Attacking Cryptosystems using Quantum Computer

Chaganti Kamaraja Siddhartha
T Mani Venkata Krishna

Indian Institute of Technology, Madras

May 8, 2023

Contents

What, why, RSA? :)

- RSA is developed by Rivest, Shamir and Adleman.
- It is the best public key scheme till date.
- Security because of high cost of factoring i.e., exponential time(for sufficiently large numbers factoring takes over a million years).
- Enryption is easier because modular exponentiation takes polynomial time$(O(num_bits^3))$.
- It is used by breaking the whole message into segments and encrypting each segment.

How to implement RSA?

- First we need to generate public(we share it to the world) and private(we keep it to ourselves) keys.
- Generate two very large prime numbers. Let us name them p, q.
- Calculate $N = p * q$. And a small number e such that it is co-prime to $(p-1)*(q-1)$.
- And let d be the modulo inverse of e modulo $(p-1)*(q-1)$.
- e is the public key, d is the private key.

Example

- Let $p = 5, q = 11$
- $N = p * q = 55$.
- $(p - 1) * (q - 1) = 40$.
- Let us choose $e = 3$ which is co-prime to 40.
- $d = 3^{-1} (mod\, 40) = 27$.(Note that 27 is one of many possiblities for that e).
- Let us choose msg $x = 13$.
- Encryption: $c = 13^3 (mod\, 55) = 52$.

How to encrypt and decrypt using those keys?

- Take a segment of data that needs to be encrypted let it be 'x'. Let 'e' be the public key of reciever.
- To encrypt, sender calculates $c = x^e (mod\, N)$. And transmits the cipher(c).
- To decrypt, reciever uses his private key(d) to calculate $x = c^d (mod\, N)$.
- Now reciever gets the segment x.

Can we break RSA without knowing private key?

How to break RSA?

- Factorise N and get p and q.
- You can d as we know public key e, p and q.
- We will get everything we need.
- But how much time does it take to factorise N? It takes $O(\sqrt{N})$ time complexity.
- RSA-4096 uses two 2048-bit prime numbers. Here $N \ 2^{4096}$. Hence trying to factorise it takes time of the order 2^{2048}.
- **If we are able to factorise it we will be able break RSA.**

Order finding problem

- Given two integers a and N finding the smallest r such that $a^r = 1(mod\, N)$ is the order finding problem.
- Example: for $N = 10$ and $a = 3$, $order = 4$ as $3^4 = 1(mod\, 10)$.

Solving factorisation problem using order finding problem

- For simplicity, let us say we have to factor **N** which is product of two distinct primes p_1 and p_2.
- We pick a random integer **a** from 2 to $N-1$ and compute $gcd(a,N)$ this can be done in polynomial time.
- If gcd is not 1 then it must either be equal to p_1 or p_2 in this case the factorisation problem is solved.
- So let us say $gcd(a,N) \ = \ 1$.
- Let **r** be the order of $a\, modulo\, N$ which can be evaluated in polynomial time(Under the assumption that order finding can be solved in poynomial time).
- We repeat the above steps until we find **r** that is even for some **a**. There is very significant fraction of **a**'s which have even order for a **N**.

CONT.

- $a^r - 1$ is a multiple of **N**. If r is even we can write $a^r - 1$ as,

$$a^r - 1 = (a^{r/2} - 1)(a^{r/2} + 1)$$

- $(a^{r/2} - 1)$ cannot be multiple of **N** because if that were the case then order would be $r/2$ not r.
- **case(i):** $(a^{r/2} + 1)$ is not a multiple of **N**. That means $(a^{r/2} - 1)$ and $(a^{r/2} + 1)$ are not multiple of **N** but their product is. That implies p_1 is a prime factor of $(a^{r/2} - 1)$ and p_2 is a prime factor of $(a^{r/2} + 1)$ or vice versa. In this case we can find p_1 by just calculating $gcd(a^{r/2} - 1, N)$ similarly p_2 by just calculating $gcd(a^{r/2} + 1, N)$.

CONT.

- **case(ii):** $(a^{r/2} + 1)$ is a multiple of **N**. In this case we cannot do anything but choose another **a** (These cases are very less frequent).
- **Hence we can say that by solving order finding problem we can solve factorisation problem.**
- Shor's algorithm is used by us to solve order finding problem.

How to implement Even-Mansour?

- We break the message into blocks and we apply encryption.
- Let us call one block as **M**.
- In encryption we use a prewhitening key K_2 and postwhitening key K_2 an a function circuit **F**.
- Here K_1,K_2 are private info and **F** is known to all.
- Let ENC(M) be the overall encryption function and C be the resulting cipher.

$$C = ENC(M) = F(K_1 \oplus M) \oplus K_2$$

How to decrypt Even-Mansour cipher?

- For the sake of decryption we need to know K_2 and K_1. If we don't have them we cannot decrypt it.
- For decryption we calculate $K_2 \oplus$ **W** followed by passing it through F^{-1} function circuit.
- Let the output be **V**.
- The message **M** will be $K_1 \oplus$ **V**.
- We use simon's algorithm to break Even-Mansour.

Overview of Quantum Computing

1. Quantum states are represented using state $|0\rangle$ and $|1\rangle$ similar to classical states 0 and 1.
2. Unlike Classical states which only exist in either 0 or 1, Quantum states exists in superposition of states. For example an arbitrary quantum state is represented as
$$a\,|0\rangle + b\,|1\rangle$$
where, a, b are complex numbers and $a^2 + b^2 = 1$.
3. Even though Quantum states exists in superposition when measured in computational basis they only output either 0 or 1. Please note that they output classical states 0 and 1 and not quantum states $|0\rangle\,, |1\rangle$.
4. If a two - one function $f(x)$ with period s is given there exists a quantum algorithm called Simon's algorithm which can find the period s in polynomial time.
5. If a Number which is product of two primes is given there exists a quantum algorithm called Shor's algorithm which makes use of order-finding and outputs the prime factors of given number.

Periodic function for Even-Mansour

Kuwakado and Morii (2012) showed that there is a function $f(M)$ that can be computed from given resource where the function $f(M)$ is periodic with period k_1. The function given by Kuwakado and Morii (2012) is

$$f(M) = Enc(M) \oplus F(M)$$

$$f(M) = F(M \oplus k_1) \oplus k_2 \oplus F(M)$$

$$f(M \oplus k_1) = F(M \oplus k_1 \oplus k_1) \oplus k_2 \oplus F(M \oplus k_1)$$

$$f(M \oplus k_1) = F(M) \oplus k_2 \oplus F(M \oplus k_1)$$

$$f(M \oplus k_1) = f(M)$$

Even-Mansour automation

Canale et al. (2022) authored a code available at period-search. (n.d.). GitHub. Retrieved May 7, 2023, from https://github.com/rub-hgi/period-search.git which can be used to generate periodic function like $f(x)$ as shown above for different ciphers like Even-Mansour, Fiestel-3,4,5 rounds and Misty-5k ciphers. The output of their code generating the function by taking encryption and resources as input is shown in results section.

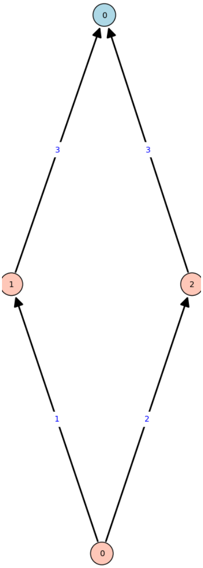


Figure: periodic function for even mansour attack

Quantum Cirucit for Even Mansour

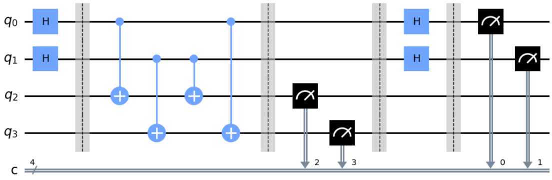


Figure: Even mansour simon

Even Mansour

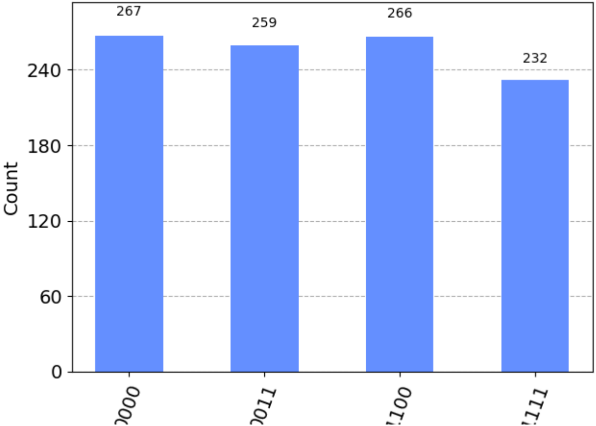


Figure: EVEN-MANSOUR RESULT

Even Mansour

The output of first two qubits is either 00 or 11 therefore, if the key is 'ab' then

$$a + b = 0$$

$$\implies ab = 00 \text{ or } 11$$

but it is periodic with non-zero period therefore, period $s = 11$.

Shor Algorithm

Let us assume there exists a Unitary operator U when acted up on any state $|y\rangle$ results in the following state.

$$U|y\rangle = |ay \mod N\rangle$$

We call this unitary opeartor as modular multiplication operator from Markov and Saeedi (2015) as shown in figure.

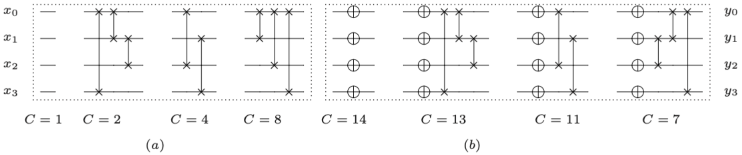


Figure: Circuit for $Cx \mod 15$.Source: Markov and Saeedi (2015)

Shor Algorithm

If there is a quantum state which is created using superposition of states as shown below,

$$|u_0\rangle = \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} \left| a^k \mod N \right\rangle \tag{1}$$

Now, if we apply the Unitary U on this state, $U|u_0\rangle = |u_0\rangle$ therefore, the state $|u_0\rangle$ is an eigen state of U with eigen value equal to one. Let us create an arbitrary state $|u_s\rangle$ as shown below,

$$|u_s\rangle = \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} e^{-2\pi iks/r} \left| a^k \mod N \right\rangle \tag{2}$$

It can be easily verified that the state $|u_s\rangle$ is also an eigen state of U with eigen value equal to $e^{2\pi is/r}$ that is

$$U|u_s\rangle = e^{2\pi is/r} |u_s\rangle$$

.

Shor Algorihtm

Now, let us create a superposition of all the states of the form $|u_s\rangle$ that is

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

As we know,

$$\sum_{k=0}^{r-1} e^{2\pi i k s/r} = 0, \ \forall s \neq 0.$$

We can show that,

$$|\varphi\rangle = |1\rangle$$

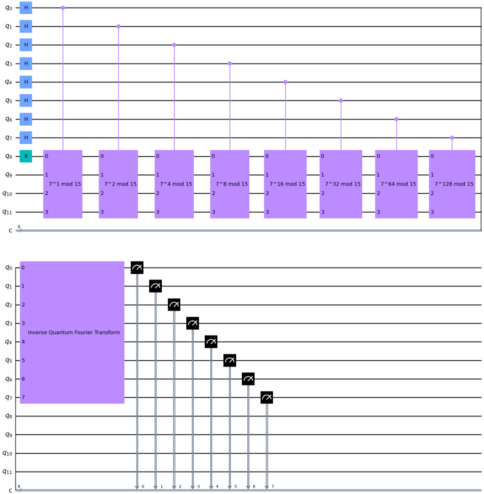
Shor Algorithm

when Unitary is operated on this state $|1\rangle$,

$$U|1\rangle = e^{2\pi is/r}|1\rangle$$

where, s is a random integer between $0$and $r-1$ because, the state $|1\rangle$ is superposition of all the states of the form $|u_s\rangle$ where s varies from 0 to $r-1$ refer to figure[6] which shows the output of the cirucit in [5].

Shor Algorithm Circuit



Shor Algorithm Result

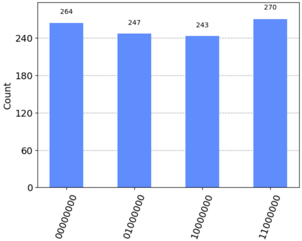


Figure: Output of shor's simulation

$$
\begin{array}{lr}
RegisterOutput & Phase \\
000000000(bin) = 000(dec) \rightarrow 000/256 = 0.00 \\
111000000(bin) = 192(dec) \rightarrow 192/256 = 0.75 \\
201000000(bin) = 064(dec) \rightarrow 064/256 = 0.25 \\
310000000(bin) = 128(dec) \rightarrow 128/256 = 0.50
\end{array}
$$

Shor result

```
ATTEMPT 1:
Register Reading: 01000000
Corresponding Phase: 0.25
Result: r = 4
Guessed Factors: 3 and 5
*** Non-trivial factor found: {guess} ***
*** Non-trivial factor found: {guess} ***
```

Figure: Result with a = 7 for N = 15

Bibliography

Canale, F., Leander, G., and Stennes, L. (2022). Simon's algorithm and symmetric crypto: Generalizations and automatized applications. Cryptology ePrint Archive, Paper 2022/782. https://eprint.iacr.org/2022/782.

Kuwakado, H. and Morii, M. (2012). Security on the quantum-type even-mansour cipher. pages 312–316.

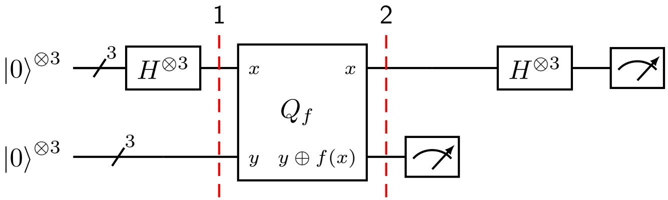
Markov, I. L. and Saeedi, M. (2015). Constant-optimized quantum circuits for modular multiplication and exponentiation.

Simon problem

If a function $f(x)$ is $2:1$ such that for every input x_1 in 2^n there exists x_2 which satisfy the condition $x_1 \oplus s = x_2$ and $f(x_1) = f(x_2)$. If the function $f(x)$ is given in a black box how many calls to the black box it will take to determine s. Here, we call s as period of function $f(x)$.

$$f(000) = f(111) = 000$$
$$f(001) = f(110) = 001$$
$$f(010) = f(101) = 010$$
$$f(011) = f(100) = 011$$

Simon Algorithm



Starting State

$$|\psi_0\rangle = |0\rangle^{\otimes 3} |0\rangle^{\otimes 3}$$

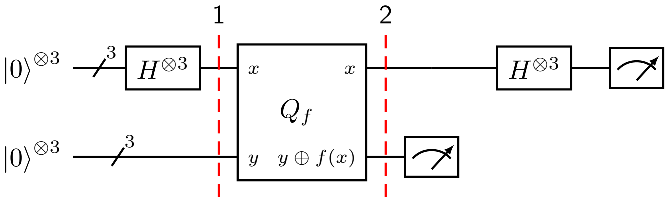
State after First Hadamard Transforms

$$|\psi_1\rangle = \frac{1}{\sqrt{2^3}} \sum_{x \in \{0,1\}^3} |x\rangle |0\rangle^{\otimes 3}$$

State after applying the oracle

$$|\psi_2\rangle = \frac{1}{\sqrt{2^3}} \sum_{x \in \{0,1\}^3} |x\rangle |f(x)\rangle$$

Simon Algorithm



State after measuring the second register

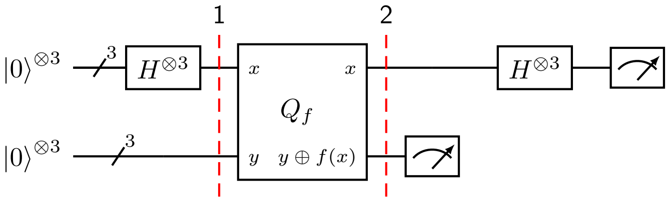
If the measurement gave $|001\rangle$

$$|\psi_3\rangle = \frac{1}{\sqrt{2}}(|x\rangle + |x \oplus s\rangle)$$

where,

$$f(x) = f(x \oplus s) = 001$$

Simon Algorithm



State after final Hadamard

$$|\psi_3\rangle = \frac{1}{\sqrt{2^7}} \sum_{z \in \{0,1\}^3} \left[(-1)^{x.z} + (-1)^{(x \oplus s).z}\right] |z\rangle$$

Simon Algorithm

Measurement of first 3 qubits of final state Measurement of first 3 qubits of final state give information about s because,
It will give output only if

$$(-1)^{x.z} = (-1)^{(x \oplus s).z}$$

which means:

$$x.z \text{ mod } 2 = (x \oplus s).z \text{ mod } 2$$
$$x.z \text{ mod } 2 = x.z \oplus s.z \text{ mod } 2$$
$$\implies s.z = 0 \text{ mod } 2$$

Simon algorithm

A string z will be measured, whose inner product with s $= 0$. Thus, repeating the algorithm $\approx$ n times, we will be able to obtain n different values of z and the following system of equation can be written:

$$\begin{cases} s.z_1 = 0 \text{ mod } 2 \\ s.z_2 = 0 \text{ mod } 2 \\ \vdots \\ s.z_n = 0 \text{ mod } 2 \end{cases}$$

From which s can be determined, for example by Gaussian elimination.

Simon Algorithm

If first run gives the output $z_1 = 011$ then, Let s = abc
$z_1 = 011$

$$s.z_1 = 0 \text{ mod } 2$$

$$b + c = 0 \text{ mod } 2$$

either, bc = 00 or bc = 11.
If the second run gives the output $z_2 = 101$ then, $z_2 = 101$

$$s.z_2 = 0 \text{ mod } 2$$

$$a + c = 0 \text{ mod } 2$$

either ac=00 or ac =11.
If c = 0 $\implies$ a = 0 and b = 0 then s = 000 but we know s $\neq$ 000 therefore, c = 1 $\implies$ a = 1, b = 1 $\implies$ s = 111. We can clearly see, Simon's algorithm determined the period s in polynomial steps.