Shor's Algorithm and Cryptography

rip discrete log

State of Cryptography

- Many cryptographic primitives based on one of two problems that are assumed to be hard

Factoring	Discrete Log
$N = p \cdot q$	y = g^x (mod N)
(where N is really big)	(where N is really big)

- Key exchange and public-key cryptography are oof'd
- Hash functions and symmetric cryptography are still safe
- Post-quantum cryptography (lattices, isogenies) is in the works :0

What is Shor's Algorithm?

- Supposed to be the main topic of this talk
- Quantum algorithm that makes factoring and discrete log really easy
- Actually practical!
    - Most quantum algorithms until then were very situational and required perfect black-boxes
    - Deutsch's algorithm: check if $f(0) = f(1)$
    - Bernstein-Vazirani: $f(x) = x \cdot s$, find s
    -
- Period-finding algorithm in disguise

Period Finding Problem

- Given some function f that repeats every s inputs (i.e. $f(x) = f(x+s)$), find s
- Measured in terms of query complexity, not time complexity
  - Intuition: usually we know how to compute f, but it's costly so we want to reduce it anyways
- How to do it classically?
  - The best we can do is to keep trying values in order until it repeats

Shor’s Algorithm: Preliminaries

Disclaimer: math :(

Factoring -> Period Finding

- Assume the RSA case where $N = p*q$
- Fun number theory fact: $a^{(\lambda(N))} \equiv 1 \pmod{N}$
  - $\lambda(N) = \mathrm{lcm}(p-1,q-1)$
- Do some math, we get:

$$a^{(\lambda(N))} = 1+kN$$

$$a^{(\lambda(N))} - 1 = kN$$

$$(a^{(\lambda(N)/2}-1) * (a^{(\lambda(N)/2)}+1) = kN$$

- With high probability, p and q get separated!
- Take gcd to get p and q to get factors

Factoring -> Period Finding: Example

- Let's try $N = 3*7 = 21$, $a = 2$
  - $\lambda(N) = (3-1)*(7-1) = 6$
- When we compute the thing from earlier:

From earlier:

$$(a^{(\lambda(N)/2}-1) * (a^{(\lambda(N)/2)}+1) = kN$$

$$(2^3-1)(2^3+1) = 7 * 9 = 63 = 21 * 3$$

- $\gcd(7, 21) = 7$, $\gcd(9, 21) = 3$
- It successfully split up p and q!
- How do we get $\lambda(N)$? It's exactly the period of $f(x) = a^x \bmod N$:

$$f(x) = a^x = a^x * a^{\lambda(N)} = a^{(x+\lambda(N))} = f(x+\lambda(N)) \pmod N$$

- Or just listing powers of 2 mod 21: 1, 2, 4, 8, 16, 11, 1

Discrete Log -> Period Finding

- Discrete log problem: given g, h, N, find x such that $g^x = h \pmod{N}$
- Tl;dr, construct this function:

$$f(a,b) = g^a \cdot h^b \pmod{N}$$

- It kind of has a periodic structure:

$$\begin{aligned} f(a+x,b-1) &= g^{(a+x)} \cdot h^{(b-1)} \\ &= g^a \cdot h^b \cdot g^x \cdot h^{(-1)} \\ &= g^a \cdot h^b \cdot h \cdot h^{(-1)} \\ &= g^a \cdot h^b \\ &= f(a,b) \end{aligned}$$

What is Classical Information Theory?

- Bits are only 0 or 1, qubits are randomly 0 or 1 – easy, right…?
- Classical systems can have randomness though
  - flipping a coin isn't quantum physics
- Think about the following states:
  - Start with a coin that's heads up
  - Flip it (but don't look at the result)
  - Flip it again
- We can describe the states using probability:
  - 100% heads
  - 50% heads, 50% tails
  - 50% heads, 50% tails

What is Quantum Information Theory?

- Qubits have *amplitudes*, not probabilities
- Usually written like α|0> + β|1>
  - |α|^2 gives you the *probability* that you measure the qubit as |0> (same for |β|^2)
  - α,β are *complex* numbers, i.e in the form a+bi
- Amplitudes have constructive and destructive interference!
- Quantum version of flipping a coin: the Hadamard Gate (H)
  - Puts 0 into an equal superposition of 0 and 1 (known as |+>)
  - Probability of measuring: (1/√2)^2 = 50%
- Consider the quantum version of the coin flipping thing:
  - Start with the qubit |0>
  - Hadamard the qubit
  - Hadamard the qubit again
- What happens to our state?

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

$$|1\rangle \mapsto \frac{|0\rangle - |1\rangle}{\sqrt{2}}$$

Quantum State Example

- Scenario:
  - Start with the qubit |0>
  - Hadamard the qubit
  - Hadamard the qubit again
- Our state is:
  - |0>
  - (|0>+|1>)/√2
  - **(** (|0>+|1>)/√2 + (|0>-|1>)/√2 **)**/√2 = (2(|0>)/√2)/√2 = |0>
- If we measure at the end, we always get 0
- The amplitudes constructively interfere at |0>, and destructively interfere at |1>

Hadamard

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

$$|1\rangle \mapsto \frac{|0\rangle - |1\rangle}{\sqrt{2}}$$

Entanglement

- In any scenario, values can be correlated but still unknown!
- Classical setting:
  - Basic gambling game: we flip a coin, if its heads I win \$1, otherwise I lose \$1
  - If you flip the coin but don't look at it, the state looks like 50% (heads, +\$1), 50% (tails, -\$1)
  - Learning something about one value (such as heads, or losing \$1) tells you about the other
- Quantum correlation = entanglement
  - We can correlate two qubits with a query: $|x\rangle|0\rangle \rightarrow |x\rangle|f(x)\rangle$
  - The most basic form is a CNOT, which does $f(0) = 0$ and $f(1) = 1$; so $|0\rangle \rightarrow |00\rangle$ and $|1\rangle \rightarrow |11\rangle$
  - The qubit doesn't need to be classical! $(|0\rangle+|1\rangle)/\sqrt{2} \rightarrow (|00\rangle+|11\rangle)/\sqrt{2}$
  - The qubits together are 50% 0 or 50% 1, but once you measure one you know what the other is
  - This is *distinctly* different from separately Hadamarding the two qubits! (no entanglement)

Basically...

- The distinction between quantum and classical things is in amplitudes
    - Specifically, constructive and destructive interference
- The only things we can do with qubits are apply operations, and measure
    - No specifying which measurement we want to see
    - No looking at all superposition states and calculating something across all of them
    - Quantum speedups *must* come from clever use of interference, entanglement, and measurements (remember we can change our behavior based on what we measure)

Shor’s Algorithm: Period Finding

Disclaimer: more math :(

(greatly oversimplified too)

Simplifications

- We usually talk about bits in larger contexts (bytes, ints, etc.)
  - Qubits easily generalize to larger contexts too, so we will use quantum registers not bits (basically the same idea, but they are more than just 0 and 1)
- f (the periodic function) can be any periodic function
  - add the restriction that f has to be exactly periodic (makes the math easier)
    - The *only* time two values are equal is if f has cycled
    - Basically, not this: 1,1,1,0,1,1,1,0,1,1,1,0, etc.
  - For clarity, we will use f(x) = a^x mod N (we will use a=2, N=21)

Shor’s Algorithm

- Start out by getting equal superposition of everything: |0> + |1> ... + |N-1>
- Use one query to f for entanglement:

  |0>|1> + |1>|2> + |2>|4> + |3>|8> + |4>|16> + |5>|11> + |6>|1>...
- Measure the second register! Suppose we get 1:

  |0>|1> + |6>|1> + |12>|1> ...
- Now suppose we got 2:

  |1>|2> + |7>|2> + |13>|2> ...
- Regardless of what we measured, the state will look like:

  |x>|f(x)> + |x+s>|f(x)> + |x+2s>|f(x)> ...
- We ignore the second register afterwards

Quantum Fourier Transform (QFT)

- Operation that roughly maps |x> to |0> + |x> + |2x> + |3x>.., but mod N
    - If N = 4 and x = 2, then this maps |2> ➔ |2> + |0> + |2> + |0>
- There are weird complex amplitudes that make this really wack (but are needed to make this a valid quantum operation)
- Intuitively, it takes a value x and gives you a cycle where the values are separated by s, but with weird complex amplitudes!
- How does this help us find s in Shor’s algorithm?

Shor’s Algorithm, continued

- Answer: inverse QFT!
  - QFT is very spooky, turns out inverse QFT is basically just QFT
  - We have a cycle where things are separated by s, taking the inverse should undo the cycle and give us s!
- Lol sike, quantum is not that simple
  - we had nice looking amplitudes, but for the inverse to work perfectly we needed the wacky complex ones
  - Also the cycle starts at something that’s not 0 (turns out this doesn’t matter, actually)
- Basically, we actually get *another* cycle that looks kind of like

$$|0\rangle + |s\rangle + |2s\rangle \ldots$$

- What happens if we measure?

Shor's Algorithm, the finale

- Measuring the output state gives us a multiple of s
- We can just run this part multiple times, and take the gcd of our answers!
  - With very high probability, this gives us s, the period
- Recall from earlier: the period allows us to break factoring and discrete log!

<u>Factoring</u>

(a^(λ(N)/2-1) * (a^(λ(N)/2)+1) = kN

This splits up p and q, usually

<u>Discrete Log</u>

f(a+x,b-1) = f(a,b)

The period, x, *is* the discrete log

- How long does this take? Uh, apparently $O\big((\log N)^2(\log\log N)(\log\log\log N)\big)$

Are We Screwed?

no

- Quantum computers are still very unreliable, and factoring large numbers that we use today will be hard for a long time
  - We could still record all the key exchanges, encryptions, etc. that are happening today, and wait until when breaking them is feasible though
- Symmetric cryptography is still secure, and post-quantum cryptography is advancing very quickly
  - We think – always the possibility that we are still too dumb to figure it out
- Even if P = NP and we can do everything in polynomial time, so what?
  - Polynomial is still not necessarily fast: O(n^3) when n = 10^4 already takes over a day to compute
  - O(n^3) is also pretty good – imagine cryptosystems where the best known attacks are O(n^6)

Totally Valid Questions bc of Simplification

- Why only RSA and discrete log? And why not hash functions, symmetric cryptography, lattices, isogenes, etc.
- Aren’t there multiple periods for the discrete log function?
- How does QFT actually work? (and how is it also inverse QFT?)
- How do we just “put everything into superposition”?
- How do we compute f?
- Quantum circuits need to be polynomial in size too, right?
- What happens if f is not exactly periodic?
- Why the “with high probability” in Shor’s algorithm?