Public Key Cryptography

Lecture 13

Quantum Cryptography

Index

- early 1980s: the roots of quantum cryptography – Conjugate Coding by Wiesner
- early 1980s: Benioff proposed a quantum mechanical model of the Turing machine
- 1984: Bennett and Brassard produced BB84 – the first quantum cryptography protocol
- 1991: Bennett et al. made operable the first experimental prototype based on this protocol
- 1998: Chuang, Gershenfeld and Kubinec created the first two-qubit quantum computer

- 2019: IBM launched Q System One, the first circuit-based commercial quantum computer (20-qubit)
- 2019: Google's Sycamore performed a task in about 200 seconds that would have taken the fastest supercomputer about 10000 years
- 2022: IBM's Osprey processor has 433 qubits
- 2024: D-Wave's Advantage 2 quantum annealer has 4400+ qubits
- 2024: Google's Willow completed a computation in under 5 minutes that would take today's fastest supercomputer (Frontier) approximately 10 septillion (10^{25}) years

- Classical computer:
  A classical bit can store either a 1 or a 0, and when measured the value observed will always be the stored value 0 or 1.
- Quantum computer:
  A *qubit* is in the 1 or the 0 state when it is measured.
  The differences between the qubit and the bit come from what sort of information a qubit can store when it is not being measured.

Any particle with a spin-$1/2$ characteristic can be characterized by its spin value, which when measured is either $+1/2$ or $-1/2$. This spin-$1/2$ particle which behaves in a quantum manner could be the fundamental building block of a quantum computer.

- **Superposition**
  Qubits exist in multiple states simultaneously.

- **Entanglement**
  Qubits are interconnected, affecting each other's state instantly.

- **Decoherence**
  Qubits are very sensitive to their environment and may lose their quantum properties.
  Decoherence leads to loss of information, making it difficult to scale up quantum systems.

- The state of a qubit may be described by a state vector in a complex linear vector space (Hilbert space).
- The usual axes of coordinates from the 3-dimensional real vector space are replaced by some perpendicular axes which correspond to each possible state that the system can be measured in.
- The Hilbert Space for a single qubit has 2 perpendicular axes, corresponding to the qubit being in the 1 and 0 states. These states which the vector can be measured to be are referred to as *eigenstates*.
- The vector which exists somewhere in this space which represents the state of our qubit is called the *state vector*.
- In general, the state of a qubit can be any combination of the base states.

Standard notation from Quantum Physics.

- The state vector is denoted by $|x\rangle$ and called a *ket vector*, where x is a list of numbers which contain information about the projection of the state vector onto its base states.
- The term *ket* and this notation come from the physicist Paul Dirac who wanted a concise shorthand way of writing formulas that occur in Quantum Physics.
- These formulas frequently took the form of the product of a row vector with a column vector.
- He referred to row vectors as *bra vectors* represented as $\langle y|$.
- The product of a *bra* and a *ket* vector would be written $\langle y|x\rangle$, and would be referred to as a *bracket*.

According to quantum physics a quantum system can exist in a mix of all of its allowed states simultaneously. This mixing of states is called *quantum superposition*, and it is key to the power of the quantum computer.

The qubit

- Let x_1 and x_0 be the eigenstates corresponding to the 1 state and to the 0 state respectively.

- Let X be the total state of our state vector, and let w_1 and w_0 be the complex numbers that weight the contribution of the base states to our total state. Then in general:

$$|X\rangle = w_0 * |x_0\rangle + w_1 * |x_1\rangle \equiv (w_0, w_1)$$

- The weighting factors w_0 and w_1 are complex numbers, and when the state of X is measured, we are guaranteed to find it to be in either the state:

$$0 * |x_0\rangle + w_1 * |x_1\rangle \equiv (0, w_1)$$

or the state

$$w_0 * |x_0\rangle + 0 * |x_1\rangle \equiv (w_0, 0)$$

- We can further restrict our state vector to be a unit vector (i.e., it has length 1) in a Hilbert space.

- A quantum system is not constrained to be a two state system, much of the above discussion for a 2-state quantum system is applicable to a general n-state quantum system.

- In an n state system our Hilbert Space has n perpendicular axes, or eigenstates, which represent the possible states the system can be measured in.

- As with the 2-state system, when we measure our n-state quantum system, we will always find it to be in exactly one of the n states, and not a superposition of the n states.

- The system is still allowed to exist in any superposition of the n states while it is not being measured.

- Mathematically, a 2-state quantum system with coordinate axes x_0, x_1 can be fully described by:

$$|X\rangle = w_0 * |x_0\rangle + w_1 * |x_1\rangle \equiv (w_0, w_1)$$

- Then an *n*-state quantum system with coordinate axes $x_0, x_1, \ldots, x_{n-1}$ can be fully described by:

$$|X\rangle = \sum_{k=0}^{n-1} w_k * |x_k\rangle$$

- In general a quantum system with *n* base states can be represented by the *n* complex numbers w_0 to w_{n-1}.
- When this is done the state may be written as:

$$|X\rangle = \begin{pmatrix} w_0 \\ w_1 \\ \vdots \\ w_{n-1} \end{pmatrix}$$

where w_k refers to the complex weighting factor for the *k*'th eigenstate.

- We can construct a *quantum memory register* out of the qubits previously described.
- We may store any number x in our quantum memory register as long as we have enough qubits, just as we may store any number x in a classical register as long as we have enough classical bits to represent that number.
- The state of the quantum register with n states is given by the formula above.
- In general a quantum register composed of m qubits requires 2^m complex numbers to completely describe its state: an m qubit register can be measured to be in one of 2^m states, and each state requires one complex number to represent the projection of that total state onto that state.
- In contrast, a classical register composed of m bits requires only m integers to fully describe its state.

- One can store an exponentially greater amount of information in a quantum register than in a classical memory register of the same number of (qu)bits.
- So, some intractable problems may become tractable!
- Most modern quantum algorithms rely on *quantum parallelism*, which arises from the ability of a quantum memory register to exist in a superposition of base states.
- A quantum memory register can exist in a superposition of states, each component of this superposition may be thought of as a single argument to a function.
- Since the number of possible states is 2^n, where n is the number of qubits in the quantum register, you can perform in one operation on a quantum computer what would take an exponential number of operations on a classical computer.

Shor's algorithm finds a hidden period of a function and is based on the Quantum Fourier Transform.

Let us first see how to derive the unknown factors of a composite number $n = pq$ from the multiplicative order $ord(x)$ of an element $x \in \mathbb{Z}_n^*$ (i.e. the smallest non-zero natural number r such that $x^r = 1$).

The multiplicative order of x is also the least period of the function defined by $f(a) = x^a$ mod n.

Choose a random integer x with $1 < x < n$. If $gcd(x, n) \neq 1$, then $gcd(x, n)$ is either p or q and the unknown factors are found. But this probability is very small if the prime factors are large and x is uniformly randomly chosen.

Assume that $gcd(x, n) = 1$. Then x is invertible modulo n, and by Lagrange's Theorem its order must divide the order of the group of invertible elements in $\mathbb{Z}_n$, hence

$$r = ord(x) \mid \varphi(n) = (p-1)(q-1).$$

By definition, $x^r = 1 \pmod{n}$. If r is even, then

$$x^r - 1 = (x^{r/2} - 1)(x^{r/2} + 1) = 0 \pmod{n},$$

and thus $n \mid (x^{r/2} - 1)(x^{r/2} + 1)$.
Since $ord(x) \neq r/2$, we have $n \nmid (x^{r/2} - 1)$ and there are two possibilities:
(1) p divides one of the two factors and q divides the other. In this case $gcd(x^{r/2} + 1, n)$ gives p or q.
(2) $n \mid (x^{r/2} + 1)$. In this case the algorithm fails and one has to choose another base x.

The algorithm is successful if r is even and $n \nmid (x^{r/2} + 1)$. A closer analysis shows that the probability of success is at least 50%.

Example. Let $n = 77$ and $x = 3$. Then $gcd(x, n) = 1$. Suppose we know the order of x mod n in the multiplicative group $\mathbb{Z}_n^*$, namely $r = 30$, which is an even number. We have:

$$x^{r/2} + 1 = 3^{15} + 1 = 35 \bmod 77.$$

We compute

$$gcd(x^{r/2} + 1, n) = gcd(35, 77) = 7,$$

and we obtain one of the prime factors of n. Note that

$$gcd(x^{r/2} - 1, n) = gcd(33, 77) = 11$$

gives the other factor of n.

- It has not been proven that factoring large numbers cannot be achieved on a classical computer in polynomial time.
- The fastest algorithm publicly available for factoring a large number n runs in $O(e^{c(\log n)^{1/3}*(\log \log n)^{2/3}})$.
- Shor (1994): polynomial time algorithm for factoring large numbers on a quantum computer
- Shor's algorithm runs in $O((\log n)^2 * \log \log n)$ on a quantum computer, and then must perform $O(\log n)$ steps of post processing on a classical computer

- It is known that if $gcd(x, n) = 1$, then $f(a) = x^a \bmod n$ is periodic (say with period r).
- One would like to factor a large integer n.
- Since $x^r \equiv 1 \pmod{n}$, it follows that

$$(x^{r/2} - 1)(x^{r/2} + 1) \equiv 0 \pmod{n}.$$

  If $|x^{r/2}| \neq 1$, then both $gcd(x^{r/2} - 1, n)$ and $gcd(x^{r/2} + 1, n)$ give a factor of n.
- Shor's algorithm tries to find r.

- It creates a quantum memory register with two parts. In the first part it places a superposition of the integers which are to be a's in the $x^a \bmod n$ function, namely 0 through $q - 1$, where q is the power of 2 such that $n^2 \leq q < 2n^2$. Then it calculates $x^a \bmod n$, where a is the superposition of the states, and places the result in the second part.

- We must calculate $x^a \bmod n$ an exponential number of times.

- The algorithm measures the state of the second register, which contains the superposition of all possible outcomes for f. Measurement of the second part results in exactly one value k, and causes the other partition to collapse into a superposition of the base states whose evaluation in $x^a \bmod n$ produce k.

- Since f is periodic, the first part of the register will contain the values $c, c + r, c + 2r \ldots$, where c is the lowest integer such that $x^c \bmod n = k$.

- The next step is to perform a discrete Fourier transform (DFT) on the contents of the first part of the register. DFT is a bijective $\mathbb{C}$-linear map on $\mathbb{C}^n$ which maps a sequence of n complex numbers (in the discrete time domain) to the frequency domain. The resulting vector of complex Fourier coefficients reveals the periodic structure of the input data. This has the effect of peaking the probability amplitudes of the first part of the register at integer multiples of the quantity q/r.

- Measuring the first part of the quantum register will yield an integer multiple of the inverse of the period with high probability. A classical computer can do analysis of this number, make a guess as to the actual value of r, and from that compute the possible factors of n.

Shor's Algorithm

- Input: a composite number n.
- Output: a non-trivial factor of n.
- Algorithm:
  1. Check if the number n is prime, even or a power of a prime. There are efficient classical methods for determining this and, if yes, providing factors for n.
     This step would be done on a classical computer.
  2. Pick an integer q that is the power of 2 such that $n^2 \leq q < 2n^2$.
     This step would be done on a classical computer.
  3. Pick a random integer x that is relatively prime to n.
     This step would be done on a classical computer.

Shor's Algorithm (continued)

4 Create a quantum register and partition it into two sets, reg1 and reg2.
Thus the state of our quantum computer can be given by:
$| \, reg1, reg2 \rangle$.
reg1 and reg2 must have enough qubits to represent integers as large as $q - 1$ and $n - 1$ respectively.

5 Load reg1 with an equally weighted superposition of all integers from 0 to $q - 1$.
Load reg2 with the 0 state.
This operation would be performed by our quantum computer.
Now the total state of the quantum memory register is:

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

Shor's Algorithm (continued)

6 Apply the transformation x^a mod n to each number stored in reg1 and store the result in reg2.
Due to quantum parallelism this will take only one step, as the quantum computer will only calculate $x^{|a\rangle}$ mod n, where $|a\rangle$ is the superposition of states created in step 5.
This step is performed on the quantum computer.
Now the state of the quantum memory register is:

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

Shor's Algorithm (continued)

7 Measure reg2 and observe some value k.
   This has the side effect of collapsing reg1 into an equal
   superposition of each value a between 0 and $q - 1$ such that

   $$x^a \bmod n = k$$

   This operation is performed by the quantum computer.
   Now the state of the quantum memory register is:

   $$\frac{1}{\sqrt{|A|}} \sum_{a'=a \in A} \left| a', k \right\rangle$$

   where A is the set of a's such that $x^a \bmod n = k$, and $|A|$ is
   the number of elements of A.

8 Compute the discrete Fourier transform on reg1.
The discrete Fourier transform when applied to a state $|a\rangle$ changes it in the following manner:

$$|a\rangle = \frac{1}{\sqrt{q}} \sum_{c=0}^{q-1} |c\rangle * e^{2\pi iac/q}$$

This step is performed by the quantum computer in one step through quantum parallelism.
After the discrete Fourier transform our register is in the state:

$$\frac{1}{\sqrt{|A|}} \sum_{a' \in A} \frac{1}{\sqrt{q}} \sum_{c=0}^{q-1} |c, k\rangle * e^{2\pi ia'c/q}$$

Shor's Algorithm (continued)

9. Measure the state of reg1, and call this value m.

   This integer m has a very high probability of being a multiple of q/r, where r is the desired period.

   This step is performed by the quantum computer.

10. Take the value m, and on a classical computer do some post processing which calculates r based on knowledge of m and q.

    There are many ways to do this post processing, they are complex and are omitted here for clarity in presentation.

    This post processing is done on a classical computer.

11. Once you have attained r, a factor of n can be determined by taking $\gcd(x^{r/2} + 1, n)$ and $\gcd(x^{r/2} - 1, n)$.

    If you have found a factor of n, then stop, if not go to step 4.

    This final step is done on a classical computer.

Remark. Shor's algorithm requires around $3size(n)$ qubits and uses $O(size(n)^3)$ operations.

Shor's algorithm can efficiently solve the Integer Factorization problem.

The Discrete Logarithm problem can also be solved with a period-finding algorithm. This can be applied to the multiplicative group of integers modulo a prime number and also to the group of points on an elliptic curve over a finite field.

Hence classical public-key cryptography is broken once sufficiently large and stable quantum computers become available.

On the other hand, symmetric algorithms are less severely affected by the capabilities of quantum computers. Grover's algorithm (or quantum search algorithm) from 1996 provides a speedup which effectively halves the security level, but attacking 256-bit AES with 128-bit post-quantum security is still unfeasible.

Selective Bibliography

📄 M. Hayward, *Quantum computing and Shor's algorithm.*
[http://alumni.imsa.edu/ matth/quant/299/paper/]

📄 H. Knospe, *A Course in Cryptography*, Amer. Math. Soc., Rhode
Island, 2019.