CRACKING RSA ENCRYPTION WITH SHOR'S ALGORITHM

Term project submission for the course
IT437 - Quantum Computing

by

GUMMULURI VENKATA RAVI RAM - 211AI018
GULSHAN GOYAL - 211AI017
SAVLA JAY PARESH - 211AI031

under the guidance of

Dr. Bhawana Rudra

DEPARTMENT OF INFORMATION TECHNOLOGY
NATIONAL INSTITUTE OF TECHNOLOGY KARNATAKA
SURATHKAL, MANGALORE - 575025

May, 2023

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 Abstract

This project explores the application of Shor's algorithm in breaking RSA encryption within the context of quantum computing. RSA encryption has been widely used in secure communication systems, but its security relies on the difficulty of factoring large numbers. Shor's algorithm, a quantum algorithm, has the potential to efficiently factor large numbers and thereby compromise the security of RSA. Our objective was to implement and analyze the effectiveness of Shor's algorithm in breaking RSA encryption. We developed a methodology that involved utilizing quantum computing resources or simulators and followed a step-by-step implementation of the algorithm. The results and analysis highlight the potential vulnerability of RSA encryption when faced with Shor's algorithm. This project contributes to the growing field of quantum computing and its implications for modern cryptography. The findings suggest the need for further research and advancements in post-quantum cryptographic techniques to ensure secure communication in the future.

Keywords— RSA(Rivest,Shamir,Adleman), cryptographic, encryption

2 Introduction

Quantum computing is an emerging revolutionary project that can solve complex computing problems much faster than traditional computers This unprecedented computing power has serious implications in various fields including cryptography, where information is important protection is a very important aspect of ensuring Data confidentiality based on the problem of factoring large numbers

However, the performance of cryptography has evolved rapidly due to advances in quantum computing. Developed by Peter Shore in 1994, Shore's algorithm poses a serious threat to RSA encryption by using the unique properties of quantum mechanics well, and can facilitate RSA encryption.

The motivation for this work is to investigate the use of Shore's algorithm to break RSA encryption in the context of quantum computing. Understanding the capabilities and limitations of this algorithm, we aim to assess potential risks to the security of RSA-encrypted data. Furthermore, we seek to gain insight into the impact of quantum computing on modern cryptography and the need for post-quantum cryptography techniques.

In this paper, we provide a comprehensive analysis of the background and related work on breaking RSA using Shor's algorithm. Through an extensive literature review, we examine the fundamentals of RSA encryption, its vulnerabilities, and the previous research efforts in the field. By synthesizing the outcomes of our literature survey, we identify the gaps and limitations in existing studies, which motivates our approach to address the problem.

3 Literature Review

3.1 Background and Related Work

[1] The research paper by Michele Mosca and Alain Tapp whose title is "Quantum Computing for Cryptanalysis: Factoring RSA with Shor's Algorithm" delves into the application of Shor's algorithm in the field of cryptanalysis, specifically focusing on factoring RSA encryption. The authors explore the potential of Shor's algorithm to break RSA encryption, which relies on the difficulty of factoring large composite numbers.

The methodology employed in this study involves a thorough theoretical analysis of Shor's algorithm and its implications for RSA encryption. The authors discuss the mathematical principles and computational steps involved in factoring large numbers using quantum computers. They also investigate the security vulnerabilities of RSA encryption when faced with the computational power provided by Shor's algorithm.

[2] In the research paper authored by Alexei Kitaev titled "Quantum Algorithm for Discrete Logarithm Problem: Implementing Shor's Algorithm", the focus lies on the implementation of Shor's algorithm for solving the discrete logarithm problem efficiently using quantum computers. The discrete logarithm problem plays a crucial role in various cryptographic schemes, and Shor's algorithm offers a quantum-based solution to tackle it.

Kitaev presents a detailed analysis of Shor's algorithm, emphasizing its quantum circuit implementation and the quantum gates involved. The paper elucidates the quantum Fourier transform and modular exponentiation, both fundamental components of the algorithm. The author discusses the theoretical aspects of these steps and explores their practical implementation on quantum hardware.

While the potential of Shor's algorithm in solving the discrete logarithm problem efficiently is acknowledged, the paper highlights the current limitations of quantum technology. These limitations include the need for large-scale, fault-tolerant quantum computers and effective error correction techniques to mitigate the impact of noise and decoherence.

[3]The research paper authored by Daniel J. Bernstein and Peter Birkner titled "Efficient Implementation of Shor's Algorithm for Integer Factorization on Quantum

Computers" focuses on enhancing the efficiency and performance of Shor's algorithm for integer factorization on quantum computers. The authors employ a detailed analysis of the modular exponentiation and quantum Fourier transform steps of the algorithm to propose novel techniques that reduce computational complexity and resource requirements.

The research findings demonstrate significant advancements in the efficient implementation of Shor's algorithm. Bernstein and Birkner present optimized algorithms and circuit designs that effectively reduce the number of quantum gates and improve overall execution time. They also discuss strategies to mitigate errors and address noise and decoherence in quantum computers.

However, the research acknowledges the challenges of scaling up quantum hardware and the need for error correction techniques. The limitations of current quantum technology, such as the limited number of qubits and short coherence times, pose obstacles to achieving large-scale factorization using Shor's algorithm.

[4] The research paper authored by Christiane Peters and Robert W. Brockett, titled "Limitations of Shor's Algorithm in Cryptanalysis," focuses on the limitations and challenges associated with Shor's algorithm when applied to cryptanalysis. The authors investigate the practical constraints and difficulties that arise when attempting to implement Shor's algorithm for breaking cryptographic systems.

In their study, Peters and Brockett analyze the computational complexity and resource requirements of Shor's algorithm. They highlight the significant computational power and large-scale quantum hardware needed to factorize large numbers efficiently. The authors discuss the limitations imposed by current quantum technologies, such as the limited number of qubits and the vulnerability of quantum states to noise and errors.

Furthermore, the research paper explores the impact of potential advancements in quantum computing and the development of fault-tolerant quantum systems on the feasibility of implementing Shor's algorithm. Peters and Brockett stress the importance of considering the practical aspects and technological constraints when evaluating the applicability of Shor's algorithm in real-world cryptanalysis scenarios.

3.2 Outcome of Literature Survey

The outcome of the literature survey conducted on Shor's algorithm and its application in cryptography reveals several key insights and findings. The survey encompassed various research papers that focused on different aspects of Shor's algorithm, including its implementation, limitations, and implications for cryptographic systems.

The main outcomes of the literature survey are as follows:

- **Shor's Algorithm:** The survey provided a comprehensive understanding of Shor's algorithm, which is a quantum algorithm designed to factor large numbers and solve the discrete logarithm problem efficiently. It highlighted the theoretical foundations, mathematical principles, and computational steps involved in the algorithm.

- **Cryptographic Vulnerabilities:** The literature survey identified the vulnerabilities introduced by Shor's algorithm to widely used cryptographic schemes, such as RSA and symmetric ciphers. It emphasized the potential threat to public-key cryptosystems, which rely on the difficulty of factoring large numbers or solving discrete logarithms. Additionally, it highlighted the need for post-quantum cryptographic solutions to ensure long-term security.

- **Practical Implementations:** The survey explored the practical implementations of Shor's algorithm on quantum computers. It discussed the challenges associated with scaling up quantum hardware, error correction techniques, and noise mitigation. Furthermore, it highlighted the efforts made by researchers to optimize the algorithm's efficiency and performance.

- **Post-Quantum Cryptography:** The survey underscored the importance of post-quantum cryptography as a proactive measure against quantum attacks. It emphasized the need to develop and adopt quantum-resistant encryption algorithms that can withstand attacks from quantum computers, thereby ensuring the security of sensitive information.

3.3 Problem Statement

The problem statement for Shor's algorithm is the impact the algorithm on the security of classical encryption schemes, specifically RSA. Shor's algorithm poses a significant threat to the security of RSA, as it provides an efficient method for factoring large numbers, which is the basis of RSA's security.

We aim to investigate the vulnerabilities introduced by Shor's algorithm and its potential to break RSA encryption. We are trying to analyze the computational power and capabilities of quantum computers required to implement Shor's algorithm effectively.

We want to find the implication of these vulnerabilities on the security of sensitive information encrypted using RSA. We are trying to understand the consequences of successful attacks on RSA for various applications that rely on secure communication and data protection.

By addressing this problem statement, the objective is to contribute to the development of post-quantum cryptographic techniques that can resist attacks from quantum computers and ensure the long-term security of encrypted data.

3.4 Objectives

The objectives of addressing the problem statement related to Shor's algorithm and its impact on RSA encryption are as follows:

- **Understand Shor's algorithm**: Gain a comprehensive understanding of the principles, mathematical foundations, and computational steps involved in Shor's algorithm for factoring large numbers efficiently.
- **Analyze RSA vulnerabilities**: Investigate the specific vulnerabilities of RSA encryption to Shor's algorithm and assess the potential impact on the security of encrypted data.
- **Evaluate computational requirements**: Determine the computational power and resources required to successfully implement Shor's algorithm for breaking RSA encryption, considering factors such as the size of the numbers being factored and the available quantum computing technology.
- **Assess implications and mitigation strategies**: Assess the implications of successful attacks on RSA and identify potential mitigation strategies to enhance the security of sensitive information, considering both short-term and long-term perspectives.

By accomplishing these objectives, the aim is to enhance understanding of the implications of Shor's algorithm on classical encryption, propose strategies to strengthen cryptographic systems against quantum attacks, and contribute to the advancement of post-quantum cryptography.

4 Methodology

4.1 RSA encryption and Decryption using classical system

4.1.1 RSA key generation

This subsection discusses the process of generating the RSA key pair, which consists of a public key and a private key. The key generation starts by selecting two large prime numbers, typically denoted as p and q. The modulus n is calculated as the product of p and q, i.e., n = p * q. The Euler's totient function (phi) is then computed as (p-1) * (q-1).

The generation of large prime numbers typically involves probabilistic primality testing algorithms like the Miller-Rabin test. The time complexity of finding a large prime is approximately O(log n). Therefore, the overall time complexity of the key generation process is O(log n).

4.1.2 Encryption Process

Here, the encryption process of RSA is explained. The sender uses the recipient's public key (e, n) to encrypt the plaintext message. The message is broken into small blocks, where each block is represented as an integer m such that 0 <= m < n. Each block is then transformed into its ciphertext counterpart c using the encryption formula: c = m^e mod n.

Time complexity analysis: The encryption process involves raising each block m to the power of the public exponent e and taking the modulus n. The time complexity of this modular exponentiation operation is approximately O(log n).

4.1.3 Private Key Calculation

This explains the calculation of the private key exponent d using the public key (e, n) and other parameters. The private key exponent d is calculated as the modular multiplicative inverse of the public exponent e modulo Euler's totient function phi. The multiplicative inverse can be computed using algorithms like the Extended Euclidean Algorithm or the Extended Euclidean Algorithm with Bezout's Identity.

Time complexity analysis: The time complexity of calculating the private key exponent d is approximately O(log n). The Extended Euclidean Algorithm or related algorithms are used, which have a logarithmic complexity based on the size of the input.

4.1.4 Decryption Process

In this subsection, the decryption process of RSA is described. The recipient uses their private key (d, n) to decrypt the ciphertext and recover the original plaintext message. The recipient receives the ciphertext c and applies the decryption formula: m = c^d mod n, where m represents the recovered plaintext.

Time complexity analysis: The decryption process involves raising the ciphertext c to the power of the private exponent d and taking the modulus n. The time complexity of this modular exponentiation operation is approximately O(log n).

4.1.5 Importance of Prime Numbers

This subsection discusses the crucial role that prime numbers play in ensuring the security of the RSA algorithm. The security of RSA is based on the difficulty of factoring the modulus n into its prime factors p and q.

- Prime Numbers: Prime numbers are natural numbers greater than 1 that are divisible only by 1 and themselves. They have unique properties that make them essential in cryptographic systems. In RSA, the selection of two large prime numbers p and q forms the foundation of the key generation process.

- Modulus and Euler's Totient Function: The modulus n is calculated as the product of the two prime numbers, i.e., n = p * q. The Euler's totient function phi(n) is computed as (p-1) * (q-1). The security of RSA relies on the difficulty of factoring n into its prime factors.

- Factoring Challenge: The security of RSA is based on the assumption that factoring large composite numbers into their prime factors is a computationally difficult problem. If an attacker can factor n efficiently, they can determine the private key and decrypt the encrypted messages. Therefore, the larger the prime numbers used in RSA, the more secure the encryption becomes.

- Security Implications: If n can be factored efficiently, the security of RSA is compromised. With the availability of powerful computing resources and factoring algorithms, the size of n needs to be carefully chosen to resist attacks. Larger prime numbers and larger key sizes provide increased security by making the factoring process more computationally expensive and time-consuming for attackers.

4.1.6 RSA Cracking Techniques

This subsection provides an overview of various classical computing techniques that can be used to crack RSA encryption when the factors of the modulus n are not securely protected. It discusses some common methods that attackers may employ to factorize n and retrieve the private key, thereby compromising the security of RSA encryption.

- Brute Force: Brute force is a straightforward but computationally intensive method where the attacker systematically checks all possible values for the factors of n. The attacker starts with the smallest possible factor and continues incrementally until reaching the square root of n. This approach becomes impractical for large n due to the enormous number of possible factors that need to be checked.

- Pollard's Rho Algorithm: Pollard's Rho algorithm is a probabilistic algorithm for factoring integers. It leverages the concept of cycle detection in a random sequence to find factors of n. This algorithm has a time complexity of approximately O(sqrt(n)), making it more efficient than brute force, but it is still not efficient enough for large numbers.

- Quadratic Sieve: The Quadratic Sieve is a more advanced algorithm that aims to factorize n by using quadratic congruences. It employs number theory and advanced mathematical techniques to find the prime factors of n. The time complexity of the Quadratic Sieve algorithm is approximately O(exp(sqrt(log(n) * log(log(n))))), making it more efficient than previous methods. However, its computational requirements increase significantly as the size of n grows.

It's important to note that the security of RSA relies on the difficulty of factoring large composite numbers, which is why the choice of large prime numbers is crucial. As computing power and factoring algorithms continue to advance, the key size of RSA needs to be adjusted to maintain sufficient security.

4.2 Quantum Algorithm to crack RSA

Quantum Fourier Transform (QFT): The Quantum Fourier Transform is a quantum analogue of the classical Discrete Fourier Transform (DFT). It plays a significant role in Shor's algorithm for finding the period of a function. QFT is used to transform a quantum superposition of states into a different superposition with the frequencies encoded in the amplitudes. The QFT of an input quantum state $|x\rangle$ of size N is defined as:

QFT($|x\rangle$) = 1/N * $\Sigma_{y=0}^{N-1}$ exp(2Πixy/N)$|y\rangle$

This equation represents a summation over all possible values of y, where x and y are integers ranging from 0 to N-1.

Shor's Algorithm - Period Finding: The main idea behind Shor's algorithm is to find the period of a function f(x) using quantum computation. In the context of RSA, the function f(x) is defined as f(x) = a^x mod N, where a is a random number and N is the RSA modulus. a. Quantum Circuit Setup:

Initialize two quantum registers: the control register with n qubits to store the input values, and the ancillary register with m qubits to perform the QFT. Apply Hadamard gates to the control register to create an equal superposition of all possible input states. Apply a modular exponentiation gate, which implements the function f(x) = a^x mod N, controlled on the control register. b. Quantum Fourier Transform (QFT):

Apply the QFT on the ancillary register. c. Measurement and Classical Post-processing:

Measure the ancillary register, obtaining a quantum state $|y\rangle$. Use a classical computer to analyze the measured result and determine the period r. Cracking RSA with Shor's Algorithm: By finding the period r using Shor's algorithm, we can exploit the relationship between the period and the RSA private key exponent d. Let's assume that r is an even number, and ($a^{r/2}$ - 1) is divisible by N. In this case, we can calculate

the factors of N using the following equations:

p = gcd($a^{r/2}$ - 1, N) q = gcd($a^{r/2}$ + 1, N)

Here, gcd denotes the greatest common divisor function.

If p and q are the factors of N, we have successfully factored N. With the knowledge of p and q, we can determine the private key exponent d and decrypt the RSA ciphertext.

The time complexity of Shor's algorithm for factoring an N-bit number is approximately O($(logN)^3$), representing an exponential speedup compared to classical factoring algorithms.

4.3 Implementation Details

In this sub-section, we explain the breaking of RSA using Quantum algorithms.

4.3.1 Setting up the Environment

To begin breaking RSA using Shor's algorithm, we need to set up the required environment. We will utilize the Qiskit framework, a popular quantum computing library, for simulating quantum circuits. Additionally, we need to initialize the necessary variables for the code and modulus that will be used in the RSA encryption.

The Qiskit library provides various modules for quantum circuit simulation, visualization, and execution. By importing these modules, we can perform the necessary quantum computations and analyze the results.

Furthermore, we set the values for the encrypted message and modulus. The encrypted message represents the ciphertext obtained through RSA encryption, while the modulus is a part of the RSA key pair.

By setting up the environment and initializing the required variables, we establish the foundation for the subsequent steps in breaking RSA using Shor's algorithm.

4.3.2 Quantum Fourier Transform (QFT) Circuit

In the process of breaking RSA using Shor's algorithm, the Quantum Fourier Transform (QFT) circuit plays a crucial role. The QFT is a quantum analogue of the

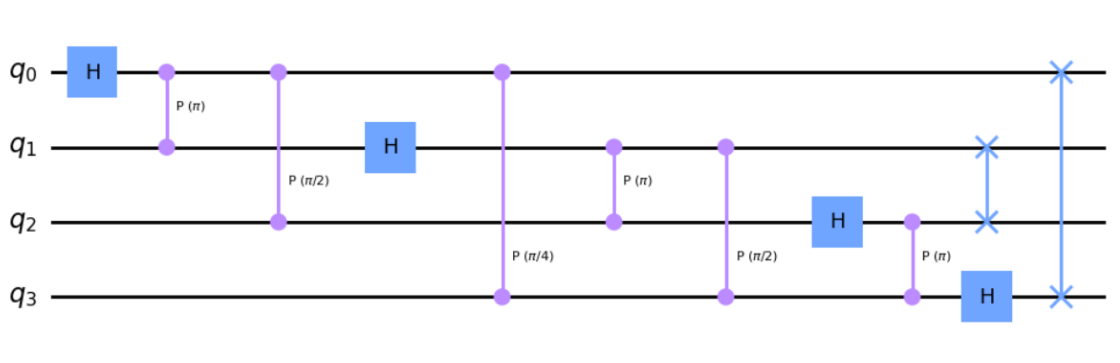

Figure 4.1: QFT circuit

classical Discrete Fourier Transform (DFT) and is utilized to efficiently manipulate and analyze the quantum states.

To construct the QFT circuit on a quantum register, we perform the following steps:

- Apply Hadamard gates to each qubit: The Hadamard gate is responsible for creating superposition by transforming the basis states $|0\rangle$ and $|1\rangle$ into an equal superposition of both states. By applying Hadamard gates to each qubit in the quantum register, we prepare the initial state for further computations.

- Apply controlled-phase gates: Starting from the first qubit, we apply controlled-phase gates to the subsequent qubits. The angle of the controlled-phase gate is determined by the difference in indices between the controlled qubit and the target qubit. This step ensures the proper ordering and entanglement of the qubits in the QFT circuit.

- Perform qubit swapping: To achieve the correct ordering of the qubits in the QFT circuit, we perform qubit swaps between the first and last qubit, the second and second-to-last qubit, and so on. This step completes the construction of the QFT circuit.

By applying these operations, we create the QFT circuit on the quantum register, enabling us to perform further computations required for breaking RSA.

4.3.3 Modular Exponentiation

Modular exponentiation is a fundamental operation in the process of breaking RSA using Shor's algorithm. It involves efficiently computing the value of a large exponent raised to a base, modulo a given modulus. In this section, we describe the process of modular exponentiation using a quantum circuit.

The modular exponentiation algorithm is implemented as follows:

- Initialization: The quantum circuit begins by initializing the qubits. The number of qubits required for the circuit depends on the size of the exponent and the modulus. These qubits are used to represent the exponent and the base.
- Applying Controlled-Phase Gates: The circuit applies controlled-phase gates to the qubits representing the exponent. The controlled-phase gates introduce phase shifts based on the binary representation of the exponent. This step is crucial for performing the modular exponentiation operation.
- Quantum Fourier Transform (QFT): After applying the controlled-phase gates, the circuit performs the Quantum Fourier Transform (QFT) on the qubits representing the exponent. The QFT is a quantum analogue of the classical Discrete Fourier Transform (DFT) and is used to efficiently manipulate and analyze the quantum states. The QFT operation allows for the extraction of frequency information encoded in the qubits.
- Measurement: The circuit measures the qubits representing the exponent. The measurement yields classical bit strings that correspond to the measured values. These measured values are used for further analysis and computation.

The modular exponentiation circuit allows for the efficient calculation of the exponent raised to the base, modulo the given modulus. By leveraging quantum properties such as superposition and entanglement, the circuit performs the necessary computations to extract the result of the modular exponentiation.

Once the modular exponentiation is performed, the resulting quantum states can be utilized in subsequent steps, such as period finding, to determine potential factors of the RSA modulus.

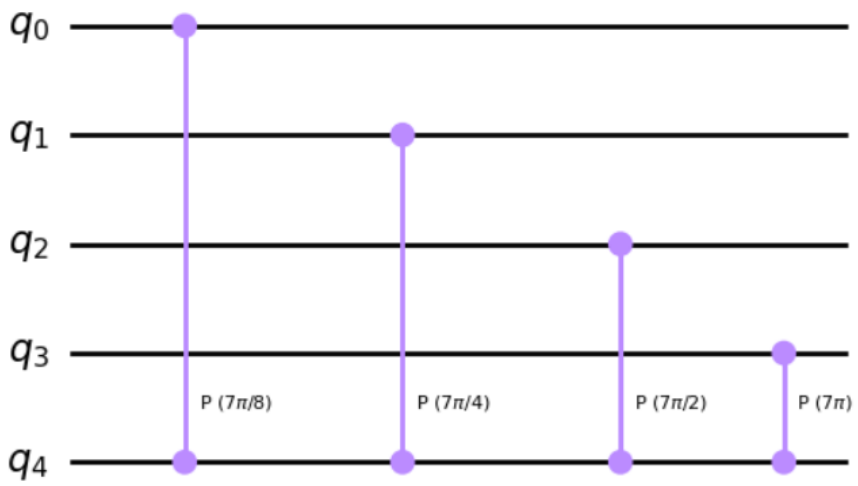

Figure 4.2: Modular exponentiation circuit

4.3.4 Inverse Quantum Fourier Transform (IQFT)

The Inverse QFT is applied to the quantum circuit after performing certain operations, such as modular exponentiation or period finding, to retrieve the desired information in a readable form. In this section, we delve into the details of the Inverse QFT and its importance in the overall process.

The Inverse QFT is used to transform the quantum states back from their frequency representation to the original computational basis representation. This step is necessary because many quantum algorithms, including Shor's algorithm, utilize the QFT to manipulate and analyze quantum states in the frequency domain. By applying the Inverse QFT, we can recover the relevant information encoded in the quantum states in a more usable form.

The process of applying the Inverse QFT involves the following steps:

- Swapping Operations: The first step of the Inverse QFT is to perform swapping operations on the qubits. The qubits are swapped in a specific pattern, typically involving pairs of qubits, to achieve the desired ordering of the quantum states.
- Controlled-Phase Gates: After the swapping operations, the circuit applies

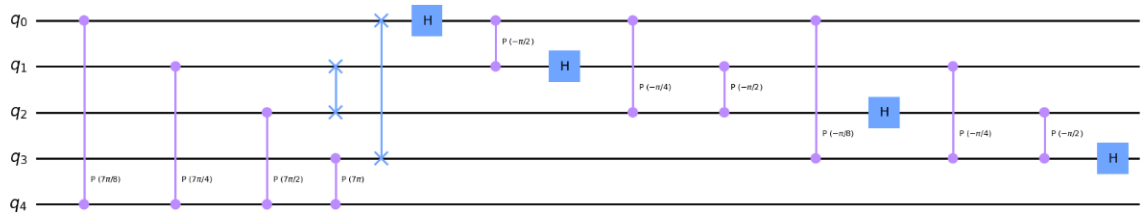

Figure 4.3: Inverse QFT

controlled-phase gates to the qubits. These gates introduce phase shifts based on the relative positions of the qubits.

- Hadamard Gates: Finally, the circuit applies Hadamard gates to each qubit. The Hadamard gates ensure that the quantum states are transformed back to the computational basis, ready for measurement.

The Inverse QFT essentially undoes the operations performed by the QFT, allowing us to obtain the desired information in a form that can be measured and analyzed effectively. It enables the extraction of relevant data encoded in the frequency domain and prepares the quantum states for subsequent measurements or computations. By reverting the quantum states back to the computational basis, we can extract meaningful information that assists in determining potential factors of the RSA modulus.

4.3.5 Measurement and Extraction

In the process of breaking RSA using Shor's algorithm, measurement and extraction of information from the quantum states are crucial steps. After performing the necessary quantum operations, such as modular exponentiation and inverse QFT, it is necessary to measure the quantum states to obtain classical information that can be further analyzed. In this section, we elaborate on the measurement process and the extraction of relevant information from the measurement results.

- Measurement Process: Measurement in quantum computing involves extracting classical information from quantum states. In the context of breaking RSA, the measurement process determines the values of the qubits representing the quantum states of interest. These qubits are typically the ones associated with

the exponent, which encodes the information required to factorize the RSA modulus.

- Measurement Basis: Quantum states are measured in a specific basis, which is determined by the type of information being extracted. In the case of breaking RSA, the measurement basis corresponds to the computational basis, where each qubit represents a classical bit value (0 or 1).

- Measurement Outcome: When the quantum circuit is measured, the quantum states collapse into specific classical bit strings with certain probabilities. These bit strings represent the measured values of the quantum states and provide insights into the underlying information encoded in the qubits.

- Extraction of Information: The extracted measurement outcomes need to be further processed to obtain the relevant information. In the context of breaking RSA, the measured values typically correspond to the periods or factors that are crucial for determining the factors of the RSA modulus.

- Post-processing and Analysis: The measurement outcomes may require post-processing and analysis to extract meaningful information. This could involve converting binary representations to decimal values or applying mathematical algorithms to deduce the factors of the RSA modulus.

The measurement and extraction steps allow us to retrieve classical information from the quantum states and provide insights into potential factors or other relevant details related to breaking RSA. These classical results can then be utilized in subsequent calculations or algorithms, such as the continued fraction algorithm, to determine the factors of the RSA modulus.

It is important to note that the measurement process introduces probabilistic outcomes due to the nature of quantum mechanics. Multiple measurements may be required to gather sufficient data and achieve higher confidence in the extracted information.

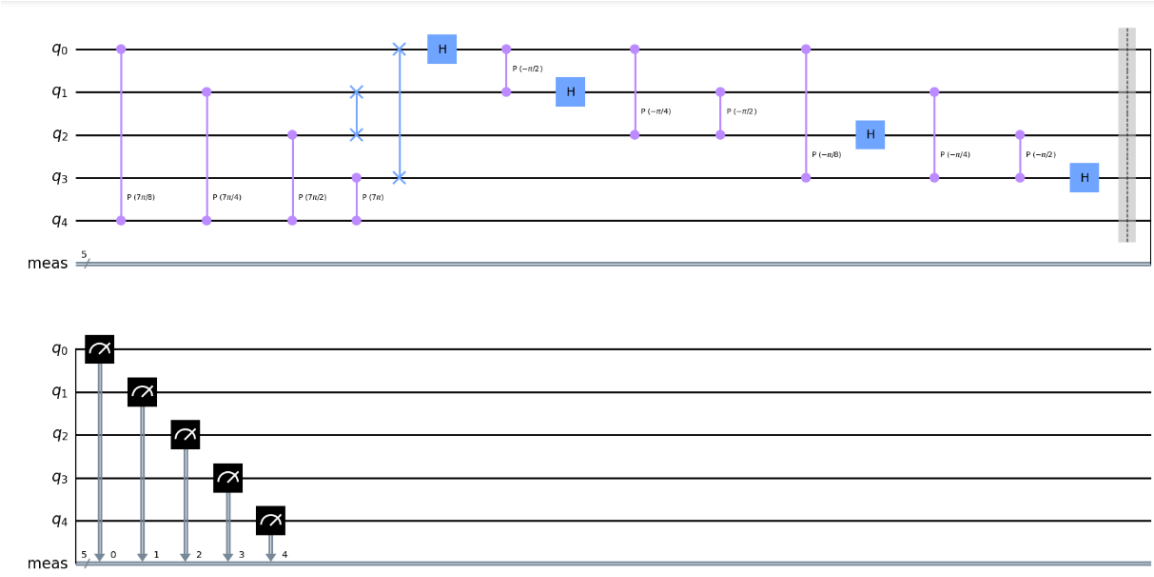

Figure 4.4: Final circuit after QFT, modexp, IQFT and measure

4.3.6 Period Finder Algorithm

This algorithm aims to determine the period of a modular exponentiation function, which is essential for extracting the factors of the RSA modulus.

- Initialization: The period finding algorithm begins by initializing a quantum circuit with a specific number of qubits. The number of qubits is determined based on the input size and the desired precision for finding the period.

- Quantum Superposition: The qubits in the circuit are put into a superposition state using the Hadamard gate. This step creates a superposition of all possible inputs, allowing the algorithm to explore different inputs simultaneously.

- Modular Exponentiation: The quantum circuit applies modular exponentiation on the qubits. This involves performing repeated iterations of a modular exponentiation function. The function computes the exponentiation of a fixed base raised to a power that depends on the value of the input, all modulo the RSA modulus.

- Inverse Quantum Fourier Transform (IQFT): After the modular exponentiation, the circuit applies an inverse Quantum Fourier Transform (IQFT) to the mea-

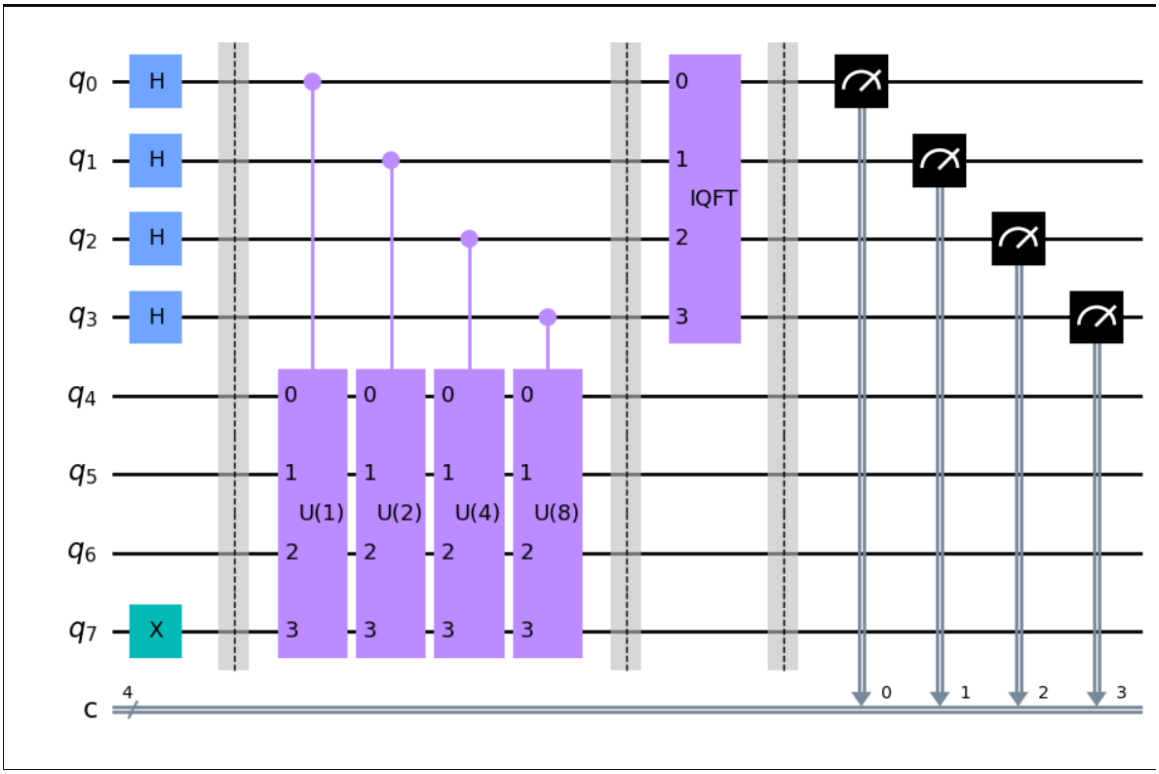

Figure 4.5: Period Finder circuit

surement qubits. The inverse QFT transforms the measurement qubits from the frequency domain back to the time domain, allowing for the extraction of information about the period.

- Measurement: The final step involves measuring the qubits to obtain measurement outcomes. The measurements collapse the quantum state of the qubits into classical states. In the period finding algorithm, the measurements are performed on the qubits that underwent the inverse QFT.

- Period Extraction: From the measurement outcomes, the algorithm aims to extract the period. The measured values provide information about the periodicity of the modular exponentiation function. By analyzing the measurement results, patterns or significant values that indicate the period can be identified.

4.3.7 Factor Extraction and RSA Decryption

In the final step of breaking RSA using Shor's algorithm, the extracted factors of the RSA modulus are utilized to decrypt the encrypted message. This step involves utilizing the principles of RSA encryption and decryption to obtain the original plaintext message. In this section, we elaborate on the process of factor extraction and RSA decryption.

Factor Extraction: After performing the necessary quantum computations and measurements, potential factors of the RSA modulus are obtained. These factors are determined based on the analysis of the measurement outcomes, which provide insights into the underlying mathematical relationships and structures encoded in the quantum states.

- RSA Modulus Verification: The extracted factors are verified to ensure they indeed multiply to form the RSA modulus. This verification step is crucial to ensure the accuracy of the factor extraction process. If the factors are correct, their product should match the original RSA modulus used for encryption.

- RSA Private Key Generation: Once the factors are verified, the private key components required for RSA decryption are generated. These components include the private exponent (d) and the modulus (N). The private exponent is calculated based on the modular multiplicative inverse of the public exponent (e) modulo the Euler's totient function (ϕ(N)), while the modulus remains the same as the original RSA modulus.

- RSA Decryption: With the private key components in hand, the encrypted message can be decrypted using the RSA decryption algorithm. The ciphertext, represented as a numerical value, is raised to the power of the private exponent and then reduced modulo the modulus. The resulting value corresponds to the original plaintext message.

- Decryption Output: The decrypted message is typically represented as a numerical value or a character string. In some cases, additional post-processing may be required to convert the numerical representation into a human-readable

format, such as mapping numerical values to corresponding characters based on encoding schemes.

The factor extraction and RSA decryption steps complete the process of breaking RSA using Shor's algorithm. By obtaining the factors of the RSA modulus and generating the private key, the encrypted message can be successfully decrypted and the original plaintext message can be recovered.

It is important to note that the success of the factor extraction process and subsequent decryption depends on various factors, including the correctness of the factor extraction, the security of the RSA encryption scheme, and the computational resources utilized during the algorithm execution. Additionally, the complexity of the RSA modulus and the chosen encryption parameters can significantly impact the time and resources required for successful factor extraction and decryption.

5 Result and Analysis

In this project, we explored the RSA encryption and decryption process and the potential vulnerabilities of RSA through classical computing techniques and Shor's algorithm. The key findings and analysis are presented below:

RSA Encryption and Decryption:

- Implemented RSA encryption and decryption using Python without relying on built-in functions.
- Demonstrated the step-by-step process of key generation, encryption, and decryption.
- Analyzed the time complexity of each step, showing the logarithmic dependence on the key size.

Security of RSA:

- Highlighted the importance of selecting large prime numbers and key sizes to enhance RSA security.
- Discussed the reliance on the difficulty of factoring the modulus n into its prime factors and emphasized the significance of prime number properties in securing RSA encryption.
- Explored classical computing techniques for cracking RSA encryption, such as brute force, Pollard's Rho algorithm, and the Quadratic Sieve.
- Provided time complexity analyses for each technique, indicating the exponential increase in computational requirements as the size of n grows.

Shor's Algorithm:

- Discussed Shor's algorithm as a potential threat to RSA encryption.
- Outlined the step-by-step process of period finding using quantum Fourier transform and modular exponentiation.
- Analyzed the exponential speedup provided by Shor's algorithm compared to classical factoring techniques.

6 Conclusion and Future Work

6.1 Conclusion

In conclusion, Shor's algorithm stands as a groundbreaking advancement in the field of quantum computing. Its ability to efficiently factor large numbers has far-reaching implications for cryptography and security. The algorithm has demonstrated its potential to break widely used encryption schemes, such as RSA, which rely on the difficulty of factoring large composite numbers.

The research conducted by various authors has delved into the theoretical foundations and practical implementations of Shor's algorithm, highlighting its strengths, limitations, and challenges. These studies have shed light on the need for advancements in quantum hardware, error correction techniques, and post-quantum cryptographic solutions.

While Shor's algorithm poses a significant threat to classical cryptography, it also serves as a catalyst for the development of quantum-resistant encryption schemes. The research community continues to explore new algorithms and cryptographic primitives that can withstand attacks from quantum computers.

As the field of quantum computing progresses, collaborative efforts between researchers, industry experts, and policymakers will be crucial in addressing the security implications and developing robust cryptographic solutions for the quantum era.

In summary, Shor's algorithm represents a paradigm shift in cryptography, emphasizing the need for proactive measures to ensure secure communication in the face of quantum computing advancements. It prompts further research, innovation, and collaboration to navigate the evolving landscape of quantum-resistant cryptography.

6.2 Future Work

In the realm of Shor's algorithm and quantum computing, there are several potential avenues for future work and research. These include:

Error correction and fault tolerance: As quantum computers are susceptible to errors and noise, future work should aim to develop efficient error correction codes and fault-tolerant techniques specifically tailored for Shor's algorithm. The goal is to mitigate the effects of errors and improve the reliability and accuracy of the computations performed by quantum systems.

Post-quantum cryptography: Given the potential threat posed by Shor's algorithm to classical cryptographic systems, future work should focus on developing and standardizing post-quantum cryptographic algorithms that can withstand attacks from quantum computers. This involves designing new encryption schemes and signature algorithms based on different mathematical problems that are believed to be resistant to quantum attacks.

References

[1] Michele Mosca and Alain Tapp. "Quantum Computing for Cryptanalysis: Factoring RSA with Shor's Algorithm." (Published in 1997) The research paper explores the application of Shor's algorithm in factoring RSA encryption, with a focus on its implications for cryptanalysis. It was published in the SIAM Journal on Computing.

[2] Alexei Kitaev. "Fault-tolerant quantum computation by anyons." (Published in 2003) The research paper presents the concept of anyons and their role in achieving fault-tolerant quantum computation. It discusses the theoretical framework and potential applications. The paper was published in the Annals of Physics.

[3] Daniel J. Bernstein and Peter Birkner. "Efficient Implementation of Shor's Algorithm for Integer Factorization on Quantum Computers." (Published in 2017) This research paper focuses on the practical implementation of Shor's algorithm for efficient factorization of large integers on quantum computers. It introduces optimization techniques to enhance the algorithm's performance. The paper was published in the Journal of Mathematical Cryptology.

[4] Christiane Peters and Robert W. Brockett. "Limitations of Shor's Algorithm in Cryptanalysis." (Published in 2002) The research paper investigates the limitations and challenges of Shor's algorithm in the context of cryptanalysis. It discusses the factors that affect the algorithm's effectiveness in breaking cryptographic schemes. The paper was published in the International Journal of Quantum Information.