The RSA Cryptosystem and Shor's factoring algorithm

Quantum Algorithms using Qniverse

Jothishwaran C A

Department of Electronics and Communication Engineering
Indian Institute of Technology Roorkee

Glossary

- **<u>Prime Number</u>:** A natural number p is prime if it is only divisible by 1 and itself. A number that is not prime is called **composite**.

- **<u>Greatest Common Divisor (GCD)</u>:** Given two natural numbers p, q; $\mathrm{GCD}(p, q)$ is the largest number that can divide both p and q.

  If $\mathrm{GCD}(p, q) = 1$ then p and q are said to be **coprime**.

- **<u>Modulo</u>:** This refers to the remainder obtained when dividing one natural number by another. Example, $5\ mod\ 2 = 1$, $13\ mod\ 5\ =\ 3$.

It can be seen that the values of the remainder when dividing by n can take values from 0 to $n - 1$. The concept of modulo generalizes to integers as well. However, we shall concern ourselves with natural numbers here.

- **<u>Modular Arithmetic</u>:** By definition it is possible to assign a unique modulus value to every natural number divided by n to partition all natural numbers into n distinct equivalence classes depending on the value of the remainder after division, i.e. for a given integer a, n:

$$a \equiv b\ mod\ n \Rightarrow a = m \cdot n + b$$

Where m is an integer and b is an integer such that $0 \leq b < n$. One may perform arithmetic with these modular values:

If $a \equiv\ b\ mod\ n$,

$$a + k \equiv (b + k)\ mod\ n$$
$$ka \equiv (kb)\ mod\ n$$
$$a^k \equiv \left(b^k\right)\ mod\ n$$

For any integer a, b and natural number k. Finally, if n divides a, then $a \equiv 0\ mod\ n$.

The RSA cryptosystem

1. Key generation:

   - Let $\boldsymbol{p}$ and $\boldsymbol{q}$ be two prime numbers and $\boldsymbol{n} = \boldsymbol{p} \cdot \boldsymbol{q}$.
   - Compute $\boldsymbol{\lambda(n) = \mathrm{LCM}(p-1, q-1)}$. → LCM = least common multiple
   - Choose a number $1 < e < \lambda(n)$, such that $\mathrm{GCD}(e, \lambda(n)) = 1$, i.e. $\boldsymbol{\lambda(n)}$ **is coprime to** $\boldsymbol{e}$.
   - Calculate d, such that $e \cdot d \equiv 1 \; mod \; \lambda(n)$, i.e. $\boldsymbol{d}$ **is the inverse of** $\boldsymbol{e}$ **modulo** $\boldsymbol{n}$.
   - n and e are made publicly available. $\boldsymbol{n}$ is known as the RSA number and $\boldsymbol{e}$ is the public key.
   - $\boldsymbol{d}$ is stored as a secret key.

2. Encryption of a number m:

   - Let the plaintext be a number m.
   - The number is encrypted by calculating the following quantity:

$$\boldsymbol{c = m^e \; mod \; n}$$

one-way function

   - This new number c is the ciphertext obtained from m.

$$e \cdot d \equiv 1 \mod \lambda(n)$$

$$e \cdot d = m \cdot \lambda(n) + 1$$

$$\Rightarrow e \cdot d - 1 = m \cdot \lambda(n)$$

The RSA cryptosystem [contd.]

3. <u>Decryption of the number c</u>:

   - The receiver of the ciphertext c can recover the original plaintext m and the secret key d using the following method:

$$\boldsymbol{m = c^d \bmod n}$$

   - This method was a widely used technique for public key exchange.

4. <u>Caveats</u>:

   - The technique described above is reliable only when when $0 \leq m < n$.
   - It is necessary for p and q to be very large to make this process practically secure.
   - Additionally, the prime numbers must be chosen at random. Any structured approach for finding the primes might enable an adversary to guess the primes using the same methods.

Why RSA works?

- The reason for the RSA cryptosystem stems form the fact that the modular exponentiation function $f_a(x) = a^x \ mod\ n$ is periodic for all natural numbers a, n and x.

- If a and n are coprime, then there exist values of r such that: Euler's theorem

$$a^r \equiv 1 \ mod\ n \quad \Rightarrow \quad a^r = k \cdot n + 1$$

- In this case, the smallest value of r that obeys this property is the period of $f_a(x)$. This statement is a consequence of modular arithmetic.

- In the case of RSA, the quantity $\boldsymbol{\lambda(n)}$ is the smallest such value. It is also known as the reduced totient function.

Why RSA works? [contd.]

- Assuming that m is coprime to the RSA number n, the ciphertext c is given as:

$$c = m^e \; mod \; n$$

- During the decryption process, we evaluate $c^d \; mod \; n$, since $e \cdot d \equiv 1 \; mod \; \lambda(n)$, this implies:

$$e \cdot d = k \cdot \lambda(n) + 1$$

For some integer k, this implies the quantity calculated during the decryption is the following

$$c^d \; mod \; n = \; (m^e)^d \; mod \; n = (\underbrace{m^{k \cdot \lambda(n)}}_{(m^{\lambda(n)})^k} \cdot m) \; mod \; n = m \; mod \; n$$

- If $0 \leq m \; < n$, then, $m \; mod \; n = m$. It is also possible to prove the working or RSA for case where m is not coprime to n. However, we shall not be covering that here.

What does it take to break RSA?

- The only publicly known parameters in RSA are the RSA number $\boldsymbol{n}$, and the public key $\boldsymbol{e}$.
- The only way to know anything further is to know the value of $\boldsymbol{\lambda(n)} = LCM((p-1)(q-1))$.
- But in order to find the value of $\lambda(n)$ is to factorize the RSA number.
- <u>*Therefore the only challenge in completely break the RSA cryptosystem is **integer factorization**.*</u>
- While integer factorization is not a hard problem, if the chosen number is large enough, the process of factorization will become extremely time consuming.

Factorization $\rightarrow$ Sub-exponential time complexity

not NP-Hard

A strategy for factorizing RSA numbers

- The let us consider a toy example with $n = 143$. We may choose $a = 21$ as the number coprime to n.
- Let us observe the values of the modular exponents of 21 with respect to 143.

r	$21^r \bmod 143$
1	21
2	12
3	109
4	1
5	21
6	12
7	109
8	1
9	21

A strategy for factorizing RSA numbers [contd.]

$143 = p \cdot q$

$(21^2 + 1) \cdot (21^2 - 1)$

$= K \cdot 143$

- From the previous table that, $\lambda(143) = 4$. This implies:

$$21^4 \equiv 1 \; mod \; 143$$

- This means, $(21^4 - 1) \; \equiv 0 \; mod \; 143 \Rightarrow (21^2 \; - 1)(21^2 + 1) \equiv 0 \; mod \; 143.$

$a^2 - b^2 = (a+b)(a-b)$

442 440

- It can be verified that 143 divides neither $(21^2 + 1)$ nor $(21^2 - 1)$, this means that for the above statement to be true, $(21^2 \pm \; 1)$ each contains one factor 143.

- By evaluating $\mathrm{GCD}\left((21^2 - \; 1), 143\right) = \mathrm{GCD}(440{,}143) = 11$. Therefore, 11 is one factor of 143, the other factor may be evaluated as $\frac{143}{11}$ = 13.

- The GCD evaluation may be done using the Extended Euclidean algorithm.

Shor's Algorithm and Quantum Parallelism

RSA - 2048

n - 2048 bits

- The previously described method is the classical version of Shor's algorithm.
- Shor's algorithm leverages quantum parallelism (or the linearity of quantum time evolution) to calculate all the values of $f_a(x)$ for a large number of values of x.

$U|\psi\rangle = U|\phi_1\rangle + U|\phi_2\rangle$

- This effectively creates a quantum state that contains all the information of the table shown before.
- To this quantum state, one may apply the Quantum Fourier Transform (QFT) to find its period.
- Once the period is known, we may proceed to evaluate the factors of the RSA number $\boldsymbol{n}$ using classical means.

Shor's Algorithm: Requirements

- As seen before, any value of a that may be used in the Shor's algorithm must be coprime to n.

$a \in (n^2, 2n^2)$ n^2 elements with $(n+1)$ multiples of n

- The periodicity of the modular exponential function ($\boldsymbol{r}$) must be even.

- The factors p and q must distribute themselves between the two factors of $(a^r - 1)$.

First Step : Choose $k < n$ if k divideds n ; stop

Shor's Algorithm: Oracle construction

→ first number factorized by Shor's Alg.

A smaller example : $n = 15$; $a = 2$

- NMR based QC
- 7 qubits

$a^x \bmod n$

$2^x \bmod 15$

$x = 0$ 00 : 1 0001

$x = 1$ 01 : 2 0010

$x = 2$ 10 : 4 0100

$x = 3$ 11 : 8 1000

$x = 4$: 1

$\Rightarrow (2^4 - 1) \equiv 0 \bmod 15$

$(2^2 + 1)(2^2 - 1) = 15$

$\quad 5 \qquad\quad 3$

RSA : 250 (829 bits) was factored in 2020

Rivest Shamin Adelmann decimals

-> Shor's Algorithm : Largest number factorized = 35

-> On an error prone QC Shor's Algorithm is guaranteed to fail asymptotically

-> RSA-2048 : 6 - 6.5 million qubits