International Journal of Sciences: Basic and Applied Research (IJSBAR)

ISSN 2307-4531
(Print & Online)

https://gssrr.org/index.php/JournalOfBasicAndApplied/index

Exploring Shor's Algorithm in Cracking RSA Encryption

Aashutosh Srivastava*

Student Delhi Public School, Navi Mumbai, Maharashtra, India

Email:aashutoshsr27@gmail.com

Abstract

The present study explores the groundbreaking possibilities of Shor's algorithm in cracking RSA encryption, hence enhancing the security consequences of this extensively employed cryptographic protocol. Due to Shor's algorithm's exponential speed advantage over classical algorithms, RSA encryption—a popular public key cryptosystem—is vulnerable to quantum attacks. To keep its security, RSA depends on the difficulty of factoring huge semi-primes. RSA's security presumptions are called into question by Shor's algorithm's quick factorization on a quantum computer, which has prompted research into post-quantum cryptography solutions to guarantee secure communication in the quantum age. Even with parallel computation, the speed and storage capacity of classical computing are constrained. A key idea in quantum computing is quantum parallelism, which allows quantum systems to carry out several computations at once. In contrast to classical portions. The scope of this paper is to understand the fundamentals of Quantum Computing, the current state of Shor's algorithm implementation, and efficiency, particularly its application in compromising public key cryptosystems like RSA. The project aims to achieve how factorization is done through classical methods and further, how is this done through Shor's algorithm and QFT techniques.

Keywords: Quantum Computing; Shor's Algorithm; RSA Encryption; Cryptography; RSA; Shor's Algorithm; Quantum Parallelism; Superposition; Hilbert Space; Quantum Error Correction.

Received: 9/5/2024
Accepted: 11/5/2024
Published: 11/15/2024

----

* Corresponding author.

1. Introduction

In the new digital age where people are using digital devices, communicating, and storing info in the cloud one of the most challenging aspects will be the processing speed of classical computers to process such a huge database. So, the next step towards technological advancement in the area of computer technology is quantum computing. So, as compared to classical computers, which utilize bits that can only be either 0 or 1, quantum computers use qubits, which can be 0,1, or both at the same time, hence, enabling them to solve problems far more quickly than conventional computers. But this will probably have a significant impact in the area of Cryptography giving a challenge to develop a more robust algorithm that can make the information quantum safe. Conventionally, the RSA algorithm is used to keep information secure, but quantum computers can break it using Shor's algorithm. This means we need to find new ways to protect our information from quantum attacks. So, there is a need to use quantum computing to make the RSA algorithm stronger. By doing this, we can add an extra layer of security to keep our data safe from future threats.

1.1. The BLOCH Sphere

To visualize how qubits operate a model called the Bloch sphere has been proposed as shown in Figure 1. The model is like a map depicting all the possible states a single qubit can be in. Consider the sphere, like a beach ball, where each spot on its surface depicts a different qubit state. Like, on the top, the North Pole, is a qubit in the state "0", while at the bottom, the South Pole, is a qubit in the state "1". Everywhere else on the sphere shows a mix of both "0" and "1" states, called a superposition. Interpreting the Bloch sphere is the best way to grasping how qubits operate. It helps to understand how quantum gates, like X, Y, and Z gates, change a qubit's state. It’s interesting to see how measurements force a qubit into either "0" or "1". In summary, the Bloch sphere is a useful tool for navigating the complex world of quantum computing.

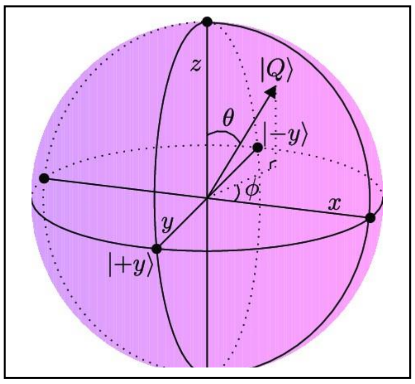


Figure 1: The Bloch sphere: showing the Hilbert space of one qubit. The eigenstates lie on the corresponding axis (x, y, or z)[15].

Usually, the Bloch sphere is depicted using the ‘*statevector’* simulator, which represents the quantum circuit's state vector.

1.2. The Quantum Phenomena

As can be seen from the Bloch sphere, what makes quantum computing a most remarkable and unique proposition is the ability of a quantum object, like an atom or photon to exist in multiple states or locations, while the classical objects can only occupy a single state or location at any given time. That means that an atom can be in two different states at once. This is called the Superposition that lies at the heart of quantum mechanics and serves as the foundation of quantum computing. So we can say that qubits (quantum bits) can exist in a linear combination of multiple states simultaneously. The actual power of superposition is more apparent when multiple qubits are considered. In a quantum computer with n qubits, it can exist in a superposition of 2n states or a computational basis. This exponential increase in information-carrying capacity is what far surpasses that of classical systems.

In quantum physics, a collection of complex numbers—more precisely, 2^n-1 complex numbers—are needed to describe the state of a system, where n is the number of qubits involved [6]. In a 2^n-1 dimensional vector space, the quantum system's state is represented by a point. The system's current state is represented by a unit-length vector in this vector space, also referred to as the Hilbert space, with 2^n states as basis vectors. We require 2^n-1 complex numbers to fully characterize the state, as multiplying this state vector by a unit-length complex phase does not change the system's behavior. Usually, this superposition of states is shown by Equation (1):

$$\sum_{i=0}^{2^n-1} a_i |S_i\rangle \qquad (1)$$

Where the amplitudes ai are complex numbers such that $\sum |ai|^2$ and each |Si⟩ is a basis vector of the Hilbert space.

$$|\psi\rangle = a_0|S0\rangle + a_1|S1\rangle + a_2\,|S2\rangle + \ldots + a_{2^n-1}|S_{2^n-1}\rangle \qquad (2)$$

In Equation (2), the values a_0, a_1, a_2 ... a_{2^n-1} are the amplitudes of complex probabilities According to Born's rule, when we measure the qubit, one of the eigenvalues of the self-adjoint operator will be the result if an observable corresponding to the self-adjoint operator is measured in a system with normalized_wave_function [1].

Following Quantum Measurement According to Equation (3), we may state that the Qubit (ψ) superposition state will collapse to a certain eigenstate |Si⟩ with the Born probability, completely characterizing the state. Usually, this superposition of states is shown by Equation (1):

Probability of collapsing on Eigenstate

$$|Si\rangle = |a_i|^2 \qquad (3)$$

The probability amplitudes a_0, a_1, a_2.... ...a_{2^n-1} also satisfies the normalization condition:

$$|a_0|^2 + |a_1|^2 + |a_2|^2 + \ldots + |a_{2^n-1}|^2 = 1 \qquad (4)$$

Equation (4) indicates that this normalization guarantees that the likelihood of measuring the Qubits in any of the states |S0⟩, |S1⟩, |S2⟩ ... $|S2^{n}-1\rangle$ is unity.

Let’s try to understand the other quantum phenomena like; Entanglement and tunnelling. Entanglement refers to a phenomenon in which the qubits in a superposition state cannot be factorized. It converts a superposition state into a Bell State. Original qubits cannot be derived from the bell state. For example,

|00> + |11> is a Bell state. It cannot be broken into qubits.

|00> +|01> = |0> (|0>+|1>) is not a Bell state as it can be broken into states.

Entanglement is another important quantum phenomenon in which two quantum objects, although separated by large distances, become linked in such a way that their properties are correlated. That means irrespective of the physical distance between two objects, measuring one feature affects the other instantaneously in real-time. Atoms can also get entangled, which means that regardless of the distance between them, the state of one atom can affect the state of another.

Quantum tunneling is the way how quantum objects pass through potential barriers, even when they lack sufficient energy to overcome those barriers according to classical physics. The atoms being in a state of superposition, exhibit behaviors that defy classical intuition. For example, they can traverse solid obstacles, such as walls, without causing any damage—a phenomenon known as "tunneling."

Uncertainty is also a very important phenomenon of quantum physics, as the Heisenberg uncertainty principle states that a particle's momentum can be determined less precisely the more precisely its position is known, and vice versa. To put it more clearly, one cannot know a particle's precise position and momentum at the same time, underscoring the inherent uncertainty at the quantum level.

1.3. RSA Algorithm

RSA (Rivest-Shamir-Adleman) encryption is the conventional method of a public-key cryptographic technique that ensures secure data transmission. The concept is simply based on the mathematical challenge of factoring large composite numbers into their prime factors. The security of RSA is based on the concept that, while multiplying two large prime numbers is computationally easy, factoring the resulting product is very difficult and time-consuming with classical computers.

1.3.1. Key Generation Steps

- Consider two large prime numbers, ‘n1’ and ‘n2’ are selected.
- The product, N = n1 x n2, is calculated where N is used as the modulus for both the public and private keys.
- The totient function $\phi(N)$ = (n1-1)(n2-1) is computed, representing the count of numbers less than N that are coprime to N
- The algorithm selects an encryption key, an integer e, is chosen such that $1 < e < \phi(N)$ and e is coprime with $\phi(N)$. ‘e’ becomes the public exponent. This pair *(e, N)* forms the public key, used for encryption.

- The private exponent d is computed as the modular multiplicative inverse of e modulo $\phi(N)$, satisfying $d \times e \equiv 1 \pmod{\phi(N)}$. The private key (d, N) is kept secret and used for decryption.

1.3.2 Encryption

- The plaintext message M is converted into an integer m to the power of e modulo N.
- The plaintext message resulting in the ciphertext $C \equiv m^e \pmod N$

1.3.3 Decryption

- The ciphertext 'C' is decrypted using the private key ((N, d) as $m \equiv C^d \pmod N$.
- The original message M is recovered from m

It can be seen that the strength of RSA encryption is very much dependent on the challenge of factoring large composite numbers into their prime factors. The public key consists of a product of two large prime numbers, and the private key is used to decrypt the message, relying on the difficulty of reverse-engineering the original primes. Classical computers find it hard to decode, especially as the key size increases, making RSA secure against current computing power. However, Quantum Computers with the advent of the algorithm like Shor's can break RSA encryption as Shor's algorithm can factor large numbers exponentially faster than the best-known classical algorithms. It does this by leveraging the properties of quantum superposition and entanglement, which allow quantum systems to explore multiple possibilities simultaneously

2. Shor's Algorithm

Shor's algorithm simplifies the tough task of breaking down big numbers into their prime factors, which is important in math and coding. Normally, this task takes a lot of time as numbers get bigger because it's complex. Using Shor's algorithm needs big and accurate quantum tools, making it a tough experiment. But Shor's algorithm is revolutionary because it makes factorization much easier by using quantum principles like doing multiple calculations at once and combining different states of data.

Shor's algorithm is composed of three parts-

- The first part turns the factoring problem into a *period-finding* problem using number theory, which can be computed on a classical computer.
- The second part finds the period using the *quantum Fourier transform* and is responsible for the quantum speedup of the algorithm.
- The third part uses the period found to *measure* the factors.

Shor's algorithm simplifies the tough task of breaking down big numbers into their prime factors, which is important in math and coding. Normally, this task takes a lot of time as numbers get bigger because it's complex. Using Shor's algorithm needs big and accurate quantum tools, making it a tough experiment. But Shor's algorithm is revolutionary because it makes factorization much easier by using quantum principles like doing multiple

calculations at once and combining different states of data.

Broadly the two quantum registers are prepared using Shor's Algorithm. The integer to be factored (designated as "a") is stored in the first register, and the "period-finding" function is assigned to the second register. The QFT is a quantum analogue of the classical discrete Fourier transform and is an essential tool for translating a function's periodic behavior into the frequency domain

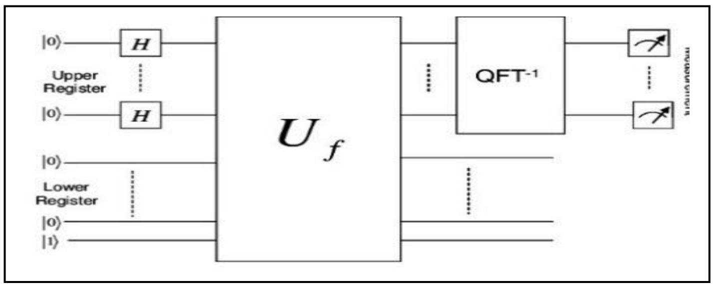


Figure 2: High-level Shor's algorithm diagram. The bottom register made up of n qubits, contains the superposition of values a^x mod N that have been computed by the U_f block; the top register, made up of 2n qubits, stores the superposition of numbers 0…. [2, N−1]. A high probability method of determining the period of the function f (x) = a^x mod N involves classical postprocessing of the measurement in the computational basis following the QFT block.

2.1. Steps in Shor's Algorithm for Factorization

1. To begin, consider a number N to be factored.

2. Choose an Appropriate "a": The next step is to choose a positive integer "a" at random that is less than "N" and roughly prime to "N." Stated otherwise, ′a′ and ′N′ should not have anything in common besides 1

$$2 < a < N - 1, \ \text{gcd}\ (a, N) = 1 \qquad (5)$$

3. Quantum Period Finding: Shor's algorithm is based on this idea. We employ a quantum computer to determine a given function's "period." The lowest positive integer "r" that is such that the period is:

$$a^r \bmod N = 1 \qquad (6)$$

4. Classical Post-Processing: To extract meaningful information from the period 'r' that the quantum computer produced, we must carry out a few classical computations. The prime components of N and the period 'r' are mathematically connected.

5. Extract Factors: Next, utilize the periods "r" and "a" to determine possible factors of 'N' by applying a few mathematical formulas. Finally, these computations will provide us with the prime factors of 'N'.

2.2. Quantum Fourier Transform (QFT)

The Quantum Fourier Transform (QFT) which plays a very important role in Shor’s algorithm, is a quantum algorithm that efficiently transforms a quantum state into its frequency components. The QFT operates on qubits and leverages quantum parallelism to achieve a significant speedup compared to its classical counterpart, making it an essential tool in quantum computing.

The discrete Fourier transform in its classical version is to be applied. Both the vectors being elements of complex space, the Classical Discrete Fourier transform applied on a vector (x_0, x_1, x_2,..., x_{n-1}), maps it to another vector (y_0, y_1, y_2,..., y_{n-1}) [1].

So we can see that the Quantum Fourier Transform is nothing but a linear transformation on a quantum bit. It can be easily compared to the traditional discrete inverse transform [1].

Here is the formula showing how the quantum Fourier transform operates on a quantum state |x⟩ and transfers it to a quantum state |y⟩:

$$y_k = \frac{1}{\sqrt{p}} \sum_{p=0}^{P-1} xp w_P^{pk} \qquad (7)$$

In equation (7), the value p varies from 0 to P-1. P is defined as the length of the vectors where $P = 2^p$[1].

The QFT is part of many Quantum computations that are related to the Hadamard gate, a quantum gate that puts a qubit in an equal ‘superposition state’ in the computational basis [3].

2.3. Period-Finding Subroutine

Next, at this stage, the quantum parallelism and superposition capabilities of quantum computers play an important role. the algorithm creates a superposition of states [3] by doing a series of modular exponentiations, with each state corresponding to a unique value of the function’s period [3]. To achieve this a modular exponentiation gate is implemented efficiently that computes

a^p mod P for different values of p.

Hence, eventually, the resulting superposition encodes the period of the function, which is an important factor in the factorization process.

2.4 Quantum Measurement

The superposition of period states is collapsed into a single value by quantum measurement. The measured value yields a rough estimate of the function's period. Because quantum measurement is probabilistic, the process may need to be run several times before a noticeable degree of certainty is reached within the measured interval.

3. Shor's Algorithm: Implementation with Examples

3.1 Process of Factorization to Period Finding

Shor's algorithm is based on number theory, which is related to periodic modulo sequences. The following number theory conclusion is required to reduce the factorization of a number N to the issue of determining the period of an integer $1<m<n$.

Consider the periodic function,

$$f(p) = m^p \bmod(N);\ m \text{ being an integer coprime to } N \text{ and } p \geq 0$$

As f(p) is a periodic function, assume the function has some period r. Knowing that (m^0 mod N) =1 implies that m^r mod N=1 since the function is periodic and thus r is just the first nonzero power where $m^r = 1$ (mod N)

$$m^r \equiv 1 \bmod N$$

$$m^{(r/2)2} \equiv 1 \bmod N$$

$$(m^{r/2})^2 - 1 \equiv 0 \bmod N$$

If r is an even number,

$$(m^{r/2} + 1)(m^{r/2} - 1) \equiv 0 \bmod N$$

=> $(m^{r/2}+1)(m^{r/2}-1)$ is an integer multiple of N

If $(m^{r/2}+1)(m^{r/2}-1)$ is not a multiple of N, then at least either one of them must have a nontrivial factor in common with N.

So, by computing the Greatest Common Divisor gcd ($m^{r/2}$-1, N) and gcd ($m^{r/2}$+1, N), it will find a factor of N which can be calculated by the Polynomial Time Euclidean Algorithm.

3.2 Classical Steps to Shor Algorithm

Consider, N=p*q; where p and q are the prime factors of N.

1. For random integer m; $1<m<N$ and computer gcd (m, N) using Euclid Algorithm.
2. If m and N have some common prime factors then

$$\gcd(m, N) = p \text{ or } q \text{ or } \gcd(m, n) = 1$$

=> m and n are co-prime.

3. Consider, r to be the period of (m^r mod N) and get it computed by the period-finding machine.
4. Repeat the above steps with random values of m until r is an even number.
5. Now p and q can be found by computing gcd ($m^{r/2} \pm 1$, N) as long as $m^{r/2} \neq 1$

Consider N =15; 1<m<15; m is co-prime with 15.

20 ≡ 1 mode 15 40 ≡ 1 mode 15

21 ≡ 2 mode 15 41 ≡ 4 mode 15

22 ≡ 4 mode 15 42 ≡ 1 mode 15

23 ≡ 8 mode 15 43 ≡ 4 mode 15

24 ≡ 1 mode 15

25 ≡ 2 mode 15

26 ≡ 4 mode 15

27 ≡ 8 mode 15

Similarly, we can find the m^p mod15 for other values of m (7,8,11,13,14) and tabulate as below-

Table 1: Period Finding

m	**m mod15**	**Period r**	**gcd($m^{r/2}$ -1,15)**	**gcd($m^{r/2}$+1,15)**
2	1,2,4,8, 12,4,8,1,2	4	3	5
4	1,4,1,4,1,4	2	3	5
7	1,7,4,13, 1,7,4,13	4	3	5
8	1,8,4,2, 18,4,2	4	3	5
11	1,11,1,11,1	2	3	5
13	1,13,4,7, 1,13,4,7	4	3	5
14	1,14,1,14	2	1	15

It is evident that the factors of 15, which are 3 and 5, will be returned for any value of m other than 14. In a specific instance like number 14, where ($m^{r/2}$+1) or ($m^{r/2}$-1) is a multiple of N, it is necessary to try a different value of m. It can be demonstrated that, on average, only two calls to the period-finding machine are necessary to account for this unique scenario because it happens infrequently.

3.3 Quantum Period Finding

Steps for the quantum period finding algorithm:

1. For two coprime integers, m and N, and output r, the period of $F(p) = x^p \bmod N$

2.Consider T=2 such that N2 ≤ T ≤ 2N2. Initialize two registers of qubits first an argument registers with t qubits and second a function register with $n= \log_2 N$ qubits. These registers start in the initial state:

$$|\psi\rangle = |0\rangle|0\rangle$$

2. Hadamard gate to be applied on each of the qubits in the argument register to get an equally weighted superposition of all integers from 0 to T :

$$|\psi_1\rangle = \frac{1}{\sqrt{T}} \sum_{a=0}^{T-1} |a\rangle|0\rangle$$

3. Modular exponentiation function m mod N to be implemented on the function register, giving the state

$$|\psi_2\rangle = \frac{1}{\sqrt{T}} \sum_{a=0}^{T-1} |a\rangle|m^a \bmod N\rangle$$

The $|\psi_2\rangle$ is highly entangled and exhibits quantum parallelism, which means the function entangled in parallel all the 0 to T input values with the corresponding values of m^a mod N, even though the function was only executed once.

4. Next, QFT (quantum Fourier transform) is to be applied on the argument register, in the below state:

$$|\psi_2\rangle = \frac{1}{\sqrt{T}} \sum_{p=0}^{T-1} \sum_{z=0}^{T-1} e^\wedge(2\pi i)\left(\frac{pz}{T}\right)|z\rangle| m^p \bmod N\rangle$$

Due to the interference, only the terms $|z\rangle$ with

$$z = qT/r \Rightarrow \mathbf{r= q(T/z)}$$

have significant amplitude where q is a random integer ranging from 0 to r-1 and r is the period of $F(p) = m^p \bmod N$

6. To find the classical result z, measure the argument register. The continuous fraction approximation of T/z will, with a fair degree of probability, be an integer multiple of the period r. Then, r may be found using Euclid's approach.

3.4. Quantum Fourier Transform

Experimental Set up:

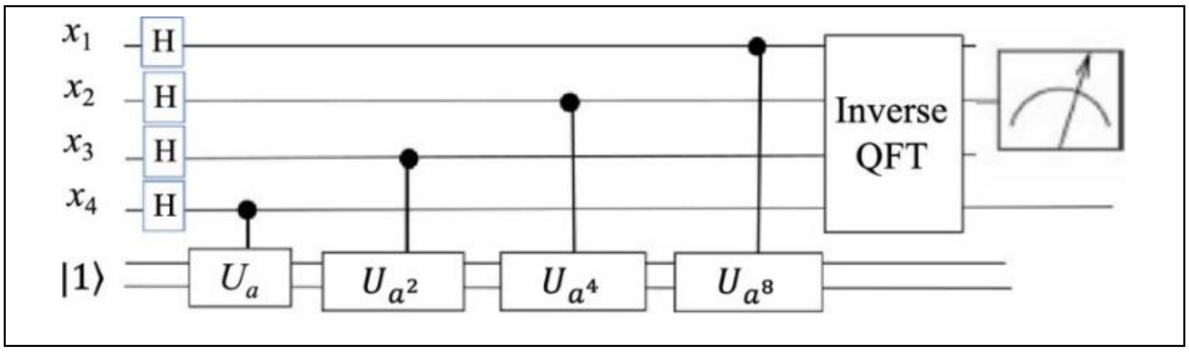


Figure 3: Shor's quantum factoring algorithm for the case of m = 4

Equation (7) describes how the Quantum Fourier Transform operates on a quantum state

$$\sum_{i=0}^{N-1} x(i)|i\rangle$$

and maps it to the quantum state $\sum_{i=0}^{N-1} y(i)|i\rangle$

$$y_k = \frac{1}{\sqrt{N}} \sum_{J=0}^{N-1} xj w_N^{jk}$$

where $w_N^{jk} = e\text{^}2\pi i\,(\frac{jk}{N})$; affects the amplitudes of the state only.

Next, the circuit can be derived for N= 2^n; QFT_N acting on the input state $|x\rangle = |x1\dots xn\rangle$; x_i is the most significant bit.

$$QFT_N\,|x = \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} w_N^{xy}|y\rangle$$

$= \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} e^{2\pi ixy/2\text{^}n}|y\rangle$; Putting the values as $w_N^{jk} = e\text{^}2\pi i\,(\frac{jk}{N})$ and N=2^n

$$= \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} e^{2\pi ixy/2\text{^}n}|y\rangle$$

$$= \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} e^{2\pi i\,(\sum_{J=0}^{N-1} \mathrm{yk}/2\text{^}\mathrm{k})x}|y1\dots yn\rangle$$

$= \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} \prod_{k=0}^{n} e^{2\pi ix\mathrm{yk}/2\text{^}\mathrm{k}}|y1\dots.yn\rangle$; After expanding the exponential of 'a'

k=n

$= \frac{1}{\sqrt{N}} \oplus (\,|0\rangle + e^{2\pi ix/2^k}|1\rangle\,)$; After rearranging the sum and products k=1

$$= \frac{1}{\sqrt{N}} (|0\rangle + e^{2\pi i|0 \cdots xn/2^k} |1\rangle) \oplus \ldots\ldots\ldots$$

$$\ldots\ldots\oplus (|0\rangle + e^{2\pi i|0.x1x2\ldots x_{n-1}x_n|}|1\rangle \text{as } e^{2\pi ix/2}$$

Before we build the circuit code for general $N=2^n$, let us look at N=8, n=3

QFT₈ | x₁ x₂ x₃ ⟩ $= \frac{1}{\sqrt{8}} (|0\rangle + e^{2\pi i|0..x_3|}|1\rangle \oplus (|0\rangle + e^{2\pi i|0..x_2x_3|}|1\rangle \oplus \rangle \oplus (|0\rangle + e^{2\pi i|0..x_1x_2x_3|}|1\rangle)$

Hence, the steps to creating the circuit for $|y_1y_2\, y_3\rangle$, remembering the controlled phase rotation gate CU, can be summarized as below:

1. Hadamard can be applied to $|x_3\rangle$, giving the state

$$= \frac{1}{\sqrt{2}} (|0\rangle + e^{2\pi i.0.x_3} |1\rangle) = \frac{1}{\sqrt{2}} (|0\rangle \;\; + (-1)^{x_3} \,|\, 1\rangle)$$

2. Apply a Hadamard to $| x_2\rangle$, then depending on k_3 (before the Hadamard gate) a $CU_1 (\frac{\pi}{2})$ depending on k_2, and $CU_1 (\frac{\pi}{4})$ depending on k_3

3. Applying the rule to measure the bits in reverse order that is $y_3 = x_1$, $y_2 = x_2$, $y_1 = x_3$

Let's implement the same through an example:

Factorize N=21 with coprime x =2, applying the above steps of the quantum period finding

algorithm which should return r= 6

1.Consider T=2t ; N2 ≤T≤2N2

For N=21, the smallest value of t is 9 => T=2t=512. Let us initialize two registers of qubits-

1. argument registers with t=9 qubits

2. a function register with $n= \log_2 N = 5$ qubits.

$$|\psi\rangle = |0\rangle|0\rangle$$

2. Hadamard gate to be applied on each of the qubits in the argument register

$$|\psi_1\rangle = \frac{1}{\sqrt{T}} \sum_{p=0}^{T-1} |p\rangle|0\rangle = \frac{1}{\sqrt{512}} \sum_{p=0}^{511} |p\rangle|0\rangle$$

3. Modular exponentiation function m^p mod N can be implemented on the function register.

$$|\psi_2\rangle = \frac{1}{\sqrt{T}}\sum_{a=0}^{T-1}|p\rangle|m^p \bmod N\rangle$$

$$= \frac{1}{\sqrt{512}}\sum_{p=0}^{511}|p\rangle|\,2^p\ mod\ 21\rangle$$

$$|\psi_2\rangle = \frac{1}{\sqrt{512}}(|0\rangle|1\rangle+|1\rangle|2\rangle+|2\rangle|4\rangle+|3\rangle|8\rangle +|4|16\rangle+|5\rangle|11\rangle+$$

$$|6\rangle|1\rangle +|7\rangle|2\rangle +|8\rangle|4\rangle + |9\rangle|8\rangle+|10\rangle|16\rangle+|11\rangle|11\rangle+\dots\dots\dots$$

$$|12\rangle|1\rangle +\dots\dots\dots..)$$

The fact that the states of the second register in each "column" are identical allows us to recognize the pattern. Thus, to obtain the second register, we rearrange the phrases.

$$=\frac{1}{\sqrt{512}}[(|0\rangle+|6\rangle+|12\rangle \dots\dots.+|504\rangle + |510\rangle)|1\rangle + \quad (8)$$

$$(|1\rangle+|7\rangle +|13\rangle\dots\dots+|505\rangle + |511) |2\rangle + \quad (9)$$

$$(|2\rangle+|8\rangle +|14\rangle\dots\dots+|506\rangle +) |4\rangle + \quad (10)$$

$$(|3\rangle+|9\rangle +|15\rangle\dots\dots+|507\rangle +) |8\rangle + \quad (11)$$

$$(|4\rangle+|10\rangle +|16\rangle\dots.+|508\rangle +) |16\rangle + \quad (12)$$

$$(|5\rangle+|11\rangle +|17\rangle\dots.+|509\rangle +) |11\rangle] \quad (13)$$

4. We will measure the function register and then apply a quantum Fourier transform on the argument register to simplify the ensuing equations. One of the following numbers will result from this, with an equal chance:

{1,2,4,6,8,16,11}. Suppose that the result of the measurement was 2, then:

$$|\psi_3\rangle = \frac{1}{\sqrt{86}}(|1\rangle + |7\rangle + |13\rangle \dots\dots. + |505\rangle) |2\rangle$$

The periodic pattern is what matters; the measurement's outcome is irrelevant. The solution to the problem is the period of the states of the first register, and the value of the period may be found using the quantum Fourier transform.

5. QFT (quantum Fourier transform) to be performed on the argument register:

$$|\psi_4\rangle = QFT(|\psi_3\rangle) = QFT(\frac{1}{\sqrt{86}}\sum_{a=0}^{85}|6a+1\rangle|2\rangle$$

$$=\frac{1}{\sqrt{512}}\sum_{j=0}^{511}(\left[\frac{1}{\sqrt{86}}\sum_{a=0}^{85}|6a+1\rangle|2\rangle$$

6. Next, measure the argument register. The probability of measuring a result j is:

$$\text{Probability (j)} = \frac{1}{512\,x\,86}\left|\sum_{a=0}^{85} e^{-2\pi i\,6ja/512}\right|^2$$

The peak at j =0,85,171, 256, 341,427. Consider that the result of the measurement yielded j=85, then following continued fraction approximation of $\frac{512}{85}$, we obtain the expected result r=6.

4. Quantum Computing Platforms

There are several Quantum Computing Platforms including Qiskit and IBM Quantum Lab. A few others are - Microsoft Quantum Development Kit, Google Cirq, Rigetti Forest, D- Wave Leap, Xanadu PennyLane, and ProjectQ. Although all provide an open-source quantum framework, Qiskit is primarily based on Python, a widely used and accessible programming language, and is closely integrated with IBM Quantum Experience, providing users with cloud access to IBM's quantum processors. This integration allows for easy experimentation and testing on real quantum hardware. It's important to note that the choice between quantum computing platforms depends on various factors, including the specific requirements of a project, ease of use, community support, available features, and compatibility with existing infrastructure These resources (Qiskit and IBM Quantum Lab) provide a foundation for understanding the fundamentals of quantum computing, including the Quantum Fourier Transform, number theory, modular arithmetic, and the quantum period finding algorithm. By mastering these concepts, we can effectively implement the QFT within Shor's algorithm, laying the groundwork for exploring the potential for factoring large numbers on a quantum computer.

4.1.The IBM-QISKIT

IBM Qiskit is simple to start up as a Python package, branch, or fork, just like other open-source projects. The software toolbox for creating applications utilizing quantum computing is incredibly user-friendly. It is incredibly small and uses very little of the resources on local computers when it is operating.

What makes it attractive is its extensive use for testing, creating, and running quantum programs. The IBM-supported Qiskit can be used to write quantum algorithms and run them on its real quantum devices simulating to mimic the quantum behavior. In addition, the Qiskit offers tools to visualize quantum circuits and analyze the computation results. In conclusion, Qiskit offers a set of resources to experiment into this fascinating field. This consists of four main components: Terra, Aer, Ignis, and Aqua [13].

5. Implementation

The next important step is the implementation of Shor’s algorithm using Python software language. However, it can be a little challenging due to the algorithm's dependency on quantum computing principles including QFT and quantum gates. While Python is used for the software program for computing simulation and algorithm development, specialized ‘Qiskit’ libraries are used to factor large numbers while implementing Shor’s algorithm.

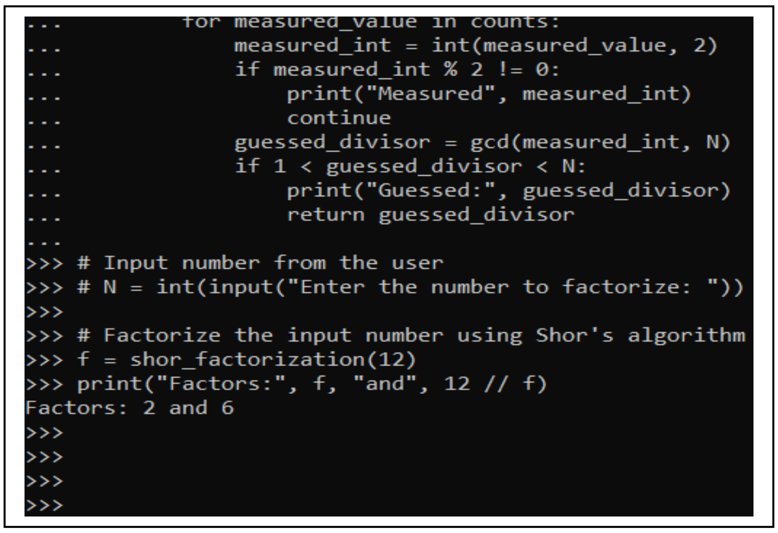
5.1 Basic implementation of Shor's algorithm using Qiskit for simulation

The Python code providing the basic implementation of Shor's algorithm using Qiskit for simulation provides the output as expected as captured in Fig 4.

However, there are certain limitations due to the non-availability of quantum hardware. Running Shor's algorithm to factor large numbers efficiently would require a quantum computer with enough qubits, which is not currently available due to the limitations of quantum hardware.

The program was run on Windows Command prompt for input number N=12 and the Factors were received as 2 and 6 as expected.

The screenshot of the run result has been captured and presented in Figure 4 for reference.



```
...         for measured_value in counts:
...             measured_int = int(measured_value, 2)
...             if measured_int % 2 != 0:
...                 print("Measured", measured_int)
...                 continue
...             guessed_divisor = gcd(measured_int, N)
...             if 1 < guessed_divisor < N:
...                 print("Guessed:", guessed_divisor)
...                 return guessed_divisor
...
>>> # Input number from the user
>>> # N = int(input("Enter the number to factorize: "))
>>>
>>> # Factorize the input number using Shor's algorithm
>>> f = shor_factorization(12)
>>> print("Factors:", f, "and", 12 // f)
Factors: 2 and 6
>>>
>>>
>>>
>>>
```

Figure 4: Output Run on Windows command prompt

6. Challenges: Current Quantum Computing Hardware

The current quantum computing hardware has certain limitations caused by various factors such as decoherence, gate imperfections, and environmental noise that lead to inaccuracies in quantum computations as explained below-

Decoherence means the loss of quantum coherence in qubits. The qubits are not able to maintain superposition or entanglement due to interaction with their environment. For Shor's algorithm to be effective, a large number of qubits needs to remain coherent over extended periods. Hence, decoherence limits the complexity of problems that current quantum computers can solve, as qubits lose their quantum state before computations can be completed.

As we have seen, Quantum gates are the operations applied to qubits to change their state. However, due to hardware material and design imperfections, the gates are not perfect, which leads to inaccuracies in the quantum operations. With high gate error rates in current quantum computers, it is difficult to implement large-scale algorithms like Shor's effectively that require precise and accurate gate operations on a large number of qubits.

Environmental noise that can arise from various sources, such as vibrations, electromagnetic fields, or temperature changes. The presence of environmental noise increases the likelihood of errors, limiting the algorithm's ability to factor large numbers with high accuracy.

Quantum error rates, the frequency of errors that occur during quantum operations, make it difficult to perform large-scale computations without significant inaccuracies. For Shor's algorithm, even small errors can lead to incorrect factorization results. Effective quantum error correction is essential for scaling up quantum computers to the point where they can handle the large numbers required by RSA encryption.

Shor's algorithm demands an exponential increase in computational resources as the size of the number to be factored grows. With today's hardware, the number of qubits and the coherence time required for this kind of large-scale computation are far beyond what current quantum systems can support.

Shor's algorithm needs to maintain error-free computation across many steps, which means that robust error correction is necessary. The overhead required for error correction is a major limiting factor for current quantum computers.

So as a next step, Quantum error correction techniques like the surface code play an important role the surface codes are designed to address these challenges by encoding qubits in a fault-tolerant manner, thereby mitigating the impact of errors on computations. The surface codes enable the detection and correction of errors, enhancing the reliability and stability of quantum systems when running algorithms like Shor's algorithm.

Overall, the next level of advancement in quantum error correction techniques is required so that the hardware limitations caused by high error rates in quantum systems can be overcome. By improvements in error correction capabilities, algorithms like Shor's algorithm can be used efficiently on quantum computers, supporting the further advancement and breakthroughs in cryptography, number theory, and other fields that benefit from quantum computing.

7. Post-Quantum Cryptography: Lattice-Based and Multivariate Cryptography vs. RSA

As the Shor Algorithm poses a threat to RSA encryption, which relies on the difficulty of factoring large integers, there is a need to find alternative techniques to have robust cryptography solutions in place. There are two schemes proposed, in a post-quantum world, as alternatives to RSA - Lattice-based and Multivariate cryptographic schemes.Lattice-based cryptography, which is in an advanced stage of research, provides robust security guarantees, scalability, and versatility, making it a strong choice for a wide range of applications.

Multivariate cryptography, although highly efficient and particularly suited for digital signatures, requires further research to establish the same level of confidence as lattice-based schemes. With the progress in quantum computing techniques, adopting and standardizing these post-quantum algorithms will be very important to maintain a high level of secure data communications and protection.

7.1 Recent Studies and Findings

Based on the recent findings of research on Lattice-based cryptography by Lyubashevsky, Peikert, and Regev, the LWE-based schemes are strong, secure, and efficient. Lattice-based schemes like ‘Kyber’ and ‘Dilithium’ have been included in NIST, the post-quantum cryptography standardization process. On the contrary, although studies have demonstrated the efficiency of multivariate signature schemes like Rainbow, which has advanced to the final round of the NIST, some multivariate schemes have been broken or found to have weaknesses. Hence it needs careful analysis and selection.

8. Additional Considerations: The Impact of Grover's Algorithm on Cryptographic Methods

Like Shor's algorithm, Grover's algorithm also poses a significant threat to cryptographic methods. While Shor's algorithm directly threatens public-key cryptosystems like RSA, Grover's algorithm necessitates adjustments to symmetric-key lengths and hash functions, Grover's algorithm can search an unsorted database or solve unstructured problems quadratically faster than classical algorithms. It uses a well-known technique that allows the use of interference to amplify certain states in quantum circuits in a way that will increase the amplitude of the value we are searching for and decrease those that we are not. Grover’s search algorithm can be broken into two main components. The first is referred to as Grover’s oracle and the second is the Grover diffusion operator[16].

The post-quantum cryptographic schemes discussed in section 7.1, like lattice-based and multivariate cryptography, remain robust and secured under the dual threats of Shor's and Grover's algorithms, provided key lengths are appropriately increased. This dual focus ensures that cryptographic methods remain secure, and performance efficient against both types of quantum threats, maintaining data security and integrity in the quantum computing age.

9. Conclusion

To sum up, this study examined the ground-breaking possibilities of Shor's algorithm for breaking RSA encryption, which is currently widely used for cryptography. The included Python code makes use of the Qiskit quantum computing framework to give a basic implementation of Shor's algorithm. This implementation shows the basic steps of Shor's algorithm, including the quantum portion that factorizes big integers using quantum Fourier transforms and controlled operations.

Keeping the theoretical aspects of Shor’s algorithm aside, because of the current limitations of Quantum computers, Shor's method faces several obstacles when it comes to breaking RSA encryption. RSA encryption, which is utilized in real-world applications, cannot be compromised by factoring big numbers at the scale that current quantum computers are unable to perform. Even if Shor's method becomes feasible in the future, RSA encryption can be reinforced by raising the key size, which will increase its resistance to attacks.

However, this study paper's examination of Shor's algorithm emphasizes how critical it is to create cryptographic protocols that are resistant to quantum computing to protect sensitive data in the future of quantum computing. Researchers and practitioners need to be on the lookout for changes in the field of cryptographic security as quantum computing advances. By broadening the scope to include Grover's algorithm as mentioned above, the analysis presents a fuller picture of the cryptographic landscape and the multifaceted challenges posed by quantum computing. This comprehensive approach highlights the need for robust, adaptable, and forward-thinking cryptographic solutions to ensure long-term security.

In conclusion, Shor's method is a major development in quantum computing with wide-ranging consequences for cryptography; yet, its actual effect on RSA encryption is still being investigated and discussed. Lattice-based schemes are primarily designed to withstand Shor's algorithm. However, they are also robust against Grover's algorithm, as the latter only provides a quadratic speedup. This means that lattice-based encryption, which might use 256-bit security levels, would need to be adjusted to 512-bit security to counteract Grover's algorithm. Similar to lattice-based cryptography, multivariate schemes must also account for Grover's algorithm, but the adjustment is less severe compared to the exponential speedup from Shor's algorithm.

To maintain the integrity and confidentiality of digital communications and transactions, we must continue to explore and innovate cryptographic solutions as we negotiate the shift to a quantum-secure future.

Acknowledgement

I would like to extend my heartfelt gratitude to my mentor, Lawrence Lin, a Ph.D. scholar at Cornell University, whose invaluable guidance and encouragement have been instrumental in shaping this research. His expertise in quantum computing and his dedication to teaching have significantly enhanced my understanding and motivated me to delve deeper into the subject. Lawrence's insights and feedback throughout this project have been an inspiration, and I am grateful for the time and effort he has invested in helping me achieve my research goals.

References

[1] W. C. Easttom II, *Quantum computing fundamentals*. Addison-Wesley Profes- sional, 2021.

[2] M. J. Nene and G. Upadhyay, "Shor's algorithm for quantum factoring," in *Advanced Computing and Communication Technologies: Proceedings of the 9th ICACCT, 2015*. Springer, 2016, pp. 325–331.

[3] V. Kasirajan, Fundamentals of quantum computing. Springer, 2021.

[4] A. Albuainain, J. Alansari, S. Alrashidi, W. Alqahtani, J. Alshaya, and N. Nagy, "Experimental implementation of shor's quantum algorithm to break rsa," in *2022 14th International Conference on Computational Intelligence and Communication Networks (CICN)*. IEEE, 2022, pp. 748–752.

[5] V. Bhatia and K. Ramkumar, "An efficient quantum computing technique for cracking rsa using shor's algorithm," in *2020 IEEE 5th international conference on computing communication and automation (ICCCA)*. IEEE, 2020, pp. 89–94.

[6] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum

computer," *SIAM review*, vol. 41, no. 2, pp. 303–332, 1999.

[7] T. Monz, D. Nigg, E. A. Martinez, M. F. Brandl, P. Schindler, R. Rines, S. X. Wang, I. L. Chuang, and R. Blatt, "Realization of a scalable shor algorithm," *Science*, vol. 351, no. 6277, pp. 1068–1070, 2016.

[8] https://medium.com/qiskit/applying-shors-algorithm-bbdfd6f05f7d

[9] https://quantum-computing.ibm.com/composer/docs/iqx/guide/ shors- algorithm

[10] https://www.nature.com/articles/s41598-021-9597

[11] Siyon Singh1, Eric Sakk, "Implementation and Analysis of Shor's Algorithm to Break RSA Cryptosystem Security", January 2024.

[12] Mavroeidis, V., Vishi, K., Zych, M. D., & Jøsang, A. (2018). The impact of quantum computing on present cryptography. arXiv preprint arXiv:1804.00200.

[13] IBM Quantum, https://learn.qiskit.org/course.

[14] Everitt, H. O. (Ed.). (2005). Experimental aspects of quantum computing. Springer Science

[15] Licentiate thesis in Physics "High Visibility of six photon entanglement"

[16] Robert Loredo, "Learn Quantum Computing with Python and IBM Quantum Experience"

Appendix -I

```
from qiskit import QuantumCircuit, Aer, transpile, assemble

from qiskit.visualization import plot_histogram

from math import gcd

from numpy.random import randint

import numpy as np

# Function to apply the controlled U gate

def cu(a, power, n):

    """Controlled multiplication by a mod N"""

    U = QuantumCircuit(n+1)

    for iteration in range(power):

        U.swap(0, 1)

        for q in range(n):

            U.x(q+1)
            U.swap(0, 1)

            U.swap(1, 0)

    U = U.to_gate()

    U.name = "%i^%i mod %i" % (a, power, N)

    c_U = U.control()

    return c_U

# Function to perform the quantum part of Shor's algorithm

def shor_quantum(n):

    # Choose a random number a

    a = randint(2, n)

    print("Chosen random number (a):", a)

    # Compute the greatest common divisor of a and n

    gcd_value = gcd(a, n)

    if gcd_value != 1:

        return gcd_value

    # Create a quantum circuit with 2 * n + 3 qubits

    qc = QuantumCircuit(n+4, n)

    # Apply Hadamard gates to the first n qubits

    qc.h(range(n))

    # Apply the controlled-U gates

    for q in range(n):

   qc.append(cu(a, 2 ** (2 ** q), n), [q] + list(range(n, n+4)))
```

```
    # Apply the quantum Fourier transform
    qc.swap(0, 3)
    for q in range(n):
        for j in range(q):
            qc.cp(np.pi / float(2 ** (q-j)), j, q)
            qc.h(q)
    # Measure the first n qubits
    qc.measure(range(n), range(n))
    # Simulate the quantum circuit
    simulator = Aer.get_backend('qasm_simulator')
    compiled_circuit = transpile(qc, simulator)
    job = simulator.run(assemble(compiled_circuit))
    result = job.result()
    counts = result.get_counts(qc)
    # Return the result
    return counts
# Main function to find the factors using Shor's algorithm
def shor_factorization(N):
    # Check if N is even
    if N % 2 == 0:
        return 2
    # Perform the quantum part of Shor's algorithm
    while True:
        counts = shor_quantum(N)
        for measured_value in counts:
            measured_int = int(measured_value, 2)
            if measured_int % 2 != 0:
                print("Measured", measured_int)
                continue
            guessed_divisor = gcd(measured_int, N)
            if 1 < guessed_divisor < N:
                print("Guessed:", guessed_divisor)
                return guessed_divisor
# Input number
N = int(input("Enter the number to factorize: "))
#Taking N = 12 as sample
f = shor_factorization (12)
print ("Factors:", f, "and", 12 // f)
```