Post-quantum cryptography

- Quantum computers can break public-key cryptography that is based on assuming hardness of factoring, discrete logs, and a few other problems.

- Post-quantum cryptography tries to design classical crypto schemes that cannot be broken efficiently by quantum algorithms.

- Classical codemakers vs quantum codebreakers.

- This tutorial: **Get to know your enemy!**
Quantum bits

- Richard Feynman, David Deutsch in early 1980s:

  Harness quantum effects for useful computations!

- Classical bit is 0 or 1; **quantum bit** is superposition of 0 and 1
  For example, can use an electron as qubit, with 0 = “spin up” and 1 = “spin down”

- 2 qubits is superposition of 4 basis states (00,01,10,11)
  3 qubits is superposition of 8 basis states (000,001, . . .)
  
  ...  
  1000 qubits: superposition of $2^{1000}$ states

- Massive space for computation! Easier said than done...
A bit of math: states

- 1-qubit basis states: $|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ and $|1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$

- Qubit: superposition $\alpha_0 |0\rangle + \alpha_1 |1\rangle = \begin{pmatrix} \alpha_0 \\ \alpha_1 \end{pmatrix} \in \mathbb{C}^2$

- 2-qubit basis state: $|10\rangle = |1\rangle \otimes |0\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$

- $n$-qubit state: $|\psi\rangle = \sum_{x \in \{0,1\}^n} \alpha_x |x\rangle \in \mathbb{C}^{2^n}$

- Axiom: measuring state $|\psi\rangle$ gives $|x\rangle$ with probability $|\alpha_x|^2$

- Hence $\sum_{x \in \{0,1\}^n} |\alpha_x|^2 = 1$, so $|\psi\rangle$ is a vector of length 1
A bit of math: operations

- Quantum operation maps quantum states to quantum states and is *linear* $\iff$ corresponds to *unitary* matrix.

- Example 1-qubit gates:
  
  $$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, \quad T = \begin{pmatrix} 1 & 0 \\ 0 & e^{\pi i/4} \end{pmatrix}$$

- More quantum: Hadamard gate $= \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$

  $$H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle), \quad H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$

  But $H\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) = \frac{1}{\sqrt{2}} H|0\rangle + \frac{1}{\sqrt{2}} H|1\rangle = |0\rangle$

  *Interference!*

- Controlled-NOT gate on 2 qubits: $|a, b\rangle \mapsto |a, a \oplus b\rangle$
Quantum circuits

- A classical Boolean circuit consists of AND, OR, and NOT gates on an \( n \)-bit register
- A quantum circuit consists of unitary quantum gates on an \( n \)-qubit register (allowing \( H \), \( T \), and CNOT gates suffices)

Example:

\[
|00\rangle \xrightarrow{H \otimes I} \frac{1}{\sqrt{2}}(|00\rangle + |10\rangle) \xrightarrow{\text{CNOT}} \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)
\]

This circuit creates an EPR-pair: entanglement!
Recap: From classical to quantum computation

- bits $\rightarrow$ qubits
- AND/OR/NOT gates $\rightarrow$ unitary quantum gates
- classical circuit $\rightarrow$ quantum circuit
- reading the output bit $\rightarrow$ measuring final state
Quantum mechanical computers

1. Start with all qubits in easily-preparable state (e.g. all $|0\rangle$)
2. Run a circuit that produces the right kind of interference: computational paths leading to correct output should interfere constructively, others should interfere destructively
3. Measurement of final state gives classical output

Two important questions:

1. Can we build such a computer?
2. What can it do?

This tutorial: 2nd question, focus on quantum algorithms
Quantum parallelism

- Suppose classical algorithm computes $f : \{0, 1\}^n \to \{0, 1\}^m$
- Convert this to quantum circuit $U : |x\rangle|0\rangle \mapsto |x\rangle|f(x)\rangle$
- We can now compute $f$ "on all inputs simultaneously"!

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

- This contains all $2^n$ function values!
- But observing gives only one random $|x\rangle|f(x)\rangle$
  All other information will be lost
- More tricks needed for successful quantum computation
  Interference!
Deutsch-Jozsa problem

- Given: function $f : \{0, 1\}^n \rightarrow \{0, 1\}$ ($2^n$ bits), s.t.
  
  (1) $f(x) = 0$ for all $x$ (constant),
  or
  
  (2) $f(x) = 0$ for $\frac{1}{2} \cdot 2^n$ of the $x$‘s (balanced)

- Question: is $f$ constant or balanced?

- Classically: need at least $\frac{1}{2} \cdot 2^n + 1$ steps ("queries" to $f$)

- Quantumly: $O(n)$ gates suffice, and only 1 query

- Query: application of unitary $O_f : |x, 0\rangle \mapsto |x, f(x)\rangle$

- More generally: $O_f : |x, b\rangle \mapsto |x, b \oplus f(x)\rangle$ ($b \in \{0, 1\}$)

- NB using $|\rightarrow\rangle = H|1\rangle$, we can get queried bit as a $\pm$-phase: $O_f |x\rangle |\rightarrow\rangle = (-1)^{f(x)} |x\rangle |\rightarrow\rangle$
Deutsch-Jozsa algorithm

- Starting state: $|0\ldots0\rangle|1\rangle$

- After first Hadamards: $\frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle|\rangle$

- Make one query: $\frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} (-1)^{f(x)} |x\rangle|\rangle$

- Forget about the last qubit $|--\rangle$
Deutsch-Jozsa (continued)

- After second Hadamard:

\[
\frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} (-1)^{f(x)} \frac{1}{\sqrt{2^n}} \sum_{y \in \{0,1\}^n} (-1)^{x \cdot y} |y\rangle
\]

- \( \alpha_{0 \ldots 0} = \frac{1}{2^n} \sum_{x \in \{0,1\}^n} (-1)^{f(x)} = \begin{cases} 1 & \text{if constant} \\ 0 & \text{if balanced} \end{cases} \)

- Measurement gives right answer with certainty

- Big quantum-classical separation: \( O(n) \) vs \( \Omega(2^n) \) steps

- But the problem is efficiently solvable by bounded-error classical algorithm (just query \( f \) at a few random \( x \) )
The meat of this tutorial: 4 quantum algorithms

1. Shor’s factoring algorithm

2. Grover’s search algorithm

3. Ambainis’s collision-finding algorithm

4. HHL algorithm for linear systems
Factoring

- Given $N = p \cdot q$, compute the prime factors $p$ and $q$
- Fundamental mathematical problem since Antiquity
- Fundamental computational problem on log $N$ bits
  - $15 = 3 \times 5$
  - $12140041 = 3413 \times 3557$
- Best known classical algorithms use time $2^{(\log N)^\alpha}$, where $\alpha = 1/2$ or $1/3$
- Its assumed computational hardness is basis of public-key cryptography (RSA)
- A quantum computer can break this, using Shor’s efficient quantum factoring algorithm!
Overview of Shor’s algorithm

- Classical reduction: choose random $x \in \{2, \ldots, N - 1\}$. It suffices to find period $r$ of $f(a) = x^a \mod N$

- Shor uses the quantum Fourier transform for period-finding

- Overall complexity: roughly $(\log N)^2$ elementary gates
Reduction to period-finding

- Pick a random integer \( x \in \{2, \ldots, N - 1\} \), s.t. \( \gcd(x, N) = 1 \)
- The sequence \( x^0, x^1, x^2, x^3, \ldots \mod N \) cycles:
  - has an unknown period \( r \) (\( \min r > 0 \) s.t. \( x^r \equiv 1 \mod N \))
- With probability \( \geq 1/4 \) (over the choice of \( x \)):
  - \( r \) is even and \( x^{r/2} \pm 1 \not\equiv 0 \mod N \)
- Then:
  \[
  x^r = (x^{r/2})^2 \equiv 1 \mod N \iff (x^{r/2} + 1)(x^{r/2} - 1) \equiv 0 \mod N \iff (x^{r/2} + 1)(x^{r/2} - 1) = kN \text{ for some } k
  \]
- \( x^{r/2} + 1 \) and \( x^{r/2} - 1 \) each share a factor with \( N \)
- This factor of \( N \) can be extracted using gcd-algorithm
Quantum Fourier transform

- **Fourier basis** (dimension $q$): $|\chi_j\rangle = \frac{1}{\sqrt{q}} \sum_{k=0}^{q-1} e^{\frac{2\pi i jk}{q}} |k\rangle$

Such a state is unentangled $|\chi_{j_0j_1j_2}\rangle = \frac{1}{\sqrt{8}} (|0\rangle + e^{2\pi i 0.j_2}|1\rangle) \otimes (|0\rangle + e^{2\pi i 0.j_1j_2}|1\rangle) \otimes (|0\rangle + e^{2\pi i 0.j_0j_1j_2}|1\rangle)$

- **Quantum Fourier Transform**: $|j\rangle \mapsto |\chi_j\rangle$

- If $q = 2^\ell$, then can implement this with $O(\ell^2)$ gates.

- For Shor: choose $q = 2^\ell$ in $(N^2, 2N^2]$
Easy case for the analysis: $r|q$

1. Apply QFT to 1st register of $|0\ldots0\rangle$ : 

$$\frac{1}{\sqrt{q}} \sum_{a=0}^{q-1} |a\rangle |0\rangle$$

2. Compute $f(a) = x^a \mod N$ (by repeated squaring)

$$\frac{1}{\sqrt{q}} \sum_{a=0}^{q-1} |a\rangle |x^a \mod N\rangle$$

3. Observing 2nd register gives $|x^s \mod N\rangle$ (random $s < r$)

1st register collapses to superposition of 

$$|s\rangle, |r + s\rangle, |2r + s\rangle, \ldots, |q - r + s\rangle$$
Easy case: \( r \mid q \) (continued)

Recall: 1st register is in superposition
\[ \sum_{j=0}^{q/r-1} |jr + s\rangle \]

4. Apply QFT once more:
\[
\sum_{j=0}^{q/r-1} \sum_{b=0}^{q-1} e^{2\pi i \frac{(jr+s)b}{q}} |b\rangle = \sum_{b=0}^{q-1} e^{2\pi i \frac{sb}{q}} \left( \sum_{j=0}^{q/r-1} e^{2\pi i \frac{rb}{q}j} \right) |b\rangle
\]

Sum \( \neq 0 \) iff \( e^{2\pi i \frac{rb}{q}} = 1 \) iff \( \frac{rb}{q} \) is an integer

Only the \( b \) that are multiples of \( \frac{q}{r} \) have non-zero amplitude!
Easy case: $r|q$ (continued)

5. Observe 1st register: random multiple $b = c \frac{q}{r}$, $c \in [0, r)$:

$$\frac{b}{q} = \frac{c}{r}$$

- $b$ and $q$ are known; $c$ and $r$ are unknown
- $c$ and $r$ are coprime with probability $\geq \frac{1}{\log \log r}$
- Then: we find $r$ by writing $\frac{b}{q}$ in lowest terms
- Since we can find $r$, we can find prime factors of $N$!

Hard case ($r \not|r q$) still works approximately: measurement gives $b$ s.t. $\frac{b}{q} \approx \frac{c}{r}$; we can find $r$ with some extra number theory
Summary for Shor’s algorithm

- Reduce factoring to finding the period $r$ of modular exponentiation function $f(a) = x^a \mod N$
- Use quantum Fourier transform to find a multiple of $q/r$, repeat a few times to find $r$
- Overall complexity:
  - QFT takes $O((\log q)^2) = O((\log N)^2)$ elementary gates
  - Modular exponentiation: $\approx (\log N)^2 \log \log N$ gates; classical computation by repeated squaring (use Schönhage-Strassen algo for fast multiplication)
  - Everything repeated $O(\log \log N)$ times
  - Classical postprocessing takes $O((\log N)^2)$ gates
- Roughly $(\log N)^2$ elementary gates in total
The search problem

- We want to search for some good item in an unordered $N$-element search space.
- Model this as function $f : \{0, 1\}^n \rightarrow \{0, 1\}$ ($N = 2^n$)
  
  $f(x) = 1$ if $x$ is a solution.

- We can query $f$:
  
  $O_f : |x\rangle|0\rangle \mapsto |x\rangle|f(x)\rangle$

  or

  $O_f : |x\rangle \mapsto (-1)^{f(x)}|x\rangle$

- Goal: find a solution.

- Classically this takes $O(N)$ steps (queries to $f$).

- Grover’s algorithm does it in $O(\sqrt{N})$ steps.
Grover’s algorithm

- Apply Grover iteration $G$ $k$ times on uniform starting state

- Idea: each iteration moves amplitude towards solutions
The good state and the bad state

- Suppose there are $t$ solutions
- Define “good” state and “bad” state:

$$|G\rangle = \frac{1}{\sqrt{t}} \sum_{x : f(x) = 1} |x\rangle \quad |B\rangle = \frac{1}{\sqrt{N - t}} \sum_{x : f(x) = 0} |x\rangle$$

- Initial uniform state is $|U\rangle = \sin(\theta)|G\rangle + \cos(\theta)|B\rangle$
  for $\theta = \arcsin(\sqrt{t/N})$
- All intermediate states will be in span$\{|G\rangle, |B\rangle\}$
- Grover iteration is a rotation over angle $2\theta$
  so after $k$ iterations the state is

$$\sin((2k + 1)\theta)|G\rangle + \cos((2k + 1)\theta)|B\rangle$$
One Grover iteration: rotation by $2\theta$

$$G = H^\otimes n RH^\otimes n \cdot O_f,$$ where $R$ reflects through $|0^n\rangle$

This $G$ is the product of two reflections:

1. $O_f$ reflects through $|B\rangle$
2. $H^\otimes n RH^\otimes n$ reflects through $|U\rangle$

Starting state: Reflect through $|B\rangle$: Reflect through $|U\rangle$:

\begin{align*}
\begin{array}{c}
|G\rangle \\
|U\rangle
\end{array} & \quad \begin{array}{c}
|G\rangle \\
|B\rangle
\end{array} & \quad \begin{array}{c}
|G\rangle \\
G|U\rangle
\end{array}
\end{align*}
How many iterations do we need?

- Success probability after $k$ iterations:
  \[
  \sin^2((2k + 1)\theta), \text{ with } \theta = \arcsin(\sqrt{t/N}) \approx \sqrt{t/N}
  \]

- If $k = \frac{\pi}{4\theta} - \frac{1}{2}$, then success probability is $\sin^2(\pi/2) = 1$

- Example: $t = N/4$ solutions $\Rightarrow k = 1$

- In general, round $k$ to nearest integer (incurs small error)

- Query complexity is $k \approx \frac{\pi}{4} \sqrt{N/t}$
  
  This is optimal for a quantum algorithm!

- Gate complexity is $O(\sqrt{N/t} \log N)$
Summary for Grover’s algorithm

- Quantum computers can search any $N$-element space with $t = \varepsilon N$ solutions, in $O(\sqrt{N/t}) = O(1/\sqrt{\varepsilon})$ iterations
  1. Set up uniform starting state $|U\rangle$
  2. Repeat the following $O(1/\sqrt{\varepsilon})$ times:
     2.1 Reflect through $|B\rangle$ (costs 1 query)
     2.2 Reflect through $|U\rangle$ (costs $O(\log N)$ gates)
  3. Measure final state to obtain an index $i$

- If we don’t know $\varepsilon = t/N$, we can try different guesses, still find a solution with expected number of $O(1/\sqrt{\varepsilon})$ iterations

- The algorithm has a small error probability, but can be modified to error 0 if we know $t$ exactly
Application: Speed up NP problems

- Given a propositional formula $f(x_1, \ldots, x_n)$
  Computable in time $\text{poly}(n)$

  Question: is $f$ satisfiable?

- This is a typical NP-complete problem
- Search space of $N = 2^n$ possibilities
- Classically: exhaustive search is the best we know.
  This takes about $N$ steps
- Quantumly: Grover finds a satisfying assignment in $\sqrt{N} \cdot \text{poly}(n)$ steps
- Because Grover is optimal, we believe that NP-hard problems cannot be efficiently computed by quantum algorithms
Classical random walks

- Explore a graph by moving to random neighbor in each step

- If $G$ is $d$-regular and connected: normalized adjacency matrix has “spectral gap” $\delta \in (0, 1)$. Starting from any vertex, $O(1/\delta)$ random walk steps produce uniform distribution

- Suppose an $\varepsilon$-fraction of the vertices are “marked” and we want to find such a marked vertex. Simple classical algorithm:
  1. Start at random vertex $v$ (setup cost $S$)
  2. Do the following $O(1/\varepsilon)$ times:
     2.1 Check if $v$ is marked (checking cost $C$)
     2.2 Rerandomize $v$ by $O(1/\delta)$ RW steps (step cost $U$)

This finds marked item w.h.p. Cost is $S + \frac{1}{\varepsilon} \left( C + \frac{1}{\delta} U \right)$
Quantum walks

- **Quantum walk**: walk in superposition over vertices (edges)
- Analogy with Grover’s algorithm:
  \[ |G\rangle = \text{uniform superposition over edges with marked endpoint} \]
  \[ |B\rangle = \text{uniform superposition over all other edges} \]
  \[ |U\rangle = \sin(\theta)|G\rangle + \cos(\theta)|B\rangle, \quad \theta = \arcsin\left(\frac{1}{\sqrt{\varepsilon}}\right) \]

1. Setup starting state \( |U\rangle \) (setup cost \( S \))
2. Repeat the following \( O(1/\sqrt{\varepsilon}) \) times:
   2.1 Reflect through \( |B\rangle \) (checking cost \( C \))
   2.2 Reflect through \( |U\rangle \)
      (can be implemented using \( 1/\sqrt{\delta} \) QW steps, each at cost \( U \))
3. Measure and check that resulting vertex is marked.

Correctness analogous to Grover. Cost is \( S + \frac{1}{\sqrt{\varepsilon}} \left( C + \frac{1}{\sqrt{\delta}} U \right) \)
Example: Ambainis’s algorithm (’03)

Suppose we want to find a collision in $h : [n] \rightarrow \mathbb{N}$

- $G =$Johnson graph: the vertices are the sets $R \subseteq [n]$ of size $r$. Edge between sets $R$ and $R'$ if they differ in 1 element
- Fraction of vertices of $G$ that contain collision: $\varepsilon \geq (r/n)^2$
- Known: spectral gap is $\delta \approx 1/r$
- With each vertex $R$, algorithm records $h(R)$; setup cost $S = r$; checking cost $C = 0$; update cost $U = O(1)$
- Total cost: $S + \frac{1}{\sqrt{\varepsilon}} \left( C + \frac{1}{\sqrt{\delta}} U \right) \overset{r=n^{2/3}}{=} O(n^{2/3})$
- Classically: $\Theta(n)$ $f$-evaluations needed

If $h$ is 2-to-1: run on random set of $\sqrt{n}$ inputs (whp 1 collision) to get complexity $O(n^{1/3})$

Classically: $\Theta(\sqrt{n})$ $f$-evaluations, by birthday paradox
HHL algorithm for “solving” large linear systems

- Solving large linear systems $Ax = b$ is one of the most important problems in science and engineering.

Goal: given matrix $A$ and vector $b$, find vector $x$

- Harrow-Hassidim-Lloyd'09: “solves” this problem exponentially faster by preparing state $|x\rangle$ IF system is well-behaved:

Assumptions

1. state $|b\rangle$ easy to prepare;
2. $A$ is well-conditioned: $\lambda_{max}/\lambda_{min}$ not too big;
3. unitary operation $e^{iA}$ is easy to apply (sparseness suffices)
How does the Harrow-Hassidim-Lloyd algorithm work?

- Input: Hermitian matrix $A \in \mathbb{R}^{N \times N}$ and vector $b \in \mathbb{R}^N$
  
  Goal: approximately prepare $|x\rangle$, where $Ax = b$

- Let $\nu_1, \ldots, \nu_N, \lambda_1, \ldots, \lambda_N$ be eigenvectors, eigenvalues of $A$

- HHL algorithm:
  
  1. Prepare quantum state $|b\rangle = \sum_{i=1}^N \beta_i |v_i\rangle$

  NB: applying $A^{-1}$ corresponds to multiplying with $\lambda_i^{-1}$

  2. Use eigenvalue estimation: $\sum_{i=1}^N \beta_i |v_i\rangle |\lambda_i\rangle$

  3. Make new qubit $\sum_{i=1}^N \beta_i |v_i\rangle |\lambda_i\rangle \left( \lambda_i^{-1} |0\rangle + \sqrt{1 - \lambda_i^{-2}} |1\rangle \right)$

  4. Uncompute $|\lambda_i\rangle$ by inverting eigenvalue estimation

  5. Amplify the $|0\rangle$-part to end with $\sum_{i=1}^N \beta_i \lambda_i^{-1} |v_i\rangle = |x\rangle$
What else can a quantum computer do?

- **Similar to Shor:** discrete logarithm, solve Pell’s equation, compute properties of number fields, ...

- **Similar to Grover:** maximum-finding, approximate counting, shortest paths in graphs, minimum spanning trees, ...

- **Similar to quantum walks:** finding small subgraphs, matrix-product verification, junta-testing, backtracking, ...

- **Similar to HHL:** quantum machine learning, principal component analysis, recommendation systems, ...

- Efficiently *simulating quantum-mechanical systems.*
  
  Could be very important for drug design, material sciences...
What quantum algorithms cannot do

- You can simulate every quantum algorithm with an exponentially slower classical computer.

  This implies that the set of computable problems doesn’t change: Church-Turing thesis remains intact.

- For many problems we can show that quantum computers give no significant speed-up, or at most a quadratic speed-up (e.g., Grover is optimal).

- NP-complete problems form a famous and important class of hard computational problems: satisfiability, Traveling Salesman Problem, protein folding,...

  Conjectured: quantum computers can’t efficiently solve them.
Conclusion

- Quantum mechanics is the best physical theory we have
- Fundamentally different from classical physics: superposition, interference, entanglement
- Quantum algorithms use these non-classical effects to solve some problems much faster
- We saw 4 important examples:
  1. Shor’s factoring algorithm
  2. Grover’s search algorithm
  3. Ambainis’s collision-finding algorithm
  4. HHL algorithm for linear systems

Much more left to be discovered...
Phase estimation

- Suppose we can apply $U$ and are given one of its eigenvectors $|v\rangle$ as a quantum state. Goal: learn eigenvalue $e^{2\pi i \theta}$
- Suppose phase $\theta = 0.\theta_1 \ldots \theta_\ell$ has $\ell$ bits of precision

- Remember QFT: $|j\rangle \mapsto |\chi_j\rangle = \frac{1}{\sqrt{2^\ell}} \sum_{k=0}^{2^\ell-1} e^{\frac{2\pi i jk}{2^\ell}} |k\rangle$

- Phase estimation algorithm:
  1. Start with $|0^\ell\rangle|v\rangle$
  2. Apply $H^{\otimes \ell}: \frac{1}{\sqrt{2^\ell}} \sum_{k\in\{0,1\}^\ell} |k\rangle|v\rangle$
  3. Conditioned on 1st register, apply $U^k$ to 2nd register:

$$\frac{1}{\sqrt{2^\ell}} \sum_{k\in\{0,1\}^\ell} |k\rangle e^{2\pi i \theta k} |v\rangle = \frac{1}{\sqrt{2^\ell}} \sum_{k\in\{0,1\}^\ell} e^{2\pi i \theta k} |k\rangle |v\rangle$$

  4. Inverse QFT on first register gives $j = \theta 2^\ell = \theta_1 \ldots \theta_\ell$

- With $O(1/\varepsilon)$ applications of $U$: $\varepsilon$-error approximation of $\theta$