Advanced Cryptography
CS 655

Week 14:

- Quantum Random Oracle Model

Homework 3: Due tonight at 11:59PM

Spring 2023

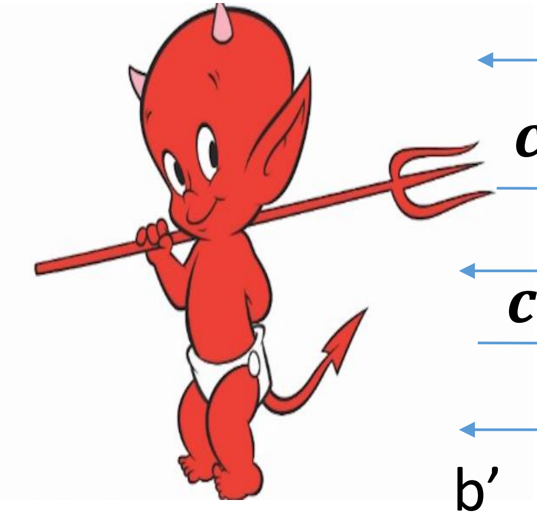
Key Encapsulation Mechanism (KEM)

- Three Algorithms
  - $\mathrm{Gen}(1^n, R)$ (Key-generation algorithm)
    - Input: Random Bits R
    - Output: $(\boldsymbol{pk}, \boldsymbol{sk}) \in \boldsymbol{\mathcal{K}}$
  - $\mathrm{Encaps}_{\mathrm{pk}}(1^n, R)$
    - Input: security parameter, random bits R
    - Output: Symmetric key $\mathrm{k} \in \{0,1\}^{\ell(n)}$ and a ciphertext c
  - $\mathrm{Decaps}_{\mathrm{sk}}(c)$ (Deterministic algorithm)
    - Input: Secret key $\mathrm{sk} \in \mathcal{K}$ and a ciphertex c
    - Output: a symmetric key$\{0,1\}^{\ell(n)}$ or ⊥ (fail)
- **Invariant**: $\mathrm{Decaps}_{\mathrm{sk}}$(c)=k whenever (c,k) = $\mathrm{Encaps}_{\mathrm{pk}}(1^n, R)$

Application: KEM

- Alice knows Bob's public key $\boldsymbol{pk_b}$ and wants to send messages $m_1, \ldots, m_k$
- Alice runs $(\mathrm{c}, \mathrm{K}) = \mathrm{Encaps}_{\boldsymbol{pk_b}}(1^n; R)$ to obtain symmetric key K
- Alice uses symmetric key to encrypt $m_1, \ldots, m_k$ i.e., $c_i = \mathrm{Enc}_{\mathrm{K}}(m_i)$
    - Example: $\mathrm{Enc}_{\mathrm{K}}$ is AES-GCM
- Alice sends $\mathrm{c}, c_1, \ldots, c_k$ to Bob
- Bob recovers $\mathrm{K} = \mathrm{Decaps}_{\boldsymbol{sk_b}}(c)$ and then can decrypt $c_1, \ldots, c_k$ to obtain $m_1, \ldots, m_k$ i.e., $m_i = \mathrm{Dec}_{\mathrm{K}}(c_i)$

KEM CCA-Security ($\mathrm{KEM}^{\mathrm{cca}}_{\mathrm{A},\Pi}(\mathrm{n})$)



$(\boldsymbol{pk}, \boldsymbol{c}, \boldsymbol{kb})$

⋮

$\boldsymbol{c_1} \neq \boldsymbol{c}$

$\mathbf{Decaps}_{\boldsymbol{sk}}(\boldsymbol{c_1})$

$\boldsymbol{c_2} \neq \boldsymbol{c}$

$\mathbf{Decaps}_{\boldsymbol{sk}}(\boldsymbol{c_2})$

b’

⋮

Random bit b

(pk,sk) = Gen(.)

$(\boldsymbol{c}, \boldsymbol{k_0}) = \mathbf{Encaps}_{\boldsymbol{pk}}(.)$

$\boldsymbol{k_1} \longleftarrow \{\mathbf{0}, \mathbf{1}\}^{\boldsymbol{n}}$

$$\forall PPT\ A\ \exists \mu\ (\text{negligible})\ \text{s.t}$$

$$\Pr\left[\mathrm{KEM}^{\mathrm{cca}}_{\mathrm{A},\Pi} = 1\right] \leq \frac{1}{2} + \mu(n)$$

KEM from RSA

- Recap: CCA-Secure KEM from RSA in Random Oracle Model

- RSA yields CPA-Secure KEM in Random Oracle Model
  - $\left(\boldsymbol{c = r^e mod\ N, K = H(r)}\right) \leftarrow \mathbf{Encaps_{pk}}(\mathbf{1}^{\boldsymbol{n}}; \boldsymbol{R})$ and $\mathbf{Decaps_{sk}}(\boldsymbol{c}) = \boldsymbol{H}\left(\boldsymbol{c^d mod\ N}\right)$
- **Security based on RSA-Inversion assumption**

- **Post-Quantum Security?**
  - **Shor's Algorithm breaks RSA by factoring N**
  - **Is random oracle model valid for quantum attacker?**

Trapdoor Permutation

- Three Algorithms
  - $\mathrm{Gen}(1^n, R)$ (Key-generation algorithm)
    - Input: Random Bits R
    - Output: $(\boldsymbol{pk}, \boldsymbol{sk}) \in \boldsymbol{\mathcal{K}}$
  - $y = \mathrm{Eval}_{\mathrm{pk}}(x)$ (Deterministic algorithm)
    - Input: x and public key pk; Output: y
  - $\mathrm{Rev}_{\mathrm{sk}}(y)$ (Deterministic algorithm)
    - Input: Secret trapdoor key $\mathrm{sk} \in \mathcal{K}$ and a ciphertex c
    - Output: x
- **Invariant**: $\mathrm{Rev}_{\mathrm{sk}}(\mathrm{Eval}_{\mathrm{pk}}(x))$=x whenever $(\boldsymbol{pk}, \boldsymbol{sk}) = \mathrm{Gen}(1^n, R)$
- **Security Game:** Challenger picks $(\boldsymbol{pk}, \boldsymbol{sk}) = \mathrm{Gen}(1^n, R)$ and generates random x. Attacker gets $\boldsymbol{pk}$ and $\mathrm{Eval}_{\mathrm{pk}}(x)$. Attacker tries to recover x.

KEM from Trapdoor Permutation

- CCA-Secure KEM from any trapdoor permutation in Random Oracle Model
- $\left(\boldsymbol{c} = \mathrm{Eval}_{\mathrm{pk}}(r), \boldsymbol{K} = \boldsymbol{H}(\boldsymbol{r})\right) \leftarrow \mathbf{Encaps}_{\mathbf{pk}}(\mathbf{1}^{\boldsymbol{n}}; \boldsymbol{R})$ and
- $\mathbf{Decaps}_{\mathbf{sk}}(\boldsymbol{c}) = \boldsymbol{H}(\mathrm{Rev}_{\mathrm{sk}}(c))$

- **Security proof in random oracle model**
  - **Any KEM attacker can break security of trapdoor permutation.**

KEM Security Reduction

- $\left(\boldsymbol{c} = \mathrm{Eval}_{\mathrm{pk}}(r)\,, \boldsymbol{K} = \boldsymbol{H}(\boldsymbol{r})\right) \leftarrow \mathbf{Encaps}_{\mathbf{pk}}(\mathbf{1}^{\boldsymbol{n}}; \boldsymbol{R})$ and
- $\mathbf{Decaps}_{\mathbf{sk}}(\boldsymbol{c}) = \boldsymbol{H}(\mathrm{Rev}_{\mathrm{sk}}(c))$

- Given KEM attacker A define Trapdoor Permutation attacker B
- B is given $\boldsymbol{pk}$ and $\mathrm{Eval}_{\mathrm{pk}}(r)$ as input
  - B simulates KEM challenger and generates $(\boldsymbol{c} = \mathrm{Eval}_{\mathrm{pk}}(r))$ and a random key $\boldsymbol{K}$
  - B sends (pk, c,$\boldsymbol{K}$) to KEM attacker A and begins simulating A.
  - For each random oracle query x_i made by A, B checks to see if $\boldsymbol{c} = \mathrm{Eval}_{\mathrm{pk}}(x_i)$; if so we have found $r = x_i$
  - B keeps track of all of A's random oracle queries $x_1, \dots x_q$ and programs random responses $r_1, \dots r_q$.
    - Caveat: If $\mathrm{Eval}_{\mathrm{pk}}(x_j) = \boldsymbol{c_i}$ for some previous query to decaps then return the associated key K_i.
  - When A queries the $\mathbf{Decaps}_{\boldsymbol{sk}}\,(\boldsymbol{c_i})$ oracle on input $\boldsymbol{c_i}$ we check to see if $\boldsymbol{c_i} = \mathrm{Eval}_{\mathrm{pk}}(x_j)$ for some j. If so we return the associated key $r_j = H(x_j)$. If not return a random key K_i.

KEM Security Reduction

- $\left(\boldsymbol{c} = \mathrm{Eval}_{\mathrm{pk}}(r), \boldsymbol{K} = \boldsymbol{H}(\boldsymbol{r})\right) \leftarrow \mathbf{Encaps}_{\mathbf{pk}}(\mathbf{1}^{\boldsymbol{n}}; \boldsymbol{R})$ and
- $\mathbf{Decaps}_{\mathbf{sk}}(\boldsymbol{c}) = \boldsymbol{H}(\mathrm{Rev}_{\mathrm{sk}}(c))$

- Given KEM attacker A define Trapdoor Permutation attacker B
- B is given $\boldsymbol{pk}$ and $\mathrm{Eval}_{\mathrm{pk}}(r)$ as input
  - B simulates KEM challenger and generates $(\boldsymbol{c} = \mathrm{Eval}_{\mathrm{pk}}(r))$, a random bit b and a random keys $\boldsymbol{K}$
  - B sends (pk, c, $\boldsymbol{K_b}$) to KEM attacker A and begins simulating A.
  - For each random oracle query x_i made by A, B checks to see if $\boldsymbol{c} = \mathrm{Eval}_{\mathrm{pk}}(x_i)$; if so we have found $r = x_i$
  - B keeps track of all of A's random oracle queries $x_1, \ldots x_q$ and programs random responses $r_1, \ldots r_q$.
    - Caveat: If $\mathrm{Eval}_{\mathrm{pk}}(x_j) = \boldsymbol{c_i}$ for some previous query to decaps then return the associated key K_i.
  - When A queries the $\mathbf{Decaps}_{\boldsymbol{sk}}\ (\boldsymbol{c_i})$ oracle on input $\boldsymbol{c_i}$ we check to see if $\boldsymbol{c_i} = \mathrm{Eval}_{\mathrm{pk}}(x_j)$ for some j. If so we return the associated key $r_j = H(x_j)$. If not return a random key K_i.

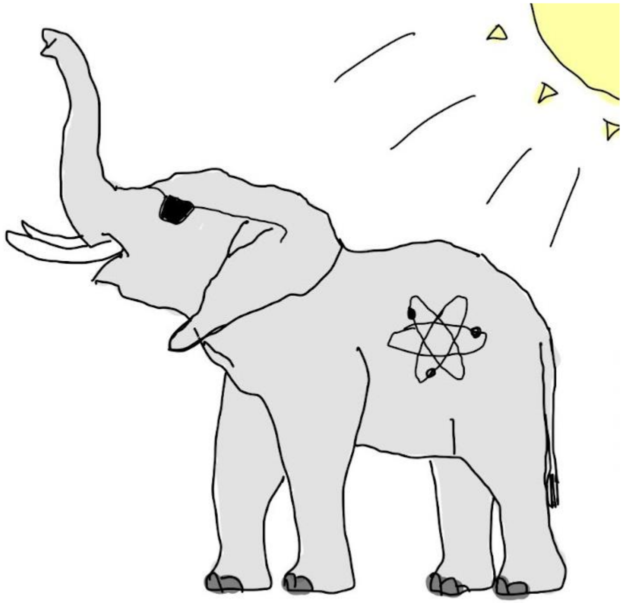
- Analysis Sketch: If A does not query $\boldsymbol{H}(\boldsymbol{r})$ then it has no advantage in original KEM game. ➔ Successful KEM attacker will query $\boldsymbol{H}(\boldsymbol{r})$ with non-negligible probability. ➔ B wins trapdoor inversion game with non-negligible probability.

KEM from Trapdoor Permutation

- CCA-Secure KEM from any trapdoor permutation in Random Oracle Model
- $\left(\boldsymbol{c} = \mathrm{Eval}_{\mathrm{pk}}(r)\,, \boldsymbol{K} = \boldsymbol{H}(\boldsymbol{r})\right) \leftarrow \mathbf{Encaps}_{\mathbf{pk}}(\mathbf{1}^{\boldsymbol{n}}; \boldsymbol{R})$ and
- $\mathbf{Decaps}_{\mathbf{sk}}(\boldsymbol{c}) = \boldsymbol{H}(\mathrm{Rev}_{\mathrm{sk}}(c))$

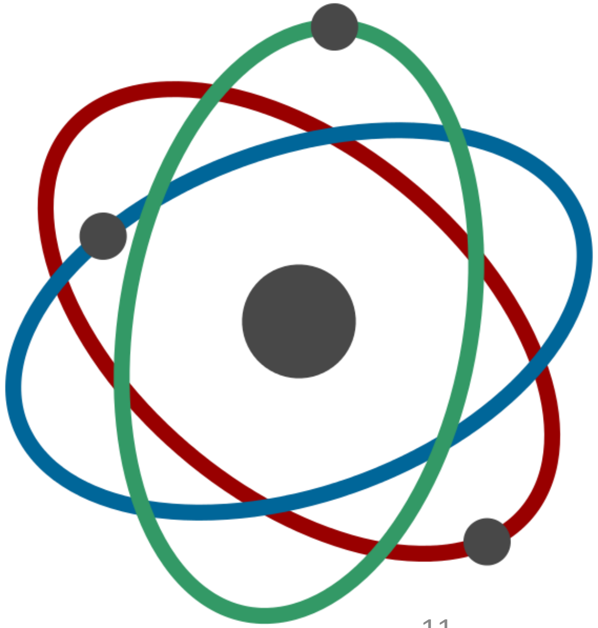
- **Security proof in random oracle model**
  - **Any KEM attacker can break security of trapdoor permutation.**
- **Post-Quantum Security?**
  - **Assume trapdoor permutation is PQ-safe e.g., based on LWE, Lattices etc…**
  - **Does reduction in classical ROM imply PQ-security?**

Elephant in the Room?

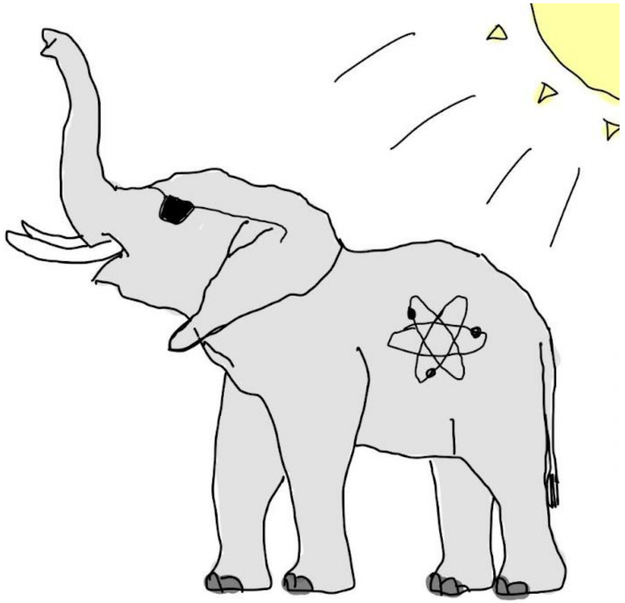


- Shor’s Factoring Algorithm
  - Breaks: RSA-OAEP, RSA-FDH, Pallier….
  - Solves Discrete Log
  - Breaks: El-Gamal, EC-DSA, Schnorr Signatures,…

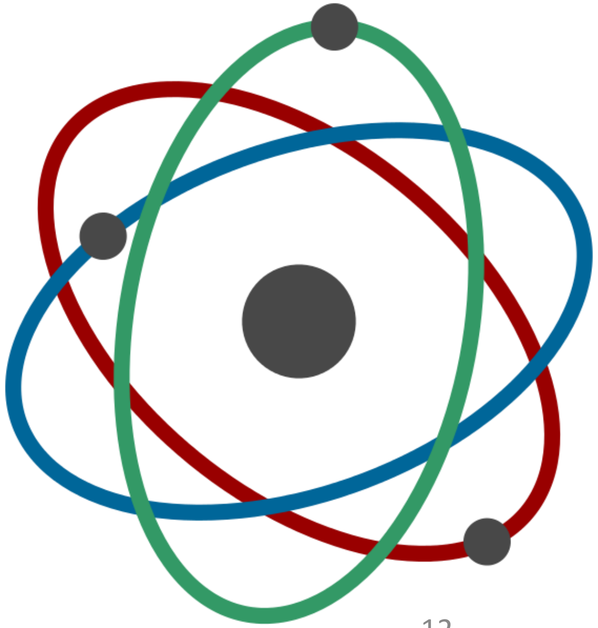
- Grover’s Algorithm
  - **Function Inversion:** Given $H:\{0,1\}^n \rightarrow \{0,1\}^n$ and $\mathrm{y} = H(x)$ find x' such that $\mathrm{y} = H(x')$
  - **Classical random oracle model:** requires $\Omega(2^n)$ queries
  - **Grover’s Search:** Runs in time $O\left(2^{n/2}\right)$



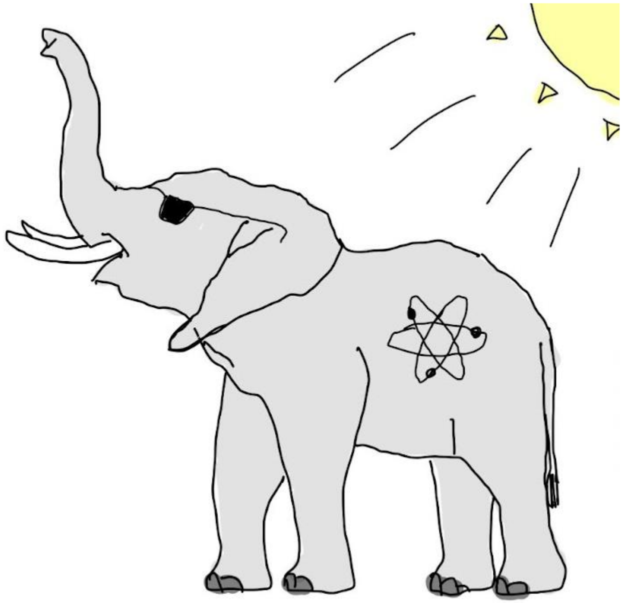
Elephant in the Room?



- Shor's Factoring Algorithm
  - Breaks: RSA-OAEP, RSA-FDH, Pallier….
  - Solves Discrete Log
  - Breaks: El-Gamal, EC-DSA, Schnorr Signatures,…
  - Basically, most deployed public key crypto
- NIST PQC Competition
  - Public Key Encryption (PKE): Crystals-Kyber
    - **Hardness:** Learning With Errors (LWE) (Specifically: Module-LWE)
    - Integration in Crypto Libraries: Cloundfare (CIRCL), Amazon (AWS Key Management), IBM
  - Digital Signatures: Three Winners
    - Crystals-Dilithium
    - Falcon (Lattice Based Signatures):
      - Hardness: Short Integer Solution (SIS) over NTRU Lattices
    - SPHINCS+ (Hash Based Construction)

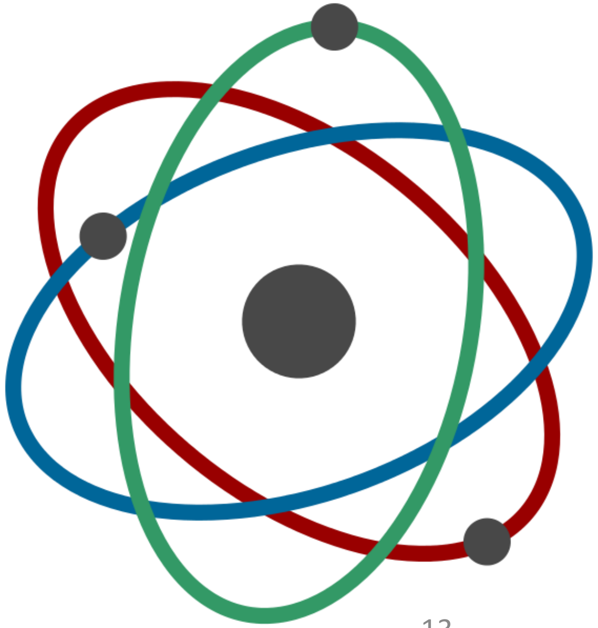


Random Oracle Model?



- Heuristic justification for Random Oracle Model
  - Security proof rules out ``generic attacks" that use a hash function like SHA3 as a black box.
  - Hash functions such as SHA3 are incredibly well designed ➔ it is difficult for an attacker to do anything but run the code for SHA3 in a black box manner…
  - **Experience:** Security proof in ROM seems to imply security in practice.

- Grover's Algorithm
  - **Function Inversion:** Given $H: \{0,1\}^n \rightarrow \{0,1\}^n$ and $\mathrm{y} = H(x)$ find x' such that $\mathrm{y} = H(x')$
  - **Classical random oracle model:** requires $\Omega(2^n)$ queries
  - **Grover's Search:** Runs in time $O\left(2^{n/2}\right)$
  - Grover's search actually uses hash function in blackbox manner!
  - What gives?



Quantum Computation (Basics)

- **Classical State (bits):** $x \in \{0,1\}^n$
- Quantum State (qubits) superposition

$$\varphi = \sum_{x \in \{0,1\}^n} \alpha_x |x\rangle$$

- **Amplitudes:** α_x is a complex number $\alpha_x = a + bi$ with magnitude

$$|\alpha_x| = \sqrt{a^2 + b^2} \Rightarrow |\alpha_x|^2 = a^2 + b^2$$

- **Measurement (in standard basis):** observe x with probability $|\alpha_x|^2$ ➔ state φ collapses to $|x\rangle$
- Sum of squared amplitudes is always 1

$$\sum_{x \in \{0,1\}^n} |\alpha_x|^2 = 1$$

Quantum Computation (Basics)

- **Classical State (bits):** $x \in \{0,1\}^n$
- Quantum State (qubits) superposition

$$\varphi = \sum_{x \in \{0,1\}^n} \alpha_x |x\rangle$$

- **Amplitudes:** α_x is a complex number $\alpha_x = a + bi$ with magnitude

$$|\alpha_x| = \sqrt{a^2 + b^2} \Rightarrow |\alpha_x|^2 = a^2 + b^2$$

- **Partial Measurement (Example): Measure first qubit**
  - observe 1 with probability $\sum_{x \in \{0,1\}^{n-1}} \alpha_{1x} |\alpha_x|^2$ ➔ state φ collapses to $c_1 \sum_{x \in \{0,1\}^{n-1}} \alpha_{1x} |x\rangle$
  - observe 0 with probability $\sum_{x \in \{0,1\}^{n-1}} \alpha_{0x} |\alpha_x|^2$ ➔ state φ collapses to $c_0 \sum_{x \in \{0,1\}^{n-1}} \alpha_{0x} |x\rangle$
- Sum of squared amplitudes is always 1

$$c_1 \sum_{x \in \{0,1\}^{n-1}} |\alpha_{1x}|^2 = 1 \quad and \quad c_0 \sum_{x \in \{0,1\}^{n-1}} |\alpha_{0x}|^2 = 1$$

Quantum Measurement (Basics)

- **Quantum (Partial) Measurement:** Necessarily alters the quantum state

Quantum Measurement (Basics)

- **Quantum (Partial) Measurement:** Necessarily alters the quantum state
- **Idea:** Replicate the state and measure the copy?
- Impossible! No-Cloning Theorem ➔ Impossible to create an independent and identical copy of an arbitrary/unknown quantum state.

Quantum Computation (Basics)

- Quantum Gate (unitary transform): $U_i|\varphi_i\rangle \Rightarrow |\varphi_{i+1}\rangle$
  - Unitary Transform: UU* = U*U = I (identity)   where U* is conjugate transpose
  - **Implication:** Quantum Computation is Invertible:

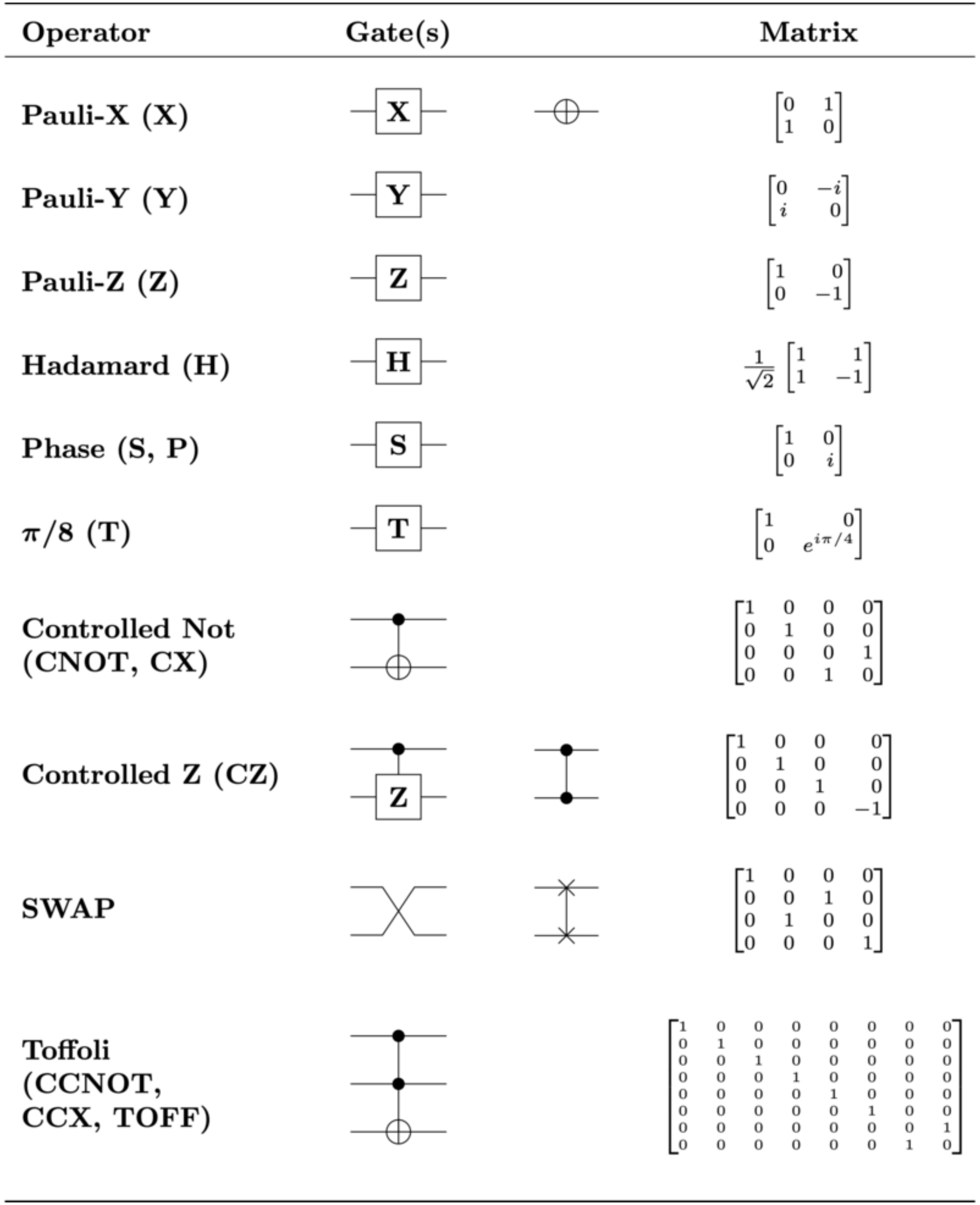
$$U_i^*|\varphi_{i+1}\rangle = U_i^*(U_i|\varphi_i\rangle) \ = |\varphi_i\rangle$$

- Quantum Logic Gates
  - Hadamard
  - (Controled Not) CNOT
  - CCNOT

Hadamard (Single Bit)

$$H|0\rangle \Rightarrow \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$$

$$H|1\rangle \Rightarrow \frac{1}{\sqrt{2}}|1\rangle - \frac{1}{\sqrt{2}}|1\rangle$$



Operator	Gate(s)		Matrix
Pauli-X (X)	X	⊕	$\begin{bmatrix}0 & 1\\1 & 0\end{bmatrix}$
Pauli-Y (Y)	Y		$\begin{bmatrix}0 & -i\\i & 0\end{bmatrix}$
Pauli-Z (Z)	Z		$\begin{bmatrix}1 & 0\\0 & -1\end{bmatrix}$
Hadamard (H)	H		$\frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\1 & -1\end{bmatrix}$
Phase (S, P)	S		$\begin{bmatrix}1 & 0\\0 & i\end{bmatrix}$
$\pi/8$ (T)	T		$\begin{bmatrix}1 & 0\\0 & e^{i\pi/4}\end{bmatrix}$
Controlled Not (CNOT, CX)			$\begin{bmatrix}1 & 0 & 0 & 0\\0 & 1 & 0 & 0\\0 & 0 & 0 & 1\\0 & 0 & 1 & 0\end{bmatrix}$
Controlled Z (CZ)	Z		$\begin{bmatrix}1 & 0 & 0 & 0\\0 & 1 & 0 & 0\\0 & 0 & 1 & 0\\0 & 0 & 0 & -1\end{bmatrix}$
SWAP			$\begin{bmatrix}1 & 0 & 0 & 0\\0 & 0 & 1 & 0\\0 & 1 & 0 & 0\\0 & 0 & 0 & 1\end{bmatrix}$
Toffoli (CCNOT, CCX, TOFF)			$\begin{bmatrix}1 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\0 & 1 & 0 & 0 & 0 & 0 & 0 & 0\\0 & 0 & 1 & 0 & 0 & 0 & 0 & 0\\0 & 0 & 0 & 1 & 0 & 0 & 0 & 0\\0 & 0 & 0 & 0 & 1 & 0 & 0 & 0\\0 & 0 & 0 & 0 & 0 & 1 & 0 & 0\\0 & 0 & 0 & 0 & 0 & 0 & 0 & 1\\0 & 0 & 0 & 0 & 0 & 0 & 1 & 0\end{bmatrix}$

Hadamard (Multiple Bits)

Qubit 1: $|0\rangle \rightarrow H \rightarrow \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$

Qubit 2: $|0\rangle \rightarrow H \rightarrow \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$

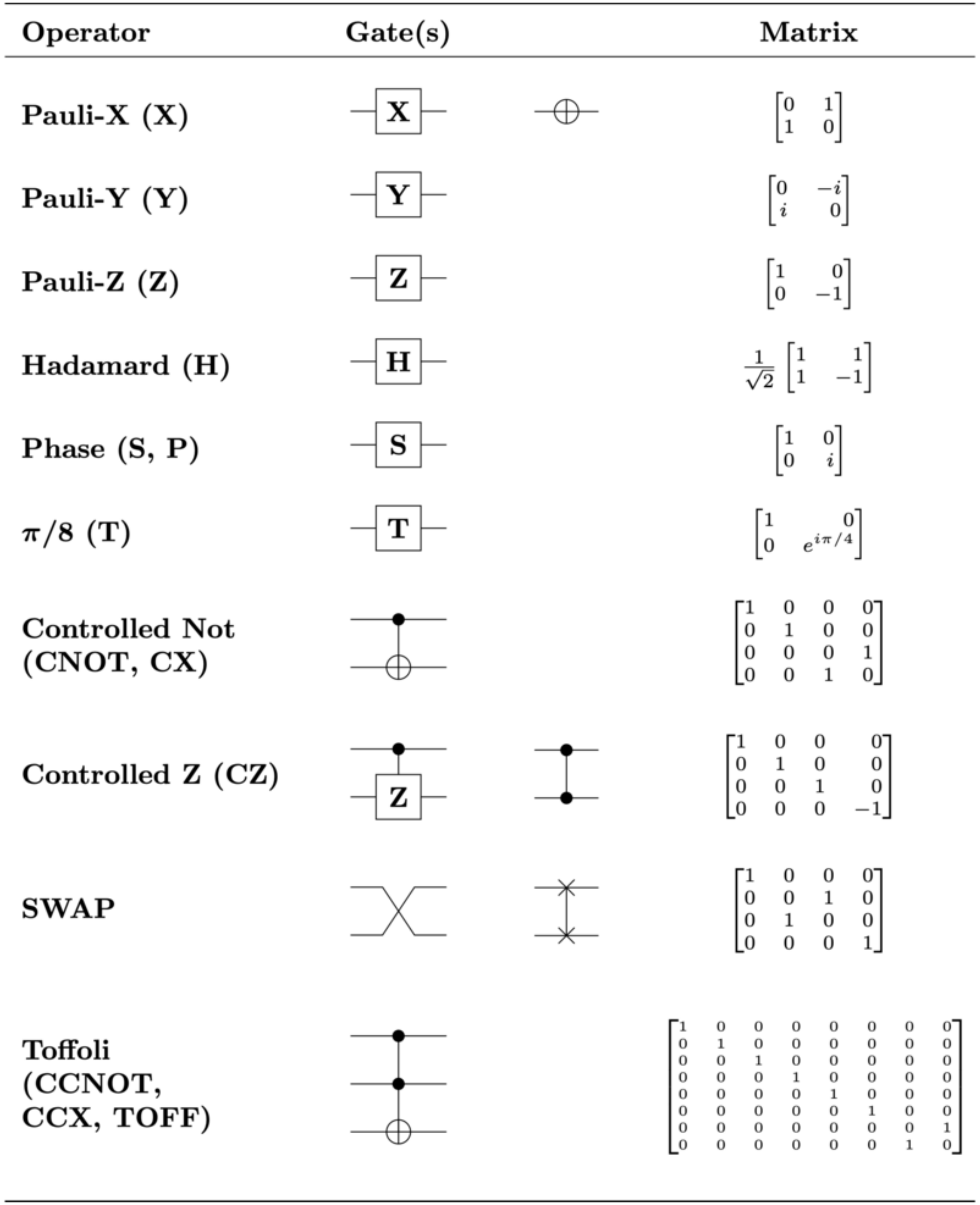
…

Qubit n: $|0\rangle \rightarrow H \rightarrow \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$

All at once

$$|0^n\rangle \rightarrow H^{\otimes n} \rightarrow \sum_{x\in\{0,1\}^n} \sqrt{2^{-n}}|x\rangle$$

(Uniform over all bitstrings)



Operator	Gate(s)		Matrix
Pauli-X (X)	X	⊕	$\begin{bmatrix}0 & 1\\1 & 0\end{bmatrix}$
Pauli-Y (Y)	Y		$\begin{bmatrix}0 & -i\\i & 0\end{bmatrix}$
Pauli-Z (Z)	Z		$\begin{bmatrix}1 & 0\\0 & -1\end{bmatrix}$
Hadamard (H)	H		$\frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\1 & -1\end{bmatrix}$
Phase (S, P)	S		$\begin{bmatrix}1 & 0\\0 & i\end{bmatrix}$
$\pi/8$ (T)	T		$\begin{bmatrix}1 & 0\\0 & e^{i\pi/4}\end{bmatrix}$
Controlled Not (CNOT, CX)	● ⊕		$\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{bmatrix}$
Controlled Z (CZ)	● Z	● ●	$\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&-1\end{bmatrix}$
SWAP		× ×	$\begin{bmatrix}1&0&0&0\\0&0&1&0\\0&1&0&0\\0&0&0&1\end{bmatrix}$
Toffoli (CCNOT, CCX, TOFF)	● ● ⊕		$\begin{bmatrix}1&0&0&0&0&0&0&0\\0&1&0&0&0&0&0&0\\0&0&1&0&0&0&0&0\\0&0&0&1&0&0&0&0\\0&0&0&0&1&0&0&0\\0&0&0&0&0&1&0&0\\0&0&0&0&0&0&0&1\\0&0&0&0&0&0&1&0\end{bmatrix}$

Quantum Oracle

- Consider real world instantiation of function $F: \{0,1\}^n \rightarrow \{0,1\}^n$
- Given efficient code to compute F ➔ Can define (quantum) circuit Q_F to compute F.

$$|x, 0^n\rangle \rightarrow Q_F \rightarrow |x, F(x)\rangle$$

- More generally

$$|x, y\rangle \rightarrow Q_F \rightarrow |x, y \oplus F(x)\rangle$$

- Reversible (Uncomputation)

$$|x, y \oplus F(x)\rangle \rightarrow |x, y \oplus F(x) \oplus F(x)\rangle = |x, y\rangle$$

Grover's Algorithm

- Consider real world instantiation of function $F: \{0,1\}^n \rightarrow \{0,1\}^n$
- **Idea:** $|0^n, 0^n\rangle \rightarrow H^{\otimes n} \otimes I^{\otimes n} \rightarrow \sum_{x \in \{0,1\}^n} \sqrt{2^{-n}} |x, 0^n\rangle$

$$\sum_{x \in \{0,1\}^n} \sqrt{2^{-n}} |x, 0^n\rangle \Rightarrow Q_F \Rightarrow \sum_{x \in \{0,1\}^n} \sqrt{2^{-n}} |x, F(x)\rangle$$

- We just evaluated F on all inputs by applying circuit Q_F once!
- Quantum Pre-Image Attack in O(1) time?
- **Problem:** We must eventually measure our quantum state…

We observe $|x', F(x') = y\rangle$ with probability $\sqrt{2^{-n}}^2 = 2^{-n}$

Quantum Oracle

- Consider real world instantiation of function $F\colon \{0,1\}^n \to \{0,1\}^n$
- **Idea 1:** $|0^n, 0^n\rangle \to H^{\otimes n} \otimes I^{\otimes n} \to \sum_{x\in\{0,1\}^n} \sqrt{2^{-n}}|x, 0^n\rangle$

$$\sum_{x\in\{0,1\}^n} \sqrt{2^{-n}}|x, 0^n\rangle \Rightarrow Q_F \Rightarrow \sum_{x\in\{0,1\}^n} \sqrt{2^{-n}}|x, F(x)\rangle$$

- **Idea 2:** Try to boost amplitude on target state(s) $|x', F(x') = y\rangle$

Quantum Oracle

- Consider real world instantiation of function $F:\{0,1\}^n \rightarrow \{0,1\}^n$
- **Idea 1:** $|0^n, 0^n\rangle \rightarrow H^{\otimes n} \otimes I^{\otimes n} \rightarrow \sum_{x\in\{0,1\}^n} \sqrt{2^{-n}}|x, 0^n\rangle$

$$\sum_{x\in\{0,1\}^n} \sqrt{2^{-n}}|x, 0^n\rangle \Rightarrow Q_F \Rightarrow \sum_{x\in\{0,1\}^n} \sqrt{2^{-n}}|x, F(x)\rangle$$

- **Negation: Can negate amplitudes where** $F(x) = y$

$$\sum_{x\in\{0,1\}^n : F(x)\neq y} \sqrt{2^{-n}}|x, F(x)\rangle - \sum_{x\in\{0,1\}^n : F(x)=y} \sqrt{2^{-n}}|x, F(x)\rangle$$

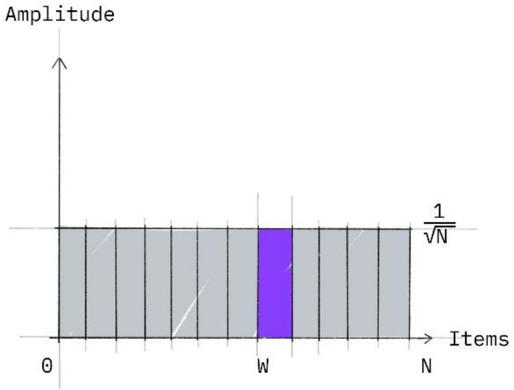
This step requires us to query oracle Q_F

Quantum Oracle

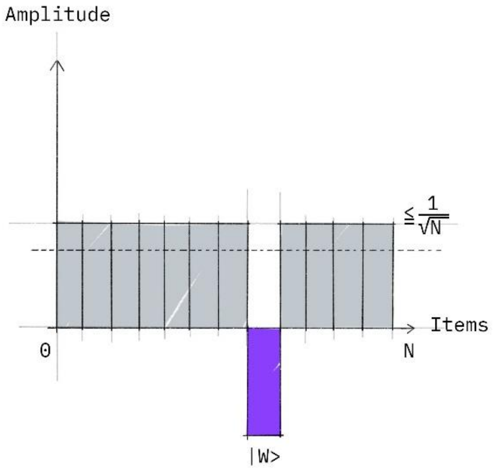
- **Negation: Can negate amplitudes where** $F(x) = y$

$$\sum_{x\in\{0,1\}^n:F(x)\neq y} \sqrt{2^{-n}}|x, F(x)\rangle - \sum_{\boldsymbol{x\in\{0,1\}^n:F(x)=y}} \boldsymbol{\sqrt{2^{-n}}|x, F(x)\rangle}$$

This step requires us to query oracle Q_F



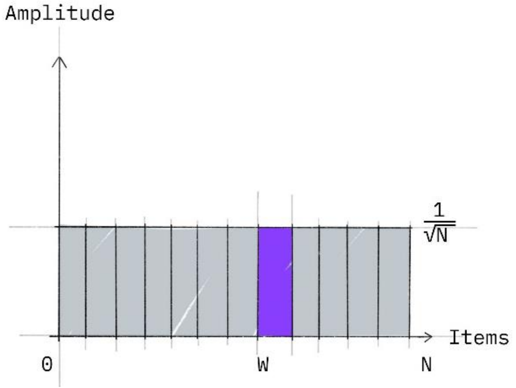
$\Rightarrow Negation \Rightarrow$



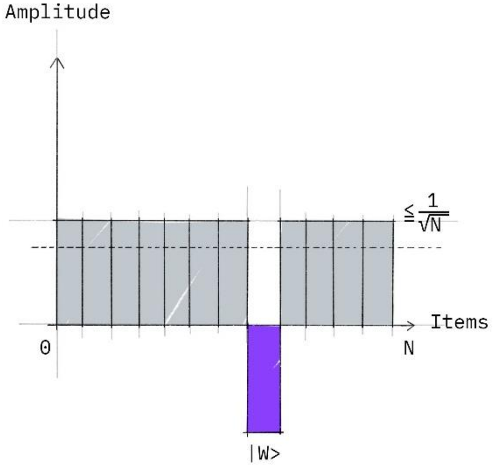
Quantum Oracle

- **Reflection: Can reflect amplitudes around mean**

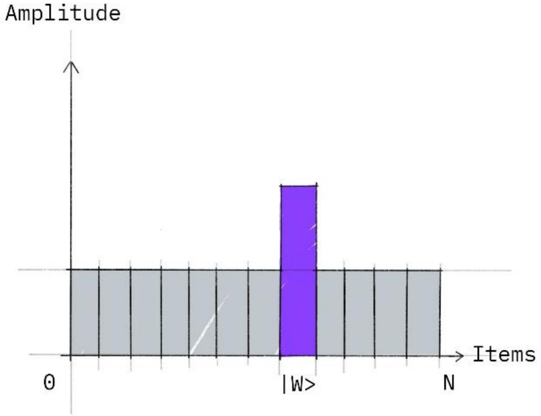
$$\sum_{x\in\{0,1\}^n:F(x)\neq y} \sqrt{2^{-n}}(1-\varepsilon)|x,F(x)\rangle + \sum_{x\in\{0,1\}^n:F(x)=y} (3-\varepsilon')\sqrt{2^{-n}}|x,F(x)\rangle$$



$\Rightarrow$ *Negation* $\Rightarrow$



$\Rightarrow$ *Reflection* $\Rightarrow$



Quantum Oracle

- Consider real world instantiation of function $F: \{0,1\}^n \rightarrow \{0,1\}^n$
- **Idea 1:** $|0^n, 0^n\rangle \rightarrow H^{\otimes n} \otimes I^{\otimes n} \rightarrow \sum_{x \in \{0,1\}^n} \sqrt{2^{-n}} |x, 0^n\rangle$

$$\sum_{x \in \{0,1\}^n} \sqrt{2^{-n}} |x, 0^n\rangle \Rightarrow Q_F \Rightarrow \sum_{x \in \{0,1\}^n} \sqrt{2^{-n}} |x, F(x)\rangle$$

- **Idea 2:** Try to boost amplitude on target state(s) $|x', F(x') = y\rangle$
  - Negate + Reflect
  - Repeat $O\left(\sqrt{2^n}\right)$ times to ensure that we reach state $\sum_{x \in \{0,1\}^n} \alpha_x |x, F(x)\rangle$ s.t

$$\sum_{x \in \{0,1\}^n : F(x) = y} |\alpha_x|^2 \geq 0.99$$

Quantum Random Oracle Model

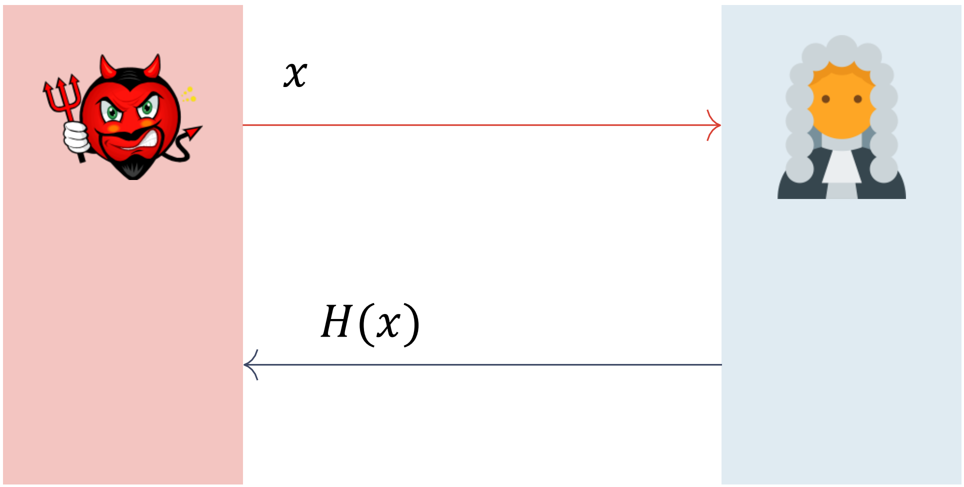
- **Motivation:** Any real world hash function can be computed efficiently by a quantum circuit ➔ we can use Grover's algorithm.

- Grover's algorithm uses hash function as random blackbox, but somehow the classical Random Oracle model does not capture power of generic quantum attacker.

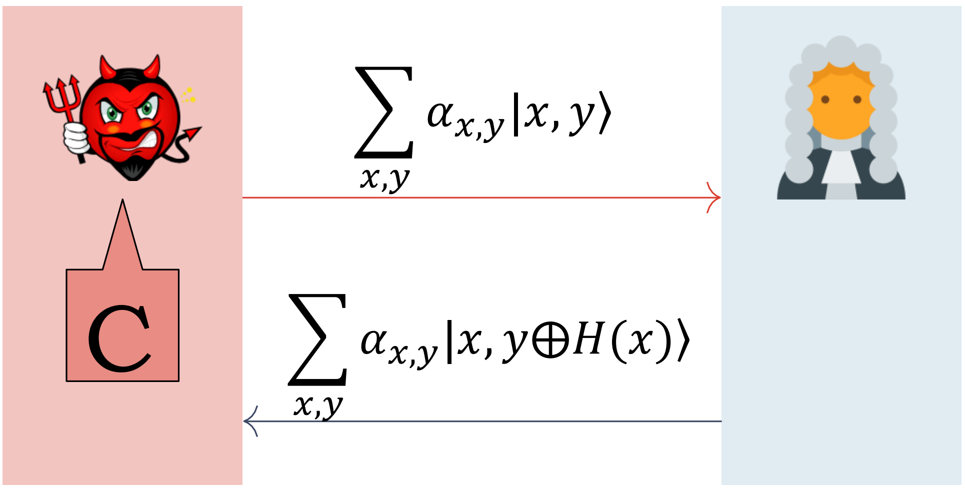
- **Goal:** Generic analysis tools to analyze the power of a quantum attacker who uses hash function as a blackbox?

ROM vs qROM [BDF+11]

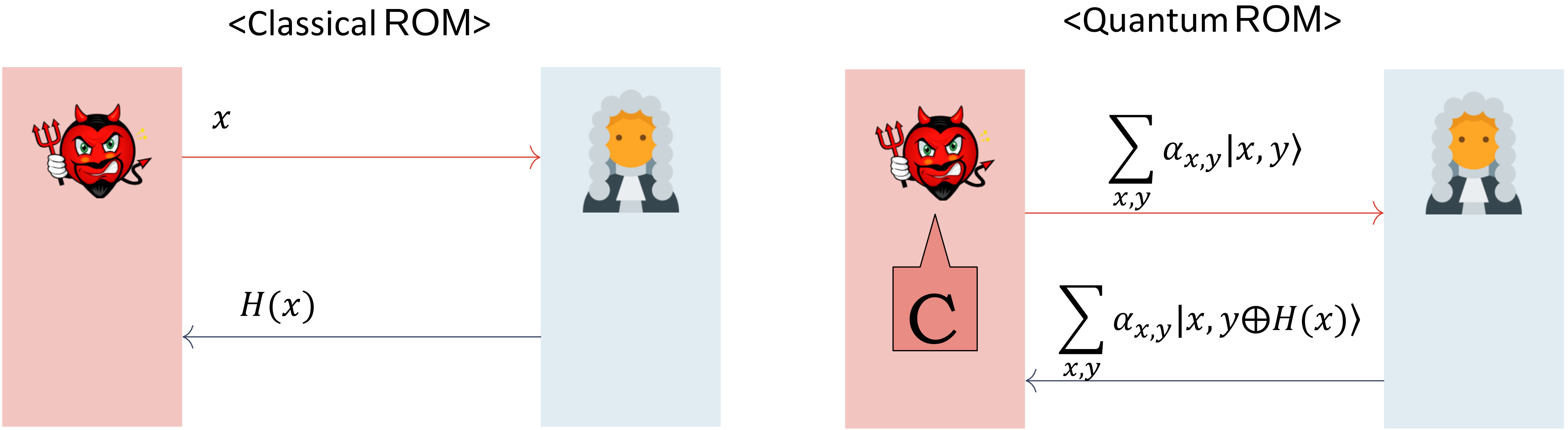
<Classical ROM>



<Quantum ROM>


$\sum_{x,y} \alpha_{x,y} |x, y\rangle$
C
$\sum_{x,y} \alpha_{x,y} |x, y \oplus H(x)\rangle$

ROM vs qROM [BDF$^+$11]


<Classical ROM>
x
$H(x)$
<Quantum ROM>
$\sum_{x,y} \alpha_{x,y} |x, y\rangle$
C
$\sum_{x,y} \alpha_{x,y} |x, y \oplus H(x)\rangle$

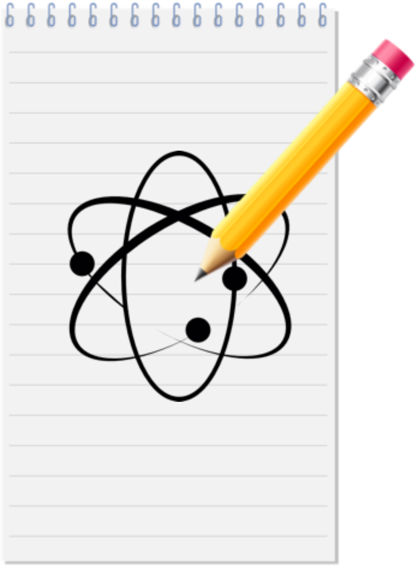
- Security proofs are much more challenging in the qROM
  - Programmability & Extractability (ROM: ✓, qROM: ✗)
  - Recording quantum queries?

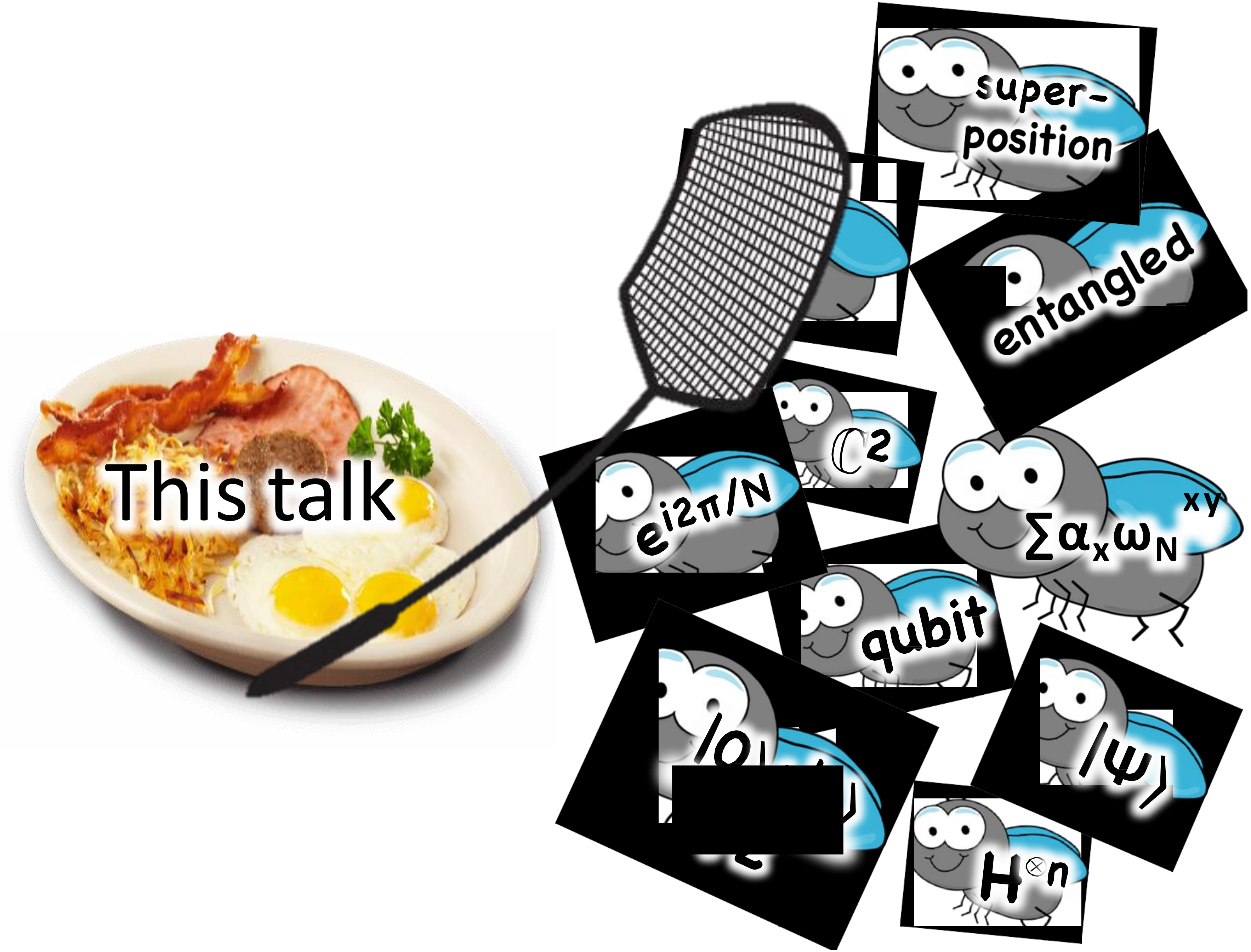
How to Record Quantum Queries

and Applications to Quantum Indifferentiability

Mark Zhandry

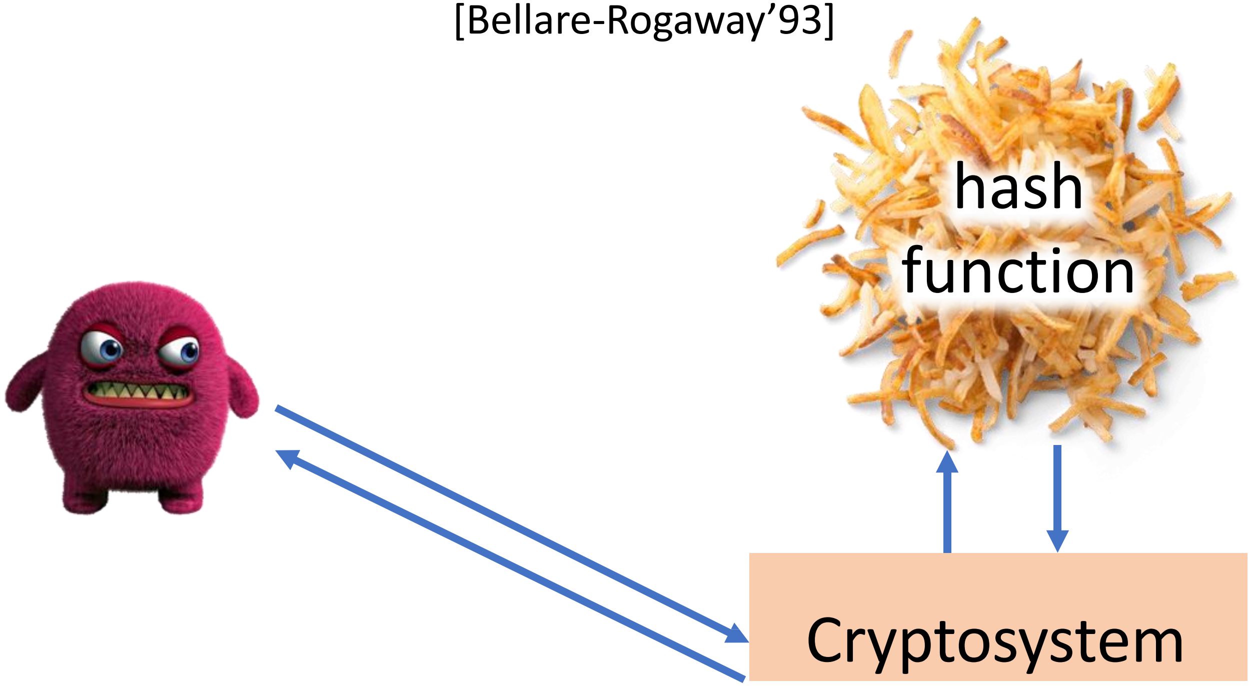
Princeton University & NTT Research




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

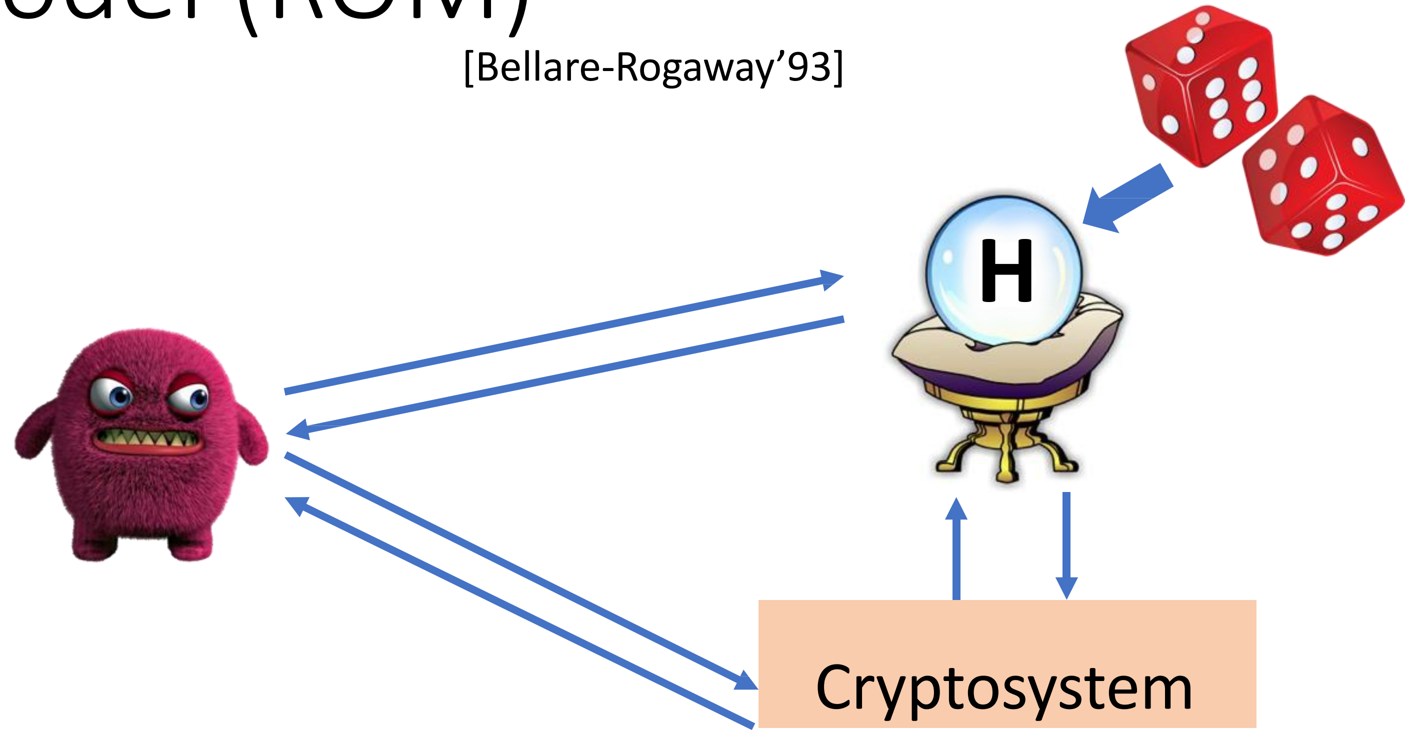
The (Classical) Random Oracle Model (ROM)

[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) ∈ D:
        Return(y,D)
    Else:
        y ß $ Y
        D’ = D+(x,y)
        Return(y,D’)
```

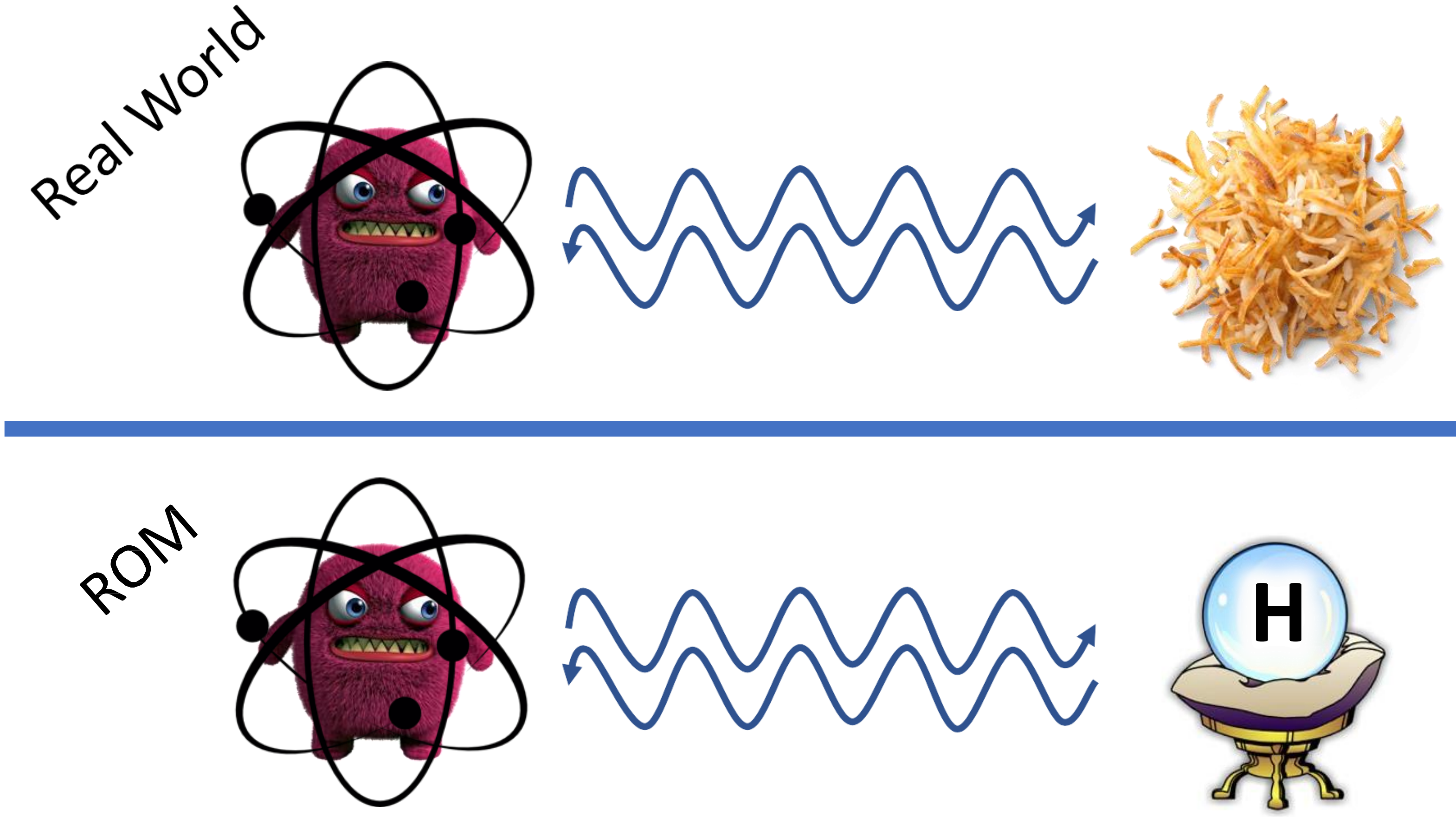
Typical ROM Proof: On-the-fly Simulation

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]



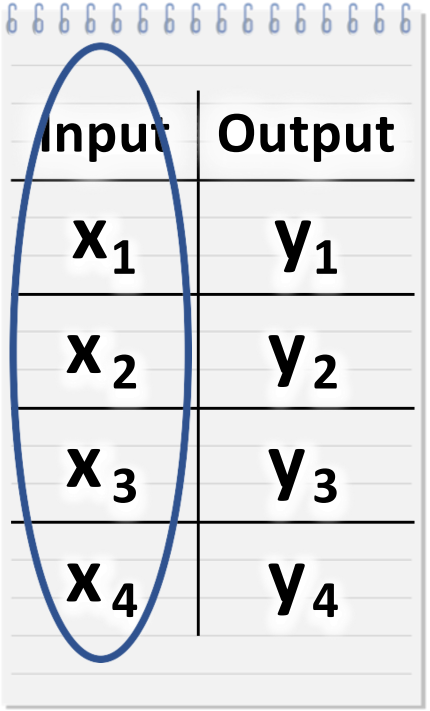
Real World

ROM

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

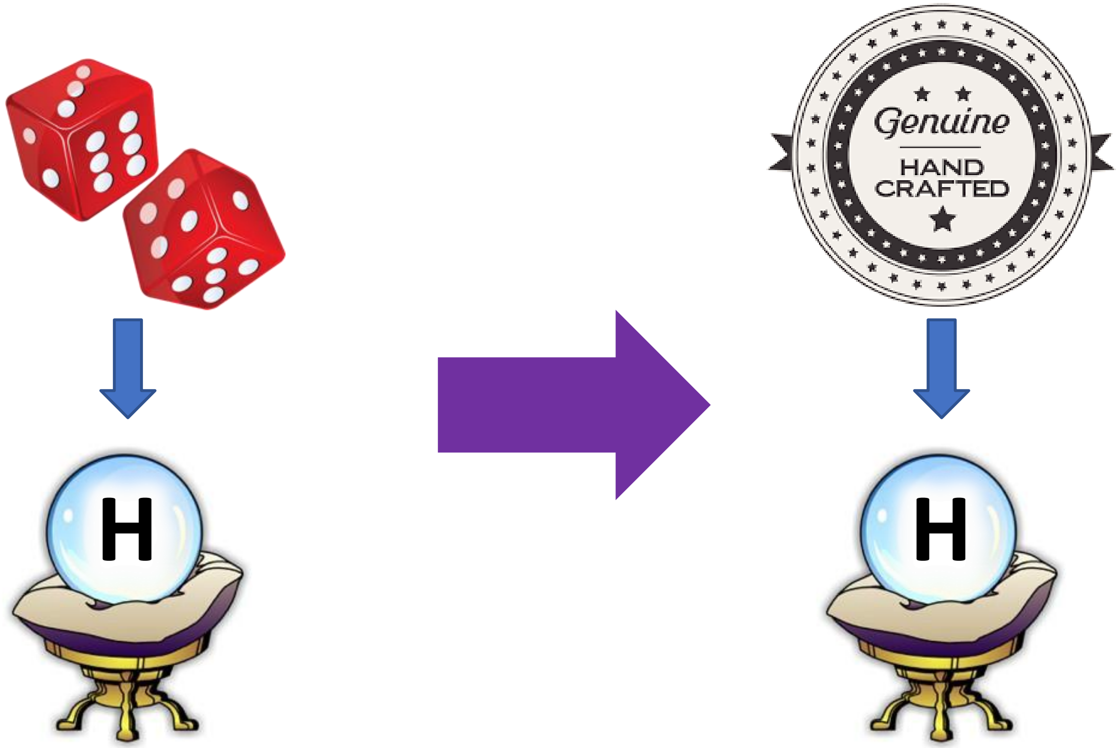
Observer Effect:
Learning anything about quantum system disturbs it

H 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

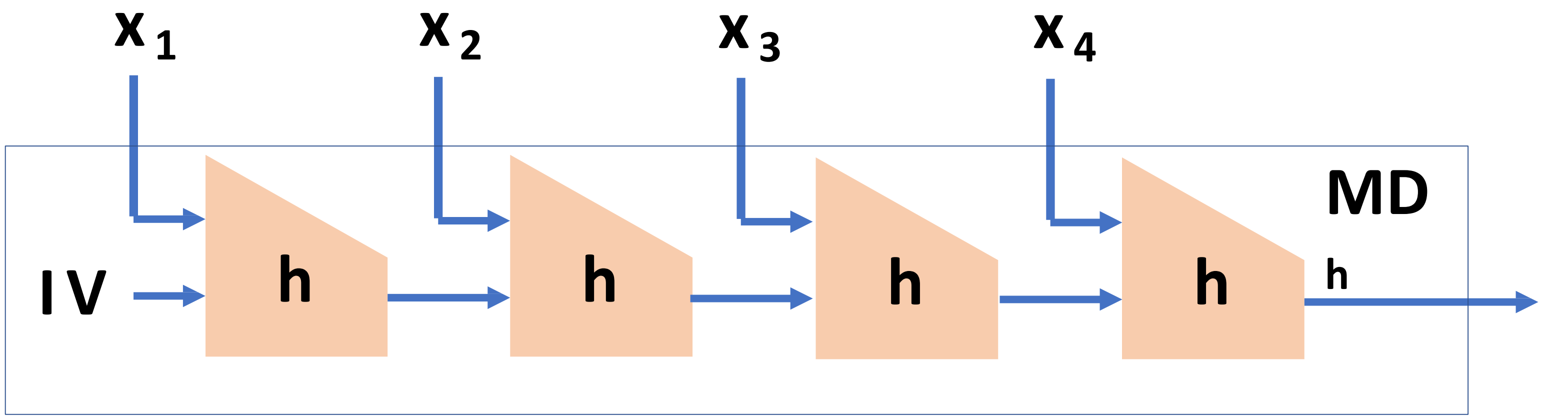
Fiat-Shamir

Luby-Rackoff

ROM è ICM

Example: Domain Extension for Random Oracles

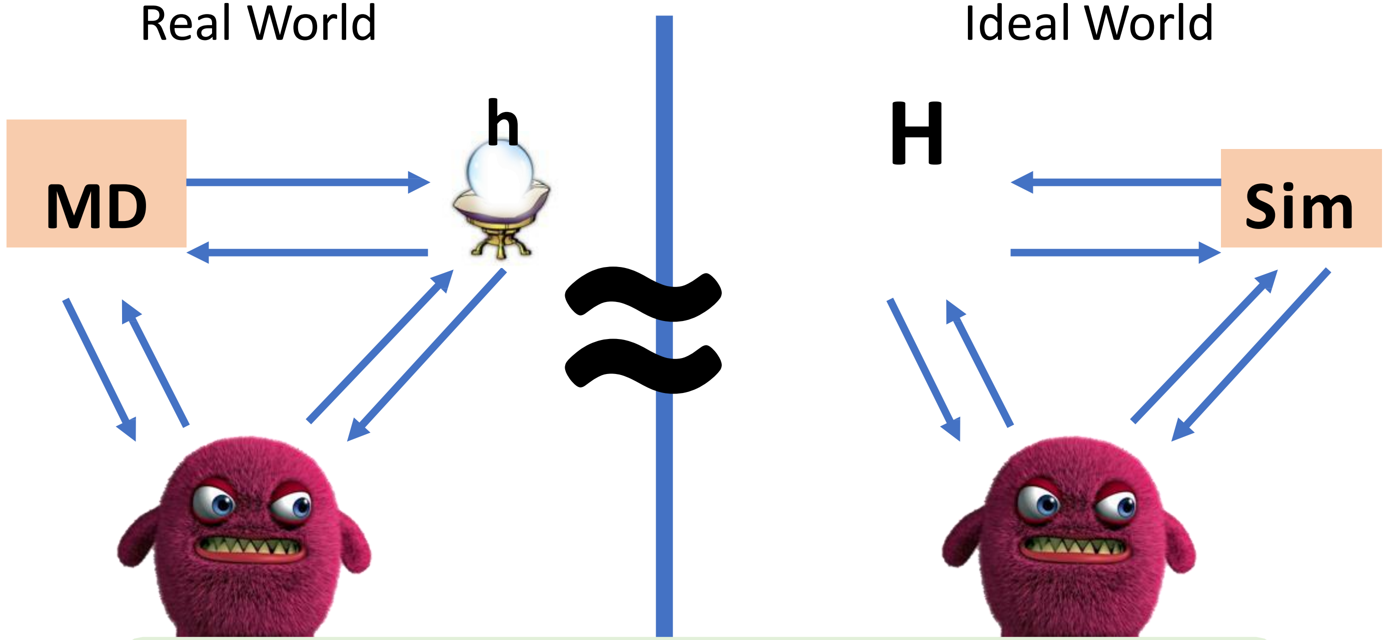
Q: Does Merkle-Damgård preserve random oracle-ness?



Example: Domain Extension for Random Oracles

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

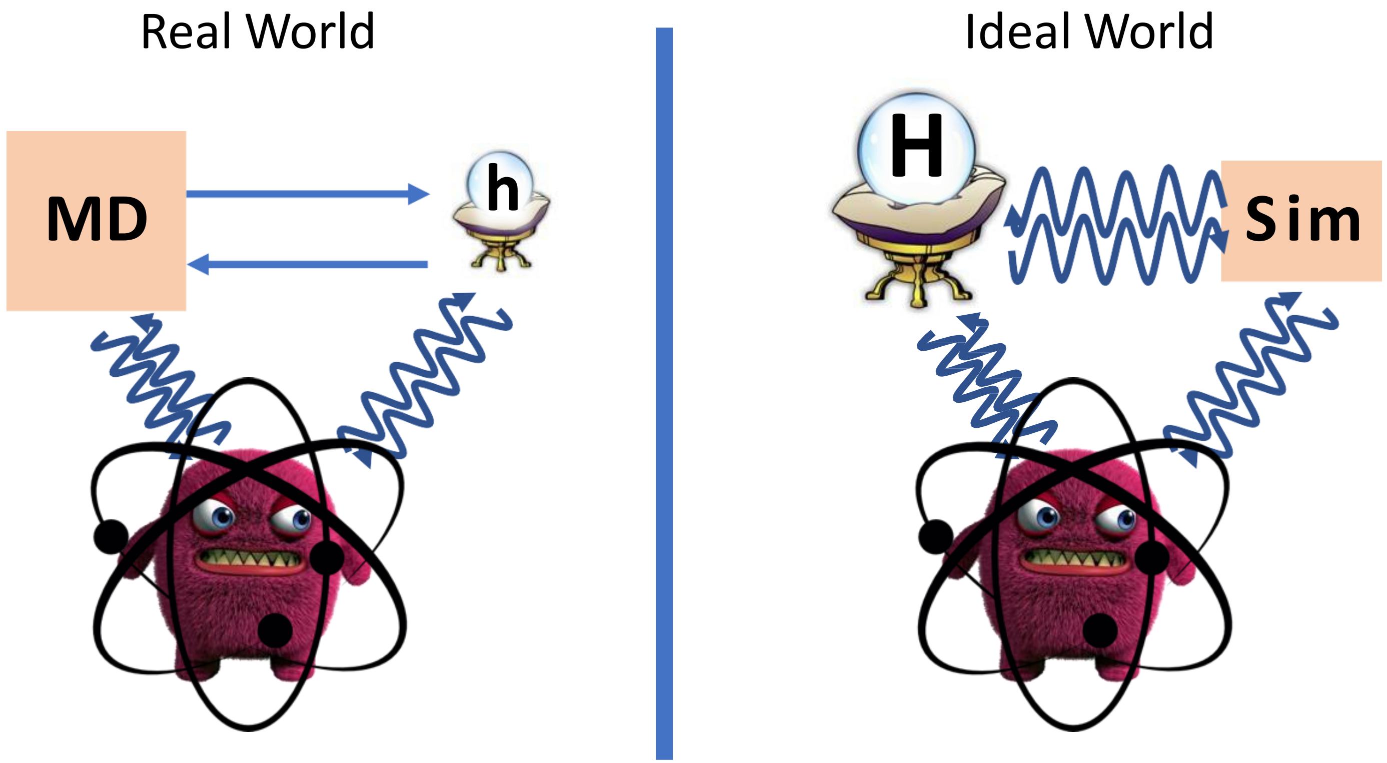
How? *Indifferentiability* [Maurer-Renner-Holenstein'04]



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

Quantum Indifferentiability?

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



Real World

MD

h

Ideal World

H

Sim

Quantum Indifferentiability?

Easy Thm

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

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

- Are we toast?

[Carstens-Ebrahimi-Tabia-Unruh'18]:

Conjecture yes

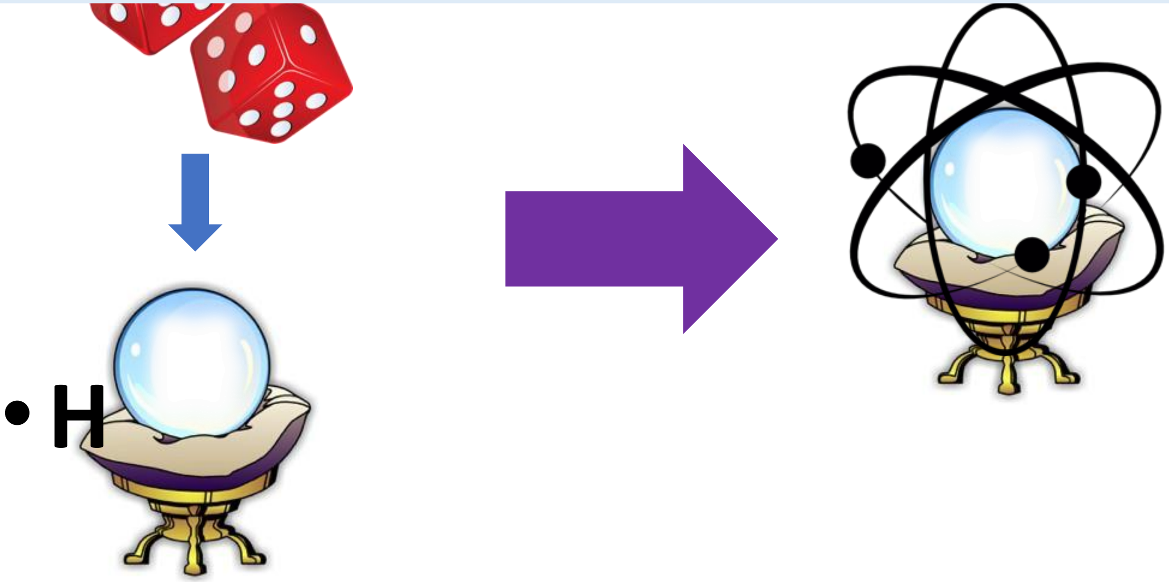
This Work:
On-the-fly simulation of quantum random oracles

(aka Compressed Oracles)

Step 1: Quantum-ify (aka Purify)

- Quantum-ifying (aka purifying) random oracle:

  → + new single quantum system



• H

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

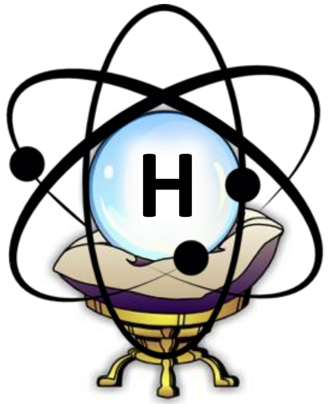
Step 1: Superposition of Oracles

Initial oracle state: H

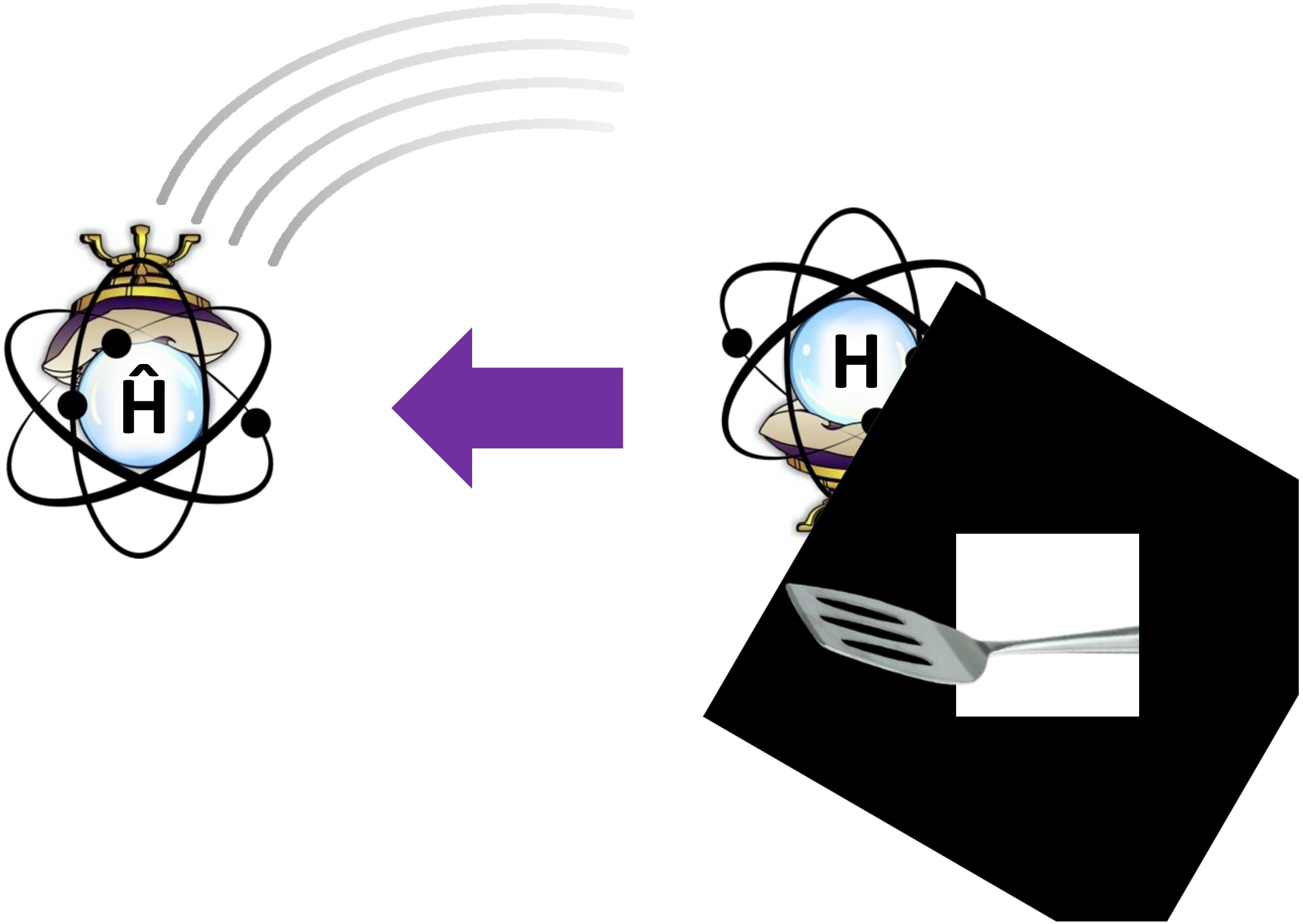
Query(x, y, H): y = y⊕H(x)

Adversary's query

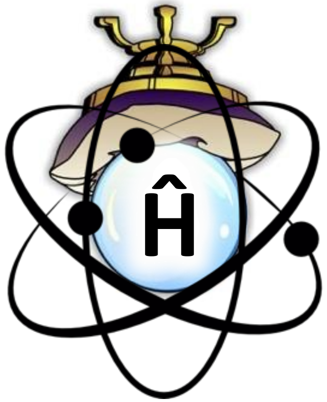
Oracle's state



Step 2: Look at Fourier Domain



Step 2: Look at Fourier Domain

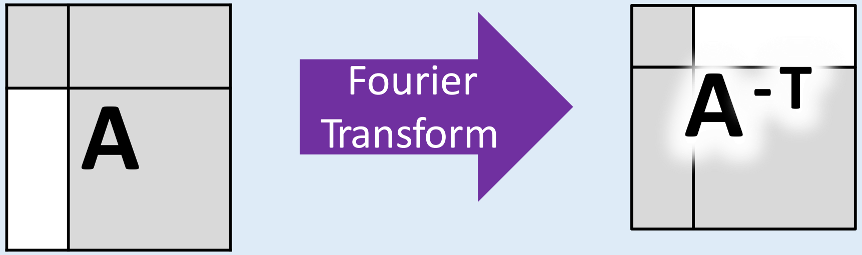


Initial oracle state: $\mathbf{Z(x) = 0}$

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

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

Proof:



A

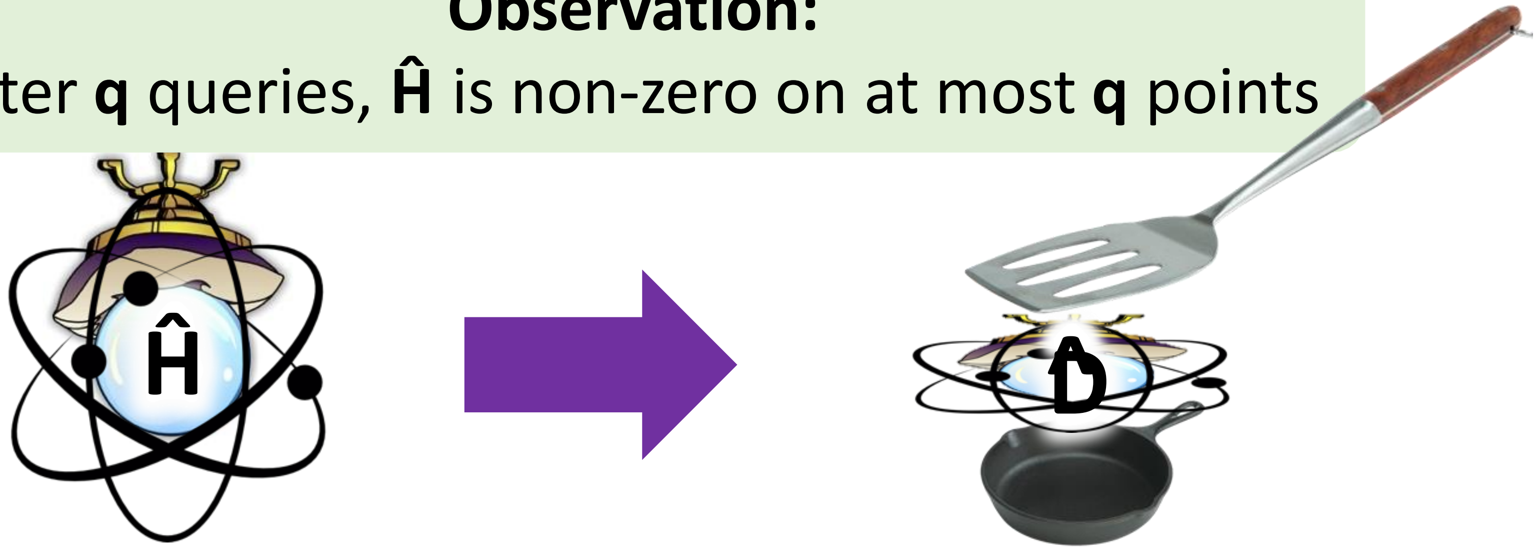
Fourier Transform

A^{-T}

Step 3: Compress

Observation:

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



Step 3: Compress

Initial oracle state: **{}**

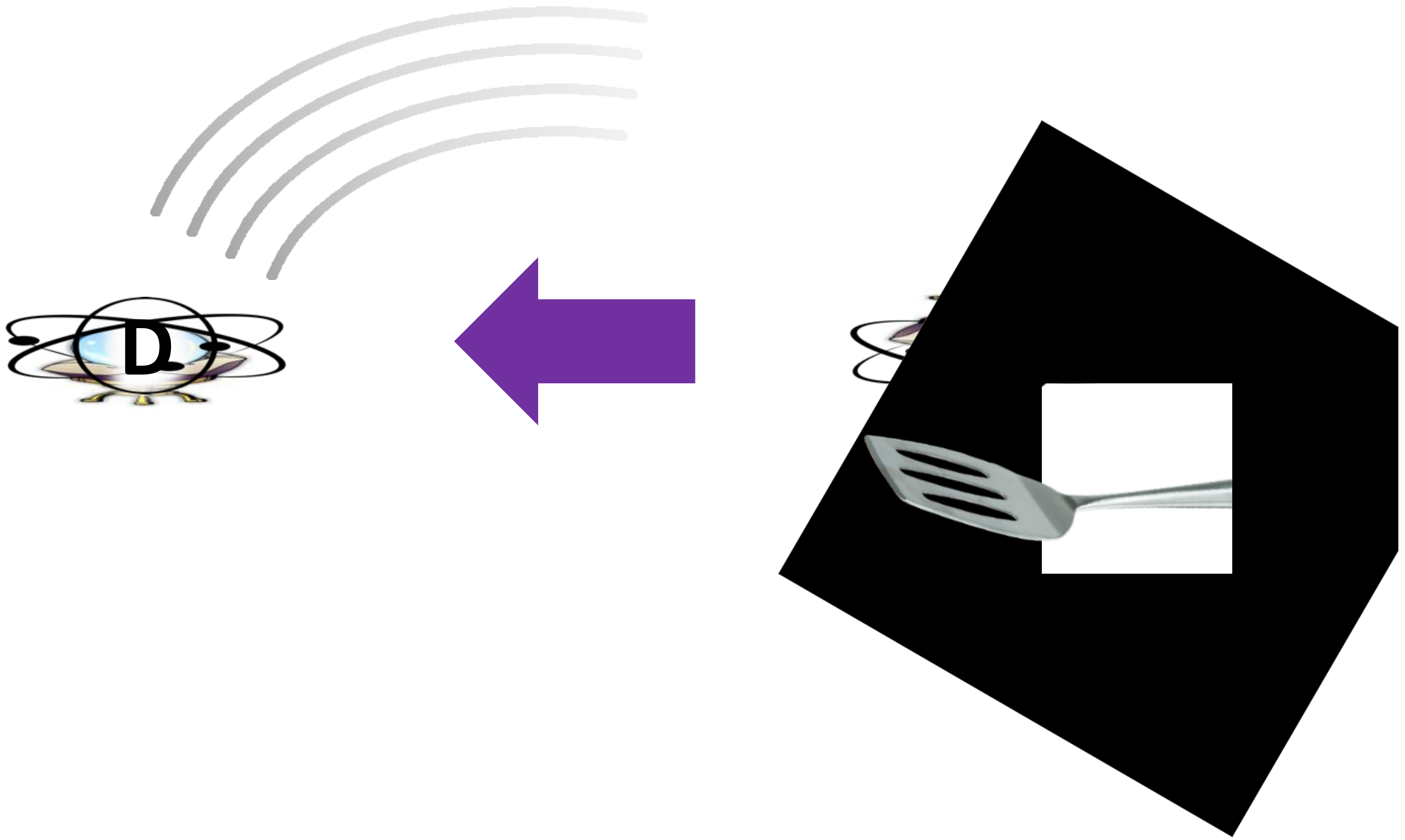
Query(x, y, D^):

(1) If ∄**(x,y')∈^D: ^D = D^+(x,0)**

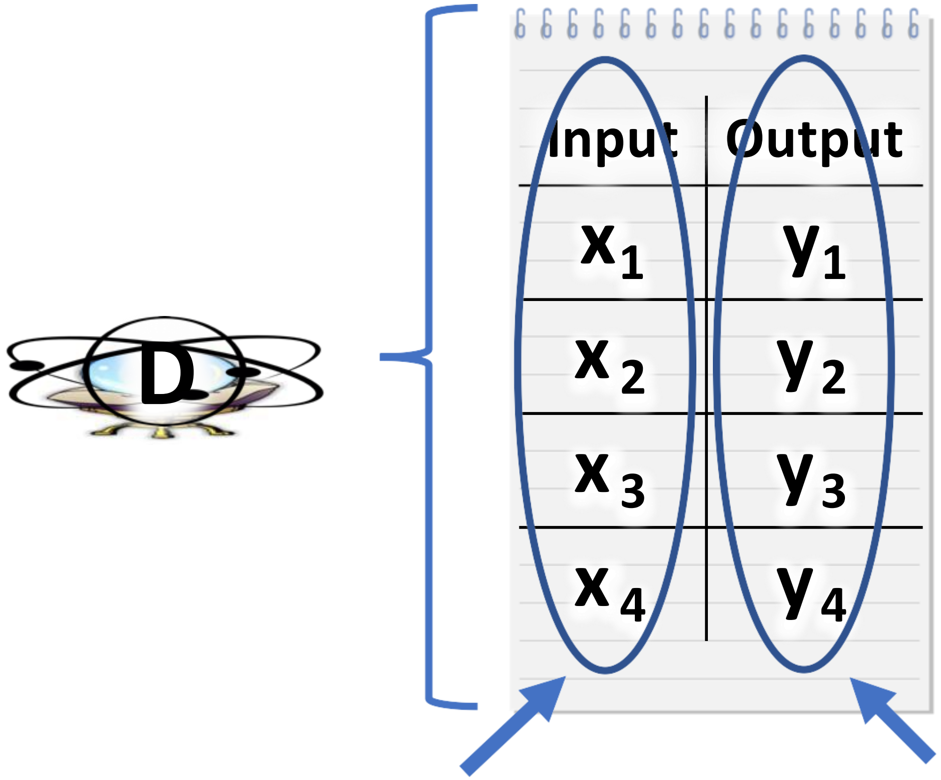
(2) Replace **(x,y')∈^D** with **(x,y'⊕y)**

(3) If **(x,0)∈^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!

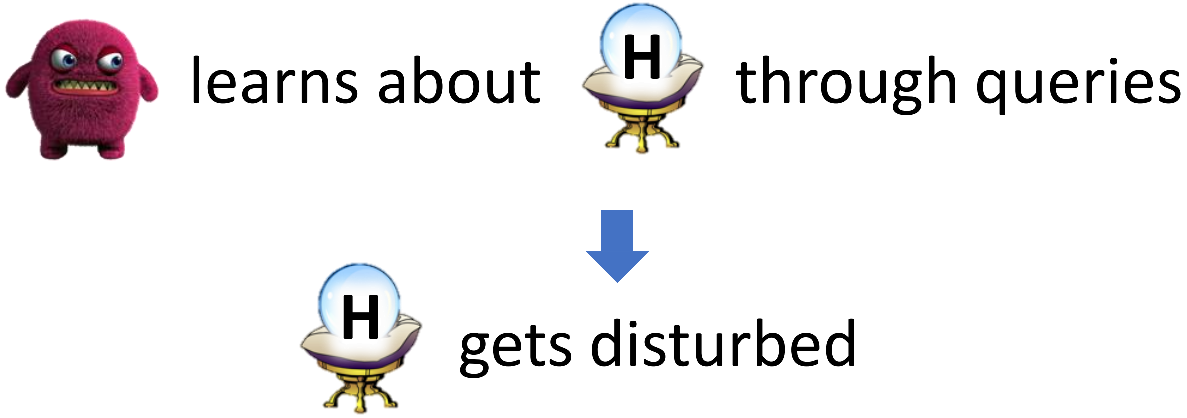
- Easy analysis of bad events (e.g. collisions)? ✓

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 **≠0** in Fourier domain
➡ **y** values aren’t exactly query outputs

Examining **x,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:

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

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

$\Omega(N^{1/(k+1)})$ 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!

Thanks for Listening