CPEN 400Q / EECE 571Q Lecture 12
RSA, order-finding and Shor's algorithm

Thursday 17 February 2022

Announcements

- Assignment 3 available sometime soon
  - second-last assignment
  - will be due $\sim$2 weeks after reading week

No class or formal office hours during reading week (still available for appointment).

We implemented the quantum Fourier transform using a *polynomial* number of gates:

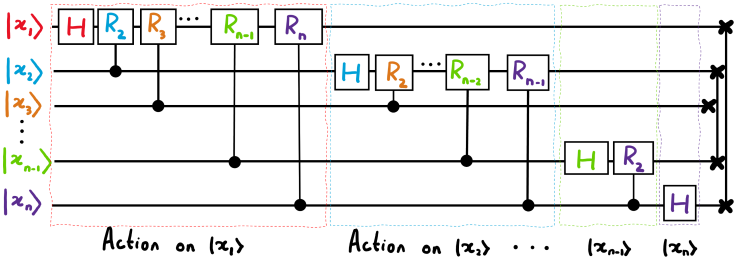


Image credit: Xanadu Quantum Codebook node F.3

We used the QFT in the quantum phase estimation subroutine to estimate the eigenvalues of unitary matrices.

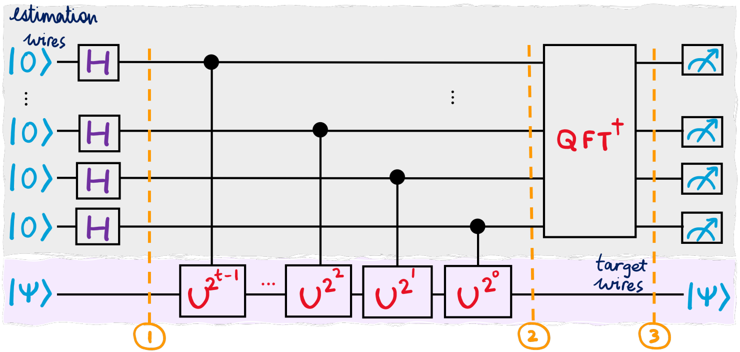


Image credit: Xanadu Quantum Codebook node P.2

Learning outcomes

- Describe the steps involved in the RSA cryptosystem, and identify the vulnerability to quantum computers
- Use QPE to implement the order finding algorithm
- Implement both classical and quantum components of Shor's algorithm to factor prime numbers

RSA

Public-key cryptosystems

Two different types of cryptosystems:

- Symmetric: the key used to decode is the same as (or can easily be obtained from) the one used to encode
- Asymmetric / public-key: the key used to decode is different than the one used to encode

Both types are used in modern infrastructure and every system has its own advantages/disadvantages, attack vectors, and vulnerabilities.

RSA (Rivest–Shamir–Adleman) is public-key cryptosystem.

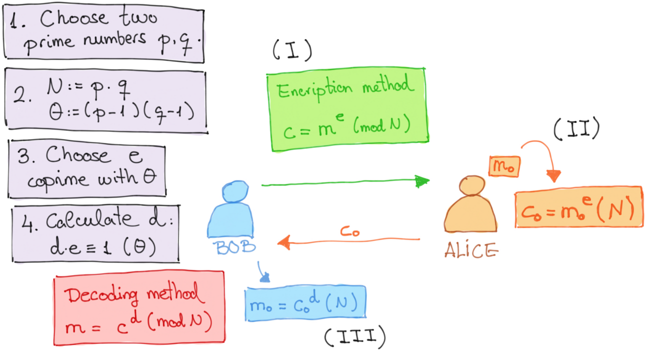


Image credit: Xanadu Quantum Codebook node S.5

RSA: brief review of number theory concepts

The math behind RSA involves modular arithmetic and a few other number-theoretic ideas:

Greatest common divisor (gcd): $gcd(a, b)$ is the largest integer factor that divides perfectly into both a and b

RSA: brief review of number theory concepts

The math behind RSA involves modular arithmetic and a few other number-theoretic ideas:

Greatest common divisor (gcd): $gcd(a, b)$ is the largest integer factor that divides perfectly into both a and b

Co-prime: a and b are co-prime if $gcd(a, b) = 1$.

RSA: brief review of number theory concepts

The math behind RSA involves modular arithmetic and a few other number-theoretic ideas:

Greatest common divisor (gcd): $gcd(a, b)$ is the largest integer factor that divides perfectly into both a and b

Co-prime: a and b are co-prime if $gcd(a, b) = 1$.

Modular inverse: Given a number a and modulus N, the inverse of a is the integer b such that $ab \equiv 1 \bmod N$. This exists *only if a and N are co-prime*.

RSA: brief review of number theory concepts

The math behind RSA involves modular arithmetic and a few other number-theoretic ideas:

Greatest common divisor (gcd): $gcd(a, b)$ is the largest integer factor that divides perfectly into both a and b

Co-prime: a and b are co-prime if $gcd(a, b) = 1$.

Modular inverse: Given a number a and modulus N, the inverse of a is the integer b such that $ab \equiv 1 \mod N$. This exists *only if a and N are co-prime*.

Fermat's little theorem: if p prime and a is an integer, then $a^p = a \mod p$. If a is not divisible by p, then $a^{p-1} \equiv 1 \mod p$.

Step 1

Choose two *prime numbers*, p and q.

Step 2

Compute:

$$N = p \cdot q$$
$$\theta = (p-1)(q-1)$$

Step 3

Choose a value e that is *co-prime* with θ.

Step 4

Compute the inverse of e mod θ, i.e., find d s.t.

$$d \cdot e \equiv 1 \bmod \theta$$

The *public key* is the pair (e, N).

The *private key* is the pair (d, N).

Encoding

Acquire the public key (e, N) of the party you wish to send something to. To send the message m, encode it as

$$c = m^d \bmod N$$

Decoding

If you receive c and have the private key (d, N), decode like so:

$$c^e \bmod N = (m^d)^e \bmod N = m$$

Two cases to consider to understand why this works.

Since $ed = 1 \bmod \theta$, there exists integer k such that $ed = 1 + k\theta$.

Case 1: m co-prime with N

$$\begin{aligned} c^e \bmod N &= (m^d)^e \bmod N \\ &= m^{1+k\theta} \bmod N \\ &= mm^{k\theta} \bmod N \end{aligned}$$

It is a known result in number theory that if m and N are co-prime, then $m^\theta \equiv 1 \bmod N$ where $\theta = (p-1)(q-1)$. Thus,

$$\begin{aligned} c^e \bmod N &= mm^{k\theta} \bmod N \\ &= m \bmod N \end{aligned}$$

Case 2: m not co-prime with N

Then, $gcd(m, N) > 1$. Must be p or q, because those are the only two factors of N.

Suppose $gcd(m, N) = p$. Then, p also divides m,

$$m \equiv 0 \bmod p, \quad m^{ed} \equiv 0 \equiv m \bmod p$$

But q does not. q is prime, so $gcd(q, m) = 1$. So by Fermat's little theorem,

$$m^{(q-1)} \equiv 1 \bmod q, \quad m^{(p-1)(q-1)} = m^{\theta} \equiv 1 \bmod q$$

Case 2: m not co-prime with N

Again, since $ed = 1 \bmod \theta$, there exists k such that $ed = 1 + k\theta$.

$$\begin{aligned} m^{ed} \bmod q &= mm^{k\theta} \bmod q \\ &= m \bmod q \end{aligned}$$

So we have

$$\begin{aligned} m^{ed} &\equiv m \bmod p \\ m^{ed} &\equiv m \bmod q \end{aligned}$$

It follows that

$$m^{ed} \equiv m \bmod N$$

- To decrypt the message, we must learn d
- We know e, and that $de \equiv 1 \bmod \theta$

But we don't know θ! We have only e, and N.

However, N and θ are based on the same two prime numbers:

$$N = p \cdot q$$
$$\theta = (p-1)(q-1)$$

So if we can factor $N = pq$, then we can decode the message!

The security of RSA relies on the fact that factoring for large numbers is computationally intractable.

Computational complexity of breaking RSA

Current recommended key (N) sizes are 2048- and 4096-bit.

The largest key size cracked so far is 829 bits in 2020. See:
`https://en.wikipedia.org/wiki/RSA_numbers`

Best-known classical algorithm:

General number field sieve

From Wikipedia, the free encyclopedia

In number theory, the **general number field sieve** (**GNFS**) is the most efficient classical algorithm known for factoring integers larger than 10^{100}. Heuristically, its complexity for factoring an integer n (consisting of $\lfloor \log_2 n \rfloor + 1$ bits) is of the form

$$\exp\left(\left(\sqrt[3]{\frac{64}{9}} + o(1)\right)(\ln n)^{\frac{1}{3}}(\ln \ln n)^{\frac{2}{3}}\right) = L_n\left[\frac{1}{3}, \sqrt[3]{\frac{64}{9}}\right]$$

On a quantum computer, there exists an algorithm that can help us solve the problem in *polynomial time*.

But it is still going to be a long time before that happens.

RSA-2048	**Old estimates**				**Current estimates**			
p_g	n_ℓ	n_p	quantum resources	time	n_ℓ	n_p	quantum resources	time
10^{-3}	6190	19.20	1.17	1.46	8194	22.27	0.27	0.34
10^{-5}	6190	9.66	0.34	0.84	8194	8.70	0.06	0.15

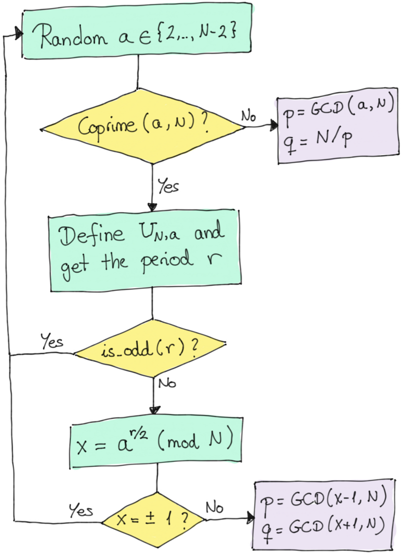
Table 2. RSA-2048 security estimates. Here n_ℓ denotes the number of logical qubits, n_p denotes the number of physical qubits (in millions), time denotes the expected time (in hours) to break the scheme, and *quantum resources* (*quantum resources*) are expressed in units of megaqubitdays. The corresponding classical security parameter is 112 bits.

Image: V. Gheorghiu and M. Mosca, *A resource estimation framework for quantum attacks against cryptographic functions - recent developments.* Feb. 15 2021.

Shor's algorithm

Shor's algorithm has both classical and quantum parts.

A quantum computer is used to compute the value of a specific order, which can then be used to extract the values of p, and q.



Non-trivial square roots

What we are ultimately searching for is a *non-trivial square root* of N, i.e., some $x \neq \pm 1$ such that

$$x^2 \equiv 1 \bmod N.$$

If we find such an x, then we know

$$\begin{aligned} x^2 &\equiv 1 \bmod N \\ x^2 - 1 &\equiv 0 \bmod N \\ (x-1)(x+1) &\equiv 0 \bmod N \end{aligned}$$

This means that

$$(x-1)(x+1) = kN$$

for some integer k.

Non-trivial square roots

If

$$(x-1)(x+1) = kN = kpq,$$

then $x-1$ is a multiple of one of p or q, and $x+1$ is a multiple of the other. If

$$x - 1 = sp$$
$$x + 1 = tq$$

we can compute the values of p and q by finding their *gcd* with N:

$$p = gcd(x-1, N)$$
$$q = gcd(x+1, N)$$

But... how do we find such an x?

Non-trivial square roots and factoring

It's actually okay to find any *even* power of x for which this holds:

$$x^r = x^{2r'} = (x^{r'})^2 \equiv 1 \bmod N$$

Let's define the function

$$f_{N,a}(m) = a^m \bmod N$$

for some integer value a, and ask the following question: for which m does

$$f_{N,a}(m) = a^m \equiv 1 \bmod N$$

The minimum integer m for which this holds is the *order* (or period) of a. If we could *find* this period, and it is an even number, then we can find an x and factor N.

Where is the quantum?

Classically, we wanted to find an m such that

$$f_{N,a}(m) = a^m \equiv 1 \bmod N$$

Let's define a unitary operation that performs

$$U_{N,a}|k\rangle = |ak \bmod N\rangle$$

If m is the period of a, and we apply U_{N_a} m times,

$$U^m_{N,a}|k\rangle = |a^m k \bmod N\rangle = |k\rangle$$

So m is also the period of $U_{N,a}$! We can find it efficiently using a quantum computer.

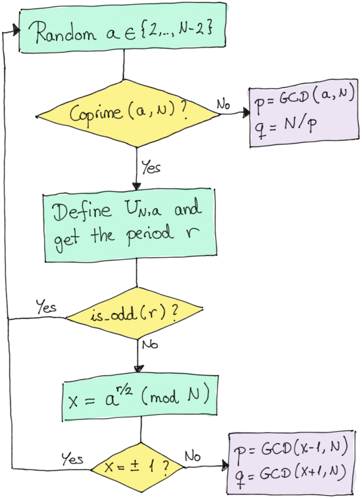


Image credit: Xanadu Quantum Codebook node S.4

Let U be an operator and $|\phi\rangle$ any state. How do we find the minimum r such that

$$U^r|\phi\rangle = |\phi\rangle$$

QPE does the trick if we set things up in a clever way:

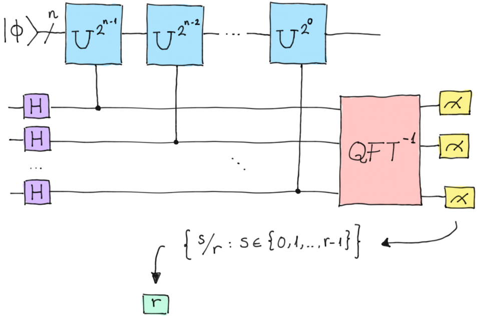


Image credit: Xanadu Quantum Codebook node S.3

Consider the state

$$|\Psi_0\rangle = \frac{1}{\sqrt{r}}\left(|\phi\rangle + U|\phi\rangle + \cdots + U^{r-1}|\phi\rangle\right)$$

If we apply U to this:

$$\begin{aligned} U|\Psi_0\rangle &= \frac{1}{\sqrt{r}}\left(U|\phi\rangle + U^2|\phi\rangle + \cdots + U^r|\phi\rangle\right) \\ &= \frac{1}{\sqrt{r}}\left(U|\phi\rangle + U^2|\phi\rangle + \cdots + |\phi\rangle\right) \\ &= |\Psi_0\rangle \end{aligned}$$

Order finding on a quantum computer

Now consider the state

$$|\Psi_1\rangle = \frac{1}{\sqrt{r}}\left(|\phi\rangle + e^{-\frac{2\pi i}{r}}U|\phi\rangle + e^{-2\cdot\frac{2\pi i}{r}}U^2|\phi\rangle + \cdots + e^{-(r-1)\cdot\frac{2\pi i}{r}}U^{r-1}|\phi\rangle\right)$$

If we apply U to this:

$$\begin{aligned}
U|\Psi_1\rangle &= \frac{1}{\sqrt{r}}\left(U|\phi\rangle + e^{-\frac{2\pi i}{r}}U^2|\phi\rangle + \cdots + e^{-(r-1)\cdot\frac{2\pi i}{r}}U^{r}|\phi\rangle\right)\\
&= \frac{1}{\sqrt{r}}\left(U|\phi\rangle + e^{-\frac{2\pi i}{r}}U^2|\phi\rangle + \cdots + e^{\frac{2\pi i}{r}}|\phi\rangle\right)\\
&= e^{\frac{2\pi i}{r}}\frac{1}{\sqrt{r}}\left(e^{-\frac{2\pi i}{r}}U|\phi\rangle + e^{-2\frac{2\pi i}{r}}U^2|\phi\rangle + \cdots + |\phi\rangle\right)\\
&= e^{\frac{2\pi i}{r}}|\Psi_1\rangle
\end{aligned}$$

This generalizes to $|\Psi_s\rangle$

$$|\Psi_s\rangle = \frac{1}{\sqrt{r}}(|\phi\rangle + e^{-s\frac{2\pi i}{r}}U|\phi\rangle + e^{-2s\cdot\frac{2\pi i}{r}}U^2|\phi\rangle + \cdots + e^{-(r-1)s\cdot\frac{2\pi i}{r}}U^{r-1}|\phi\rangle)$$

It has eigenvalue

$$U|\Psi_s\rangle = e^{\frac{2\pi i s}{r}}|\Psi_s\rangle$$

Idea: if we can create *any* one of these $|\Psi_s\rangle$, we could run QPE and get an estimate for s/r, and then recover r.

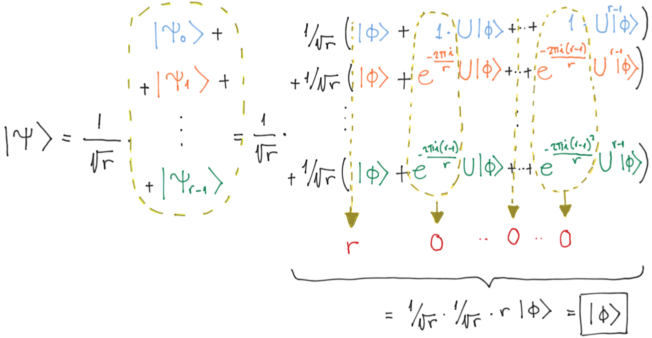
Problem: to construct any $|\Psi_s\rangle$, we would need to know r in advance!

Solution: construct the uniform superposition of all of them.

$$|\Psi\rangle = \frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} |\Psi_s\rangle$$

But what does this equal?

The superposition of all $|\Psi_s\rangle$ is just our original state $|\phi\rangle$!



If we run QPE, the output phase we get will be s/r for one of these states.

Image credit: Xanadu Quantum Codebook node S.3

Shor's algorithm

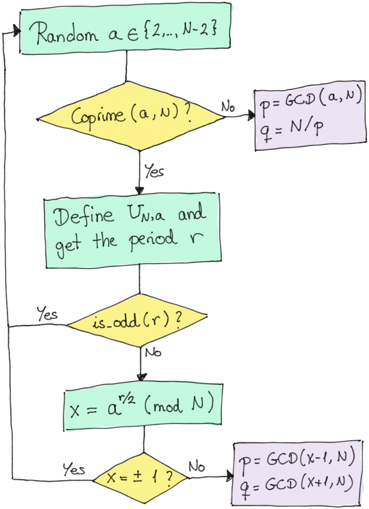
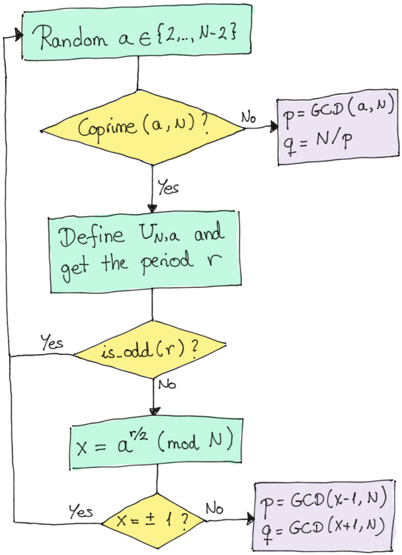


Image credit: Xanadu Quantum Codebook node S.4

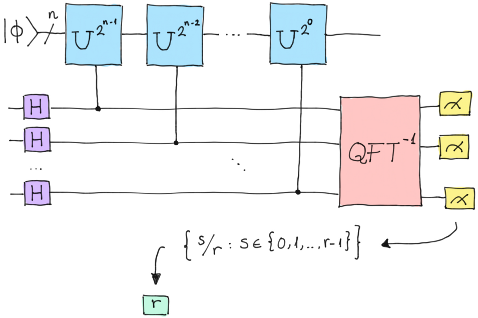
GCD: polynomial w/Euclid's algorithm

Modular exponentiation: can use exponentiation by squaring, other methods to reduce number of operations and memory required



Is this really efficient?

Quantum part: let $L = \lceil \log_2 N \rceil$.



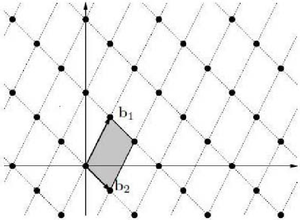
QFT: polynomial in number of qubits $O(L^2)$

Controlled-U gates: implemented using something called *modular exponentiation* in $O(L^3)$ gates.

What can / should we do about this?

RSA will not be cracked tomorrow: needs far more quantum resources than we have today. BUT, it's only a matter of time. We need to use this time wisely to re-tool our infrastructure. One option: post-quantum cryptography.

Figure 2 A 2-dimensional lattice generated by the basis $B = [b_1, b_2]$

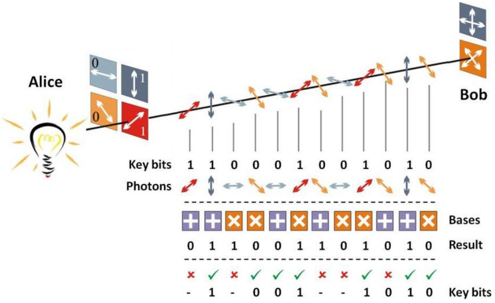


Use hard problems that *don't* have (known) efficient quantum solutions.

Image credit: G. Zhang, J. Qin. *Lattice-based threshold cryptography and its applications in distributed cloud computing*. Int. J. High Perform. Comput. Netw. 2015

What can / should we do about this?

Symmetric crypto remains largely secure; use quantum key distribution to perform key exchange for it rather than RSA.



Theoretically secure, but can be challenging to implement, and other potential attack vectors such as hardware.

Image credit: Carrasco-Casado, Marmol, Denisenko. (2016) *Free-Space Quantum Key Distribution*. Optical Wireless Communications - An Emerging Technology (pp.589-607)

Next time

Content:

- Moving onto variational algorithms (rest of the course)

Action items:

1. Start working on prototype implementation for project

Recommended reading:

- Codebook nodes S.1-S.5
- Nielsen & Chuang 5.3, Appendix A.5