
```

Suppose Attacker Learns Memory Access pattern was (100, 101).

What can attacker conclude?

Side Channel: Memory Access Pattern

When could attacker learn memory access pattern?

Scenario 1: User is storing (encrypted) array on an untrusted server

Scenario 2: Program 1 (trusted) is running on the same machine as program 2 (untrusted)

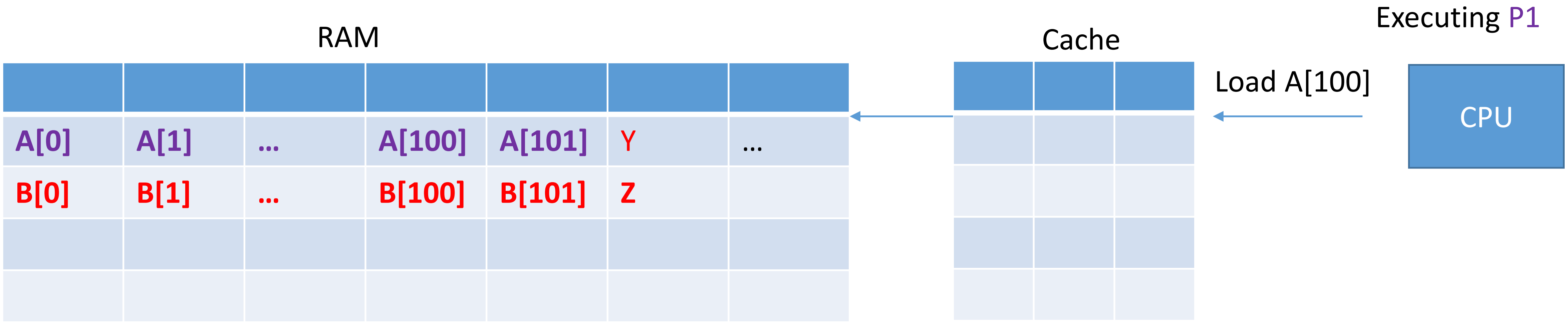
- Operating System ensures that program 2 cannot access program 1's memory. What is the problem?

Side Channel: Memory Access Pattern

When could attacker learn memory access pattern?

Scenario 2: Program 1 (trusted) is running on the same machine as program 2 (untrusted)

- Operating System ensures that program 2 cannot access program 1's memory. What is the problem?

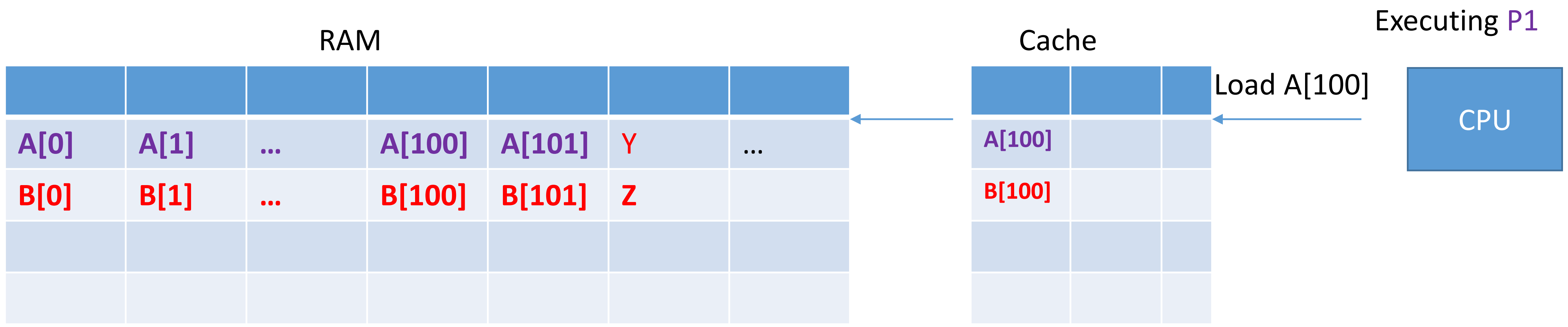


Side Channel: Memory Access Pattern

When could attacker learn memory access pattern?

Scenario 2: Program 1 (trusted) is running on the same machine as program 2 (untrusted)

- Operating System ensures that program 2 cannot access program 1's memory. What is the problem?

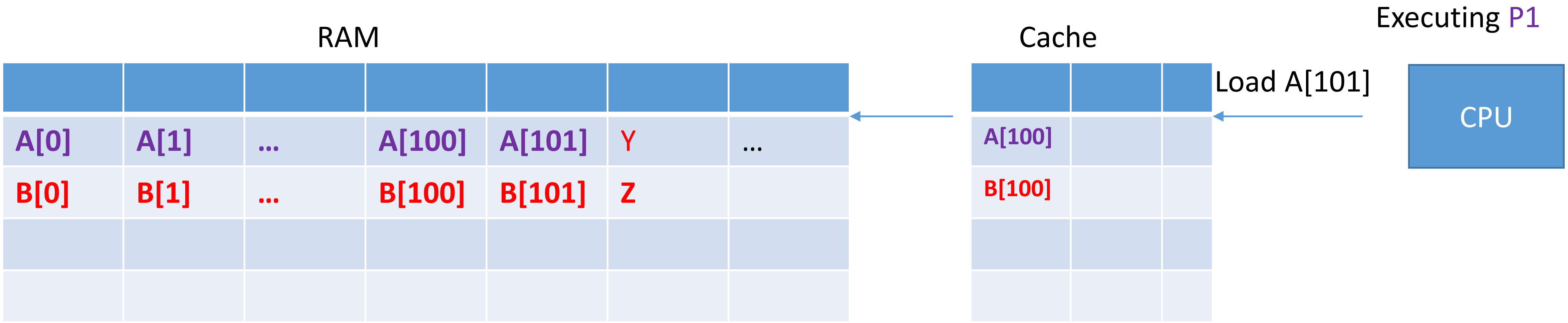


Side Channel: Memory Access Pattern

When could attacker learn memory access pattern?

Scenario 2: Program 1 (trusted) is running on the same machine as program 2 (untrusted)

- Operating System ensures that program 2 cannot access program 1's memory. What is the problem?

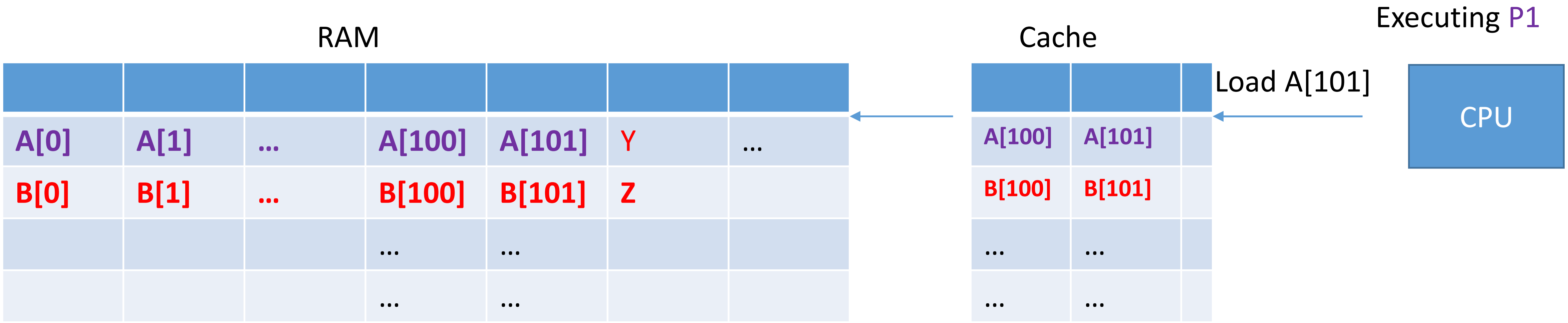


Side Channel: Memory Access Pattern

When could attacker learn memory access pattern?

Scenario 2: Program 1 (trusted) is running on the same machine as program 2 (untrusted)

- Operating System ensures that program 2 cannot access program 1's memory. What is the problem?

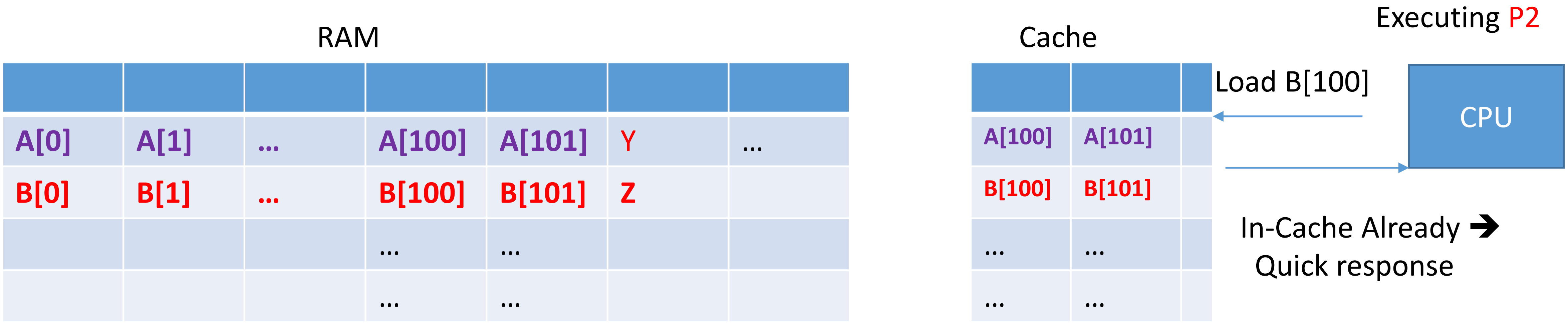


Side Channel: Memory Access Pattern

When could attacker learn memory access pattern?

Scenario 2: Program 1 (trusted) is running on the same machine as program 2 (untrusted)

- Operating System ensures that program 2 cannot access program 1's memory. What is the problem?

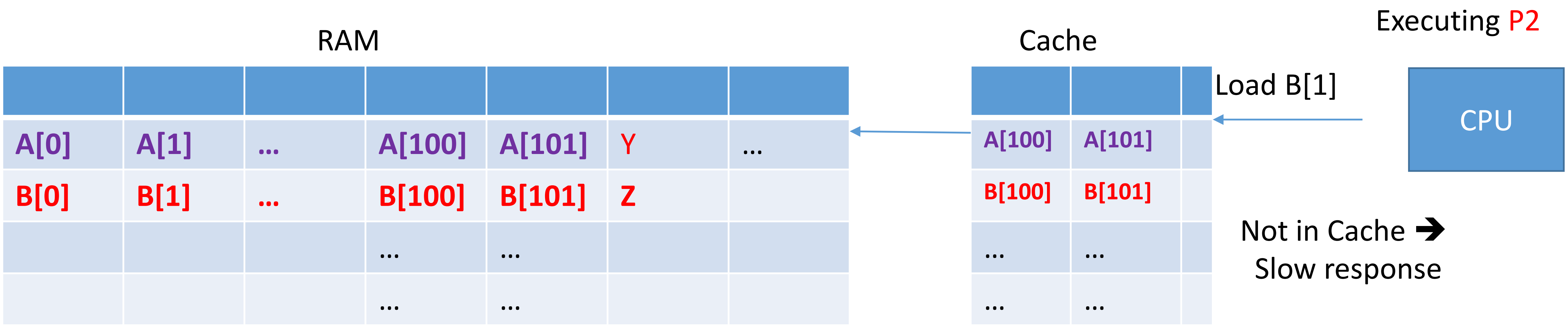


Side Channel: Memory Access Pattern

When could attacker learn memory access pattern?

Scenario 2: Program 1 (trusted) is running on the same machine as program 2 (untrusted)

- Operating System ensures that program 2 cannot access program 1's memory. What is the problem?



ORAM

Slides: Adapted from TCC 2022 Test of Time Talk

“A Walk in the ORAM Forest: About oblivious RAMs and something about trees” (Jesper Buus Nielsen)

Definition: ORAM = ObliviousArray

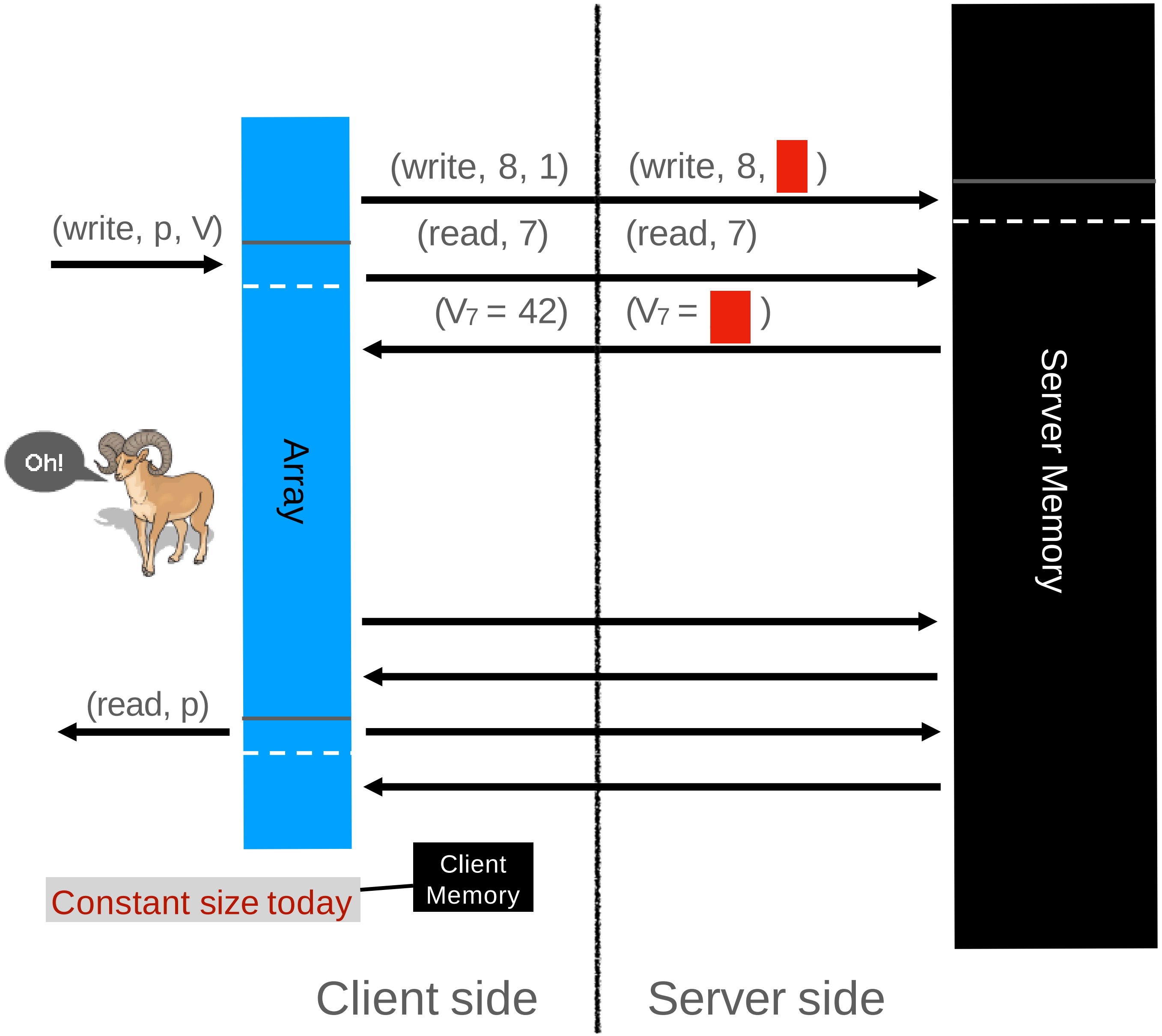
- An ORAM implements an array of N words of log(N) bits
  - Operations on array: operations
  - Operations on server memory: probes

- $\mathbf{o} = o_1, o_2, \ldots, o_n$
  - $o_i \in \{$ (read, p), (write, p, v) $\}$
- $A(o_i)$ = list of server positions probed
- $A(\mathbf{o}) = (A(o_1), A(o_2), \ldots, A(o_n))$

- **Oblivious**: $|\mathbf{o}^1|=|\mathbf{o}^2| \Rightarrow \Delta(A(\mathbf{o}^1),A(\mathbf{o}^2)) \leq \varepsilon$
  - **Perfect**: $\varepsilon=0$
  - **Statistical**: ε=negl.
  - **Computational**: $|\Delta|$=poly

- **Online**: Simulate one operation at a time
- **Offline**: Gets **o** up front and can plan

- **Amortised overhead**: $\lim_n |A(\mathbf{o})|/|\mathbf{o}|$
- **Worst-case overhead**: $\max_i |A(o_i)|$



Genesis

- **Oded Goldreich**: *Towards a Theory of Software Protection and Simulation by Oblivious RAMs*. STOC 1987
  - “We show how to implement n fetch instructions to a [RAM] memory of size m by making less than $n \cdot m^{\varepsilon}$ actual accesses, for every fixed $\varepsilon > 0$.”

- **Rafail Ostrovsky**: *An Efficient Software Protection Scheme*. CRYPTO 1989
  - Poly-logarithmic overhead

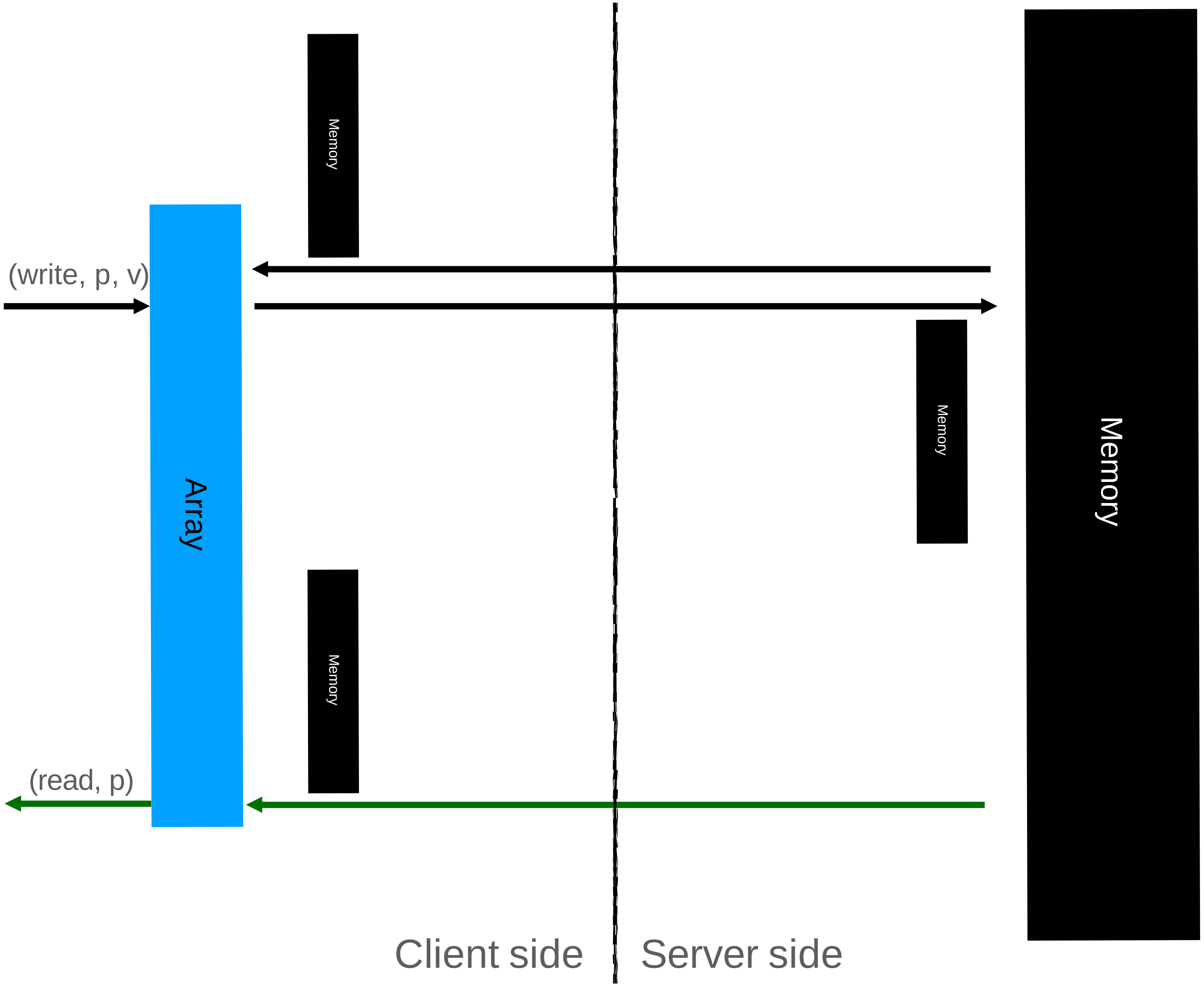
- **Oded Goldreich, Rafail Ostrovsky**: *Software Protection and Simulation on Oblivious RAMs*. J. ACM 43(3). 1996
  - Logarithmic lower bound in balls-in-bins model

[1] The rest is footnotes

Trivial ORAM

- Probe entire memory on each operation

- Perfect
- Overhead = *N*



Trivial ORAM

- Alice sends $c_1 = \mathrm{Enc}_{\mathrm{K}}(x_1), \dots, c_n = \mathrm{Enc}_{\mathrm{K}}(x_n)$ to Bob for storage

- When Alice wants to load/write x_i she requests for Bob to send all ciphertexts.

- **Subtle problem for write operations:**
  - What if Alice sends back $(c_1' = \mathrm{Enc}_{\mathrm{K}}(x_1{}'), c_2, \dots c_n)$?

    ➔ Bob learns that Alice was writing to location 1.
- **Solution:** Alice generates fresh ciphertexts $c_i' = \mathrm{Enc}_{\mathrm{K}}(x_i{}')$ for every (even if $x_i' = x_i$ is unchanged)

Trivial ORAM

- Alice sends $c_1 = \mathrm{Enc}_\mathrm{K}(x_1), \dots, c_n = \mathrm{Enc}_\mathrm{K}(x_n)$ to Bob for storage

- In remainder of this lecture we will assume that items are encrypted and that Alice always remembers to re-encrypt files every time it is touched.

- We can now focus only on the memory access pattern (i.e., probes).
- **Memory Access Pattern: o** = $o_1, o_2, \dots, o_n$
  - $o_i \in$ { (read, p), (write, p, v) }
  - Length of access pattern **|o| = n**

Trivial ORAM

- **Memory Access Pattern: o** = $o_1, o_2, \ldots, o_n$
  - $o_i \in$ { (read, p), (write, p, v) }
  - Length of access pattern **|o| = n**
- A(**o**) memory access pattern (probes) induced by ORAM compiler
  - Note: A(**o**) is a random variable
- **Trivial ORAM:**
  - $A(o_i)=(1,\ldots,n)$
  - A(**o**) = $(A(o_1),\ldots, A(o_n)) = ((1,\ldots,n),\ldots, (1,\ldots,n))$

Definition: ORAM = Oblivious Array

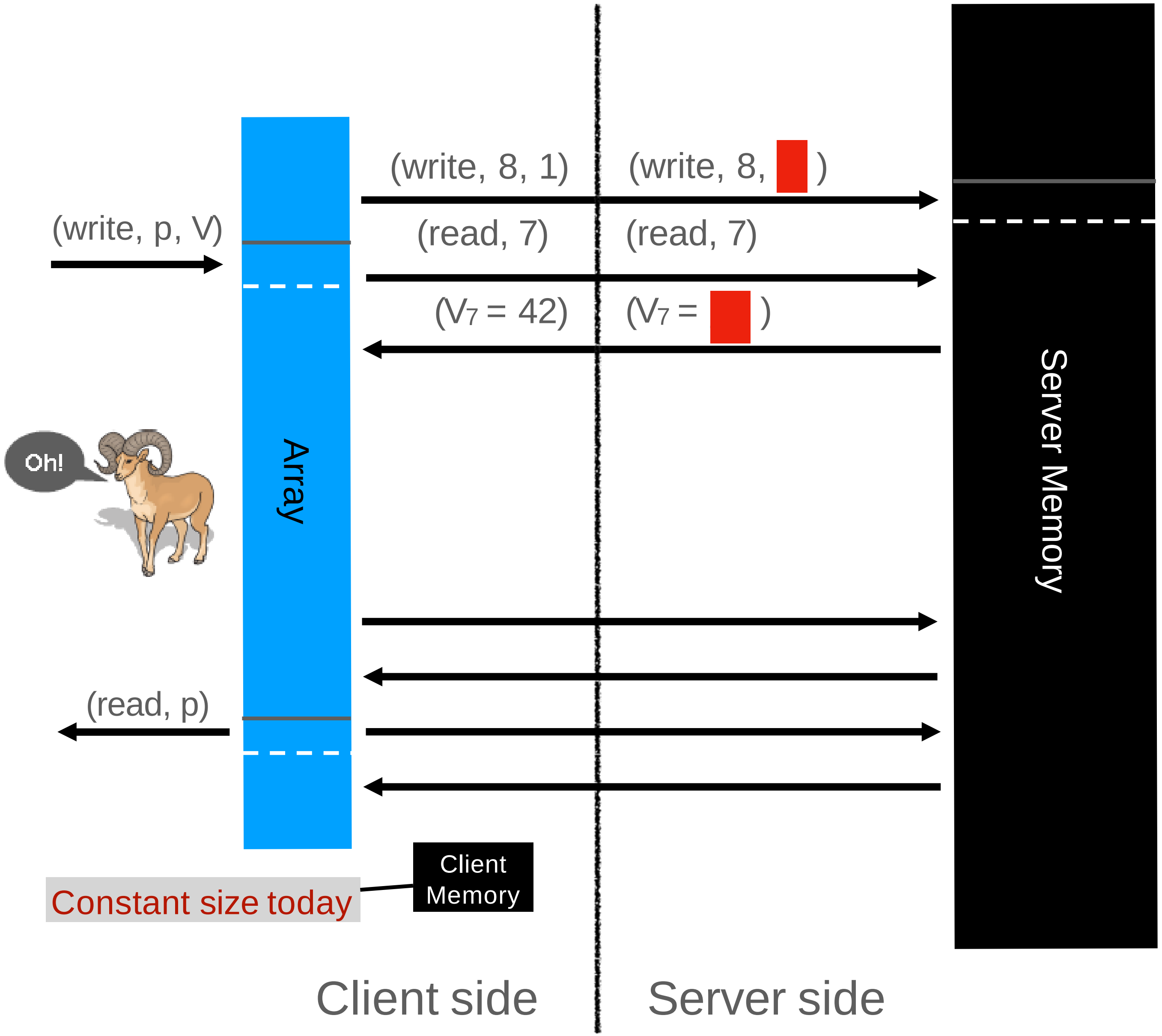
- An ORAM implements an array of N words of log(N) bits
  - Operations on array: operations
  - Operations on server memory: probes

- $\mathbf{o} = o_1, o_2, \ldots, o_n$
  - $o_i \in \{ (\text{read}, p), (\text{write}, p, v) \}$
- $A(o_i)$ = list of server positions probed
- $A(\mathbf{o}) = (A(o_1), A(o_2), \ldots, A(o_n))$

- **Oblivious**: $|\mathbf{o}^1|=|\mathbf{o}^2| \Rightarrow \Delta(A(\mathbf{o}^1),A(\mathbf{o}^2)) \leq \varepsilon$
  - **Perfect**: $\varepsilon=0$
  - **Statistical**: ε=negl.
  - **Computational**: $|\Delta|$=poly

- **Online**: Simulate one operation at a time
- **Offline**: Gets **o** up front and can plan

- **Amortised overhead**: $\lim_n |A(\mathbf{o})|/|\mathbf{o}|$
- **Worst-case overhead**: $\max_i |A(o_i)|$



ORAM Security/Limitation

- **Memory Access Pattern: o** = $o_1, o_2, \ldots, o_n$
  - $o_i \in$ { (read, p), (write, p, v) }
  - Length of access pattern **|o| = n**

- **Security Guarantee:** For any two access patterns $|\mathbf{o}^1|=|\mathbf{o}^2|$ of the same length an attacker cannot distinguish between $A(\mathbf{o}^1)$ and $A(\mathbf{o}^2)$
  - Identical Distributions/Statistically Close/Computationally Indistinguishable

- Limitation: If $|\mathbf{o}^1|>|\mathbf{o}^2|$ there are no guarantees.
  - Can append $|\mathbf{o}^2|$ with dummy operations
  - Must append to maximum running time
  - Efficiency bottleneck.

Oblivious Shuffling

- Initial Array: A[0],…,A[n-1]
- Shuffled Array: $A[\pi(0)], \ldots, A[\pi(n-1)]$
- Security Goals:
  - π is random permutation
  - Attacker who observes memory access pattern during shuffling cannot distinguish between π and π' (random unrelated permutation)
- Can do this using $\tilde{O}(N)$ probes

Oblivious Shuffling

- Initial Array: A[0],…,A[n-1]
- Shuffled Array: $A[\pi(0)], \dots, A[\pi(n-1)]$

- Shuffle via Merge Sort?
  - Suppose we have obliviously shuffled $\mathrm{L} = A[0], \dots, A\left[\frac{n}{2} - 1\right]$ and $\mathrm{R} = A\left[\frac{n}{2}\right], \dots, A[n-1]$
  - Obtained $\pi_L(\mathrm{L}) = A[\pi_L(0)], \dots, A[\pi_L(n-1)]$ and $\pi_R(R) = A\left[\pi_R\left(\frac{n}{2}\right)\right], \dots, A[\pi_R(n-1)]$
  - Merge with $\tilde{O}(N)$ probes?
  - Trickier than it seems…

Oblivious Merge (Attempt 1)

$$\mathrm{Merge}\left(i, A[\pi_L(0)], \ldots, A[\pi_L(n-1)], j, A\left[\pi_R\left(\frac{n}{2}\right)\right], \ldots, A[\pi_R(n-1)]\right)$$

Set $b = \begin{cases} 0 & w.p\ \frac{|L|-i}{|L|-i+|R|-j} \\ 1 & otherwise \end{cases}$ and $Y = \begin{cases} A[\pi_L(i)] & if\ b = 0 \\ A\left[\pi_R\left(\frac{n}{2}+j\right)\right] & otherwise \end{cases}$

$$\mathrm{Z} := \mathrm{Merge}\left(i+1-b, A[\pi_L(0)], \ldots, A[\pi_L(n-1)], |R|-b, A\left[\pi_R\left(\frac{n}{2}\right)\right], \ldots, A[\pi_R(n-1)]\right)$$

Return $\mathrm{Y} \circ \mathrm{Z}$

Initial Run with i=0, j=0. Problem?

Oblivious Merge (Attempt 1)

$$\mathrm{Merge}\left(i, A[\pi_L(0)], \dots, A[\pi_L(n-1)], j, A\left[\pi_R\left(\frac{n}{2}\right)\right], \dots, A[\pi_R(n-1)]\right)$$

Set $b = \begin{cases} 0 & w.p\ \frac{|L|-i}{|L|-i+|R|-j} \\ 1 & otherwise \end{cases}$ and $Y = \begin{cases} A[\pi_L(i)] & if\ b = 0 \\ A\left[\pi_R\left(\frac{n}{2}+j\right)\right] & otherwise \end{cases}$

$$\mathrm{Z} := \mathrm{Merge}\left(i+1-b, A[\pi_L(0)], \dots, A[\pi_L(n-1)], |R|-b, A\left[\pi_R\left(\frac{n}{2}\right)\right], \dots, A[\pi_R(n-1)]\right)$$

Return $\mathrm{Y} \circ \mathrm{Z}$

Initial Run with i=0, j=0. Problem? Red probes leak information about the final permutation!

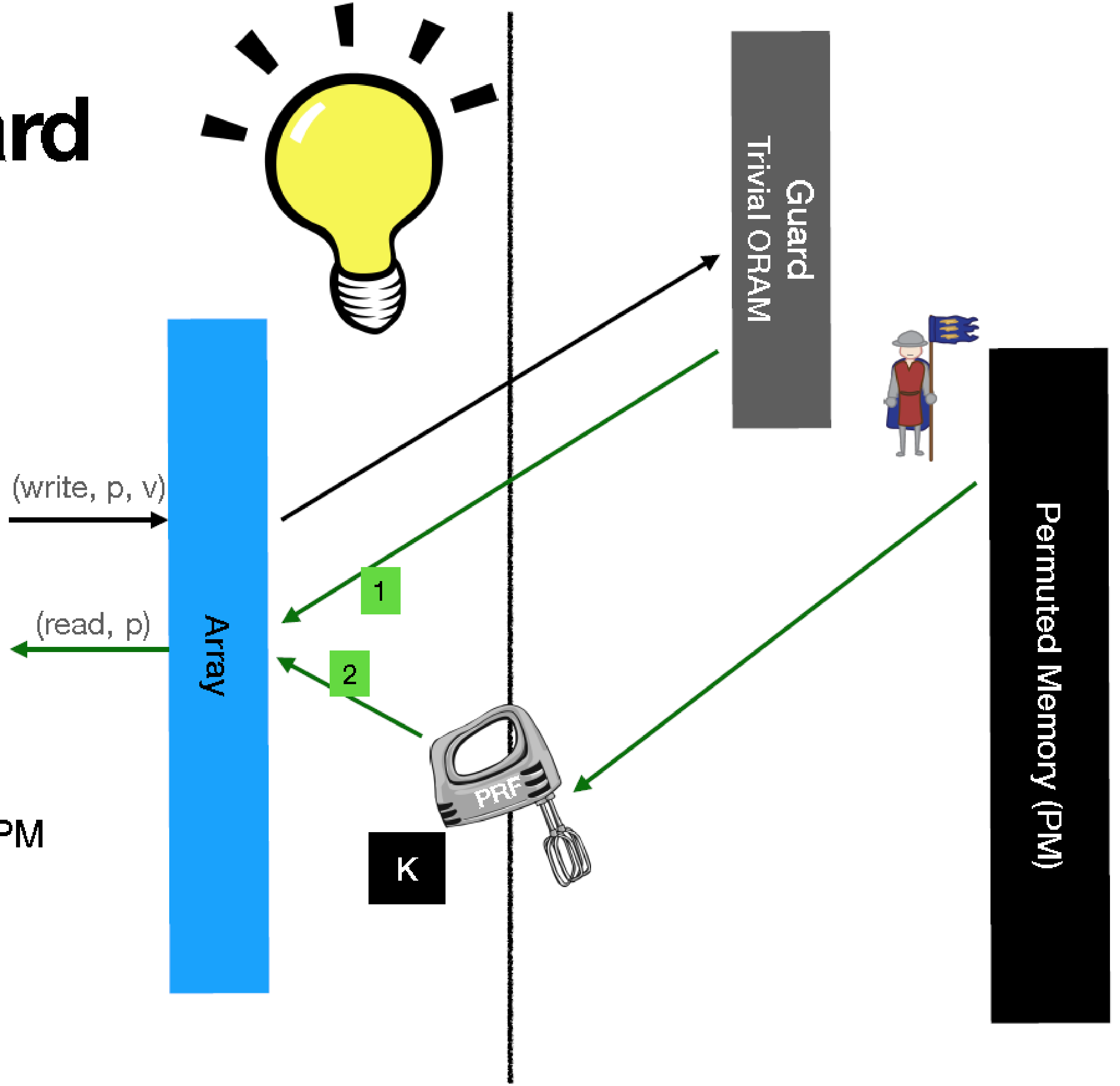
Permute and Guard

- Guard of size *G*
- Store array + *G* dummy elements uniformly random permuted on server in PM

- **Write**: Write to Guard

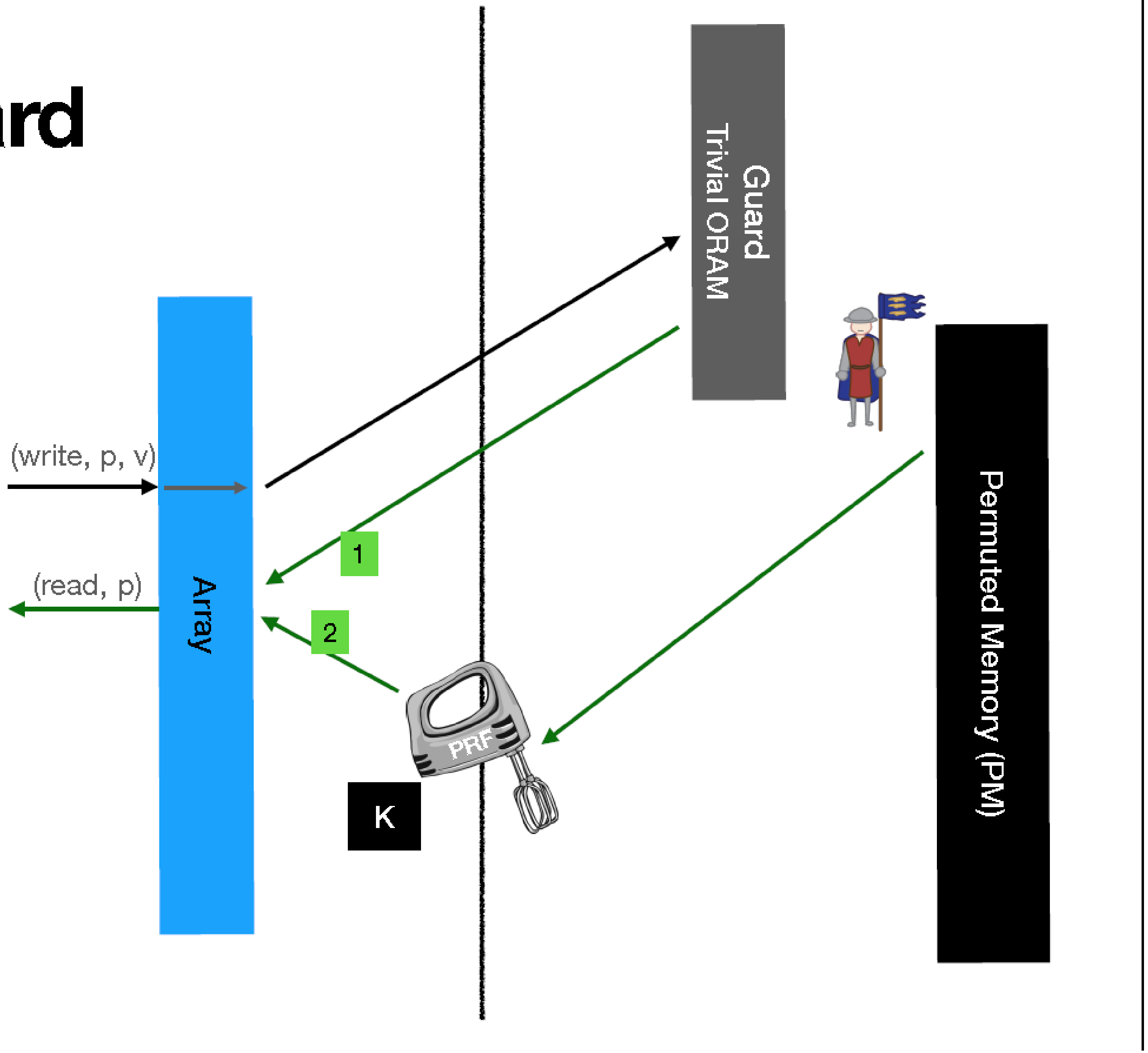
- **Read**
  - First read from Guard
    - If not found, read from PM
    - If found read dummy from PM
  - Write to Guard

- **Refresh**: After *G* operations
  - Join Guard and PM and create new fresh PM with fresh *K*



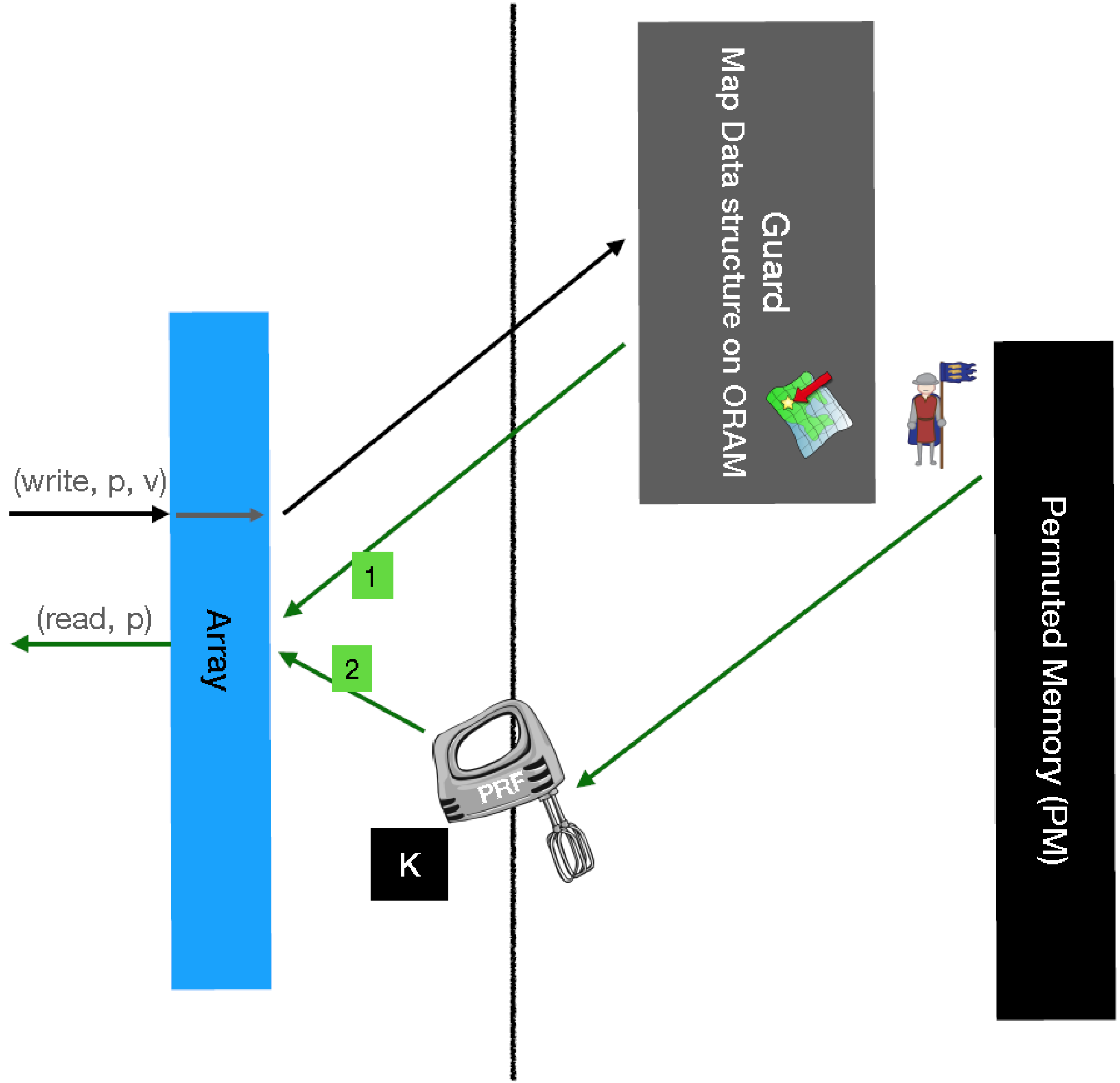
Permute and Guard

- Computational
- Price of G operations:
  - Operations: G*|Guard| ≈ G^2
  - Permute ≈ N
- Balances at $G = \sqrt{N}$
- Amortised overhead: $AOH \approx \sqrt{N}$



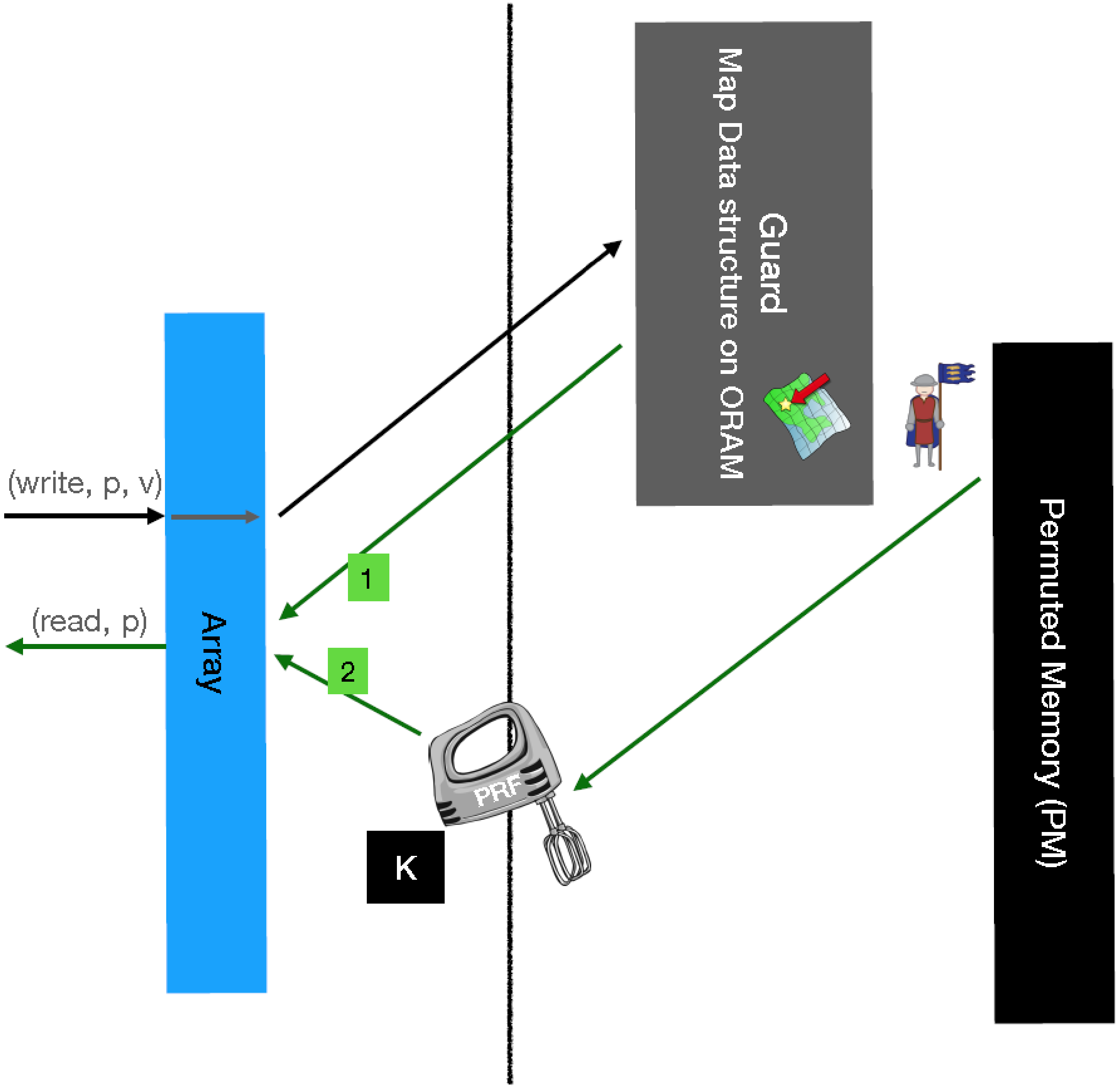
Recursive

- Use a larger Guard implemented as a Map data structure on a smaller ORAM
- We can do ORAM with AOH = $N^{1/2}$
- Set the smaller ORAM to size $\approx N^{2/3}$ and capacity $N^{2/3}$ in the map
- Price of G operations:
  - Op.s: $G^{*}OH_{Guard} = G\,(N^{2/3})^{1/2} = G\,N^{1/3}$
  - Permute $\approx N$
- Balances at $G \approx N / N^{1/3} \approx N^{2/3}$
- Amortised overhead: AOH $\approx N/N^{2/3} = N^{1/3}$



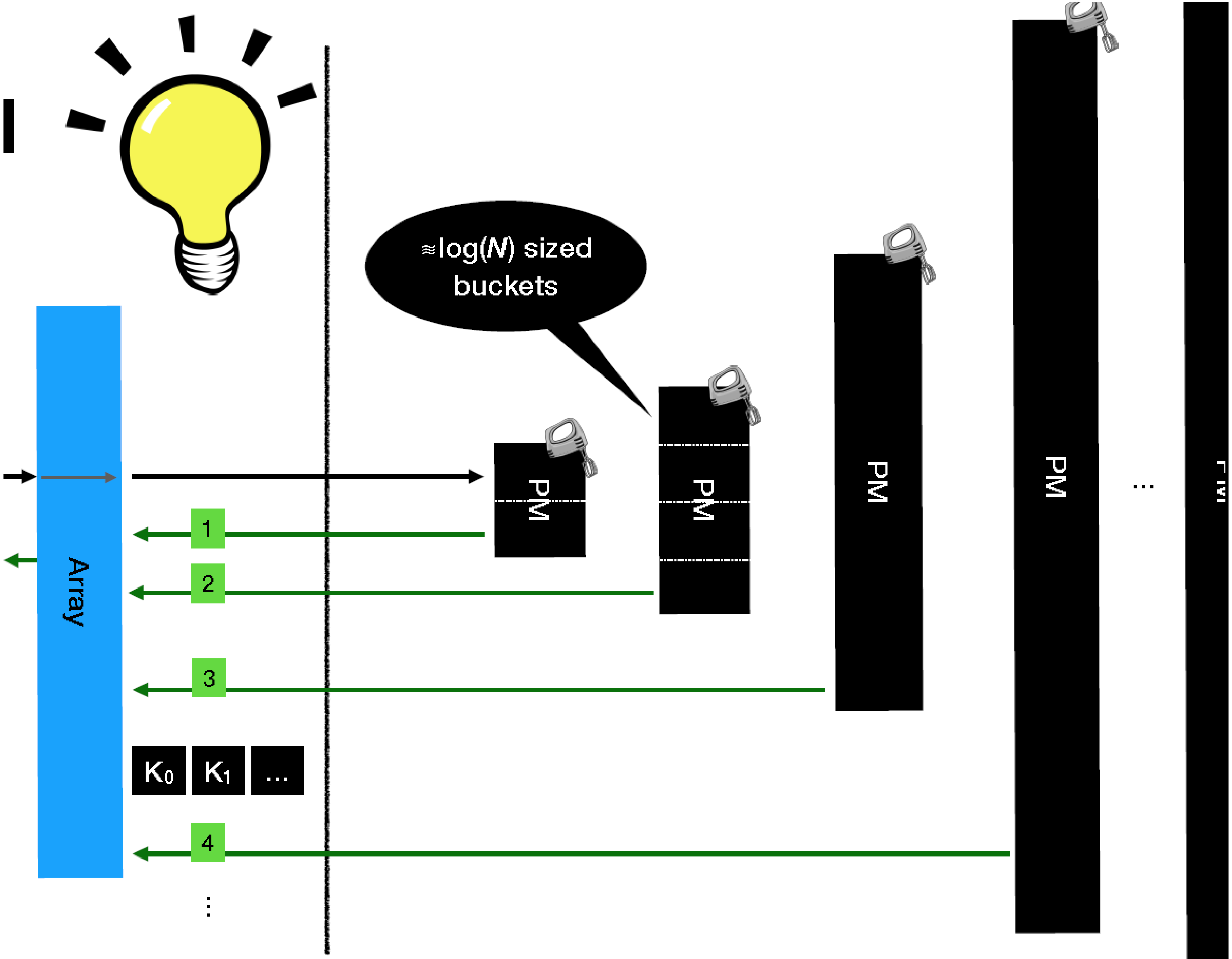
Recursive

- Use a larger Guard implemented as a Map data structure on a smaller ORAM
- We can do ORAM with AOH = $N^{1/(k-1)}$
- Set the smaller ORAM to size $\approx N^{(k-1)/k}$ and capacity $N^{(k-1)/k}$
- Price of G operations:
  - Op.s: $G^{*}\mathrm{OH}_{\mathrm{Guard}} = G\, N^{1/k}$
  - Permute $\approx N$
- Balances at G $\approx N / N^{1/k} \approx N^{(k-1)/k}$
- Amortised overhead: AOH $\approx N/N^{(k-1)/k} = N^{1/k}$



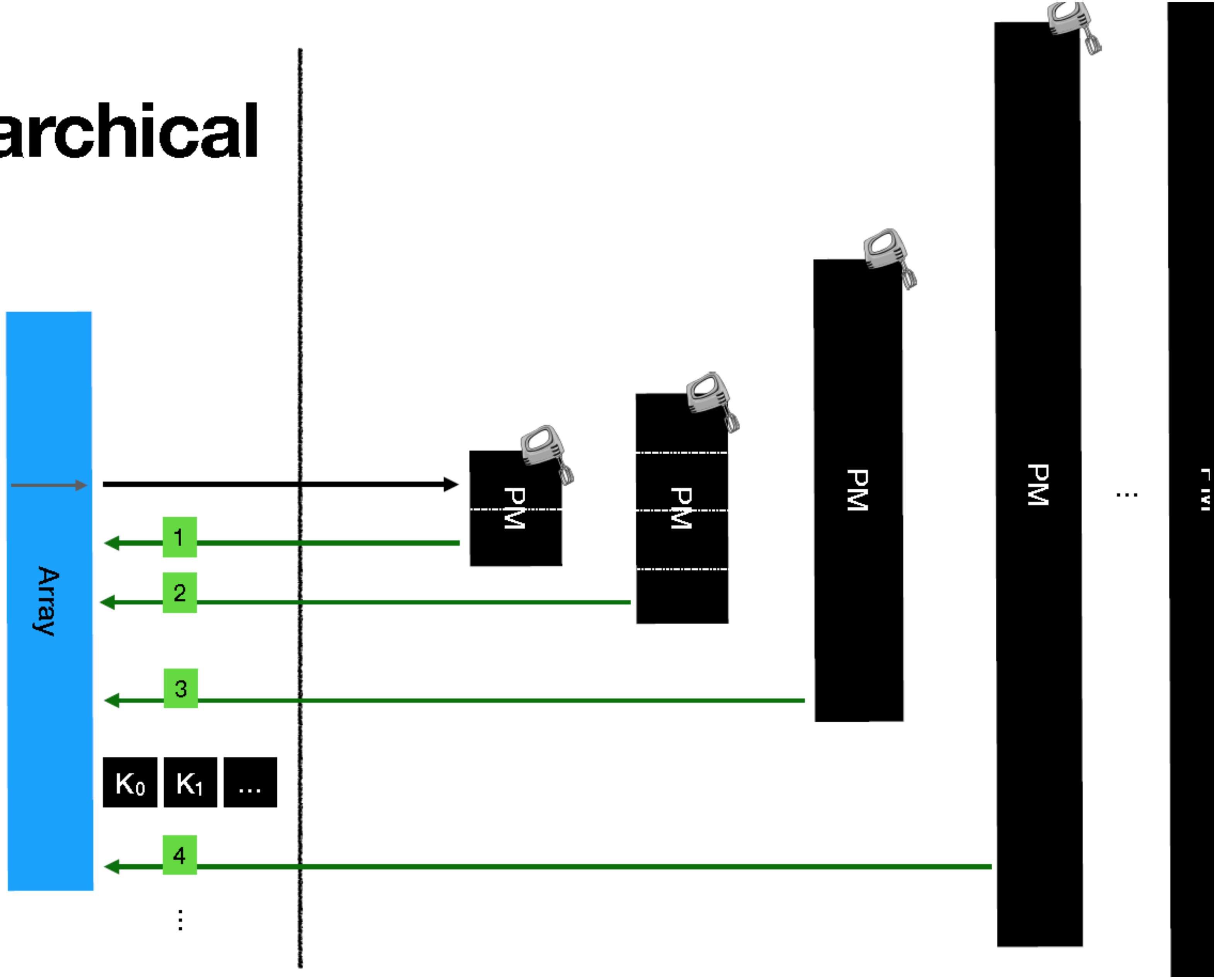
Hierarchical

- After each 2^i operations shuffle the elements in levels 1, …, i and store them in level i using fresh PRF_K
- **Lemma**: All guards get emptied before they are full
- Some nitty-gritty stuff with dummies and enough room for the map data structure and dummies



AOH of Hierarchical

- Guard i has size about $2^i\ i$
- Guard i is sorted every 2^{-i} operations at a price of $2^i i^2$
- Guard i costs amortised i^2 per operation
- $1 + 2^2 \ldots + \log(N)^2 \approx \log(N)^3$
- AOH: $\log(N)^3$



Ball in Bins

- **Lower bound [GO'96]:** To implement N operations you must make $N \log(N)$ probes (if you use a balls-in-bins ORAM)

- **Balls-in-Bins:**
  - The ORAM construction does not look at the data being stored, it is treated atomically
  - Can only confuse the adversary by swapping balls around

Now What?

- By 1996 some very basic question are open:
  - Is computational security inherent?
  - Is amortisation inherent or can we get poly-log worst-case overhead?
  - Can we go below OH *log(N)* using a non-balls-in-bins ORAM?
  - Is the OH *log(N)* or *log(N)*^{3} or somewhere in between?

March 2010: Information Theoretic Solutions

- **Ivan Damgård**, **Sigurd Meldgaard**, **Jesper Buus Nielsen**: *Perfectly Secure Oblivious RAM without Random Oracles*. TCC 2011
  - Posted on IACR ePrint on 2 March 2010
- **AOH: log(N)^3**
- **Perfect** security
- Also:
  - Method for ORAM in MPC
  - N log(N) **lower bound** on amount of **randomness** ORAM must store

- **Milkós Ajtai.** *Oblivious RAMs without cryptographic assumptions*. STOC 2010
  - Posted on ECCC on 6 March 2010
  - STOC had deadline November 5 2009
- **Poly-logarithmic overhead**, constant in exponent not explicitly given
- **Statistical** error for t operations: $t^{-\log(t)}$

2011: Worst-Case Poly-Log OH

- **Elaine Shi**, **T.-H. Hubert Chan**, **Emil Stefanov**, **Mingfei Li**: *Oblivious RAM with O((log N)^3)* ***Worst-Case Cost***. ASIACRYPT 2011
  - **Worst case *log(N)^3***
  - Amortised overhead *log(N)^2* when using DMN'11 as a tool: ORAM on *small* internal buckets
  - **Statistical**

- **Michael T. Goodrich**, **Michael Mitzenmacher, Olga Ohrimenko**, **Roberto Tamassia**: *Oblivious RAM simulation with efficient* ***worst-case*** *access overhead*. CCSW 2011
  - **Worst case *N$^{1/2}$log(N)^2***
  - ***Worst case log(N)^2*** with large client memory.
  - **Computational**

- **Eyal Kushilevitz**, **Steve Lu**, **Rafail Ostrovsky**: *On the (in)security of hash-based oblivious RAM and a new balancing scheme*. SODA 2012
  - IACR ePrint August 2011
  - **Worst case *log(N)^2 / log(log(N))***
  - **Computational**

- The last two paper uses "**deamortisation**":
  - Have two hash maps on each level
  - Shuffle one while using the other and swap when the shuffling is done

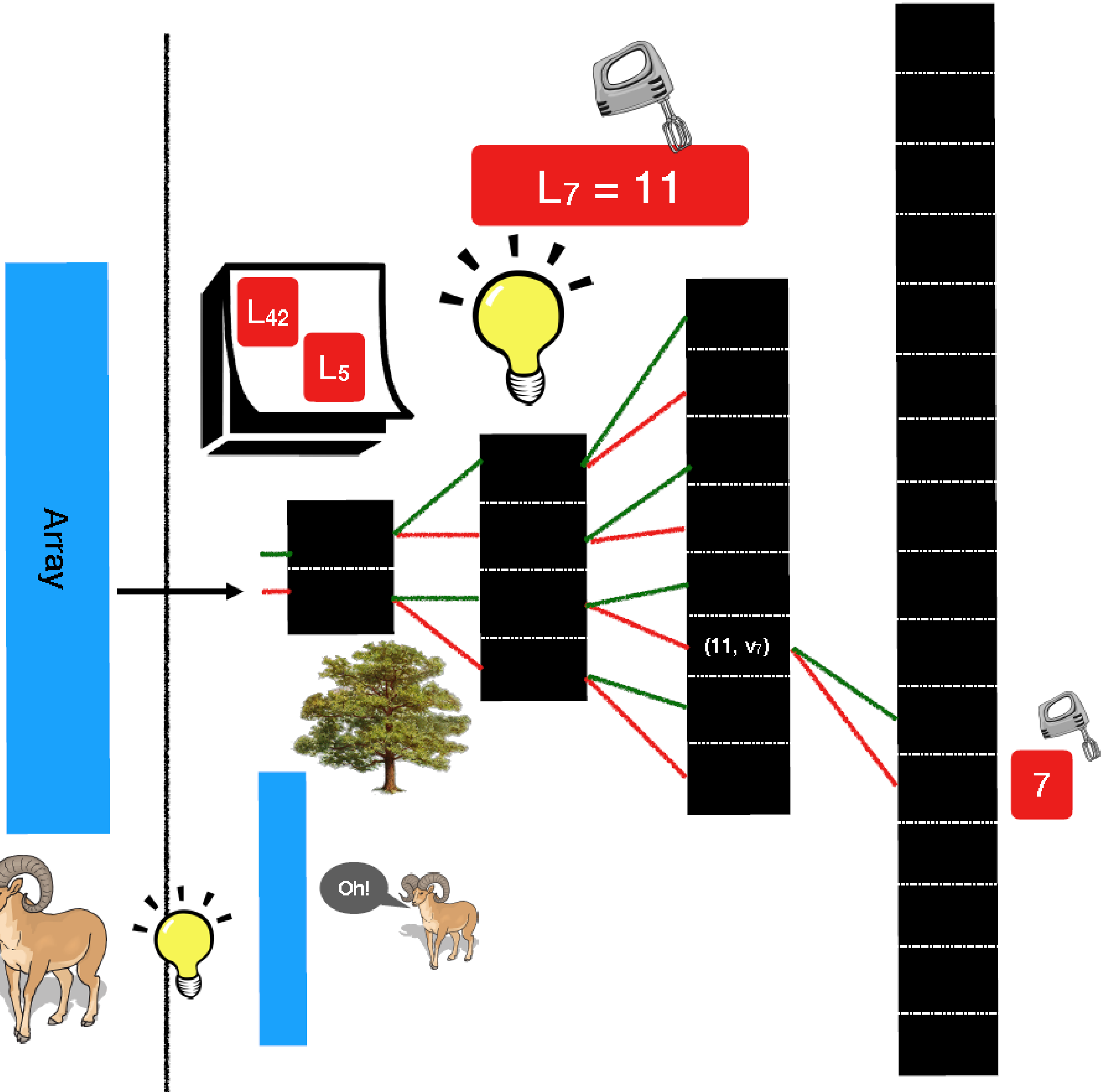
2011-2013: Path ORAM

- **Elaine Shi**, **T.-H. Hubert Chan**, **Emil Stefanov**, **Mingfei Li**: *Oblivious RAM with O((log N)^3) Worst-Case Cost*. ASIACRYPT 2011
  - Worst case overhead $log(N)^3$
- **Emil Stefanov**, **Marten van Dijk**, **Elaine Shi**, **Christopher W. Fletcher**, **Ling Ren**, **Xiangyao Yu**, **Srinivas Devadas**: *Path ORAM: an extremely simple oblivious RAM protocol*. CCS 2013
  - **Worst case overhead $log(N)^2$**

- **PathORAM Idea**:
  - <u>**Fix the tree**</u> (do not shuffle the levels as Ajtai and DMN'11)
  - Store V_p on path to ***uniformly random*** **leaf** L_p
  - **Use smaller ORAM L with size $N' = N/2$ and word size $w = 2 \log(N)$**
  - For all $p' = p \text{ div } 2$ store (L_{2p+0}, L_{2p+1}) in $L[p]$

Path ORAM

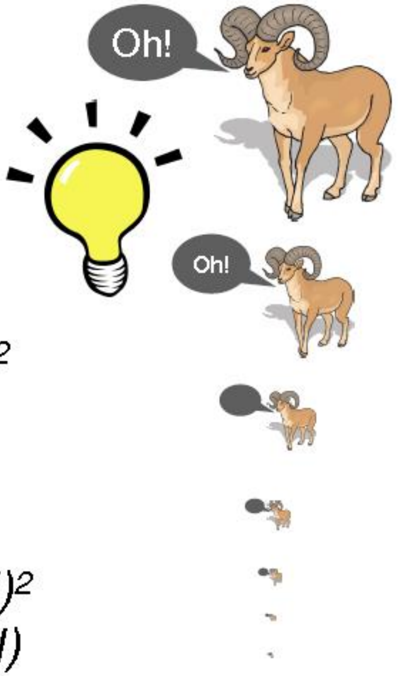
- Divide layer i into 2^i buckets of size $O(1)$
- Impose a BST on the buckets
- ***Do not shuffle the levels!***
- Assign position p of array to a **uniformly random leaf** L_p of the tree
- **Invariant: (L_p, V_p) is always to be found in bucket on path to L_p**
- Inject fresh (L_p, V_p) at root after access
- **Read/write** by r/w **entire path** to L_p
  - Push (L, V) pairs as low as possible before writing the path back
- Use a **stash** to store overflow from buckets
- **Worst-case OH: $log(N)$**



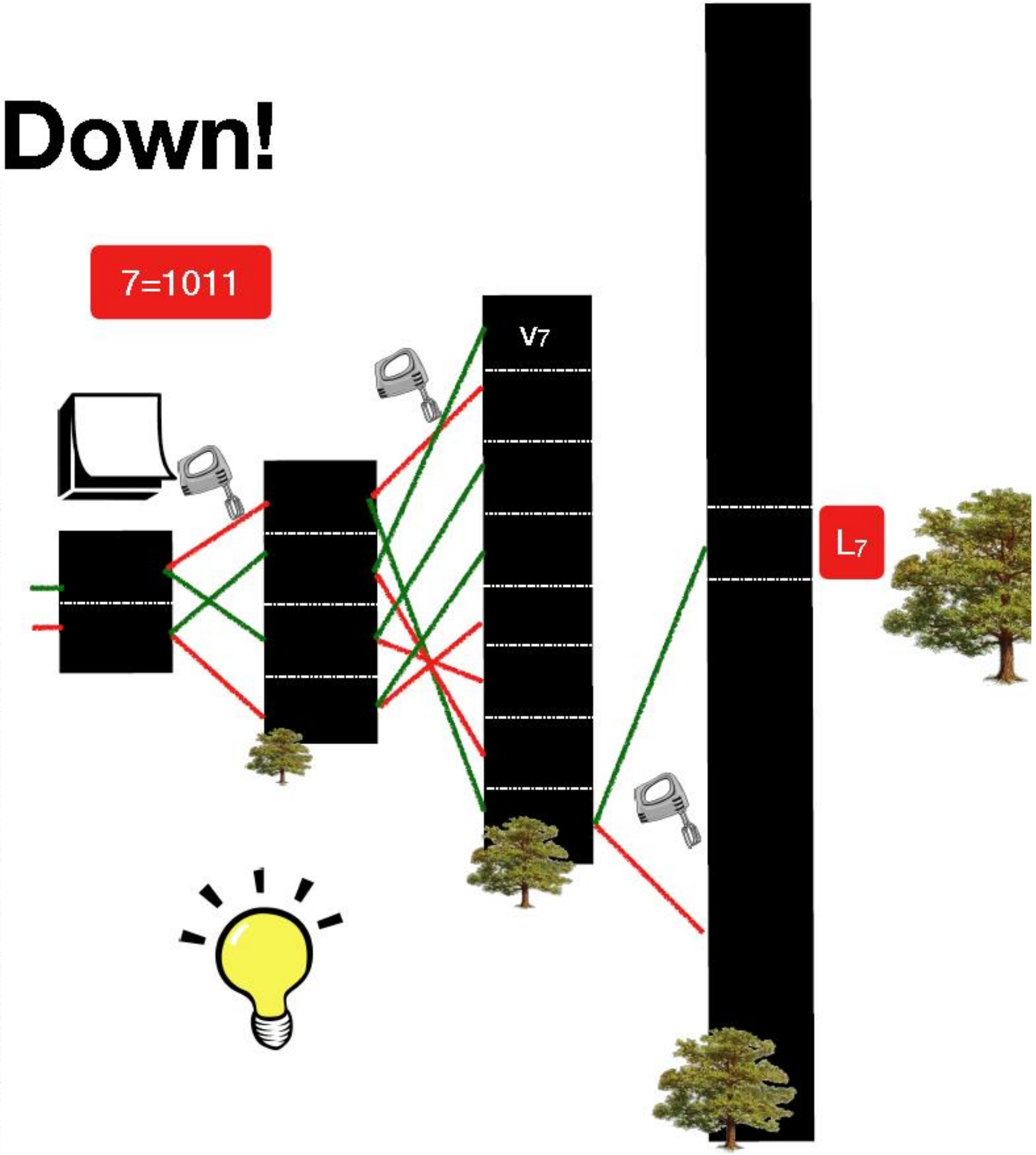
Its ORAMs All the Way Down!

- Recursively store position maps for levels of size N_i in ORAMs of size $N_{i-1} = N_i/2$

- **Statistical security**
- **Worst-case OH: $\boldsymbol{log(N)^2}$**
  - $\sum_{i=1,\ldots,log(N)} log(2^i) \approx log(N)^2$



- If the word size is w = $log(N)^2$ then the OH becomes $log(N)$
  - Not trivial
  - Not unreasonable in practice



2016-2018: What About that Lower Bound?!

- **Elette Boyle**, **Moni Naor**: *Is There an Oblivious RAM Lower Bound*? ITCS 2016
- Points out the following about the Goldreich-Ostrovsky lower bound:
  - It only applies to "**balls-in-bins**" algorithms, i.e., algorithms where the ORAM may only shuffle stored values around and not apply any sophisticated encoding of the data
  - It only applies to **computationally <u>un</u>bounded** adversaries
  - But it applies even to **off-line** algorithms and improving it will involve switching to considering on-line or proving unconditional lower bounds of circuits for sorting

- **Kasper Green Larsen**, **Jesper Buus Nielsen**: *Yes, There is an Oblivious RAM Lower Bound!* CRYPTO 2018
  - Applies to **all types of on-line** algorithms
  - Applies also to **computationally bounded** adversaries

2016-2018: What About that Lower Bound?!

- **Kasper Green Larsen**, **Jesper Buus Nielsen**: *Yes, There is an Oblivious RAM Lower Bound!* CRYPTO 2018
  - Applies to **all types of on-line** algorithms
  - Applies also to **computationally bounded** adversaries

- **Mihai Patrascu**, **Erik D. Demaine**: **Logarithmic** *Lower Bounds in the Cell-Probe Model*. SIAM J. Comput. 35(4) 2006
  - Introduced the “Information transfer” technique
  - On-line algorithms turns time into location by putting events on a time-line
  - Reasoning about how information moves around in space is much, much easier than reasoning about computational complexity
  - Put a binary tree on top to reason about how information is moved

- **LN’18**: The “Information transfer” technique normally does not apply to array maintenance but when combined with *obliviousness* suddenly it does

PanORAMa

After the Cuckoo:

- Lookup is $log(N)$
- Amortised OH is $log(N)^2$ because of having to **shuffle the levels**

Sarvar Patel, **Giuseppe Persiano**, **Mariana Raykova**, **Kevin Yeo**: *PanORAMa: Oblivious RAM with Logarithmic Overhead*. FOCS 2018

- Amortised Overhead: $log(N)\ log(log\ (N))$
- <u>No need to shuffle merged levels</u>:

  The remaining, untouched elements are already randomly permuted!

  - Extract the untouched elements
    - One can do this in O(N log(log(N)))
    - Sorting small buckets of size O(log(N))
  - Randomly merge the permuted untouched elements
    - Only has to add O(N) randomness but suffers log(log(N)) to do it obliviously

OptORAMa

Gilad Asharov, **Ilan Komargodski**, **Wei-Kai Lin**, **Kartik Nayak**, **Enoch Peserico**, **Elaine Shi**: *OptORAMa: Optimal Oblivious RAM*. EUROCRYPT 2020

- Amortised Overhead: O(*log N*)
- **Has O(*N*) oblivious, deterministic tight compaction algorithm!**
- **Tight compactions**: Sort elements marked *0* or *1* such that all marked *0* appear first
  - Circumvents 0-1 lower bound by doing non-comparison operations
- Extract the unused elements using tight compaction
- Merge-shuffle: Just a “reverse tight compaction” which is O(*N*)
- Paper is 73 pages so I must have simplified somewhere :-)

All at Once!?!?!

- Perfect, worst-case, OH = *log(N)* ?
  - **Michael A. Raskin**, **Mark Simkin**: *Perfectly Secure Oblivious RAM with Sublinear Bandwidth Overhead*. ASIACRYPT 2019
  - Worst case *OH* = $\sqrt{N}$
- Computational, worst-case, OH = *log(N)* ?
  - **Gilad Asharov**, **Ilan Komargodski**, **Wei-Kai Lin**, **Elaine Shi**: *Oblivious RAM with Worst-Case Logarithmic Overhead*. CRYPTO 2021
  - New deamortisation technique compatible with merge-shuffle and compaction
- Perfect, amortised, OH = *log(N)* ?
  - **T.-H. Hubert Chan**, **Elaine Shi**, **Wei-Kai Lin**, **Kartik Nayak**: *Perfectly Oblivious (Parallel) RAM Revisited, and Improved Constructions*. ITC 2021
  - AOH = $O(\log(N)^3/\log(\log(N)))$
- Perfect, worst-case, OH = *log(N)* ?
- Statistical, worst-case, OH = *log(N)*
  - Path ORAM for word-size $\log(N)^2$

ORAMs with Special Properties

- **ORAMs good for MPC**
  - **Marcel Keller**, **Peter Scholl**: *Efficient, Oblivious Data Structures for MPC*. ASIACRYPT 2014
  - **Xiao Wang**, **T.-H. Hubert Chan**, **Elaine Shi**: *Circuit ORAM: On Tightness of the Goldreich-Ostrovsky Lower Bound*. CCS 2015

- **Parallel ORAM**
  - **Elette Boyle**, **Kai-Min Chung**, **Rafael Pass**: *Oblivious Parallel RAM and Applications*. TCC 2016

- **Round Complexity**
  - **David Cash**, **Andrew Drucker**, **Alexander Hoover**: *A Lower Bound for One-Round Oblivious RAM*. TCC 2020
  - $\sqrt{N}$ Overhead
  - Oh a √ again!?

- **Random-index ORAM**
  - **Shai Halevi**, **Eyal Kushilevitz**: *Random-Index Oblivious RAM*. TCC Yesterday.
  - This one is One-Round…

- …

Other Oblivious Data Structures

- **Xiao Shaun Wang**, **Kartik Nayak**, **Chang Liu**, **T.-H. Hubert Chan**, **Elaine Shi**, **Emil Stefanov**, **Yan Huang**: *Oblivious Data Structures*. CCS 2014
- **Riko Jacob**, **Kasper Green Larsen**, **Jesper Buus Nielsen**: *Lower Bounds for Oblivious Data Structures*. SODA 2019
  - Ω(log *N*) lower bounds for oblivious stacks, queues, deques, priority queues and search trees
- **Giuseppe Persiano**, **Kevin Yeo**: *Lower Bounds for Differentially Private RAMs*. EUROCRYPT 2019
  - **Constant DP security of a single operation implies Ω(log *N*) OH**
  - Information transfer does not work here, introduces chronogram technique to ORAM
    - **Fredman, M.**, **Saks, M.**: *The cell probe complexity of dynamic data structures*. STOC 1989
- **Kasper Green Larsen**, **Mark Simkin**, **Kevin Yeo**: *Lower Bounds for Multi-server Oblivious RAMs*. TCC 2020
  - *K* servers of which the adversary can see the access pattern to only one
  - If better than approx *1/K* security then OH Ω(log *N*)
- **Zahra Jafargholi**, **Kasper Green Larsen**, **Mark Simkin**: *Optimal Oblivious Priority Queues*. SODA 2021
  - OH = *10 log(N)*
- **Ilan Komargodski, Elaine Shi:** *Differentially Oblivious Turing Machines*. ITCS 2021
  - OH *O(log log N)*

- **Differentially private stack can be done with OH *O(log log N).***

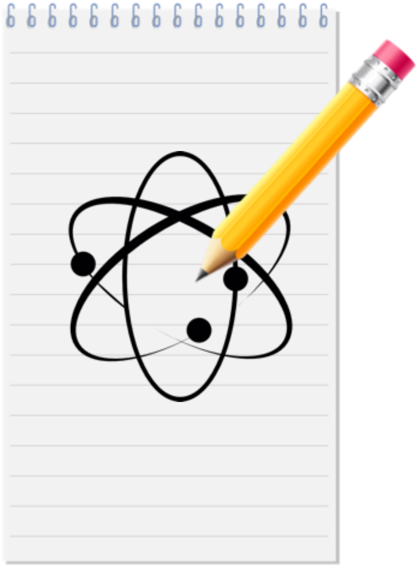
Thanks for Listening

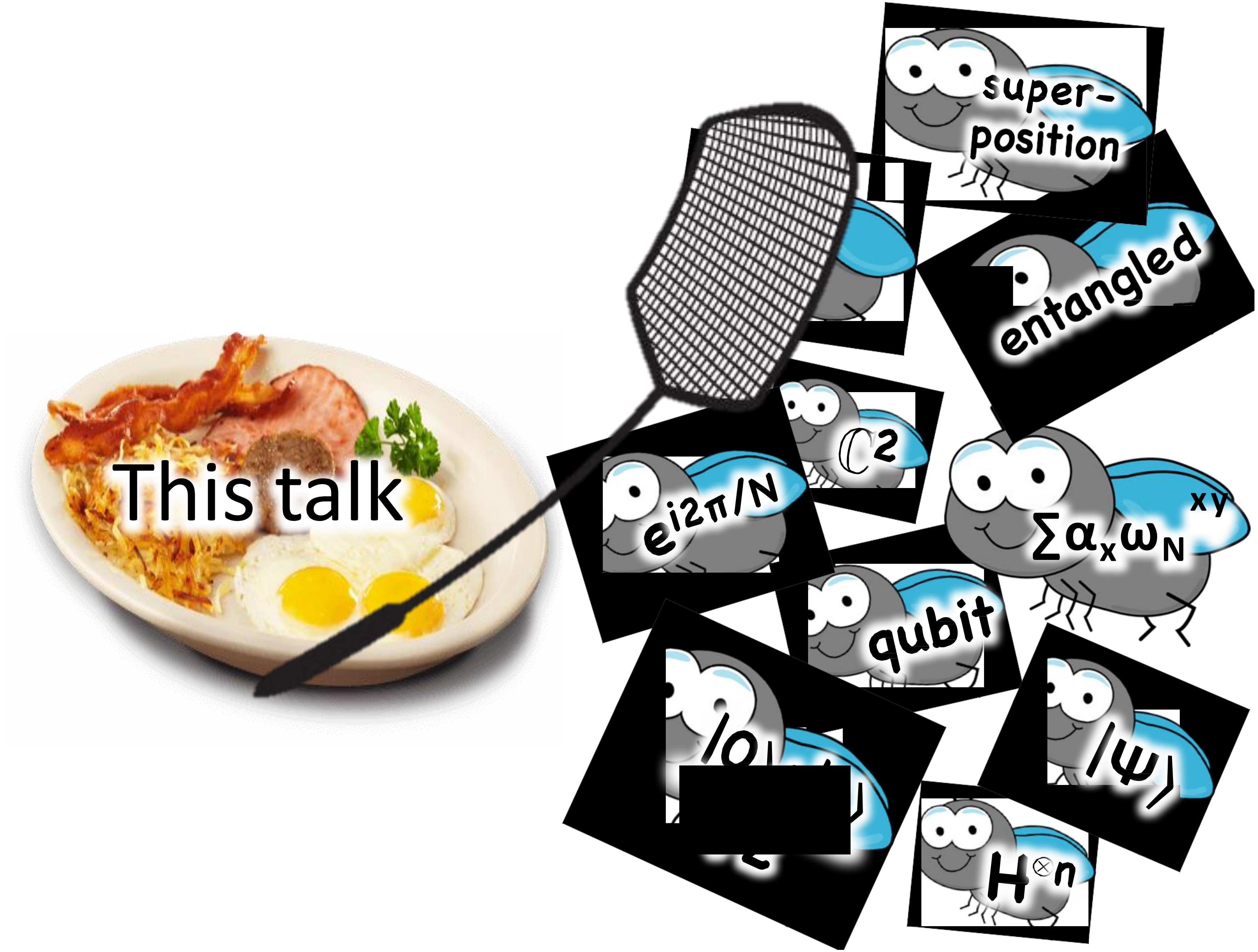
How to Record Quantum Queries

and Applications to Quantum Indifferentiability

Mark Zhandry

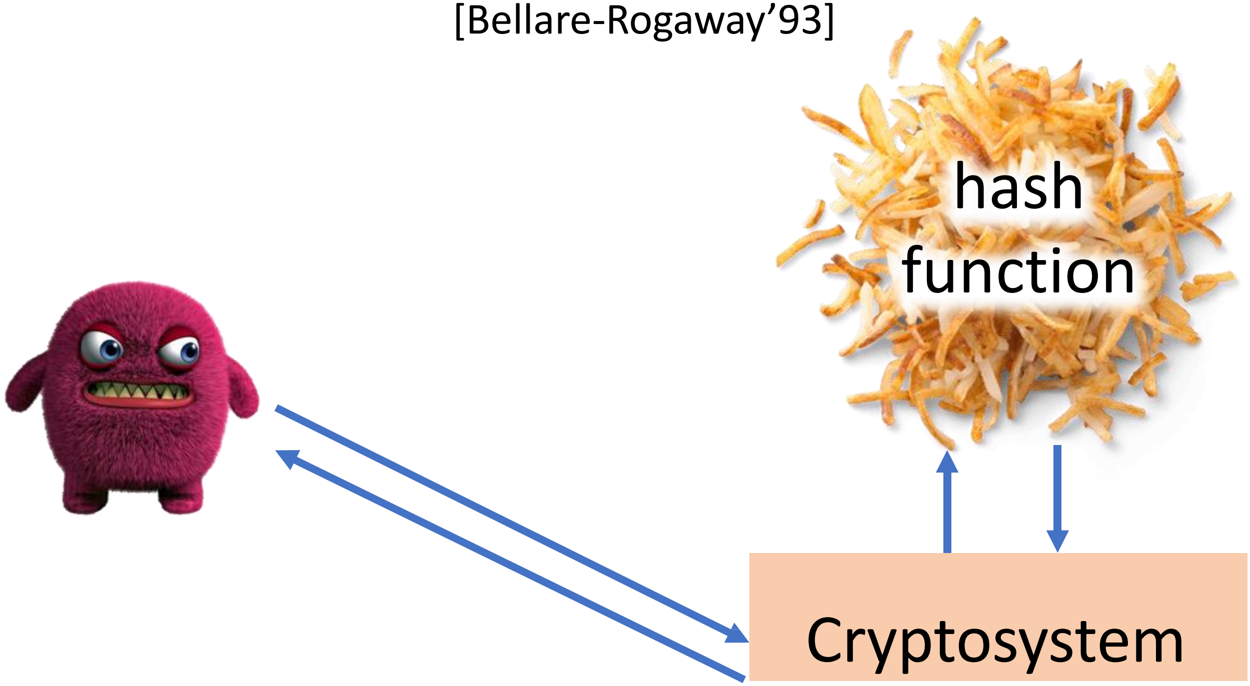
Princeton University & NTT Research




This talk
super-
position
entangled
$\mathbb{C}^2$
$e^{i2\pi/N}$
$\Sigma\alpha_x\omega_N^{xy}$
qubit
$|0\rangle$ [redacted]
$|\psi\rangle$
$H^{\otimes n}$

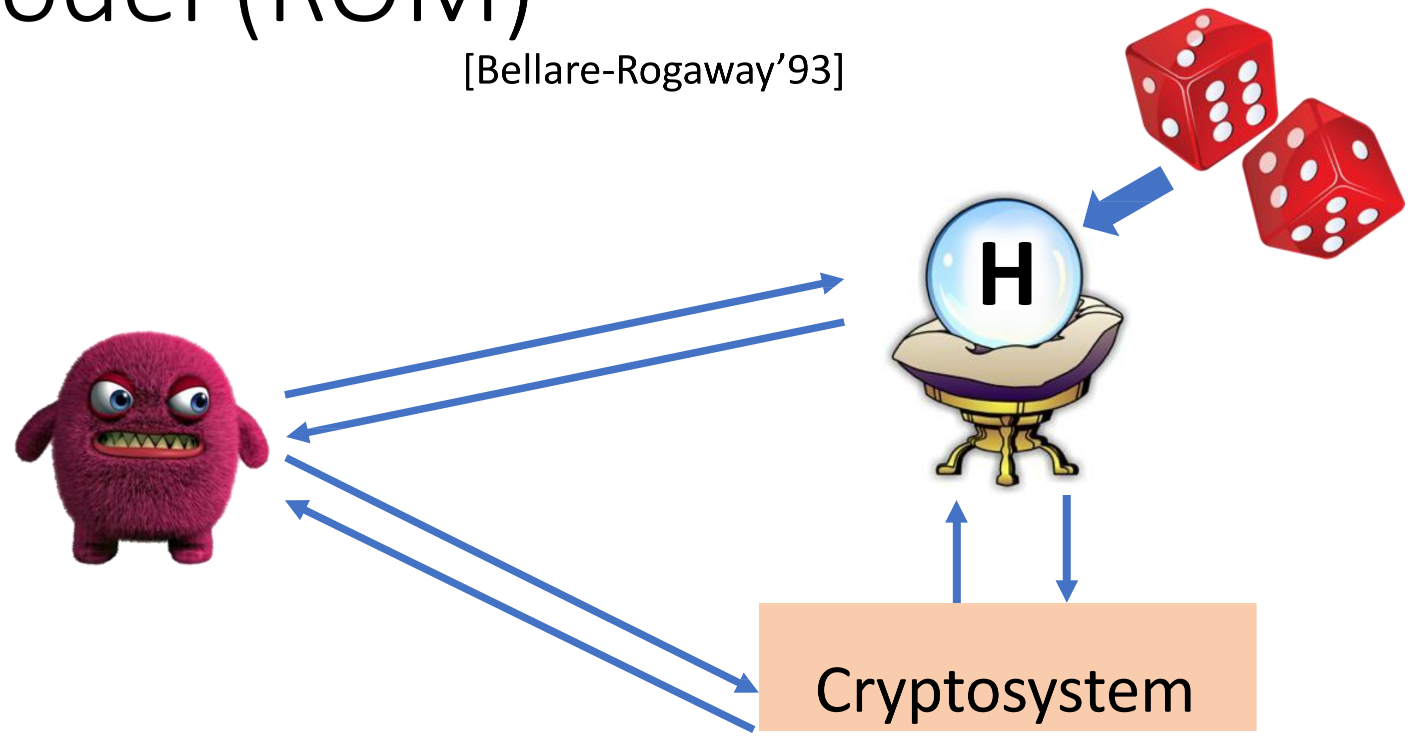
The (Classical) Random Oracle Model (ROM)

[Bellare-Rogaway’93]



The (Classical) Random Oracle Model (ROM)

[Bellare-Rogaway'93]



Typical ROM Proof: On-the-fly Simulation

Input	Output
x_1	y_1
x_2	y_2
x_3	y_3
x_4	y_4

Query(x, D):

If **(x,y)** $\in$ **D**:

Return(y,D)

Else:

y ß $ Y

D’ = D+(x,y)

Return(y,D’)

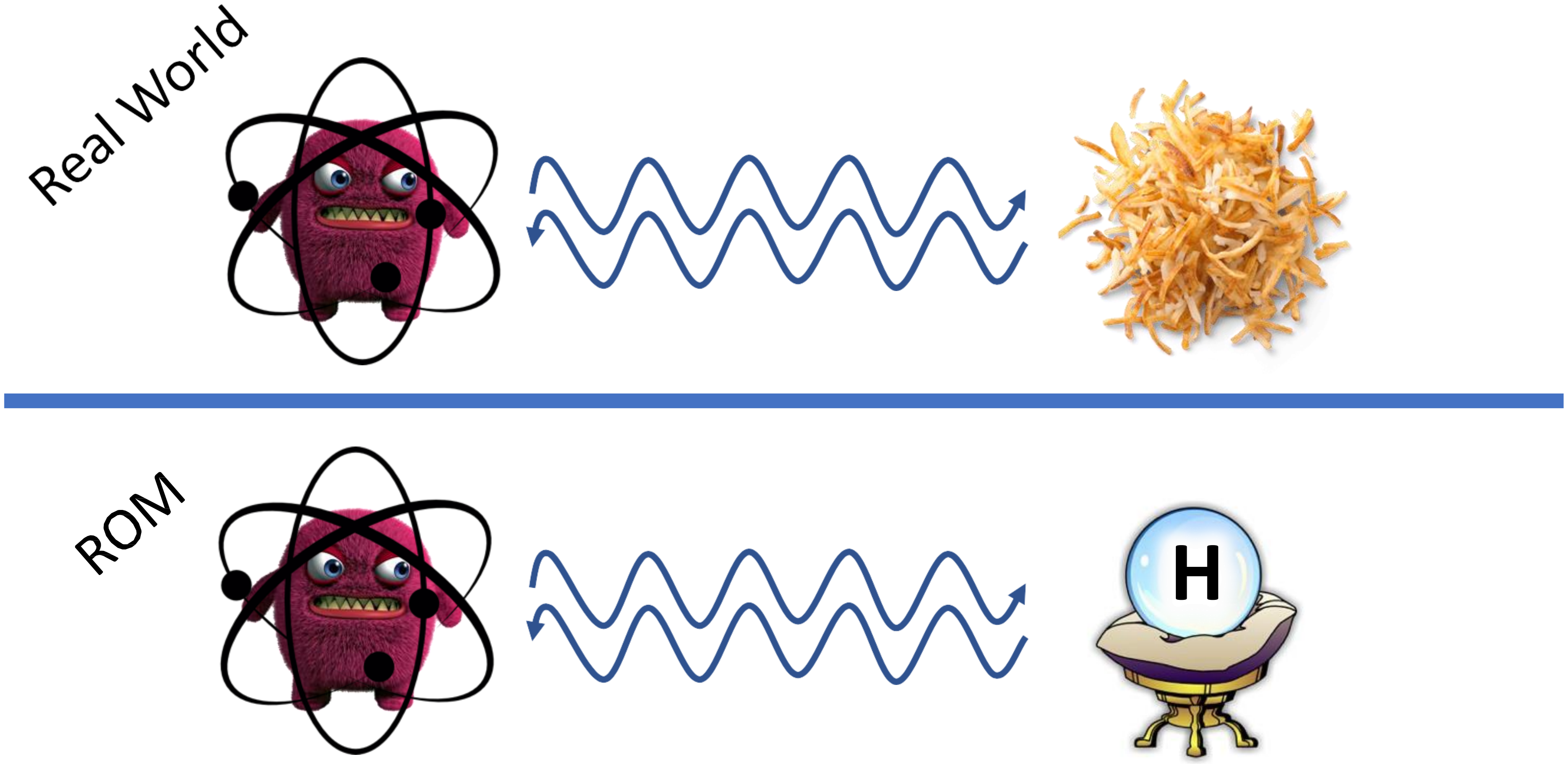
Typical ROM Proof: On-the-fly Simulation

Allows us to:

- Know the inputs adversary cares about ✓
- Know the corresponding outputs ✓
- (Adaptively) program the outputs ✓
- Easy analysis of bad events (e.g. collisions) ✓

The Quantum Random Oracle Model (QROM)

[Boneh-Dagdelen-Fischlin-Lehmann-Schaffner-Z’11]



Now standard in post-quantum crypto

Problem with Classical Proofs in QROM

How do we record the **x** values?

Input	Output
x_1	y_1
x_2	y_2
x_3	y_3
x_4	y_4

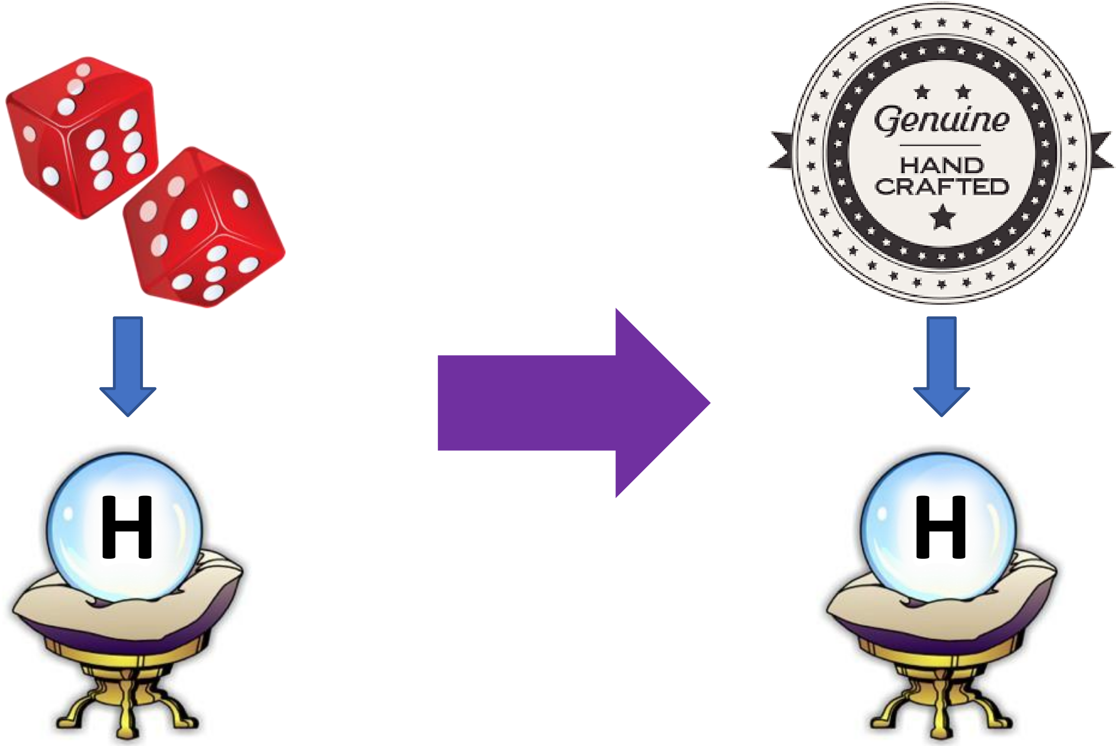
Problem with Classical Proofs in QROM

Observer Effect:
Learning anything about quantum system disturbs it

answers obliviously, so no disturbance

Reduction must answer obliviously, too?

Typical QROM Proof



H fixed once and for all at beginning

Limitations

Allows us to:

- Know the inputs adversary cares about?
- Know the corresponding outputs?
- (Adaptively) program the outputs?
- Easy analysis of bad events (e.g. collisions)?

Limitations

Allows us to:

- ~~Know the inputs adversary cares about?~~ ✘
- ~~Know the corresponding outputs?~~ ✘
- ~~(Adaptively)~~ program the outputs? ✓ / ✘
- ~~Easy analysis of bad events (e.g. collisions)?~~ ✘

Limitations

Good News: Numerous positive results (30+ papers)

Bad News: Still some major holdouts

Indifferentiable domain extension

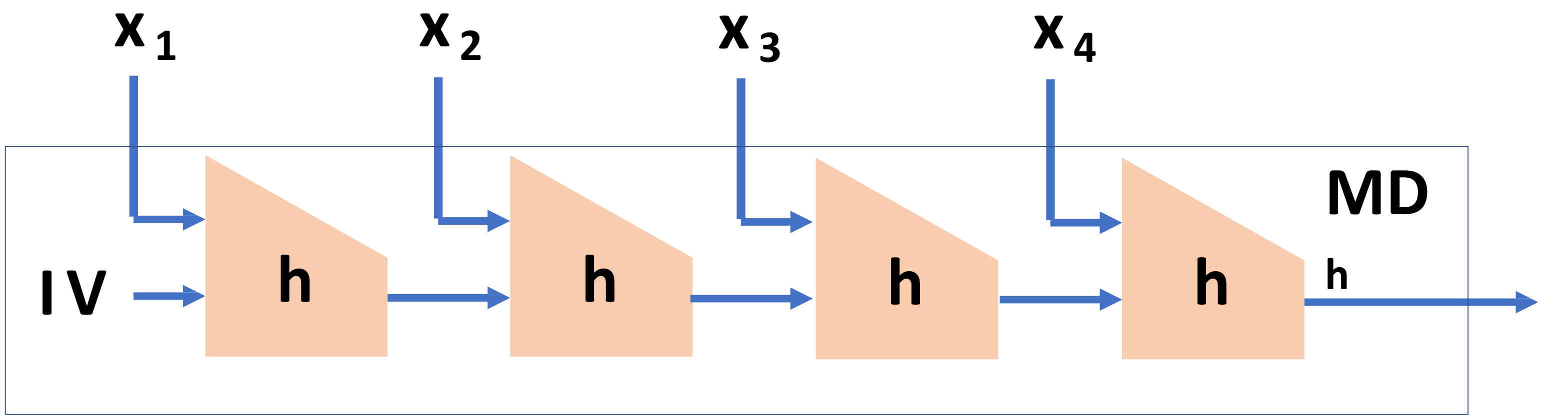
Fiat-Shamir

Luby-Rackoff

ROM è ICM

Example: Domain Extension for Random Oracles

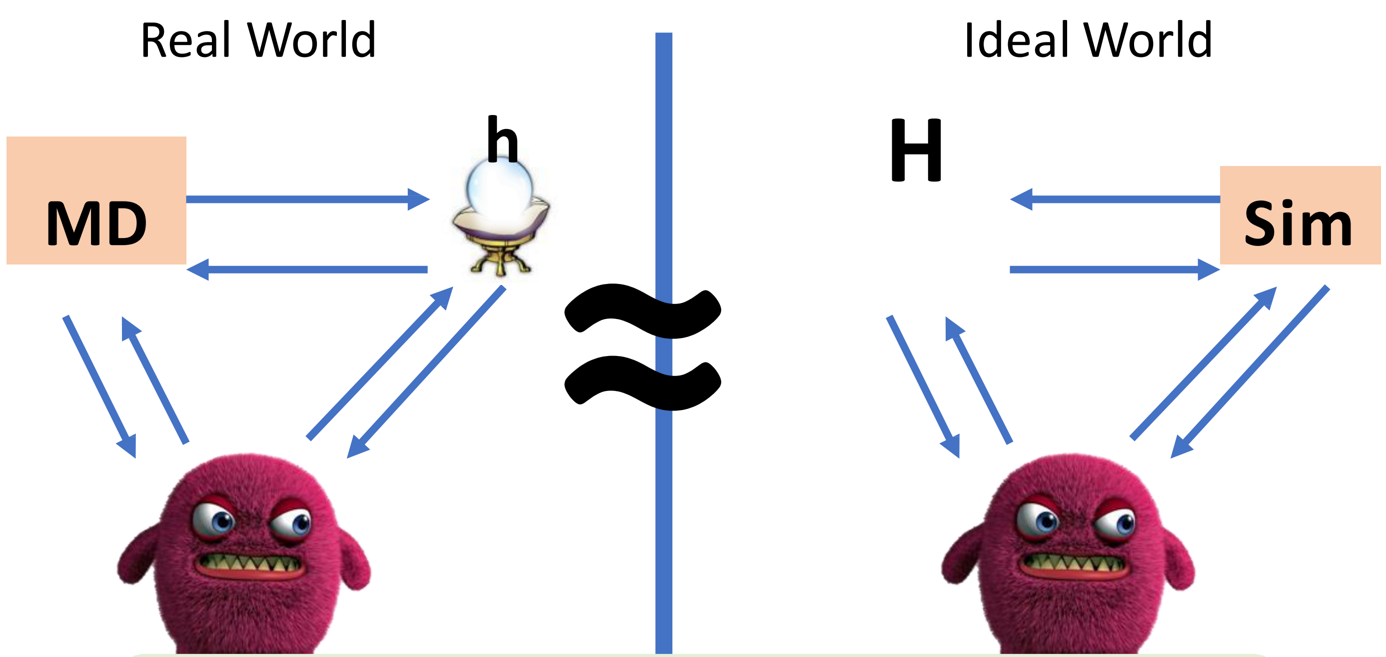
Q: Does Merkle-Damgård preserve random oracle-ness?



Example: Domain Extension for Random Oracles

A: Yes(ish) [Coron-Dodis-Malinaud-Puniya'05]

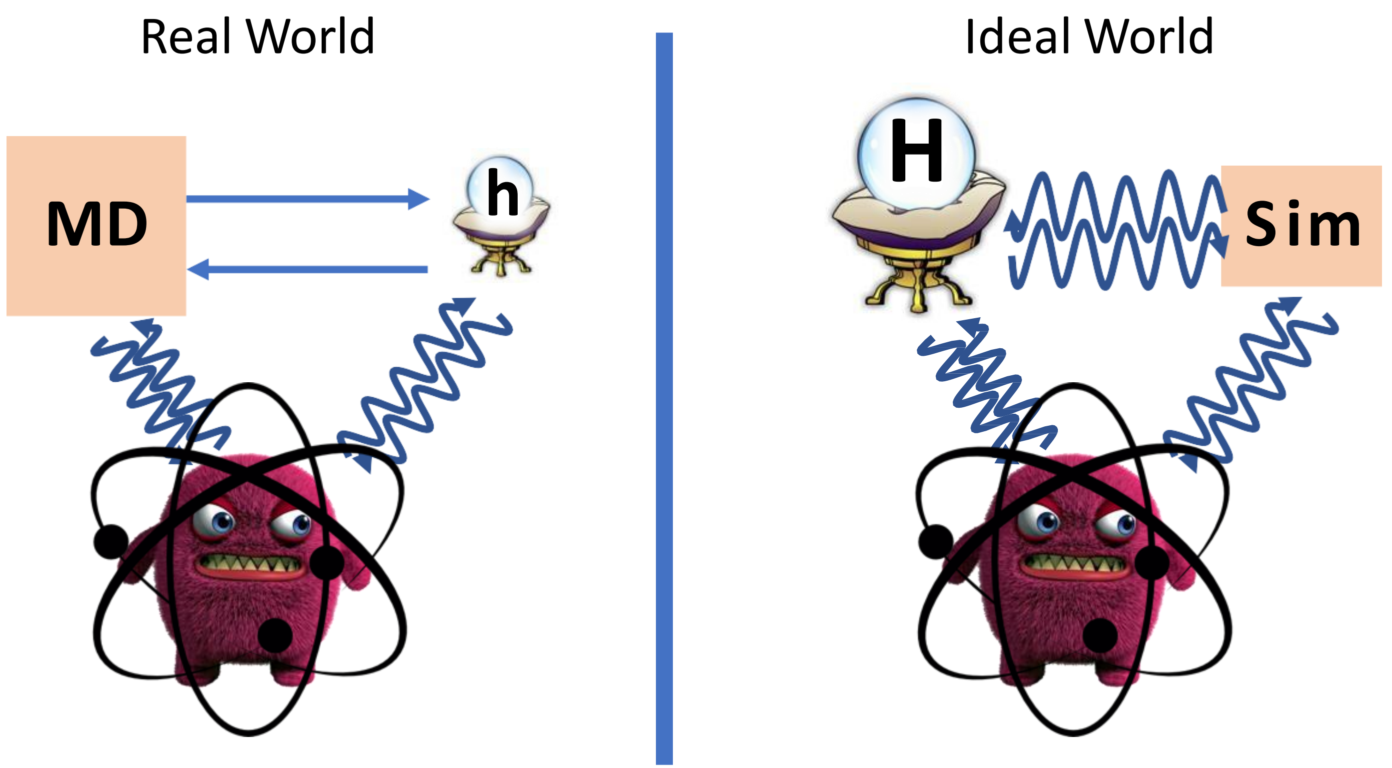
How? *Indifferentiability* [Maurer-Renner-Holenstein'04]



Thm [Ristenpart-Shacham-Shrimpton'11]:
Indifferentiability ⇒ as good as RO for "single stage games"

Quantum Indifferentiability?

Concurrently considered by [Carstens-Ebrahimi-Tabia-Unruh’18]



Quantum Indifferentiability?

Easy Thm

- Stateless simulation for domain extension is impossible, classically or quantumly

Proof idea: Compress truth table of random **H**

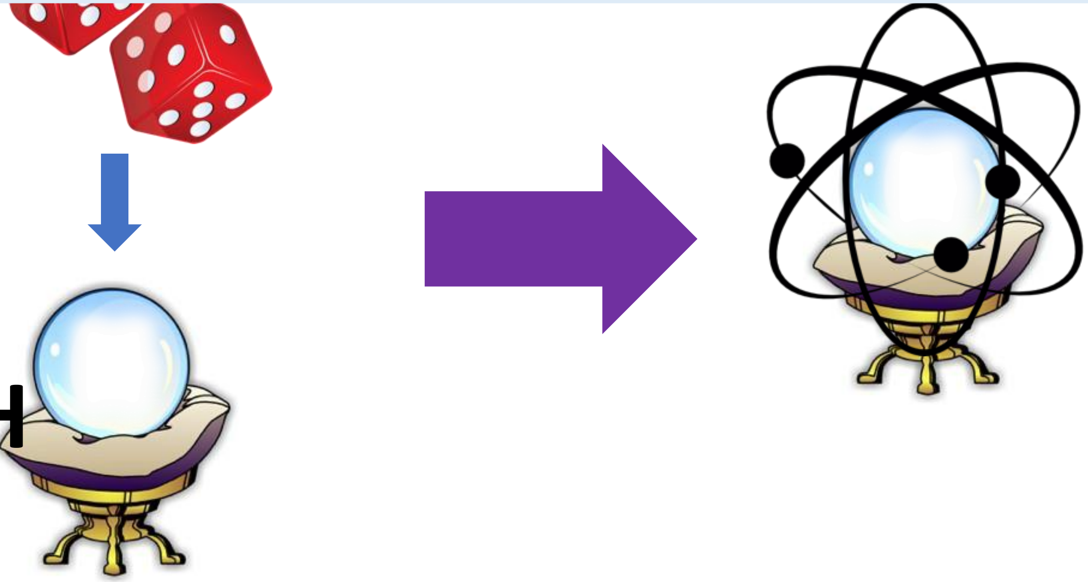
This Work:

On-the-fly simulation of quantum random oracles

(aka Compressed Oracles)

Step 1: Quantum-ify (aka Purify)

- Quantum-ifying (aka purifying) random oracle:
  - + new single quantum system



- **H**
- **H**

Reminiscent of old impossibilities for unconditional quantum protocols [Lo'97,Lo-Chau'97,Mayers'97,Nayak'99]

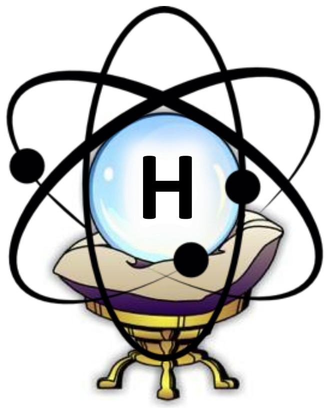
Step 1: Superposition of Oracles

Initial oracle state: 

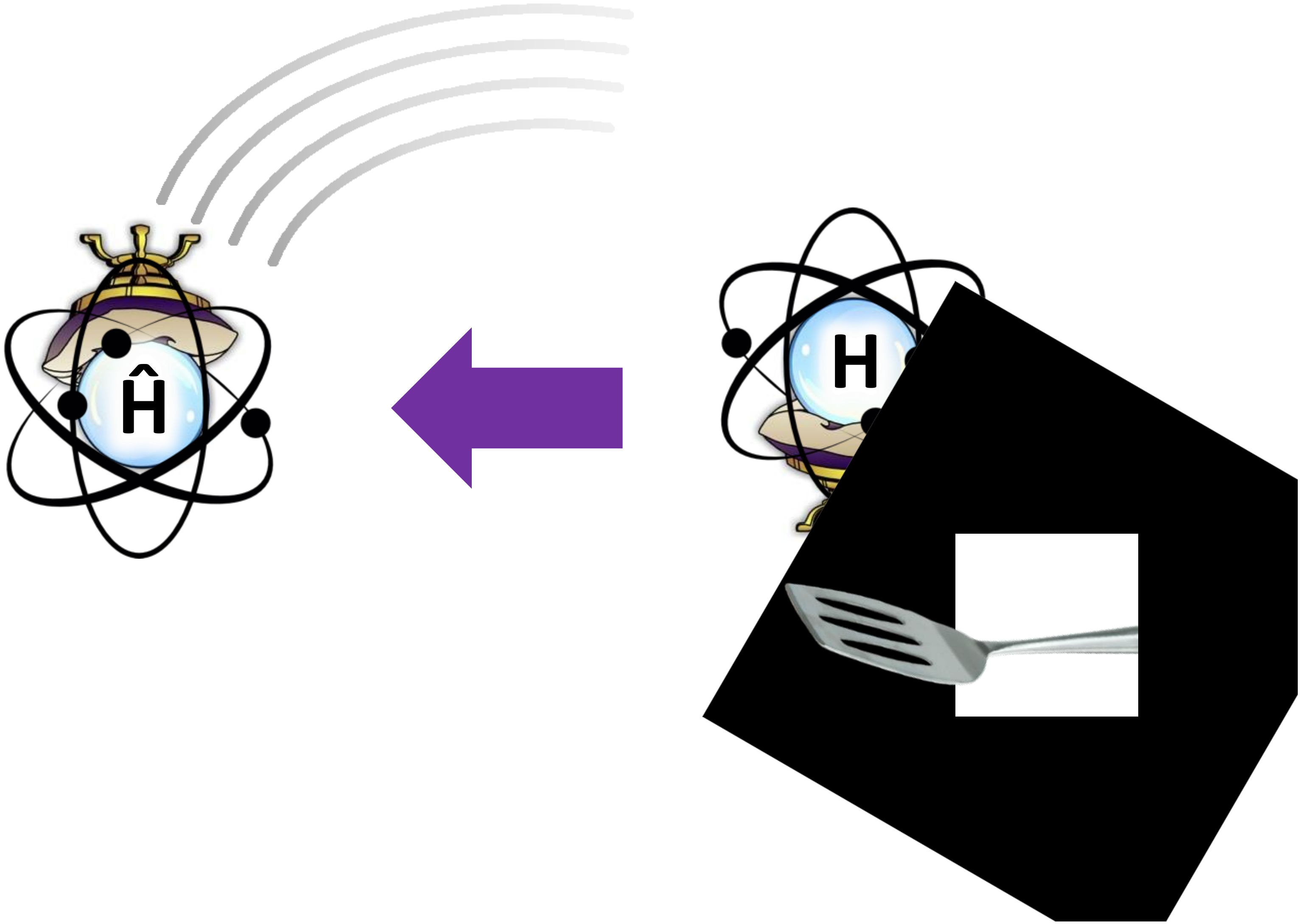
Query(x, y, H): $y = y \oplus H(x)$

Adversary's query

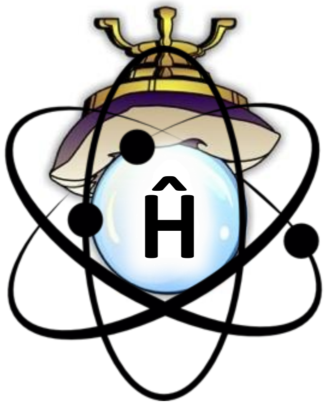
Oracle's state



Step 2: Look at Fourier Domain



Step 2: Look at Fourier Domain

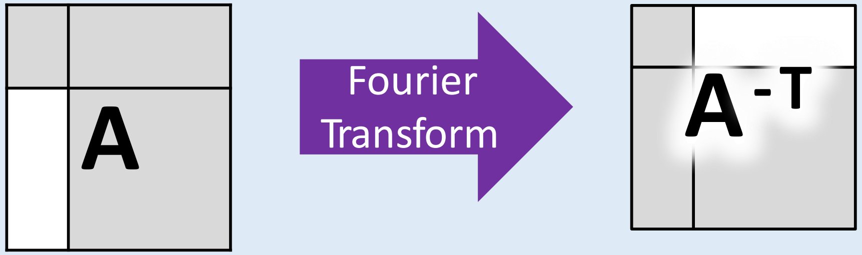


Initial oracle state: $\mathbf{Z(x) = 0}$

$\mathbf{Query(x,\ y,\ \hat{H}):\ \hat{H} = \hat{H} \oplus P_{x,y}}$

$$\mathbf{P_{x,y}(x') = \begin{cases} y & \text{if } x = x' \\ 0 & \text{else} \end{cases}}$$

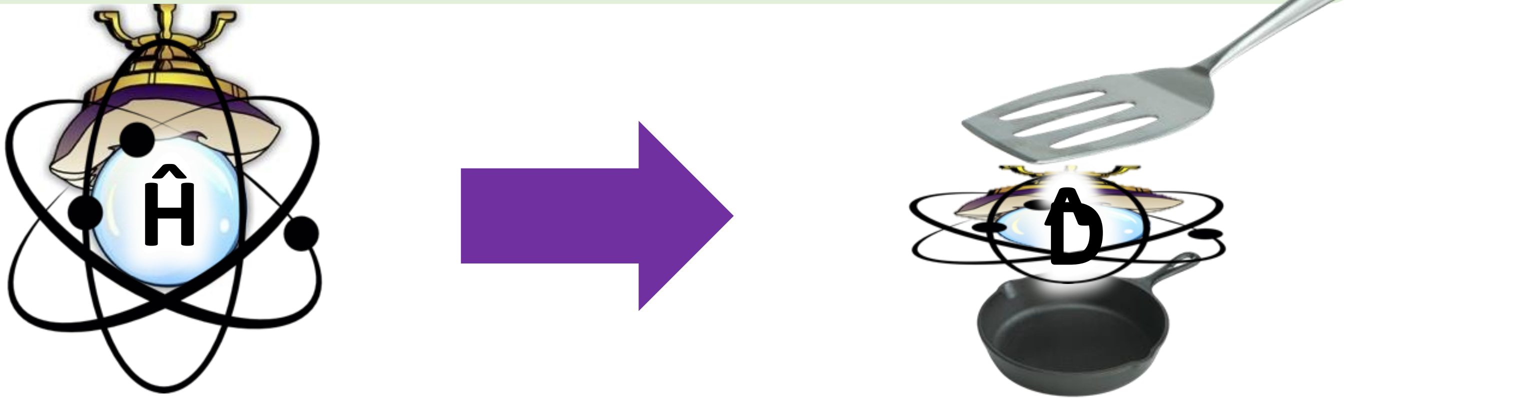
Proof:



Step 3: Compress

Observation:

After **q** queries, $\hat{\mathbf{H}}$ is non-zero on at most **q** points



Step 3: Compress

Initial oracle state: **{}**

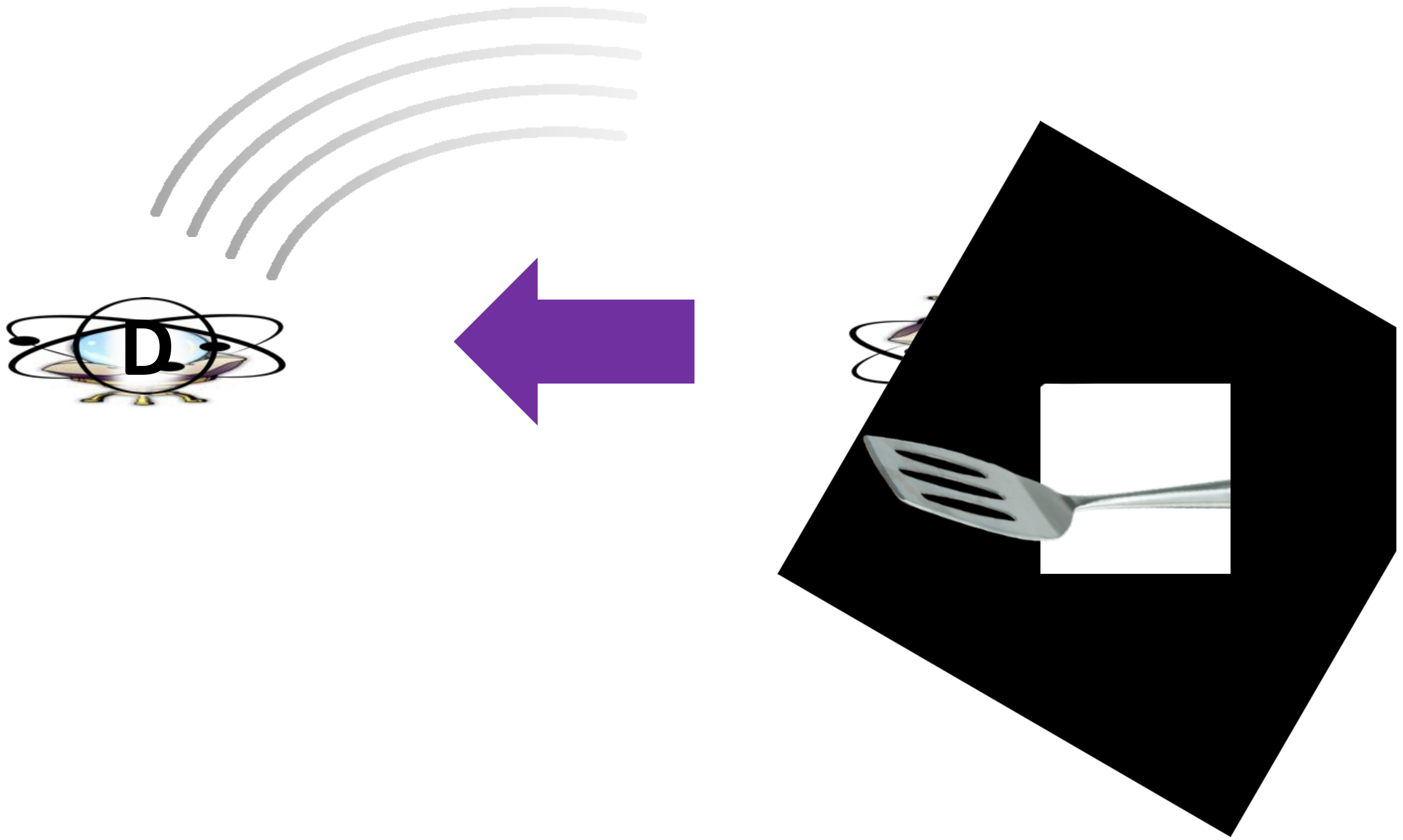
Query$(x, y, \hat{D})$:

(1) If $\nexists (x,y') \in \hat{D}$: $\hat{D} = \hat{D} + (x,0)$

(2) Replace $(x,y') \in \hat{D}$ with $(x, y' \oplus y)$

(3) If $(x,0) \in \hat{D}$: remove it

Step 4: Revert back to Primal Domain



Step 4: Revert back to Primal Domain

Input	Output
x_1	y_1
x_2	y_2
x_3	y_3
x_4	y_4

Roughly analogous to classical on-the-fly simulation

Points adversary cares about

≈Corresponding outputs

Compressed Oracles

Allows us to:

- Know the inputs adversary cares about? ✓
- Know the corresponding outputs? ✓
- ~~(Adaptively) program the outputs?~~ ✘

  Fixed by [Don-Fehr-Majenz-Schaffner’19,Liu-Z’19], later this session!
- Easy analysis of bad events (e.g. collisions)? ✓

So, what happened?

Recall…

Observer Effect:

Learning anything about quantum system disturbs it

 learns about 

through queries

gets disturbed

Compressed oracles decode such disturbance

Caveats

Outputs in database $\neq \mathbf{0}$ in Fourier domain

➡ $\mathbf{y}$ values aren't exactly query outputs

Examining $\mathbf{x},\mathbf{y}$ values perturbs state

➡ Still must be careful about how we use them

But, still good enough for many applications…

Applications In This Work

Quantum Indiff. of
Merkle-Damgård

Easily re-prove quantum lower bounds:

$\mathbf{\Omega(N^{1/2})}$ queries needed for Grover search

$\mathbf{\Omega(N^{1/3})}$ queries needed for collision finding

$\mathbf{\Omega(N^{1/(k+1)})}$ queries needed for **k**-SUM

CCA-security of plain
Fujisaki-Okamoto

Further Applications

[Alagic-Majenz-Russell-Song’18]:
Quantum-secure signature separation

[Liu-Z’19a]: Tight bounds
for multi-collision problem

[Liu-Z’19b]: Fiat-Shamir
([Don-Fehr-Majenz-Schaffner’19]: direct proof)

[Czajkowski-Majenz-Schaffner-Zur’19]:
Indifferentiability of Sponge

[Hosoyamada-Iwata’19]:
4-round Luby-Rackoff

[Chiesa-Manohar-Spooner’19]:
zk-SNARKs

[Bindel-Hamburg-Hülsing-Persichetti’19]:
Tighter CCA security proofs

Lessons Learned

Always purify your oracles!