Shor's Algorithm

Andrey Vlasov

Outline

The RSA problem

Shor's algorithm

Implementing the algorithm

So why isn't RSA broken?

Section 1

The RSA problem

The RSA problem

- Given the public keys N and e, and the publicly transmitted ciphertext $C \equiv M^e \pmod{N}$, find the original message M.
- Currently an open problem, but it's believed that factoring N is the best approach.
- Factoring N will also give us the private key d, so we can decode any messages sent with this key pair.
- Thus, our problem becomes: Given a semiprime $N = pq$, find the prime factors p and q of N.

Section 2

Shor's algorithm

A definition

- Let a and N be integers such that a is coprime to N.
- The order of $a \pmod N$ is the smallest positive integer r such that $a^r \equiv 1 \pmod N$.
- By Euler's Theorem, $a^{\phi(N)} \equiv 1 \pmod N$. Thus, $r | \phi(N)$.
- In Shor's algorithm, the problem of factoring N is replaced with the problem of finding r.

Step 1: RNG

- Pick a random number a with $1 < a < N$, and calculate $\gcd(a, N)$.
- If $\gcd(a, N) \neq 1$, we got really lucky. Since N has only two factors, $p = \gcd(a, N)$ and $q = N/p$.
- If $\gcd(a, N) = 1$, we move on to step 2.

Step 2: Order-Finding

- Find the multiplicative order of $a \pmod N$ and label it r.
- $a^r \equiv 1 \pmod N \Rightarrow\ N|(a^r - 1) \Rightarrow\ N|(a^{\frac{r}{2}} - 1)(a^{\frac{r}{2}} + 1)$
- If r is odd, we can't use this formula, so we go back to step 1.
- It's impossible that $N|(a^{\frac{r}{2}} - 1)$, because then $a^{\frac{r}{2}} \equiv 1 \pmod N$, but the order of $a \pmod N$ is $r > \frac{r}{2}$.
- Ideally, N shares a factor with $(a^{\frac{r}{2}} - 1)$ that's less than N.

Step 3: Factor

- Compute $g = \gcd(N, a^{\frac{r}{2}} - 1)$.
- If $g = 1$, then $N|(a^{\frac{r}{2}} + 1)$, which tells us nothing about the factors of N. So, we go back to step 1.
- If $g > 1$, then g is one of the prime factors of N, and the other factor is N/g.
- In his original paper, Shor proved that the probability of the algorithm working, given a random a, is at least $\frac{1}{2}$. The exact probability is an open problem!

Section 3

Implementing the algorithm

Classical part

- A classical computer is used for steps 1 and 3, because it can calculate the gcd of two numbers very quickly.
- In fact, this calculation never needs more steps than 5 times the number of digits!

Proof of gcd complexity

- Claim: Worst-case for Euclidean algorithm is two Fibonacci numbers
- More specifically, if the algorithm requires N steps to find $\gcd(a, b)$ with $a > b$, then $a \geq F_{N+2}$ and $b \geq F_{N+1}$
- Base case: $N = 1$. The smallest pair is $a = 2 = F_3$ and $b = 1 = F_2$
- Inductive step: Assume claim is true for all $N \leq K - 1$
- The first step in an algorithm with K steps is $a = qb + r$. Consider the pair b and r $(b > r)$, which takes $K - 1$ steps
- By the inductive hypothesis, $b \geq F_{K+1}$ and $r \geq F_K$
- $a = qb + r \geq b + r \geq F_{K+1} + F_K = F_{K+2}$

Proof of gcd complexity

- A property of the golden ratio: $\phi^2 = \phi + 1$
- This gives a property of Fibonacci numbers: $\phi^N = \phi F_N + F_{N-1}$
- $\phi^N = \phi F_N + F_{N-1} \leq \phi F_N + F_N = (\phi + 1) F_N = \phi^2 F_N$
- $\phi^{N-2} \leq F_N$

Proof of gcd complexity

- If the Euclidean algorithm takes $N-1$ steps, then $b \geq F_N \geq \phi^{N-2}$
- $N-2 \leq \log_\phi(b)$ and $\log_{10}(\phi) > \frac{1}{5}$, so

$$\frac{N-2}{5} \leq \log_{10}(\phi)\log_\phi(b) = \log_{10}(b)$$

$$N-1 \leq 1+5\log_{10}(b) < 5\log_{10}(b)$$

- Time complexity is $O(\log(b))$ - only depends on the smaller number!

Quantum part

- Finding the order of a number is hard for classical computers.
- The best known algorithm (repeated squaring) is subexponential.
- So, we plug the output from step 1 into a quantum computer.
- The quantum circuit complexity is $O(n^3)$ - much better!

Section 4

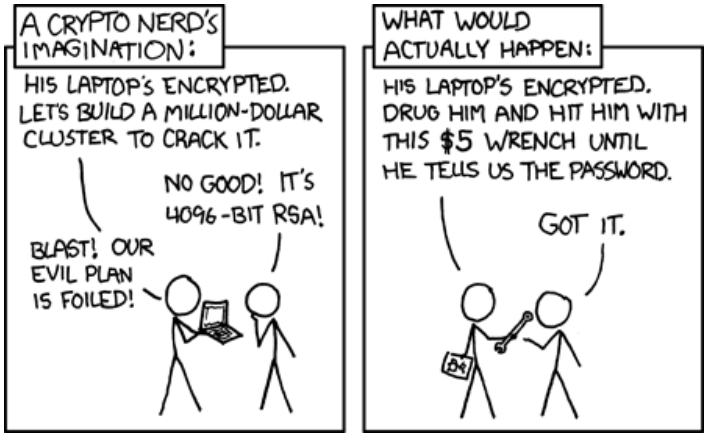
So why isn't RSA broken?

So why isn't RSA broken?

- Shor's algorithm needs perfect qubits, with no errors.
- In real quantum systems, errors are really hard to correct.
  (see meetings 3 and 4 from Fall '23)
- Order-finding algorithm requires $2n + 3$ qubits for an n-bit number.
- For 2048-bit RSA, would need 4,099 perfect qubits.
- Current best: IBM's Condor with 1,121 noisy qubits.

Questions?

Relevant xkcd



Bibliography

Peter Shor.

Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer.

https://arxiv.org/pdf/quant-ph/9508027.pdf, 1996.

Thomas Wong.

Introduction to Classical and Quantum Computing.

https://www.thomaswong.net/introduction-to-classical-and-quantum-computing-1e3p.pdf, 2022.