Cryptography and quantum computing

Cryptography

- Cryptography relies on the **assumption of <u>consistently</u> hard problems**
  - NP-completeness is not suitable for cryptography
  - Average complexity is not good enough for cryptography
  - We cannot prove that these problems are hard
- Hard problems can give rise to **trapdoor functions**
  - Given *x*, finding *f(x)* is easy
  - Given *f(x)*, finding *x* is hard
  - **Given f(x) and a secret k, finding x is easy**

Hard problem #1: Integer factorization

- Given pq, find p and q (large primes)
- *Not* believed to be NP-hard
- Best algorithm: Number sieves
  - Exponential time, but much faster than brute force
  - Very large primes are needed to maintain security
- Used to create RSA

Hard problem #1: Integer factorization -> RSA

- Public parameters: $n = pq$ (but not p and q)
- Alice generates e and d such that

$$ed \equiv 1 \ (mod\ (p-1)(q-1))$$

- e is the public key. Encryption of message m:

$$Enc(m) = m^e \ mod\ n$$

- Decryption of ciphertext x:

$$Dec(x) = x^d \ mod\ n$$

- This works because of Euler's theorem:

$$m^{ed} \equiv m \ (mod\ n)$$

If it was easy to find p and q, it would be easy to find d given *e!*

Trapdoor: only easy if given d

Hard problem #2: Discrete logarithm

- Given $m^x \bmod p$, p and x, find m
- Computing logarithm is easy normally
- However, it is (assumed) hard in modulo space
- Similarly, not believed to be NP-hard
- Used to create Diffie-Hellman Key Exchange and ElGamal Encryption

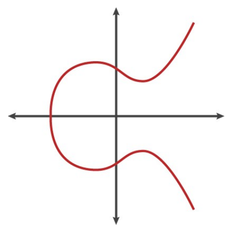
Hard problem #2: Discrete logarithm -> DH

- Diffie-Hellman key exchange:
- Public parameter prime p, generator g
  - Generator means that $\{g \bmod p, g^2 \bmod p, \ldots, g^{p-1} \bmod p\}$ are all different numbers
- Alice chooses random integer $A < p$, sends $g^A \bmod p$
- Bob chooses random integer $B < p$, sends $g^B \bmod p$
- Now both of them can compute a shared secret $g^{AB} \bmod p$
  - No one else can compute it

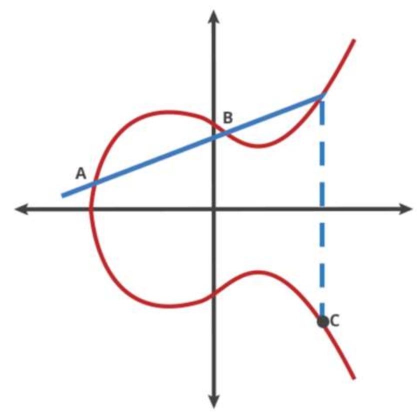
If it was easy to find A given $g^A \bmod p$, an attacker could compute the shared secret too

Hard problem #3: Elliptic Curves

- We can define new “addition” operations on elliptic curves:



- “Addition” is now:
  - Draw a line from A to B
  - Reflect it along the curve
  - The result is C = A+B
  - What is A + A?
  - What is nA?



Nick Sullivan, CloudFlare blog

Hard problem #3: Elliptic Curves

- Everything is done on finite fields instead of real numbers
  - Easy example of finite fields: integer modulo p
  - Maths on board
- Hard problem (Elliptic Curve Discrete Logarithm): Given A and B, find nA = B
  - Brute force = keep adding A until it becomes B
  - Baby step giant step algorithms: For $k = \sqrt{n}$, there must exist n_1 and n_2 such that
$$n = n_1 + n_2 k$$
  - We can **compute and store** all values of n_1 and then test all possible values for n_2 to get $\sqrt{n}$ computation time

Hard problem #3: Elliptic Curves -> ECDH

- We can rebuild the Diffie-Hellman key exchange using ECC
- Public parameters: elliptic curve, point P on curve
- Alice chooses a, sends aP
- Bob chooses b, sends bP
- Shared secret is abP

Quantum computers

- None of these problems are hard under quantum cryptography
  - Is this a coincidence…?
- Euclid's algorithm: given a and b, it is easy to find gcd(a, b)
- A trick for integer factorization of N:
  - Start with a guess a; determine if it is coprime with N (Euclid's algorithm)
  - If it is coprime, then there exists r where:

$$a^r \equiv 1 \ (mod\ N)$$

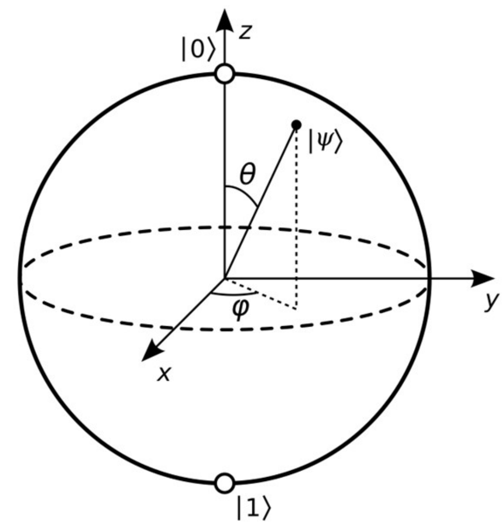
  - Furthermore if r is even (else restart the algorithm), we have

$$N \mid (a^{\frac{r}{2}} - 1)\ (a^{\frac{r}{2}} + 1)$$

  - Use Euclid's algorithm to compute $\gcd(N, a^{\frac{r}{2}} + 1)$; if it is N, restart

Quantum computers

- Let's start with some quantum computing basics
- A qubit is a superposition of states 0 and 1, and can be represented as a point on Bloch's sphere:
$$|\psi\rangle = \cos\theta|0\rangle + e^{i\varphi}\sin\theta\,|1\rangle$$
- Two qubits can be entangled, e.g.:
$$\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$$
- If we measure the second qubit, it will collapse the first qubit into the same value!



Incredible properties of quantum computers

- There is a Hadamard gate that can transform a string of zeroes into the superposition of all possibilities:

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

- Quantum computers can implement all classical functions, so we can use a second set of qubits to superimpose all possible solutions:

$$\frac{1}{\sqrt{q}}\sum_{a=0}^{q}|a\rangle\,|f(a)\rangle$$

- Measuring will still only give us one solution, but we can take advantage of this

Shor’s algorithm

- Shor’s algorithm can quickly find the smallest r where $a^r \equiv 1 \pmod N$
- We put all possible values of r in the first set of registers, and apply $a^x \bmod N$ to the second set of registers
- Now when we measure the second register and obtain f(s), it will **collapse** the first registers into a superposition of
$$|s\rangle, |r+s\rangle, |2r+s\rangle, \ldots$$
- We can then use a well known algorithm called *quantum phase estimation* to obtain r!

Shor’s algorithm

- Shor’s algorithm takes trivial time and requires 2n+3 logical qubits (where n is the number of bits in the key)
  - But Gidney and Eker (2021) estimate it needs about 20 million noisy “physical qubits”
- A slight alteration allows it to also break the discrete log problem as well as the elliptic curve problem
- Basically all of public key cryptography would break!

Post-quantum cryptography

- There are other hard problems which have no quantum solution at all
  - This is no guarantee for the future, though experts would be surprised to see a quantum solution
- Shortest vector problem: Given an n-dimensional lattice, find the shortest vector in that lattice
- This hard problem can be used to build another trapdoor, creating cryptography based on Learning With Errors (not related to AI)