Quantum Computing

Advanced Topics in ~~Cybersecurity~~Cryptography (7CCSMATC)

Martin R. Albrecht

Outline

Programme

Teach you just enough quantum computing to get a sense of how it affects cryptography.

References

- Thomas Debris-Alazard. *Lecture 1: Introduction to Quantum Computing.* INF587 Quantum computer science and applications `https://tdalazard.io/S1.pdf`
- Noson S Yanofsky and Mirco A Mannucci. **Quantum Computing for Computer Scientists**. Cambridge University Press, 2008, esp. Chapters 1, 2, 3 and 6
- Fermi Ma's talk Quantum Secure Commitments and Collapsing Hash Functions delivered as part of the *Quantum Cryptography for Dummies* reading group at the *Lattices: Algorithms, Complexity, and Cryptography* special semester at the Simons Institute, 2020

Qubits

Classical Bit

$$b \in \{0, 1\}$$

Probabilistic Bit

- Probabilistic bit: $\begin{pmatrix} p \\ q \end{pmatrix}$ where $p := \Pr(b = 0)$ and $q := \Pr(b = 1)$
- Computing on probabilistic bits

$$\begin{pmatrix} p \\ q \end{pmatrix} \to \begin{pmatrix} p' \\ q' \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \cdot \begin{pmatrix} p \\ q \end{pmatrix} \text{ where } \begin{cases} a + c = 1 \\ b + d = 1 \end{cases} \text{ and } a, b, c, d \geq 0$$

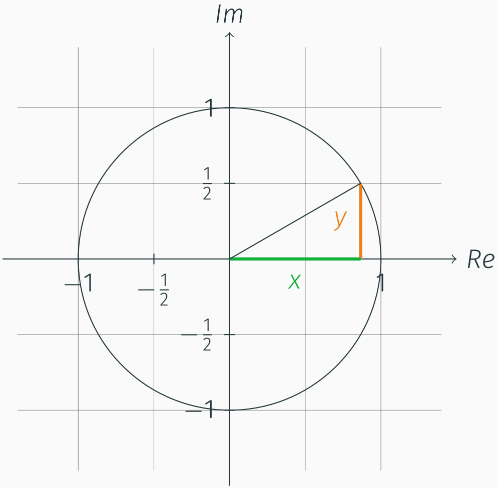
Example: $b \to b \oplus b$

$$\begin{pmatrix} p \\ q \end{pmatrix} \to \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} p \\ q \end{pmatrix}$$

Example: $b \to b \oplus 1$

$$\begin{pmatrix} p \\ q \end{pmatrix} \to \begin{pmatrix} q \\ p \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} p \\ q \end{pmatrix}$$

Complex Numbers



- $i := \sqrt{-1}$
- $z := x + i\,y$
- $Re(z) := x$
- $Im(z) := y$
- $|z| = \sqrt{x^2 + y^2}$

Quantum Bit (Qubit): "Probabilistic Bits with Complex Probabilities"

- **A qubit** $|\psi\rangle$ is an element of $\mathbb{C}^2$ with Euclidean norm 1:

$$|\psi\rangle = \alpha \cdot |0\rangle + \beta \cdot |1\rangle \text{ with } \alpha, \beta \in \mathbb{C} \text{ (called amplitude) and } |\alpha|^2 + |\beta|^2 = 1$$

- $|0\rangle, |1\rangle$ is an orthonormal basis of $\mathbb{C}^2$. Usually defined as

$$|0\rangle := \begin{pmatrix} 1 \\ 0 \end{pmatrix} \text{ and } |1\rangle := \begin{pmatrix} 0 \\ 1 \end{pmatrix} \text{ which then implies } \alpha \cdot |0\rangle + \beta \cdot |1\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}.$$

- We call this a **superposition** of $|0\rangle$ and $|1\rangle$.

Measurements

We cannot "see" qubits, we can only measure in their classical states.

Measurement: probabilistic orthogonal projection. Given $|0\rangle, |1\rangle \in \mathbb{C}^2$:

$$|\psi\rangle = \alpha \cdot |0\rangle + \beta \cdot |1\rangle \xrightarrow{\text{measure}} \begin{cases} |0\rangle & \text{with probability } |\alpha|^2 \\ |1\rangle & \text{with probability } |\beta|^2 \end{cases}$$

Computation

- A **unitary matrix in $\mathbb{C}^{2\times 2}$** is any matrix such that $\mathbf{U} \cdot \mathbf{U}^\dagger = \mathbf{U}^\dagger \cdot \mathbf{U} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ where $\mathbf{U}^\dagger$ is the conjugate transpose of $\mathbf{U}$:

$$\mathbf{U} = \begin{pmatrix} u_{00} + v_{00} \cdot i & u_{01} + v_{01} \cdot i \\ u_{10} + v_{10} \cdot i & u_{11} + v_{11} \cdot i \end{pmatrix}, \quad \mathbf{U}^\dagger := \begin{pmatrix} u_{00} - v_{00} \cdot i & u_{10} - v_{10} \cdot i \\ u_{01} - v_{01} \cdot i & u_{11} - v_{11} \cdot i \end{pmatrix}$$

- **Computation:** $|\psi\rangle \to \mathbf{U} \cdot |\psi\rangle$
  - All quantum computations are reversible:

$$|\psi\rangle \xrightarrow{\mathbf{U}} \mathbf{U} \cdot |\psi\rangle \xrightarrow{\mathbf{U}^\dagger} \mathbf{U}^\dagger \cdot \mathbf{U} \cdot |\psi\rangle = |\psi\rangle$$

- We call $\mathbf{U}$ **quantum gates**

Quantum Computers ...

... are linear-algebra machines.

QUANTUM COMPUTERS ...

... are linear-algebra machines.

Mathematics is the art of reducing any problem to linear algebra.

— William Stein

Examples of Quantum Gates

Example NOT-gate $b \to$ “$b \oplus 1$”

$$|\psi\rangle = \alpha\,|0\rangle + \beta \cdot |1\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix} \longrightarrow \begin{pmatrix} \beta \\ \alpha \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \cdot |\psi\rangle$$

Examples of Quantum Gates

Example NOT-gate $b \to$ "$b \oplus 1$"

$$|\psi\rangle = \alpha\,|0\rangle + \beta \cdot |1\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix} \longrightarrow \begin{pmatrix} \beta \\ \alpha \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \cdot |\psi\rangle$$

Example $b \to b \oplus b$

Computation is not reversible!

Hadamard-gate H I

$$\mathsf{H} \cdot |\psi\rangle = \mathsf{H} \cdot (\alpha\,|0\rangle + \beta \cdot |1\rangle) = \mathsf{H} \cdot \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \frac{1}{\sqrt{2}} \cdot \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \cdot \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \frac{1}{\sqrt{2}} \cdot \begin{pmatrix} \alpha + \beta \\ \alpha - \beta \end{pmatrix}$$

Hadamard-gate H I

$$\mathsf{H} \cdot |\psi\rangle = \mathsf{H} \cdot (\alpha\,|0\rangle + \beta \cdot |1\rangle) = \mathsf{H} \cdot \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \frac{1}{\sqrt{2}} \cdot \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \cdot \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \frac{1}{\sqrt{2}} \cdot \begin{pmatrix} \alpha + \beta \\ \alpha - \beta \end{pmatrix}$$

- $\mathsf{H} \cdot |0\rangle = \frac{1}{\sqrt{2}} \cdot \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \frac{1}{\sqrt{2}} \cdot \begin{pmatrix} 1+0 \\ 1-0 \end{pmatrix} = \begin{pmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{pmatrix}$
  - measure $|0\rangle$ or $|1\rangle$ with probability 1/2!

Hadamard-gate H I

$$\mathsf{H} \cdot |\psi\rangle = \mathsf{H} \cdot (\alpha\,|0\rangle + \beta \cdot |1\rangle) = \mathsf{H} \cdot \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \frac{1}{\sqrt{2}} \cdot \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \cdot \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \frac{1}{\sqrt{2}} \cdot \begin{pmatrix} \alpha + \beta \\ \alpha - \beta \end{pmatrix}$$

- $\mathsf{H} \cdot |0\rangle = \frac{1}{\sqrt{2}} \cdot \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \frac{1}{\sqrt{2}} \cdot \begin{pmatrix} 1+0 \\ 1-0 \end{pmatrix} = \begin{pmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{pmatrix}$
  - measure $|0\rangle$ or $|1\rangle$ with probability 1/2!
- $\mathsf{H} \cdot \begin{pmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{pmatrix} = \frac{1}{\sqrt{2}} \cdot \begin{pmatrix} 1/\sqrt{2} + 1/\sqrt{2} \\ 1/\sqrt{2} - 1/\sqrt{2} \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} = |0\rangle$
  - and we're back!

Hadamard-gate H II

The outputs of the Hadamard gate applied to $|0\rangle$ and $|1\rangle$ are so important we give them names:

$$|+\rangle := \frac{1}{\sqrt{2}} \cdot \Big(|0\rangle + |1\rangle \Big) = \frac{1}{\sqrt{2}} \cdot \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$
$$|-\rangle := \frac{1}{\sqrt{2}} \cdot \Big(|0\rangle - |1\rangle \Big) = \frac{1}{\sqrt{2}} \cdot \begin{pmatrix} 1 \\ -1 \end{pmatrix}$$

We cannot achieve the Hardmard gate with probabilistic bits $\Rightarrow$ quantum advantage

Many Qubits

Tensor Products

Let $\mathbf{v} \in \mathbb{C}^n$ and $\mathbf{w} \in \mathbb{C}^m$, their tensor product is $\mathbf{v} \otimes \mathbf{w} := (v_0 \cdot \mathbf{w}, \ldots, v_{n-1} \cdot \mathbf{w})$

- This is the same as the rows of:

$$\begin{pmatrix} v_0 \\ v_1 \\ \vdots \\ v_{n-1} \end{pmatrix} \cdot \begin{pmatrix} w_0 & w_1 & \cdots & w_{m-1} \end{pmatrix} = \begin{pmatrix} v_0 \cdot w_0 & v_0 \cdot w_1 & \cdots & v_0 \cdot w_{m-1} \\ v_1 \cdot w_0 & v_1 \cdot w_1 & \cdots & v_1 \cdot w_{m-1} \\ \vdots & \vdots & \ddots & \vdots \\ v_{n-1} \cdot w_0 & v_{n-1} \cdot w_1 & \cdots & v_{n-1} \cdot w_{m-1} \end{pmatrix}$$

Tensor Products

Let $\mathbf{v} \in \mathbb{C}^n$ and $\mathbf{w} \in \mathbb{C}^m$, their tensor product is $\mathbf{v} \otimes \mathbf{w} := (v_0 \cdot \mathbf{w}, \ldots, v_{n-1} \cdot \mathbf{w})$

- This is the same as the rows of:

$$\begin{pmatrix} v_0 \\ v_1 \\ \vdots \\ v_{n-1} \end{pmatrix} \cdot \begin{pmatrix} w_0 & w_1 & \cdots & w_{m-1} \end{pmatrix} = \begin{pmatrix} v_0 \cdot w_0 & v_0 \cdot w_1 & \cdots & v_0 \cdot w_{m-1} \\ v_1 \cdot w_0 & v_1 \cdot w_1 & \cdots & v_1 \cdot w_{m-1} \\ \vdots & \vdots & \ddots & \vdots \\ v_{n-1} \cdot w_0 & v_{n-1} \cdot w_1 & \cdots & v_{n-1} \cdot w_{m-1} \end{pmatrix}$$

- For any scalar z, we have $z \cdot (\mathbf{v} \otimes \mathbf{w}) = (z \cdot \mathbf{v}) \otimes \mathbf{w} = \mathbf{v} \otimes (z \cdot \mathbf{w})$
- For any $\mathbf{v}_0, \mathbf{v}_1 \in \mathbb{C}^n$, we have $(\mathbf{v}_0 + \mathbf{v}_1) \otimes \mathbf{w} = \mathbf{v}_0 \otimes \mathbf{w} + \mathbf{v}_1 \otimes \mathbf{w}$
- For any $\mathbf{w}_0, \mathbf{w}_1 \in \mathbb{C}^m$, we have $\mathbf{v} \otimes (\mathbf{w}_0 + \mathbf{w}_1) = \mathbf{v} \otimes \mathbf{w}_0 + \mathbf{v} \otimes \mathbf{w}_1$

Tensor Products of Spaces

Consider

- $\mathbb{C}^n = \mathrm{Span}_{\mathbb{C}}(\mathbf{v}_0, \dots \mathbf{v}_{n-1})$ where e.g. $\mathbf{v}_0 := (1, 0 \dots, 0)$ etc, or some other basis of $\mathbb{C}^n$
- $\mathbb{C}^m = \mathrm{Span}_{\mathbb{C}}(\mathbf{w}_0, \dots \mathbf{w}_{m-1})$ where e.g. $\mathbf{w}_0 := (1, 0 \dots, 0)$ etc, or some other basis of $\mathbb{C}^m$

We have:

- $\mathbb{C}^n \otimes \mathbb{C}^m := \mathrm{Span}_{\mathbb{C}}(\mathbf{v}_i \otimes \mathbf{w}_j : 0 \leq i < n; 0 \leq j < m)$
- $\mathbb{C}^n \otimes \mathbb{C}^m$ has dimension $n \times m$
- $\mathbf{x} \in \mathbb{C}^n \otimes \mathbb{C}^m \iff \exists\, \alpha_{i,j} : \sum_{0 \leq i < n, 0 \leq j < m} \alpha_{i,j} \cdot \mathbf{v}_i \otimes \mathbf{w}_j$

Tensor Products of Matrices

$$A := \begin{pmatrix} a_{0,0} & \cdots & a_{0,n-1} \\ \vdots & \ddots & \vdots \\ a_{m-1,0} & \cdots & a_{m-1,n-1} \end{pmatrix}, \quad B := \begin{pmatrix} b_{0,0} & \cdots & b_{0,q-1} \\ \vdots & \ddots & \vdots \\ b_{p-1,0} & \cdots & b_{p-1,q-1} \end{pmatrix}$$

$$A \otimes B := \begin{pmatrix} a_{0,0} \cdot B & \cdots & a_{0,m-1} \cdot B \\ \vdots & \ddots & \vdots \\ a_{n-1,0} \cdot B & \cdots & a_{n-1,m-1} \cdot B \end{pmatrix} \in \mathbb{C}^{mp \times nq}$$

Examples

1. $\begin{pmatrix} 1 \\ 2 \end{pmatrix} \otimes \begin{pmatrix} 2 \\ 3 \end{pmatrix} = \begin{pmatrix} 1 \cdot 2 \\ 1 \cdot 3 \\ 2 \cdot 2 \\ 2 \cdot 3 \end{pmatrix} = \begin{pmatrix} 2 \\ 3 \\ 4 \\ 6 \end{pmatrix}$
2. $\mathsf{X} \otimes \mathsf{H} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \otimes \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \\ 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \end{pmatrix}$

n Qubit States

- Recall that a qubit $|\psi\rangle$ is an element in $\mathbb{C}^2$ with norm 1.
- A **register of n qubits** $|\psi\rangle$ is an element in $\underbrace{\mathbb{C}^2 \otimes \cdots \otimes \mathbb{C}^2}_{n \text{ times}} = \mathbb{C}^{2^n}$.
- Let $|0\rangle, |1\rangle$ be an orthonormal basis of $\mathbb{C}^2$, then

$$(|b_0\rangle \otimes |b_1\rangle \otimes \cdots \otimes |b_{n-1}\rangle : b_0, \ldots, b_{n-1} \in \{0, 1\})$$

is an orthonormal basis of $\underbrace{\mathbb{C}^2 \otimes \cdots \otimes \mathbb{C}^2}_{n \text{ times}} = \mathbb{C}^{2^n}$.

```
ket0 = matrix(2, 1, [1,0])
ket1 = matrix(2, 1, [0,1])
ket0.tensor_product(ket0), \
ket0.tensor_product(ket1), \
ket1.tensor_product(ket0), \
ket1.tensor_product(ket1)
```

```
(
[1]  [0]  [0]  [0]
[0]  [1]  [0]  [0]
[-]  [-]  [-]  [-]
[0]  [0]  [1]  [0]
[0], [0], [0], [1]
)
```

Notation

- For $b_0, \dots, b_{n-1} \in \{0,1\}$ we write

$$|b_0\, b_1\, \dots\, b_{n-1}\rangle := |b_0\rangle \otimes |b_1\rangle \otimes \dots \otimes |b_{n-1}\rangle$$

- For $|\psi_0\rangle, |\psi_1\rangle, \dots, |\psi_{n-1}\rangle \in \mathbb{C}^2$, we write

$$|\psi_0\rangle\, |\psi_1\rangle \dots |\psi_{n-1}\rangle := |\psi_0\rangle \otimes |\psi_1\rangle \otimes \dots \otimes |\psi_{n-1}\rangle$$

- Any $|\psi\rangle \in \mathbb{C}^{2^n}$ of n qubits can be written as

$$|\psi\rangle = \sum_{\mathsf{x} \in \{0,1\}^n} \alpha_{\mathsf{x}} |\mathsf{x}\rangle \quad \text{where } \alpha_{\mathsf{x}} \in \mathbb{C} \text{ (called amplitude) and } \sum_{\mathsf{x} \in \{0,1\}^n} |\alpha_{\mathsf{x}}|^2 = 1$$

Separable States

Definition

An n-qubit state $|\psi\rangle$ is called separable if it can be decomposed as $|\psi\rangle = |\psi_0\rangle \otimes |\psi_1\rangle$.

Examples

- $|00\rangle = |0\rangle \otimes |0\rangle$
- $\frac{1}{2} \cdot (|00\rangle + |01\rangle + |10\rangle + |11\rangle) = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$

```
ket0 = matrix(2, 1, [1,0]); ket1 = matrix(2, 1, [0,1])
(1/sqrt(2)*ket0 + 1/sqrt(2)*ket1).tensor_product(1/sqrt(2)*ket0 + 1/sqrt(2)*ket1)
```

```
[1/2]
[1/2]
[---]
[1/2]
[1/2]
```

Entangled States

Definition

An n-qubit state $|\psi\rangle$ is called **entangled** if it cannot be decomposed as $|\psi\rangle = |\psi_0\rangle \otimes |\psi_1\rangle$.

Example

$$\frac{1}{\sqrt{2}} \cdot (|00\rangle + |11\rangle)$$

```
ket0 = matrix(2, 1, [1,0]); ket1 = matrix(2, 1, [0,1])
(1/sqrt(2)) * (ket0.tensor_product(ket0) + ket1.tensor_product(ket1))
```

```
[1/2*sqrt(2)]
[          0]
[          0]
[1/2*sqrt(2)]
```

Measuring n Qubit States

Measuring the state:

$$|\psi\rangle = \sum_{i_0,\ldots,i_{n-1}\in\{0,1\}^n} \alpha_{i_0\ldots i_{-n}} \left|e_{i_0} \cdots e_{i_{n-1}}\right\rangle \xrightarrow{\text{measure}} \left|e_{j_0} \ldots e_{j_{n-1}}\right\rangle \text{ with probability } \left|\alpha_{j_0\ldots j_{n-1}}\right|^2$$

Measuring the first register:

$$|\psi\rangle = \alpha_0 \cdot |e_0\rangle\,|\psi_0\rangle + \alpha_1 \cdot |e_1\rangle\,|\psi_1\rangle \xrightarrow{\text{measure}} \begin{cases} |e_0\rangle\,|\psi_0\rangle & \text{with prob. } |\alpha_0|^2 \\ |e_1\rangle\,|\psi_1\rangle & \text{with prob. } |\alpha_1|^2 \end{cases}$$

We necessarily have $|\alpha_0|^2 + |\alpha_1|^2 = 1$

- A unitary matrix $\mathbf{U} \in \mathbb{C}^{2^n \times 2^n}$, i.e. $\mathbf{U}^\dagger \cdot \mathbf{U} = \mathbf{I}_{2^n}$ is called a quantum circuit
- Any classical circuit f on n-bits can be written as a unitary $\mathbf{U}_f$ on $2n$-qubits

$$\mathbf{U}_f \cdot |\psi\rangle |0\rangle = |\psi\rangle |f(\psi)\rangle$$

Measuring Entangled Registers

Consider $\mathbf{U}_f \cdot |\mathbf{x}, y\rangle = |\mathbf{x}, f(\mathbf{x}) \oplus y\rangle$ for some $f : \{0,1\}^n \to \{0,1\}$.

- Measure the last register $|f(\mathbf{x}) \oplus y\rangle$ to obtain v
- The first register $|\mathbf{x}\rangle$ **collapses** to those $\mathbf{x} \in \{0,1\}^n$ s.t. $f(\mathbf{x}) \oplus y = v$.
- The first n registers hold a superposition of the preimages of v under f.
- Measurements must stay consistent. Turns out this is quite powerful!

No-Cloning Theorem

Cannot copy a quantum state

- We can "cut" and "paste" a quantum state, but we cannot "copy" and "paste".
- "Move is possible. Copy is impossible."

Proof Sketch:

- There is no linear map C from $|\psi\rangle \otimes |0\rangle$ to $|\psi\rangle \otimes |\psi\rangle$.
- It would need to map $\frac{|x\rangle+|y\rangle}{\sqrt{2}} \otimes |0\rangle$ to $\frac{|x\rangle+|y\rangle}{\sqrt{2}} \otimes \frac{|x\rangle+|y\rangle}{\sqrt{2}}$
- By linearity, we'd need: $C\left(\frac{|x\rangle+|y\rangle}{\sqrt{2}} \otimes |0\rangle\right) =$
  - $\frac{1}{\sqrt{2}} \cdot C((|x\rangle + |y\rangle) \otimes |0\rangle)$
  - $\frac{1}{\sqrt{2}} \cdot (C(|x\rangle \otimes |0\rangle) + C(|y\rangle \otimes |0\rangle))$
  - $\frac{1}{\sqrt{2}} \cdot (|x\rangle \otimes |x\rangle + |y\rangle \otimes |y\rangle)$
  - $\neq \frac{|x\rangle+|y\rangle}{\sqrt{2}} \otimes \frac{|x\rangle+|y\rangle}{\sqrt{2}}$

Grover's Algorithm

Problem Statement

Given some function $f : \{0,1\}^n \to \{0,1\}$, we want to find some special element x_0.

- For example, given an plaintext-ciphertext pair (p, c) for AES, we might write $f : k \to AES(k, p) \stackrel{?}{=} c$.
- Classically, we'd need to call f about 2^n times to find x_0
- Grover's algorithm only needs $\sqrt{2^n} = 2^{n/2}$ queries

Example

- $n = 2$; $x_0 = 10$; $\mathbf{U}_f \cdot |\mathbf{x}, y\rangle = |\mathbf{x}, f(\mathbf{x}) \oplus y\rangle$

$$\mathbf{U}_f := \begin{array}{cc} & \begin{array}{cccccccc} 00,0 & 00,1 & 01,0 & 01,1 & 10,0 & 10,1 & 11,0 & 11,1 \end{array} \\ \begin{array}{c} 00,0 \\ 00,1 \\ 01,0 \\ 01,1 \\ 10,0 \\ 10,1 \\ 11,0 \\ 11,1 \end{array} & \begin{pmatrix} 1 & & & & & & & \\ & 1 & & & & & & \\ & & 1 & & & & & \\ & & & 1 & & & & \\ & & & & & 1 & & \\ & & & & 1 & & & \\ & & & & & & 1 & \\ & & & & & & & 1 \end{pmatrix} \end{array}$$

First Attempt

Let's apply the Hadamard gate on the first n registers, apply $\mathbf{U}_f$ and measure

$$|\psi_0\rangle = |\mathbf{0}, 0\rangle$$

$$|\psi_1\rangle = \left(\mathbf{H}^{\oplus n} \otimes \mathbf{I}\right) \cdot |\mathbf{0}, 0\rangle = \left[\frac{\sum_{\mathbf{x} \in \{0,1\}^n} |\mathbf{x}\rangle}{\sqrt{2^n}}\right] |0\rangle$$

$$|\psi_2\rangle = \mathbf{U}_f \cdot \left[\frac{\sum_{\mathbf{x} \in \{0,1\}^n} |\mathbf{x}\rangle}{\sqrt{2^n}}\right] |0\rangle = \frac{\sum_{\mathbf{x} \in \{0,1\}^n} |\mathbf{x}, \ f(\mathbf{x})\rangle}{\sqrt{2^n}}$$

Measuring the last qubit will produce 1 with probability $1/2^n$. If that event happens, then measuring the first n qubits will output the correct answer x_0.

Trick 1: Phase Inversion I

Apply the Hadamard gate on the last register and apply $\mathbf{U}_f$

$$
\begin{aligned}
|\psi_0\rangle &= |\mathbf{x}, 1\rangle \\
|\psi_1\rangle &= (\mathbf{I}_n \otimes \mathbf{H}) \cdot |\mathbf{x}, 1\rangle = |\mathbf{x}\rangle \left[\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right] = \left[\frac{|\mathbf{x}, 0\rangle - |\mathbf{x}, 1\rangle}{\sqrt{2}}\right] \\
|\psi_2\rangle &= \mathbf{U}_f \cdot \left(|\mathbf{x}\rangle \left[\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right]\right) = |\mathbf{x}\rangle \left[\frac{|f(\mathbf{x}) \oplus 0\rangle - |f(\mathbf{x}) \oplus 1\rangle}{\sqrt{2}}\right] = |\mathbf{x}\rangle \left[\frac{|f(\mathbf{x})\rangle - |\overline{f(\mathbf{x})}\rangle}{\sqrt{2}}\right] \\
&= (-1)^{f(\mathbf{x})} \cdot |\mathbf{x}\rangle \left[\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right] = \begin{cases} -1\,|\mathbf{x}\rangle \left[\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right], & \text{if } \mathbf{x} = \mathbf{x}_0 \\ +1\,|\mathbf{x}\rangle \left[\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right], & \text{if } \mathbf{x} \neq \mathbf{x}_0 \end{cases}
\end{aligned}
$$

Trick 1: Phase Inversion II

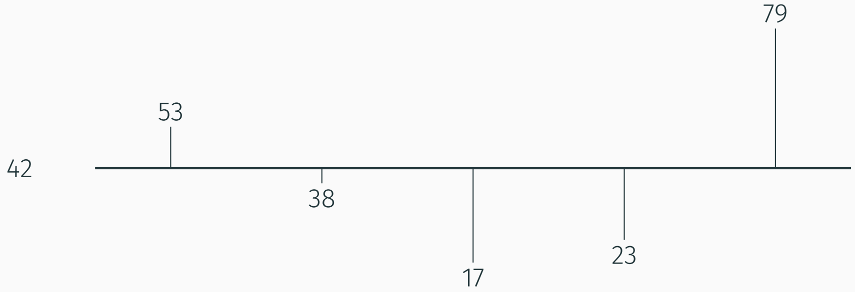
Applying:

$$U_f \cdot (I_n \otimes H) \cdot \begin{pmatrix} 1/2 \\ 1/2 \\ 1/2 \\ 1/2 \end{pmatrix} = \begin{pmatrix} 1/2 \\ 1/2 \\ -1/2 \\ 1/2 \end{pmatrix}$$

Useless?

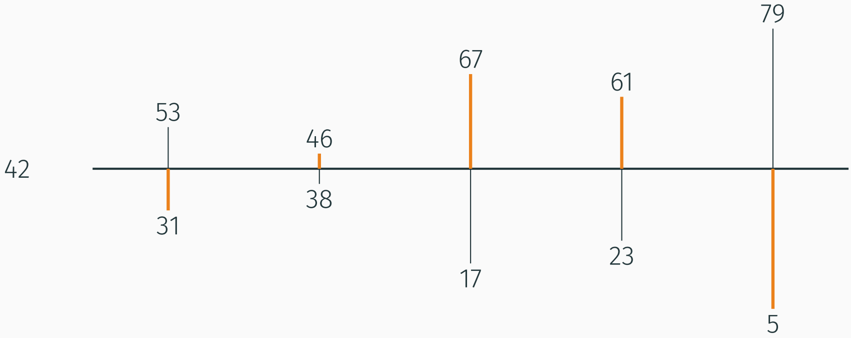
$$|(-1/2)|^2 = |1/2|^2$$

Trick 2: Inversion about the mean (Example)



$$y = \mu + (\mu - x) = -x + 2\mu$$

Trick 2: Inversion about the mean (Example)



$$y = \mu + (\mu - x) = -x + 2\mu$$

Computing the mean:

$$
M = \begin{pmatrix} 1/2^n & 1/2^n & \dots & 1/2^n & 1/2^n \\ 1/2^n & 1/2^n & \dots & 1/2^n & 1/2^n \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 1/2^n & 1/2^n & \dots & 1/2^n & 1/2^n \\ 1/2^n & 1/2^n & \dots & 1/2^n & 1/2^n \end{pmatrix}
\qquad
M \cdot v = \begin{pmatrix} 1/2^n \cdot \sum_{i=0}^{2^n} v_i \\ 1/2^n \cdot \sum_{i=0}^{2^n} v_i \\ \vdots \\ 1/2^n \cdot \sum_{i=0}^{2^n} v_i \\ 1/2^n \cdot \sum_{i=0}^{2^n} v_i \end{pmatrix}
$$

Inversion about the mean:

$$-\mathbf{I} + 2\,\mathbf{M} = \begin{pmatrix} (-1 + {}^{2}\!/_{2^n}) & {}^{2}\!/_{2^n} & \dots & {}^{2}\!/_{2^n} & {}^{2}\!/_{2^n} \\ {}^{2}\!/_{2^n} & (-1 + {}^{2}\!/_{2^n}) & \dots & {}^{2}\!/_{2^n} & {}^{2}\!/_{2^n} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ {}^{2}\!/_{2^n} & \dots & {}^{2}\!/_{2^n} & (-1 + {}^{2}\!/_{2^n}) & {}^{2}\!/_{2^n} \\ {}^{2}\!/_{2^n} & \dots & {}^{2}\!/_{2^n} & {}^{2}\!/_{2^n} & (-1 + {}^{2}\!/_{2^n}) \end{pmatrix}$$

$$(-\mathbf{I} + 2\,\mathbf{M}) \cdot \mathbf{v} = \begin{pmatrix} v_0 + {}^{2}\!/_{2^n} \cdot \sum_{i=0}^{2^n-1} v_i \\ v_1 + {}^{2}\!/_{2^n} \cdot \sum_{i=0}^{2^n-1} v_i \\ \vdots \\ v_{2^n-2} + {}^{2}\!/_{2^n} \cdot \sum_{i=0}^{2^n-1} v_i \\ v_{2^n-1} + {}^{2}\!/_{2^n} \cdot \sum_{i=0}^{2^n-1} v_i \end{pmatrix}$$

Trick 2: Inversion about the mean III

A straight-forward calculation shows that $-\mathsf{I} + 2\,\mathsf{M}$ is indeed unitary an thus a quantum circuit!

Combining the two tricks

```
def phasef(v, i):
    v = enumerate(v)
    v = [v_ * (-1)**int(i==j) for j,v_ in v]
    return vector(v)

phasef(vector(ZZ, 4, [1,1,1,1]), 2)
```

```
(1, 1, -1, 1)
```

```
def imeanf(v):
    N = ZZ(len(v))
    M = matrix(QQ, len(v), len(v), [1/N]*N**2)
    I = identity_matrix(N)
    return (-I + 2*M)*v

imeanf(vector(ZZ, 5, [53,38,17,23,79]))
```

```
(31, 46, 67, 61, 5)
```

Combining the two tricks

```
def phasef(v, i):
    v = enumerate(v)
    v = [v_ * (-1)**int(i==j) for j,v_ in v]
    return vector(v)

phasef(vector(ZZ, 4, [1,1,1,1]), 2)
```

```
(1, 1, -1, 1)
```

```
def imeanf(v):
    N = ZZ(len(v))
    M = matrix(QQ, len(v), len(v), [1/N]*N**2)
    I = identity_matrix(N)
    return (-I + 2*M)*v

imeanf(vector(ZZ, 5, [53,38,17,23,79]))
```

```
(31, 46, 67, 61, 5)
```

```
v = vector(RealField(prec=12), 5, [10]*5)
print(f"  input: {v}")
v = phasef(v, i=3); print(f"step 1a: {v}")
v = imeanf(v); print(f"step 1b: {v}, Δ: {max(v)-min(v)}")
v = phasef(v, i=3); print(f"step 2a: {v}")
v = imeanf(v); print(f"step 2b: {v}, Δ: {max(v)-min(v)}")
v = phasef(v, i=3); print(f"step 3a: {v}")
v = imeanf(v); print(f"step 3b: {v}, Δ: {max(v)-min(v)}")
```

```
  input: (10.0, 10.0, 10.0, 10.0, 10.0)
step 1a: (10.0, 10.0, 10.0, -10.0, 10.0)
step 1b: (2.00, 2.00, 2.00, 22.0, 2.00), Δ: 20.0
step 2a: (2.00, 2.00, 2.00, -22.0, 2.00)
step 2b: (-7.60, -7.60, -7.60, 16.4, -7.60), Δ: 24.0
step 3a: (-7.60, -7.60, -7.60, -16.4, -7.60)
step 3b: (-11.1, -11.1, -11.1, -2.33, -11.1), Δ: 8.80
```

Combining the two tricks

```
def phasef(v, i):
    v = enumerate(v)
    v = [v_ * (-1)**int(i==j) for j,v_ in v]
    return vector(v)

phasef(vector(ZZ, 4, [1,1,1,1]), 2)
```

```
(1, 1, -1, 1)
```

```
def imeanf(v):
    N = ZZ(len(v))
    M = matrix(QQ, len(v), len(v), [1/N]*N**2)
    I = identity_matrix(N)
    return (-I + 2*M)*v

imeanf(vector(ZZ, 5, [53,38,17,23,79]))
```

```
(31, 46, 67, 61, 5)
```

```
v = vector(RealField(prec=12), 5, [10]*5)
print(f"  input: {v}")
v = phasef(v, i=3); print(f"step 1a: {v}")
v = imeanf(v); print(f"step 1b: {v}, Δ: {max(v)-min(v)}")
v = phasef(v, i=3); print(f"step 2a: {v}")
v = imeanf(v); print(f"step 2b: {v}, Δ: {max(v)-min(v)}")
v = phasef(v, i=3); print(f"step 3a: {v}")
v = imeanf(v); print(f"step 3b: {v}, Δ: {max(v)-min(v)}")
```

```
  input: (10.0, 10.0, 10.0, 10.0, 10.0)
step 1a: (10.0, 10.0, 10.0, -10.0, 10.0)
step 1b: (2.00, 2.00, 2.00, 22.0, 2.00), Δ: 20.0
step 2a: (2.00, 2.00, 2.00, -22.0, 2.00)
step 2b: (-7.60, -7.60, -7.60, 16.4, -7.60), Δ: 24.0
step 3a: (-7.60, -7.60, -7.60, -16.4, -7.60)
step 3b: (-11.1, -11.1, -11.1, -2.33, -11.1), Δ: 8.80
```

The optimal number of repetitions is $\sqrt{\dim(v)}$

1. Start with $|\mathbf{0}\rangle$
2. Apply $\mathbf{H}^{\otimes n}$
3. Repeat $\sqrt{2^n}$ times:
    3.1 Apply phase inversion $\mathbf{U}_f \cdot (\mathbf{I} \otimes \mathbf{H})$
    3.2 Apply inversion about the mean $-\mathbf{I} + 2\mathbf{M}$
4. Measure the qubits

Recap: Grover vs AES

Best known quantum algorithms for attacking symmetric cryptography are based on Grover's algorithm.

- Search key space of size 2^n in $2^{n/2}$ operations: AES-256 $\rightarrow$ 128 "quantum bits of security".
- Taking all costs into account: $> 2^{152}$ classical operations for AES-256.[1]
- Assuming a max depth of 2^{96} for a quantum circuit: overall AES-256 cost is $\approx 2^{190}$.
- Does not parallelise: have to wait for 2^X steps, cannot buy 2^{32} quantum computers and wait $2^X/2^{32}$ steps.

[1] Samuel Jaques, Michael Naehrig, Martin Roetteler, and Fernando Virdia. Implementing Grover Oracles for Quantum Key Search on AES and LowMC. In: *EUROCRYPT 2020, Part II*. ed. by Anne Canteaut and Yuval Ishai. Vol. 12106. LNCS. Springer, Cham, May 2020, pp. 280–310. DOI: `10.1007/978-3-030-45724-2_10`.

Shor's Algorithm

Task

Given $N = p \cdot q$ for p, q prime find p or q.

A Magical New Operation

Consider a function $f_{a,N}(x)$ for any $0 < a < N$, which computes $f_{a,N}(x) := a^x \bmod N$

Example:

```
p, q = 13, 15
N = p*q
a = 2

def f(x):
    return power_mod(a, x, N)

f(13)
```

```
p, q = 3, 5
N = p*q
a = 2

[list(range(N)), None, [f(i) for i in range(N)]]
```

```
2
```

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	2	4	8	1	2	4	8	1	2	4	8	1	2	4

Theorem (Euler's Theorem)

For any modulus N and any coprime integer a, it holds that

$$a^{\phi(n)} \equiv 1 \bmod N$$

where $\phi(n)$, Euler's totient function, counts the integers up to n relatively prime to n.

- So $f_{a,N}(\cdot)$ should have some period r: $f_{a,N}(x) \equiv f_{a,N}(x+r)$.
- We can implement $f_{a,N}(\cdot)$ efficiently on classical and on quantum computers
- On a quantum computer, we can find this period efficiently but this assumed hard on classical computers.

A Magical New Operation

Let $\mathcal{P}(a, N)$ be an oracle that outputs r s.t. $f_{a,N}(x) \equiv f_{a,N}(x+r)$.

Factoring with that Magical New Operation

1. Pick a random $2 \leq a < N$.
2. If $\textbf{gcd}(a, N) \neq 1$, output a as a factor of N.
3. **Call $\mathcal{P}(a, N)$ and retrieve r.**
4. If r is not even, start over.
5. We have $a^r \equiv 1 \textbf{ mod } N$ and thus $N \mid (a^r - 1)$.
6. Write $a^r - 1 = (\sqrt{a^r} + 1) \cdot (\sqrt{a^r} - 1)$.[2]
7. So we get $N \mid (a^{r/2} - 1) \cdot (a^{r/2} + 1)$, i.e. any factor of N is a factor of $(a^{r/2} - 1)$, $(a^{r/2} + 1)$ or both
    7.1 It can't be that $N \mid a^{r/2} - 1$ because the period is r and not $r/2$
    7.2 It could be that $N \mid a^{r/2} + 1$ and then the algorithm fails
8. Compute $d := \textbf{gcd}(N, a^{r/2} + 1)$

[2] $x^2 - y^2 = (x - y) \cdot (x + y)$

The Magical New Operation

1. We can implement $f_{a,N}(\cdot)$ as a quantum circuit $\mathbf{U}_{f_{a,N}(\cdot)}$ acting on $m := \lceil \log N^2 \rceil$ qubits
2. We can apply Hadamard gates on the inputs before applying $\mathbf{U}_{f_{a,N}(\cdot)}$
3. This gives us a state[3]

$$|\phi_2\rangle := \frac{\sum_{\mathbf{x}\in\{0,1\}^m} |\mathbf{x}, f_{a,N}(\mathbf{x})\rangle}{\sqrt{2^m}} = \frac{\sum_{\mathbf{x}\in\{0,1\}^m} |\mathbf{x}, a^{\mathbf{x}} \bmod N\rangle}{\sqrt{2^m}}.$$

4. The final ingredient is a Quantum Fourier Transform (QFT) which more or less extracts the period from such a state.[4]

[3] I'm identifying the binary representation $\mathbf{x}$ of x with x here.
[4] I have yet to find a simple way of explaining it :(

Recap: Shor vs RSA, DH, ...

Commitment Schemes

Alice		Bob
$\mathsf{r} \leftarrow\!\!\$ \{0,1\}^\lambda$	$\xleftarrow{\quad\mathsf{k}\quad}$	
$\mathsf{m} \in \{0,1\}^\lambda$ $\mathrm{com}_k(\mathsf{m}, \mathsf{r}) = \mathsf{c}$	$\xrightarrow[\text{(commitment)}]{\quad\mathsf{c}\quad}$	
	$\xrightarrow[\text{(opening)}]{\quad\mathsf{r}, \mathsf{m}\quad}$	

Statistically Hiding:

$$\Pr\left[b' = b \;\middle|\; \begin{array}{r} (\mathsf{m}_0, \mathsf{m}_1) \leftarrow \mathcal{A}(\mathsf{k}) \\ b \leftarrow \{0,1\} \\ \mathsf{r} \leftarrow\!\!\$ \{0,1\}^\lambda \\ \mathsf{c} \leftarrow \mathrm{com}_k(\mathsf{m}_b, \mathsf{r}) \\ b' \leftarrow \mathcal{A}(\mathsf{c}) \end{array} \right] = \frac{1}{2}$$

for any $\mathcal{A}$.

Computationally Binding: “PPT adversary cannot change its mind after sending c”

Alice		Bob
$\mathsf{r} \leftarrow\!\!\$ \{0,1\}^\lambda$	$\xleftarrow{\quad\mathsf{k}\quad}$	
$\mathsf{m} \in \{0,1\}^\lambda$	$\xrightarrow[\text{(commitment)}]{\quad\mathsf{c}\quad}$	
$\mathsf{com}_k(\mathsf{m},\mathsf{r}) = \mathsf{c}$	$\xrightarrow[\text{(opening)}]{\quad\mathsf{r},\mathsf{m}\quad}$	

Statistically Hiding:

$$\Pr\left[b' = b \;\middle|\; \begin{array}{r} (\mathsf{m}_0,\mathsf{m}_1) \leftarrow \mathcal{A}(\mathsf{k}) \\ b \leftarrow \{0,1\} \\ \mathsf{r} \leftarrow\!\!\$ \{0,1\}^\lambda \\ \mathsf{c} \leftarrow \mathsf{com}_k(\mathsf{m}_b,\mathsf{r}) \\ b' \leftarrow \mathcal{A}(\mathsf{c}) \end{array} \right] = \frac{1}{2}$$

for any $\mathcal{A}$.

Computationally Binding: “PPT adversary cannot change its mind after sending c”

How should we formalise this?

"PPT adversary cannot change its mind after sending c"

Classical Definition

PPT $\mathcal{A}$ cannot find $(\mathsf{m}, \mathsf{r}, \mathsf{m}', \mathsf{r}')$ where $\mathsf{m} \neq \mathsf{m}'$ and

$$\mathsf{com}_{\mathsf{k}}(\mathsf{m}, \mathsf{r}) = \mathsf{com}_{\mathsf{k}}(\mathsf{m}', \mathsf{r}').$$

In particular, any collision-resistant hash function implies a binding commitment scheme.

Interlude

- A commitment scheme cannot be statistically hiding and statistically binding at the same time
  - If it is statistically hiding this means that for any $c = \mathrm{com}_k(m, r)$ there exists some r' such that $c = \mathrm{com}_k(m', r')$ for any m'.
  - If it is statistically binding this means that for any $c = \mathrm{com}_k(m, r)$ there exists no r' such that $c = \mathrm{com}_k(m', r)$ for any $m' \neq m$.
- Any IND-CPA secure encryption scheme is a hiding commitment scheme
- Any perfectly-correct encryption scheme is a binding commitment scheme, otherwise decryption might fail

"PPT adversary cannot change its mind after sending c"

Classical Definition

PPT $\mathcal{A}$ cannot find $(\mathsf{m}, \mathsf{r}, \mathsf{m}', \mathsf{r}')$ where $\mathsf{m} \neq \mathsf{m}'$ and

$$\mathsf{com}_\mathsf{k}(\mathsf{m}, \mathsf{r}) = \mathsf{com}_\mathsf{k}(\mathsf{m}', \mathsf{r}').$$

In particular, any collision-resistant hash function implies a binding commitment scheme.

Punchline

This is not true if $\mathcal{A}$ is a quantum adversary.

Attack on Classical Definition I

There exists a quantum-secure collision-resistant hash function H where $\mathcal{A}$ can open $\mathsf{com}_\mathsf{k}(\mathbf{m},\mathbf{r}) := H(\mathbf{m},\mathbf{r})$ to any $\mathbf{m}$.[5]

- Quantum adversary cannot find two pairs $(\mathbf{m},\mathbf{r}), (\mathbf{m}',\mathbf{r}')$ that agree on $\mathsf{com}_\mathsf{k}(\mathbf{m},\mathbf{r}) = \mathsf{com}_\mathsf{k}(\mathbf{m},\mathbf{r})$
- But it can open to some message $\mathbf{m}$ even if it learns it after sending $\mathbf{c}$.

Caveat

The attack depends on an oracle that we do not know how to build. But even with this oracle collision resistance holds.

[5] Andris Ambainis, Ansis Rosmanis, and Dominique Unruh. Quantum Attacks on Classical Proof Systems: The Hardness of Quantum Rewinding. In: *55th FOCS*. IEEE Computer Society Press, Oct. 2014, pp. 474–483. DOI: 10.1109/FOCS.2014.57; Dominique Unruh. Computationally Binding Quantum Commitments. In: *EUROCRYPT 2016, Part II*. ed. by Marc Fischlin and Jean-Sébastien Coron. Vol. 9666. LNCS. Springer, Berlin, Heidelberg, May 2016, pp. 497–527. DOI: 10.1007/978-3-662-49896-5_18.

Attack on Classical Definition II

1. Prepare a quantum state

$$|\phi\rangle := \left[\frac{\sum_{\mathsf{m},\mathsf{r}\in\{0,1\}^\lambda\times\{0,1\}^\lambda} |\mathsf{m}\rangle\,|\mathsf{r}\rangle}{\sqrt{2^{2\,\lambda}}}\right] \quad |0\rangle$$

2. Apply H on the first two registers and add result to the third

$$|\phi\rangle := \frac{\sum_{\mathsf{m},\mathsf{r}\in\{0,1\}^\lambda\times\{0,1\}^\lambda} |\mathsf{m}\rangle\,|\mathsf{r}\rangle\,|H(\mathsf{m},\mathsf{r})\rangle}{\sqrt{2^{2\,\lambda}}}$$

3. Measure the third register to obtain some value h

$$|\phi\rangle := \frac{\sum_{(\mathsf{m},\mathsf{r})\,|\,\mathsf{h}=H(\mathsf{m},\mathsf{r})} |\mathsf{m}\rangle\,|\mathsf{r}\rangle}{\sqrt{|\,\{(\mathsf{m},\mathsf{r})\mid h = H(\mathsf{m},\mathsf{r})\}\,|}}\,|\mathsf{h}\rangle$$

The first register now contains all preimages of h.

Attack on Classical Definition III

4. Use the magic oracle[6] to filter $\left\{(\mathsf{m}, \mathsf{r}) \mid \mathsf{h} = H(\mathsf{m}, \mathsf{r})\right\}$ to

$$\left\{(\mathsf{m}, \mathsf{r}) \mid \mathsf{h} = H(\mathsf{m}, \mathsf{r}) \wedge \mathsf{m} = \mathsf{m}_0\right\}$$

   for any chosen m_0.
5. Measure the first register to obtain $(\mathsf{m}_0, \mathsf{r})$ and submit as an opening.

Collision Resistance

This does not violate collision resistance because we are "using up" our state, i.e. we can only measure once, still.

[6] This is a variant of Grover's algorithm but we don't know how to implement the required steps.

Corrected Definition: Formalising the Attacker

Can write down our attacker like this:

Alice		Bob
	$\xleftarrow{\;k\;}$	
$\lvert S, M, R\rangle$	$\xrightarrow[\text{(commitment)}]{\;c\;}$	
··		
$\lvert S, M, R\rangle$ measure $\lvert S, M, R\rangle$ obtain $\mathbf{m}, \mathbf{r}$	$\xrightarrow[\text{(opening)}]{\;\mathbf{m}, \mathbf{r}\;}$	

Corrected Definition: What does ideal look like?

Collapse-binding Commitment

$b \leftarrow\!\!\$\ \{0,1\};\ k \leftarrow\!\!\$\ \{0,1\}^{\lambda}$

$\mathsf{c}, |\mathsf{S}, \mathsf{M}, \mathsf{R}\rangle \leftarrow \mathcal{A}(k)$

compute $|\mathsf{S}, \mathsf{M}, \mathsf{R}, V_{\mathsf{c}}(\mathsf{M}, \mathsf{R})\rangle$ // $V_{\mathsf{c}}(\mathsf{M}, \mathsf{R}) = 1$ iff $\mathsf{com}_{\mathsf{k}}(\mathsf{M}, \mathsf{R}) = \mathsf{c}$

measure $|V_{\mathsf{c}}(\mathsf{M}, \mathsf{R})\rangle = v$

// measurement has no effect if $|\mathsf{M}\rangle = |\mathsf{m}\rangle$, i.e. “collapsed”

if $v = 1 \wedge b = 0$ **then** measure $|\mathsf{M}\rangle$

$b' \leftarrow \mathcal{A}\left(|\mathsf{S}, \mathsf{M}, \mathsf{R}, V_{\mathsf{c}}(\mathsf{M}, \mathsf{R})\rangle\right)$

return $b = b'$

Dominique Unruh. Computationally Binding Quantum Commitments. In: *EUROCRYPT 2016, Part II*. ed. by Marc Fischlin and Jean-Sébastien Coron. Vol. 9666. LNCS. Springer, Berlin, Heidelberg, May 2016, pp. 497–527. DOI: 10.1007/978-3-662-49896-5_18

Collapsing Hash Functions

Collapsing Hash Function H

$b \leftarrow\!\!\$ \; \{0,1\}$

$|\psi\rangle_0 := |\mathsf{S}\rangle \sum_{\mathsf{x}} |\mathsf{x}, \mathbf{0}\rangle \leftarrow \mathcal{A}(H)$

$|\psi\rangle_1 := |\mathsf{S}\rangle \sum_{\mathsf{x}} |\mathsf{x}, H(\mathsf{x})\rangle$

if $b = 0$ **then**

  measure $|\mathsf{x}\rangle \in |\psi\rangle_1 \rightarrow |\psi\rangle_2$

else

  measure $|H(\mathsf{x})\rangle \in |\psi\rangle_1 \rightarrow |\psi\rangle_2$

$b' \leftarrow \mathcal{A}\left(|\psi\rangle_2\right)$

return $b = b'$

Figure 1: Collapsing Hash Function

Game indeed differs:

- $b = 0$: collapses to a single input-output pair
- $b = 1$: collapses to all preimages of measured value $H(\mathsf{x})$

[Unr16]: This implies collapse-binding commitments.

A New Classical Definition

Any somewhere statistically binding hash function is collapsing.

Somewhere Statistically Binding (SSB)

- Consider $H(\mathsf{x}_0 \mid \mathsf{x}_1 \mid \ldots \mid \mathsf{x}_{\ell-1})$
- There are "modes" $H^{(i)}(\mathsf{x}_0 \mid \mathsf{x}_1)$ that are **statistically binding** to block x_i
- We also have "index hiding": $H \approx_c H^{(i)} \approx_c H^{(j)}$ for any i, j.

- Since $H()$ is compressing it it cannot be statistically binding to its input
- But it can be be statistically binding for one small block
- If cannot tell which block it is statistically binding to, have an SSB hash function
- Can build this from a perfectly correct fully-homomorphic encryption scheme

Fin

If you take nothing else from this lecture: quantum computers won't solve hard problems instantly by just trying all solutions in parallel.

Credit: `https://scottaaronson.blog/`

References I

[ARU14] Andris Ambainis, Ansis Rosmanis, and Dominique Unruh. Quantum Attacks on Classical Proof Systems: The Hardness of Quantum Rewinding. In: *55th FOCS*. IEEE Computer Society Press, Oct. 2014, pp. 474–483. DOI: 10.1109/FOCS.2014.57.

[JNRV20] Samuel Jaques, Michael Naehrig, Martin Roetteler, and Fernando Virdia. Implementing Grover Oracles for Quantum Key Search on AES and LowMC. In: *EUROCRYPT 2020, Part II*. Ed. by Anne Canteaut and Yuval Ishai. Vol. 12106. LNCS. Springer, Cham, May 2020, pp. 280–310. DOI: 10.1007/978-3-030-45724-2_10.

[Unr16] Dominique Unruh. Computationally Binding Quantum Commitments. In: *EUROCRYPT 2016, Part II*. Ed. by Marc Fischlin and Jean-Sébastien Coron. Vol. 9666. LNCS. Springer, Berlin, Heidelberg, May 2016, pp. 497–527. DOI: 10.1007/978-3-662-49896-5_18.

References II

[YM08] Noson S Yanofsky and Mirco A Mannucci. Quantum Computing for Computer Scientists. Cambridge University Press, 2008.