3-25-2024

Factorization in Cybersecurity: a Dual Role of Defense and Vulnerability in the Age of Quantum Computing

Himateja Arukala
Illinois State University, arukala.himateja@gmail.com

FACTORIZATION IN CYBERSECURITY: A DUAL ROLE OF DEFENSE AND VULNERABILITY IN THE AGE OF QUANTUM COMPUTING

HIMATEJA ARUKALA

83 Pages

One of the most critical components of modern cryptography and thus cybersecurity is the ability to factor large integers quickly and efficiently. RSA encryption, one of the most used types, is based largely on the assumption that factoring for large numbers is computationally infeasible for humans and computers alike. However, with quantum computers, people can use an algorithm like Shor's algorithm to perform the same task exponentially faster than any normal device ever could. This investigation will go into the strength and vulnerability of RSA encryption using the power of factorization in an age of quantum computers.

We start by looking at the foundations of both classical and quantum factoring with greater detail at number field sieve (NFS) and Shor's. We examine the mathematical background of each topic and the associated algorithms. We conclude with theoretical analysis and experimental simulations that address the difficulty and implications of the above-mentioned algorithms in cryptography.

The final thing that I will be discussing is where quantum computing is at present and how this could pose a threat to the current type of cryptographic systems, we use every day. I will be mentioning how we need post-quantum cryptography and how people are currently creating algorithms that are designed to be attack-resistant even to large-scale quantum computers.

This investigation has shown the changing dynamics of cybersecurity in the quantum era and helps us understand the challenges and the need to innovate the current cryptographic systems.

KEYWORDS: factorization; cybersecurity; quantum computing; Shor's algorithm; Number Field Sieve; post-quantum cryptography.

FACTORIZATION IN CYBERSECURITY: A DUAL ROLE OF DEFENSE AND VULNERABILITY IN THE AGE OF QUANTUM COMPUTING

HIMATEJA ARUKALA

A Thesis Submitted in Partial
Fulfillment of the Requirements
for the Degree of

MASTER OF SCIENCE

School of Information Technology

ILLINOIS STATE UNIVERSITY

2024

FACTORIZATION IN CYBERSECURITY: A DUAL ROLE OF DEFENSE AND VULNERABILITY IN THE AGE OF QUANTUM COMPUTING

HIMATEJA ARUKALA

COMMITTEE MEMBERS:

Yongning Tang, Chair

Chung-Chih Li, Co-Chair

Rudra Prasad Baksi

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to all those who have supported and guided me throughout the process of completing this thesis. This work would not have been possible without the unwavering support, encouragement, and expertise of several individuals.

First and foremost, I would like to extend my sincere thanks to my esteemed thesis committee members, Dr. Yongning Tang, Dr. Chung-Chih Li, and Dr. Rudra Prasad Baksi. Your invaluable insights, constructive feedback, and constant guidance have been instrumental in shaping this research. I am truly grateful for your time, patience, and dedication in helping me navigate through the complexities of this study.

Dr. Yongning Tang, your mentorship and guidance as my committee co-chair have been invaluable throughout this journey. Your extensive experience and deep understanding of cryptography have greatly enriched my research. I am grateful for your patient explanations, meticulous reviews, and the countless hours you have dedicated to helping me refine my work. Your encouragement and belief in my abilities have been a constant source of motivation.

Dr. Chung-Chih Li, as my committee co-chair, your profound knowledge and expertise in the field of quantum cryptography have been an immense source of inspiration. Your thought-provoking questions and insightful suggestions have challenged me to think critically and explore new avenues of research. Thank you for your unwavering support and for always being available to discuss ideas and provide guidance.

Dr. Rudra Prasad Baksi, I am deeply appreciative of your contributions as a committee member. Your expertise in cybersecurity and your keen eye for detail have significantly strengthened this thesis. Thank you for your insightful comments, probing questions, and valuable suggestions, which have helped me improve the quality and coherence of my work.

I would also like to extend my heartfelt thanks to my dear friends, Mahima Arukala, Srivalli Chilukuri, Akhil Raghava Kalal, Yashaswini Mekala, Praneel Reddy, and Yashwanth Rachala, Durgamatha Krishna Prasad. Your unwavering support, encouragement, and friendship have been a lifeline throughout this challenging process. Whether it was engaging in thought-provoking discussions, providing moral support during long hours of research, or simply being there to lend a listening ear, your presence has made this journey more enjoyable and meaningful. I am truly fortunate to have such amazing friends who have stood by me every step of the way.

I would like to extend my gratitude to Illinois State University and the School of Information Technology for providing me with the resources, facilities, and academic support necessary to conduct this research. The vibrant academic community and the opportunities for intellectual growth have been instrumental in my development as a researcher.

Finally, I would like to express my deepest appreciation to my family for their unconditional love, understanding, and support. Your unwavering belief in me has been a constant source of strength and motivation. Thank you for always being there, celebrating my successes, and encouraging me through the challenges.

To everyone who has contributed to this thesis in any way, I extend my heartfelt thanks. Your support and guidance have been invaluable, and I am truly grateful for the opportunity to learn from each and every one of you. Thank you for being a part of this incredible journey.

H. A.

CONTENTS

Page

ACKNOWLEDGMENTS i

TABLES vii

FIGURES viii

CHAPTER I: INTRODUCTION 1

1.1 Background Information 1

1.1.1 The Role of Factorization in Cybersecurity 1

1.1.2 Introduction to Quantum Computing 2

1.2 Problem Statement 3

1.3 Research Objectives 3

1.4 Thesis Structure 3

CHAPTER II: LITERATURE REVIEW 5

2.1 Traditional Factorization Techniques and Their Limitations 5

2.1.1 Mathematical Foundations of Factorization 5

2.1.1.1 Prime Numbers and Factorization. 6

2.1.1.2 Factorization Algorithms. 6

2.1.1.3 Modular Arithmetic. 7

2.1.1.4 Hardness Assumptions. 7

2.1.2 Historical Development of Factorization-based Cryptography 8

2.1.2.1 The RSA Cryptosystem. 8

2.1.2.2 Early Factorization Challenges. 9

2.1.2.3 Increasing Key Sizes. 10

2.1.2.4 Quantum Computing and Shor's Algorithm. 10

2.1.2.5 Post-Quantum Cryptography. 10

2.1.2.6 Continued Importance of Factorization. 11

2.2 Evolution of Quantum Computing 12

2.2.1 Theoretical Foundations 12

2.2.1.1 Superposition. 13

2.2.1.2 Entanglement. 13

2.2.2 Quantum Algorithms for Factorization 13

2.3 Quantum Computing's Threat to Cryptography 14

2.4 Current Cybersecurity Practices and Quantum Vulnerabilities 15

CHAPTER III: THEORETICAL FRAMEWORK 17

3.1 Classical Factorization: Number Field Sieve (NFS) 17

3.2 Quantum Factorization: Shor's Algorithm 20

3.2.1 Classical Reduction 21

3.2.2 Quantum Period Finding 21

3.2.3 Quantum Gates in Shor's Algorithm 22

3.2.4 Quantum computing's speed advantage in factorization 24

3.3 Fast Fourier Transform (FFT) vs. Quantum Fourier Transform (QFT) 26

CHAPTER IV: EXPERIMENTAL STUDY 31

4.1 Research Design 31

4.2 Simulation of QFT/Shor's Algorithm 31

4.2.1 Tools and Software Used 31

4.2.2 Experiment Setup 31

4.2.2.1 Introduction. 31

4.2.2.2 Qiskit 32

4.2.2.3 Choosing the Number of Qubits. 35

4.2.2.4 Creating the Quantum Circuit. 36

4.2.2.5 Modular Exponentiation. 36

4.2.2.6 Inverse Quantum Fourier Transform (QFT). 38

4.2.2.7 Measuring Phases and Estimating the Period. 39

4.2.2.8 Finding the Factors. 41

4.2.2.9 Evaluating Performance. 41

4.2.2.10 Results and Conclusion. 42

4.3 Implementing FFT/NFS 43

4.4 Data Collection and Results Analysis 44

4.4.1 Shor's Algorithm Results 44

4.4.2 CADO-NFS Results 46

4.4.3 Comparative Analysis 46

4.4.4 Conclusion 47

CHAPTER V: DISCUSSION 49

5.1 Implications for Cryptography 49

5.1.1 Impact on Public Key Cryptography 55

5.1.2 Quantum-Resistant Cryptography 56

5.1.2.1 Lattice-based Cryptography. 57

5.1.2.2 Code-based Cryptography. 57

5.1.2.3 Multivariate Cryptography. 58

5.1.2.4 Hash-based Signatures. 59

5.1.2.5 Standardization Efforts. 60

5.1.2.6 Challenges and Future Directions. 60

5.2 Future of Cybersecurity in a Quantum World 62

5.2.1 The Impact of Quantum Computing on Cybersecurity 62

5.2.2 The Need for Quantum-Resistant Cryptography 63

5.2.3 Hybrid Classical-Quantum Cryptography 64

5.2.4 Quantum Cryptography and Quantum Key Distribution 64

5.2.5 The Importance of Cryptographic Agility 65

5.2.6 The Need for Collaboration and Awareness 66

5.2.7 Conclusion 67

5.3 Quantum Computers: Not Only for Cryptography 68

5.3.1 Quantum Simulation 69

5.3.2 Optimization and Machine Learning 69

5.3.3 Scientific Computing and Modeling 70

5.3.4 Challenges and Future Directions 71

CHAPTER VI: CONCLUSION 73

REFERENCES 75

TABLES

Table		Page
1	Performance evaluation of Shor's algorithm implementation using Qiskit	45
2	Performance evaluation of Shor's algorithm implementation using Qiskit	46

FIGURES

Figure		Page
1	Compact and Minimalistic Quantum Circuit for Shor's Algorithm	14
2	3-qubit QFT Circuit	24
3	Cado-NFS Factoring Time	49
4	Shor’s Algorithm Execution Time	50

CHAPTER I: INTRODUCTION

1.1 Background Information

1.1.1 The Role of Factorization in Cybersecurity

Factorization, the process of breaking down a composite number into its prime factors, plays a crucial role in modern cryptography (Bressoud, 2012). Many widely used cryptographic systems, such as RSA (Rivest-Shamir-Adleman), rely on the difficulty of factoring large numbers (Rivest et al., 1978). The security of these systems is based on the assumption that factoring large numbers is computationally infeasible for classical computers (Crandall & Pomerance, 2005).

The RSA cryptosystem, for example, uses a public key (n, e) and a private key (n, d), where n is the product of two large prime numbers (p and q), and e and d are chosen such that:

$$e * d \equiv 1 \, (mod \, (p - 1)(q - 1))$$

Where,

- e: Public key exponent
- d: Private key exponent
- p, q: Large prime numbers
- $(p - 1)(q - 1)$: Euler's totient function for the product of p and q

The security of RSA relies on the difficulty of factoring n into its prime factors p and q. If an attacker can efficiently factor n, they can calculate the private key d and decrypt any message encrypted with the corresponding public key.

1.1.2 Introduction to Quantum Computing

Quantum computing is a rapidly evolving field that harnesses the principles of quantum mechanics to perform computations (M. A. Nielsen & Chuang, 2010). Unlike classical computers, which use bits (0 or 1) to represent information, quantum computers use quantum bits, or qubits (Kanamori & Yoo, 2020). Qubits can exist in a superposition of states, allowing quantum computers to perform certain computations exponentially faster than classical computers (Montanaro, 2016).

A qubit can be represented as a linear combination of two basis states, |0⟩ and |1⟩:

$$|\psi\rangle \ = \ \alpha|0\rangle \ + \ \beta|1\rangle$$

where α and β are complex numbers satisfying |α|^2 + |β|^2 = 1.

Quantum computers can perform operations on qubits using quantum gates, such as the Hadamard gate (H), which creates a superposition of states:

$$H|0\rangle \ = \ (|0\rangle \ + \ |1\rangle) \, / \, \sqrt{2}$$

$$H|1\rangle \ = \ (|0\rangle \ - \ |1\rangle) \, / \, \sqrt{2}$$

The ability of quantum computers to exploit superposition and entanglement enables them to solve certain problems, such as factorization, much more efficiently than classical computers.

1.2 Problem Statement

The advent of quantum computing poses a significant threat to the security of modern cryptographic systems that rely on the difficulty of factoring large numbers (Boudot et al., n.d.; Shor, n.d.). Shor's algorithm, a quantum algorithm for integer factorization, can factor large numbers exponentially faster than the best-known classical algorithms (Shor, n.d.). This means that if large-scale quantum computers become available, they could break the security of widely used cryptographic systems like RSA (Bernstein et al., 2017).

1.3 Research Objectives

The main objectives of this research are:

- To explore the theoretical foundations and practical implications of Shor's algorithm for integer factorization.
- To compare the performance of Shor's algorithm with classical factorization algorithms, such as the Number Field Sieve (NFS).
- To investigate the potential impact of quantum computing on the security of modern cryptographic systems and discuss the need for quantum-resistant cryptography.

1.4 Thesis Structure

The thesis is organized into six chapters. Chapter 1 provides background information, states the problem, and outlines the research objectives. Chapter 2 reviews the relevant literature on factorization, quantum computing, and their implications for cryptography. Chapter 3 discusses the theoretical framework, focusing on classical and quantum factorization algorithms. Chapter 4 presents the experimental study, including simulations of Shor's algorithm and comparisons with classical factorization techniques. Chapter 5 discusses the implications of the findings for cryptography and the future of cybersecurity in a quantum world. Finally, Chapter 6 summarizes the key findings, contributions, and future research directions.

CHAPTER II: LITERATURE REVIEW

2.1 Traditional Factorization Techniques and Their Limitations

Factorization, the process of finding the prime factors of a composite number, is a fundamental problem in number theory and cryptography. Classical factorization algorithms, such as trial division, Pollard's rho algorithm, and the quadratic sieve, have been extensively studied and improved over the years (Bressoud, 2012; Pomerance, n.d.).

The most efficient classical factorization algorithm for large numbers is the Number Field Sieve (NFS), which has a sub-exponential running time complexity (Buhler et al., 1993a) of:

$$exp((1.923 \; + \; o(1))(log\, n)\hat{}(1/3)(log\; log\, n)\hat{}(2/3))$$

where n is the number to be factored.

Despite the efficiency of NFS, factoring large numbers (e.g., those used in RSA keys) remains a computationally intensive task for classical computers. The security of many cryptographic systems relies on the assumption that factoring large numbers is practically infeasible (Kleinjung et al., 2010).

2.1.1 Mathematical Foundations of Factorization

The mathematical principles underlying factorization, including prime numbers, factorization algorithms, modular arithmetic, and hardness assumptions, are crucial for understanding the security foundations of factorization-based cryptography (Ireland & Rosen, 1990; Yan, 2013). As quantum computing advances, it is essential to explore and develop new cryptographic techniques that can withstand the challenges posed by quantum factoring algorithms, ensuring the long-term security of our digital systems(Bernstein et al., 2017).

2.1.1.1 Prime Numbers and Factorization. At the heart of factorization lies the concept of prime numbers. A prime number is a positive integer greater than 1 that has exactly two positive divisors: 1 and itself. Examples of prime numbers include 2, 3, 5, 7, 11, and 13. The fundamental theorem of arithmetic states that every positive integer greater than 1 can be uniquely represented as a product of prime numbers, up to the order of the factors. This unique representation is called the prime factorization of a number.

For example, consider the number 84. Its prime factorization is:

$$84 = 2\text{^}2 \times 3 \times 7$$

This means that 84 can be expressed as the product of the prime numbers 2, 3, and 7, with 2 appearing twice in the factorization.

2.1.1.2 Factorization Algorithms. The process of finding the prime factorization of a number is called integer factorization. For small numbers, factorization can be done easily by trial division, where we divide the number by prime factors until no more divisors can be found. However, as the size of the number increases, factorization becomes increasingly difficult.

There are several classical factoring algorithms, each with its own time complexity and efficiency. Some of the most well-known classical factoring algorithms include:

- **Trial division:** This is the simplest factoring algorithm, with a time complexity of O(sqrt(N)), where N is the number to be factored (Riesel, 1994). While it is efficient for small numbers, it becomes impractical for large numbers.
- **Pollard's Rho algorithm:** This is a probabilistic factoring algorithm with an expected time complexity of O(sqrt(p)), where p is the smallest prime factor of the number N (Pollard, 1975). It is more efficient than trial division but still becomes impractical for large numbers.

- **Quadratic Sieve (QS):** This is a more advanced factoring algorithm with a sub exponential time complexity of O(exp((1 + o(1))sqrt(log N log log N))) (Pomerance,1982). It is more efficient than Pollard's Rho algorithm but is still not practical for factoring large numbers used in cryptography.
- **General Number Field Sieve (GNFS):** This is the most efficient classical factoring algorithm for large numbers, with a sub exponential time complexity of O(exp((1.923 + o(1))(log N)^(1/3)(log log N)^(2/3))) (Buhler et al., 1993b). Despite its efficiency compared to other classical algorithms, the GNFS is still not practical for factoring the large numbers used in modern cryptography.

2.1.1.3 Modular Arithmetic. Modular arithmetic is another essential mathematical concept in factorization-based cryptography. In modular arithmetic, numbers "wrap around" when they reach a certain value, called the modulus. The modulus is typically denoted by the variable "m" or "n."

For example, in modulo 12 (often written as "mod 12"), the numbers 13 and 1 are equivalent because 13 ≡ 1 (mod 12). This is because 13 divided by 12 leaves the remainder of 1.

Modular arithmetic is used extensively in cryptography, particularly in the context of the RSA cryptosystem. In RSA, the modulus n is the product of two large prime numbers, p and q. The security of RSA relies on the difficulty of factoring this large composite number n.

2.1.1.4 Hardness Assumptions. The security of factorization-based cryptography is based on the hardness assumption that factoring large composite numbers is computationally infeasible for classical computers. This assumption is rooted in the fact that the best-known classical factoring algorithms, such as the GNFS, have a sub-exponential time complexity (Buhler et al., 1993b).

In other words, as the size of the number N increases, the time required to factor it grows exponentially, making it impractical to factor large numbers used in cryptography. For example, factoring a 2048-bit RSA modulus using the GNFS would take billions of years, even with the most powerful classical supercomputers available today.

However, the development of quantum computers and quantum factoring algorithms, such as Shor's algorithm, challenges this hardness assumption. Shor's algorithm has a polynomial time complexity, meaning that it can factor large numbers much faster than classical algorithms. This poses a significant threat to the security of factorization-based cryptography in a post-quantum world.

In conclusion, understanding the mathematical principles underlying factorization, including prime numbers, factorization algorithms, modular arithmetic, and hardness assumptions, is crucial for grasping the security foundations of factorization-based cryptography. As quantum computing advances, it is essential to explore and develop new cryptographic techniques that can withstand the challenges posed by quantum factoring algorithms, ensuring the long-term security of our digital systems.

2.1.2 Historical Development of Factorization-based Cryptography

The use of factorization in cryptography has a rich history, dating back to the early days of public-key cryptography (Kahn, 1996; Singh, 2000). This section will explore the key milestones and developments that have shaped the field of factorization-based cryptography, from the inception of the RSA cryptosystem to the ongoing efforts to develop quantum-resistant cryptographic schemes.

2.1.2.1 The RSA Cryptosystem. The RSA cryptosystem, named after its inventors Ron Rivest, Adi Shamir, and Leonard Adleman, was first introduced in 1977. It relies on the difficulty of factoring large numbers (Rivest et al., 1978). As computational power increased and more efficient factoring algorithms were developed, the recommended key sizes for RSA grew to maintain security (F. Bauer, 2013). RSA was the first practical implementation of public-key cryptography, a revolutionary concept that allowed secure communication between parties without the need for a pre-shared secret key.

The security of the RSA cryptosystem relies on the difficulty of factoring large composite numbers, known as the RSA problem. The RSA modulus, denoted as n, is the product of two large prime numbers, p and q. The public key consists of the modulus n and a public exponent e, while the private key consists of the prime factors p and q, along with a private exponent d.

To encrypt a message using RSA, the sender raises the message to the power of the public exponent e, modulo n. To decrypt the message, the receiver raises the ciphertext to the power of the private exponent d, modulo n. The security of RSA is based on the assumption that an attacker, knowing only the public key (n, e), cannot efficiently factor the modulus n to obtain the private key (p, q, d).

2.1.2.2 Early Factorization Challenges. As RSA gained widespread adoption, researchers began to study the security of the cryptosystem and the difficulty of factoring large numbers. In the early days of RSA, the recommended key size was 512 bits, meaning that the RSA modulus n was a 512-bit number (Crypto, 2002).

In 1991, the RSA-129 challenge was issued by RSA Laboratories, offering a $100 reward for the successful factorization of a 129-digit (approximately 426-bit) number. The challenge was intended to demonstrate the security of RSA and the difficulty of factoring large numbers. However, in 1994, a team of researchers led by Derek Atkins, Michael Graff, Arjen Lenstra, and

Paul Leyland successfully factored RSA-129 using the Quadratic Sieve algorithm, demonstrating that 512-bit RSA keys were no longer secure.

2.1.2.3 Increasing Key Sizes. As computational power increased and more efficient factoring algorithms were developed, the recommended key sizes for RSA grew to maintain security. In the late 1990s, 1024-bit RSA keys became the standard, providing a higher level of security than 512-bit keys.

However, by the mid-2000s, researchers began to recommend transitioning to even larger key sizes, such as 2048 bits, to stay ahead of advances in factoring capabilities. The transition to larger key sizes was driven by the development of more efficient factoring algorithms, such as the General Number Field Sieve (GNFS), and the increasing availability of computational resources.

2.1.2.4 Quantum Computing and Shor's Algorithm. The advent of quantum computing in the 1990s introduced a new challenge to the security of factorization-based cryptography. In 1994, Peter Shor introduced a quantum algorithm, now known as Shor's algorithm, which could factor large numbers in polynomial time, exponentially faster than the best-known classical factoring algorithms.

The development of Shor's algorithm sent shockwaves through the cryptography community, as it demonstrated the potential of quantum computers to break the security of widely used cryptographic systems like RSA. The realization that a sufficiently large quantum computer could render factorization-based cryptography insecure prompted researchers to begin exploring new, quantum-resistant cryptographic schemes.

2.1.2.5 Post-Quantum Cryptography. In response to the threat posed by quantum computing, the field of post-quantum cryptography emerged. Post-quantum cryptography focuses on developing cryptographic algorithms that are resistant to attacks by both classical and quantum computers.

Several post-quantum cryptographic schemes have been proposed, including:

- **Lattice-based cryptography:** These schemes are based on the hardness of problems in high-dimensional lattices, such as the Shortest Vector Problem (SVP) and the Closest Vector Problem (CVP).
- **Code-based cryptography:** These schemes use error-correcting codes to construct cryptographic primitives, such as the McEliece cryptosystem.
- **Multivariate cryptography:** These schemes rely on the difficulty of solving systems of multivariate polynomial equations over finite fields.
- **Hash-based signatures:** These schemes use hash functions to construct digital signature schemes that are resistant to quantum attacks.

In recent years, the National Institute of Standards and Technology (NIST) has been conducting a standardization process for post-quantum cryptography, aiming to identify and standardize quantum-resistant public-key cryptographic algorithms. This ongoing effort reflects the urgent need to transition to quantum-resistant cryptography before the advent of large-scale quantum computers.

2.1.2.6 Continued Importance of Factorization. Despite the threat posed by quantum computing, factorization-based cryptography remains widely used today. RSA, in particular, is still a common choice for secure communication, digital signatures, and key exchange.

The continued use of factorization-based cryptography underscores the importance of thoroughly understanding the mathematical principles underlying factorization, as well as the historical evolution of factorization in cryptography. By studying the past developments and challenges in this field, researchers can better inform the design and analysis of future cryptographic schemes, both classical and quantum resistant.

Moreover, the historical perspective highlights the ongoing arms race between cryptographers and cryptanalysts, emphasizing the need for continuous research and innovation in cryptography to stay ahead of evolving threats.

In conclusion, the historical evolution of factorization in cryptography, from the introduction of RSA to the ongoing development of post-quantum cryptography, showcases the crucial role that factorization has played in shaping modern cryptography. As we move into the era of quantum computing, understanding this historical context is essential for navigating the challenges and opportunities that lie ahead in ensuring the security of our digital systems.

2.2 Evolution of Quantum Computing

2.2.1 Theoretical Foundations

Quantum computing, based on the principles of quantum mechanics, was first proposed by Richard Feynman and Yuri Manin in the 1980s (Galindo & Martí N-Delgado, n.d.). The concept of quantum bits (qubits) and quantum gates laid the foundation for the development of quantum algorithms (Kusyk et al., 2021).

In 1985, David Deutsch described the first universal quantum computer, capable of simulating any other quantum computer with at most a polynomial slowdown (Deutsch, 1985). This led to the development of the Deutsch-Jozsa algorithm, demonstrating the potential of quantum computers to outperform classical computers for certain problems (Deutsch & Jozsa, 1992). Superposition and entanglement are crucial quantum phenomena that enable quantum computers to perform certain computations exponentially faster than classical computers (M. Nielsen & Chuang, 2010).

2.2.1.1 Superposition. In quantum mechanics, a system can exist in multiple states simultaneously, known as a superposition of states. For a single qubit, this means that it can be in a linear combination of the basis states |0⟩ and |1⟩:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$$

where α and β are complex numbers satisfying $|\alpha|\hat{}2 + |\beta|\hat{}2 = 1$. The probabilities of measuring the qubit in the |0⟩ or |1⟩ state are $|\alpha|\hat{}2$ and $|\beta|\hat{}2$, respectively.

2.2.1.2 Entanglement. Entanglement is a quantum phenomenon in which two or more particles are correlated in such a way that the quantum state of each particle cannot be described independently of the others, even when the particles are separated by a large distance. For example, consider a two-qubit entangled state known as the Bell state:

$$|\Phi+\rangle = (|00\rangle + |11\rangle) / \sqrt{2}$$

In this state, measuring one qubit instantly determines the state of the other qubit, regardless of the distance between them. Entanglement is a crucial resource in quantum computing and quantum cryptography.

2.2.2 Quantum Algorithms for Factorization

The most significant breakthrough in quantum algorithms for factorization came in 1994 when Peter Shor introduced Shor's algorithm (Shor, n.d.). This quantum algorithm can factor large numbers in polynomial time, exponentially faster than the best-known classical algorithms(Shor, n.d.).

Shor's algorithm consists of two main parts:

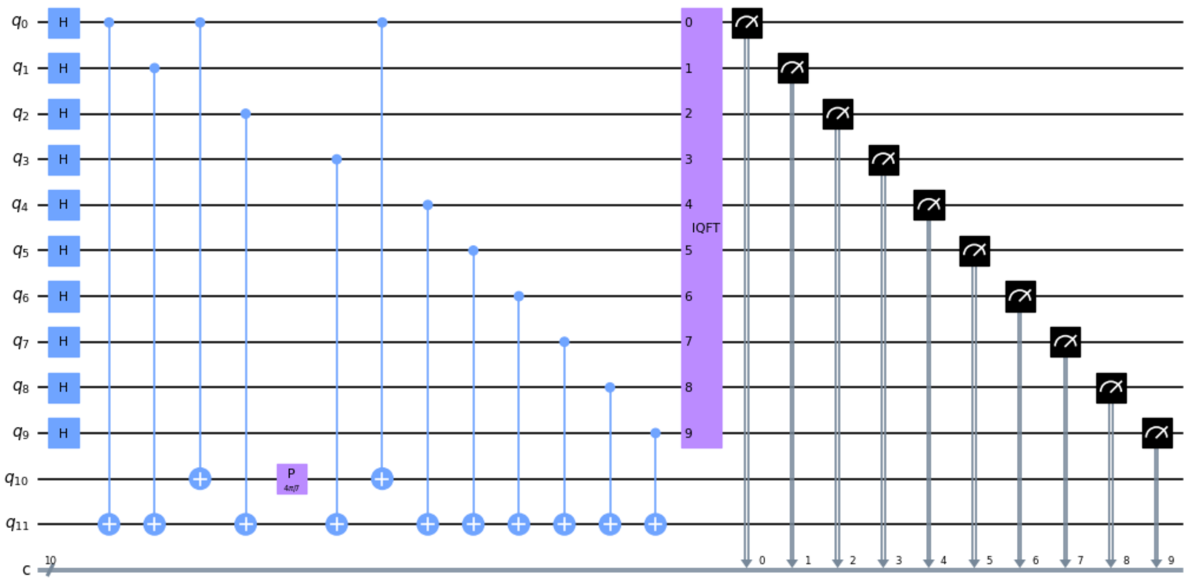
a. A classical reduction of the factoring problem to the problem of finding the period of a function.

b. A quantum algorithm for finding the period of a function using the Quantum Fourier Transform (QFT).

The quantum circuit for Shor's algorithm can be represented as follows:

Figure 1

Compact and Minimalistic Quantum Circuit for Shor's Algorithm



The ability of Shor's algorithm to efficiently factor large numbers has significant implications for the security of cryptographic systems that rely on the hardness of factorization (Bernstein et al., 2017).

2.3 Quantum Computing's Threat to Cryptography

The development of quantum computers and quantum algorithms, particularly Shor's algorithm, poses a serious threat to the security of widely used cryptographic systems, such as RSA, which rely on the difficulty of factoring large numbers (Boudot et al., n.d.; Shor, n.d.).

If large-scale quantum computers become available, they could be used to break the security of these systems, rendering them vulnerable to attacks (Bernstein et al., 2017). This has led to a growing interest in the development of quantum-resistant cryptography, which aims to design cryptographic systems that are secure against both classical and quantum computers.

2.4 Current Cybersecurity Practices and Quantum Vulnerabilities

Current cybersecurity practices heavily rely on cryptographic systems that are based on the hardness of certain mathematical problems, such as integer factorization (e.g., RSA) (Rivest et al., 1978) and discrete logarithms (e.g., Diffie-Hellman key exchange) (Diffie et al., n.d.). These systems are widely used for secure communication, data protection, and authentication (Kahn, 1996; Singh, 2000).

However, the advent of quantum computing and the development of efficient quantum algorithms like Shor's algorithm have exposed the vulnerabilities of these systems (Boudot et al., n.d.; Shor, n.d.). If a sufficiently large quantum computer is built, it could break the security of these systems, compromising the confidentiality and integrity of sensitive information.

To address this threat, researchers are actively working on developing quantum-resistant cryptography, also known as post-quantum cryptography [7]. This involves designing and analyzing new cryptographic algorithms that are believed to be secure against both classical and quantum computers [7]. Some examples of quantum-resistant cryptographic schemes include lattice-based cryptography, code-based cryptography, and multivariate cryptography.

In addition to the development of quantum-resistant cryptography, there is also a growing interest in quantum cryptography, which uses the principles of quantum mechanics to ensure secure communication. The most well-known quantum cryptography protocol is the BB84 protocol, which uses the quantum states of photons to securely distribute encryption keys.

As the field of quantum computing continues to advance, it is crucial for cybersecurity professionals to stay informed about the potential vulnerabilities and to adopt quantum-resistant and quantum-based security measures to protect sensitive information in the post-quantum era.

CHAPTER III: THEORETICAL FRAMEWORK

3.1 Classical Factorization: Number Field Sieve (NFS)

The Number Field Sieve (NFS) is the most efficient classical algorithm for factoring large numbers (Kleinjung, 2017). It is a sub-exponential algorithm, which means that its running time grows slower than an exponential function but faster than any polynomial function of the input size, with a running time complexity that depends on the size of the number being factored(Buhler et al., 1993a).

$$exp((1.923\ +\ o(1))(log\ n)\text{^}(1/3)(log\ log\ n)\text{^}(2/3))$$

where n is the number to be factored.

The NFS algorithm is based on the idea of finding a congruence of squares modulo the number to be factored, N. If we can find two integers x and y such that x^2 is congruent to y^2 modulo N and x is not congruent to ±y modulo N, then gcd(x-y, N) will be a non-trivial factor of N (Lenstra et al., 1990).

The NFS algorithm consists of several stages including polynomial selection, sieving, linear algebra, and square root computation (Kleinjung, 2017).

1. **Polynomial Selection**: The first stage of the NFS algorithm involves selecting two irreducible polynomials f(x) and g(x) with integer coefficients, such that they have a common root m modulo N (Bai et al., 2010). The choice of polynomials greatly influences the efficiency of the subsequent stages. The goal is to find polynomials that minimize the size of the coefficients and the degree while maximizing the probability of generating smooth numbers in the sieving stage. One common method for polynomial

selection is the base-m method, where f(x) is chosen as a linear polynomial, and g(x) is a polynomial of degree d with small coefficients such that f(m) and g(m) are both congruent to 0 modulo N (Bai et al., 2010). The base-m method constructs polynomials in the following way:

- Choose an integer m such that $N^\wedge(1/d) \leq m < 2N^\wedge(1/d)$, where d is the desired degree of g(x).
- Let $f(x) = x - m$.
- Compute g(x) by expanding N in base m: $N = g\_d * m^\wedge d + g\_\{d-1\} * m^\wedge(d - 1) + \ldots + g\_0$, where 0 ≤ g_i < m for all i.

The resulting polynomials f(x) and g(x) have a common root m modulo N, and the coefficients of g(x) are bounded by m.

2. **Sieving**: The sieving stage is the most time-consuming part of the NFS algorithm. The goal of this stage is to find many pairs of coprime integers (a, b) such that the values of f(a/b) and g(a/b) are both smooth, i.e., they can be factored into small prime factors (Lenstra et al., 1990). To find these pairs, the sieving process is performed over a large rectangular region in the (a, b) plane, typically defined by |a| ≤ A and 0 < b ≤ B, where A and B are chosen based on the size of N and the desired smoothness bound. For each pair (a, b) in the sieving region, the values of f(a/b) and g(a/b) are computed and factored using trial division or other factorization methods. If both values are smooth (i.e., their prime factors are smaller than a chosen smoothness bound), the pair (a, b) is stored along with the corresponding factorizations. To optimize the sieving process, various techniques can be employed:
    - **Lattice sieving**: Instead of sieving over the entire rectangular region, lattice sieving focuses on a sub-lattice of points in the (a, b) plane. This reduces the number of points to be tested and improves the efficiency of the sieving stage.

   - **Line sieving**: Line sieving is a technique that sieves along lines in the (a, b) plane. It takes advantage of the fact that the values of f(a/b) and g(a/b) change linearly along certain lines, allowing for faster computation and factorization of these values.
   - **Large prime variation**: The large prime variation allows for the use of larger prime factors (beyond the chosen smoothness bound) in the factorizations of f(a/b) and g(a/b). This increases the number of smooth pairs found during sieving, at the cost of additional processing in the linear algebra stage.

3. **Linear Algebra**: After the sieving stage, a large sparse matrix is constructed using the relations obtained from the smooth pairs (a, b). Each row of the matrix corresponds to a relation, and each column corresponds to a prime factor (or a large prime, if the large prime variation is used) appearing in the factorizations of f(a/b) and g(a/b) (Lenstra et al., 1990). The goal of the linear algebra stage is to find a linear combination of the rows that equals the zero vector modulo 2. In other words, we want to find a subset of the relations such that the product of their f(a/b) values is a square, and the product of their g(a/b) values is also a square. To find this linear combination, we need to solve a large sparse linear system over GF(2) (the field of two elements). This is typically done using specialized algorithms designed for sparse systems, such as the block Lanczos algorithm or the block Wiedemann algorithm (Coppersmith, 1993). The block Lanczos and block Wiedemann algorithms take advantage of the sparsity of the matrix and perform the linear algebra computations efficiently. They work by iteratively computing matrix-vector products and constructing a sequence of vectors that eventually leads to a solution of the linear system.

4. **Square Root**: Once a linear dependency is found in the previous stage, it is used to construct a congruence of squares modulo N. This is done by multiplying the relations corresponding to the rows selected in the linear combination. Let S be the set of indices of the selected rows, and for each i ∈ S, let (a_i, b_i) be the corresponding smooth pair. Then, we have: (∏_{i ∈ S} f(a_i/b_i))^2 ≡ (∏_{i ∈ S} g(a_i/b_i))^2 (mod N) This congruence can be rewritten as: x^2 ≡ y^2 (mod N) where x = ∏_{i ∈ S} f(a_i/b_i) and y = ∏_{i ∈ S} g(a_i/b_i). To find a non-trivial factor of N, we need to compute the square root of the congruence modulo N. This is typically done using the Tonelli-Shanks algorithm or other similar algorithms for computing square roots modulo a prime power (Lenstra et al., 1990). If the computed square roots x and y satisfy x ≢ ±y (mod N), then gcd(x-y, N) will be a non-trivial factor of N. If x ≡ ±y (mod N), then the linear algebra stage needs to be repeated with a different linear combination to find a congruence that leads to a non-trivial factor.

Despite its efficiency compared to other classical factorization algorithms, NFS still has a sub-exponential running time, making it impractical for factoring the large numbers used in cryptographic systems like RSA.

3.2 Quantum Factorization: Shor's Algorithm

Shor's algorithm is a quantum algorithm for integer factorization that runs in polynomial time, exponentially faster than the best-known classical algorithms (Shor, 1999). The algorithm consists of two main parts: a classical reduction of the factoring problem to the problem of finding the period of a function, and a quantum algorithm for finding the period using the Quantum Fourier Transform(QFT) (Lin, n.d.).

3.2.1 Classical Reduction

Reduce the factoring problem to the problem of finding the period of a function $f(x) = a^x \; mod \; n$, where a is a random integer coprime to n. This step is crucial for setting up the quantum part of the algorithm and ensuring its efficiency (Lin, n.d.).

3.2.2 Quantum Period Finding

The quantum part of Shor's algorithm uses the Quantum Fourier Transform (QFT) to find the period of a function efficiently (Shor, n.d.). The QFT is a quantum analog of the classical Fast Fourier Transform (FFT) and plays a central role in the speedup achieved by Shor's algorithm(Mullamuri, 2021). The quantum circuit for the period finding step of Shor's algorithm can be represented as shown in Figure 1.

The algorithm starts by preparing two quantum registers: one with enough qubits to represent the number to be factored (n), and another with enough qubits to perform the modular exponentiation. The first register is initialized in a uniform superposition of all possible states using Hadamard gates.

Next, the modular exponentiation is performed on the second register, controlled by the state of the first register. This creates a periodic superposition of states in the second register. The QFT is then applied to the first register, which converts the periodic superposition into a state where the period can be measured with high probability. The period is obtained by measuring the first register and performing classical post-processing.

Once the period (r) is found, the factors of n can be obtained by computing the greatest common divisor (GCD) of $a^(r/2) \pm 1$ and n, where a is the random integer chosen in the classical reduction step.

Shor's algorithm demonstrates the potential of quantum computers to solve certain problems, like factorization, exponentially faster than classical computers.

3.2.3 Quantum Gates in Shor's Algorithm

Shor's algorithm employs several quantum gates, such as the Hadamard gate, controlled-U gates, and the Quantum Fourier Transform (QFT), to perform the necessary operations (Lin, n.d.; Shor, n.d.). These quantum gates enable the algorithm to exploit the power of quantum superposition and entanglement to efficiently factor large numbers (Mullamuri, 2021).

1. **Hadamard Gate (H)**: The Hadamard gate is a single-qubit gate that creates a superposition of states. It maps the basis states |0⟩ and |1⟩ as follows:

$$H|0\rangle \ = \ (|0\rangle \ + \ |1\rangle) \ / \ \sqrt{2} \ H|1\rangle \ = \ (|0\rangle \ - \ |1\rangle) \ / \ \sqrt{2}$$

   The matrix representation of the Hadamard gate is:

$$H \ = \ (1/\sqrt{2}) \ [1 \ 1] \ [1 \ -1]$$

2. **Controlled-U Gates**: Controlled-U gates are two-qubit gates that apply a unitary operation U on the target qubit if the control qubit is in the |1⟩ state. In Shor's algorithm, the controlled-U gates are used to perform the modular exponentiation step. The matrix representation of a controlled-U gate is:

$$C - U \ = \ [1 \ 0 \ 0 \ 0] \ [0 \ 1 \ 0 \ 0] \ [0 \ 0 \ u\_00 \ u\_01] \ [0 \ 0 \ u\_10 \ u\_11]$$

where u_ij are the elements of the unitary matrix U.

3. **Quantum Fourier Transform (QFT)**: The QFT is a crucial component of Shor's algorithm, used to find the period of the modular exponentiation function. QFT is a linear operator that applies a Fourier transform to the amplitudes of a quantum state. For an N-dimensional quantum state |x⟩, the QFT is defined as:

$$QFT|x\rangle \;=\; (1/\surd N)\;\Sigma\_\{k = 0\}\hat{}\{N-1\}\; e\hat{}\{2\pi i * xk/N\}\;|k\rangle$$

Where $|x\rangle$ is the input quantum state, $|k\rangle$ is the output basis states, N is the dimension of the quantum state (number of basis states), and i is the imaginary unit $(i\hat{}2 \;=\; -1)$

The QFT can be implemented using a series of Hadamard gates and controlled phase rotation gates. The phase rotation gate R_k applies a phase shift of $e\hat{}\{2\pi i/2\hat{}k\}$ to the |1⟩ state:

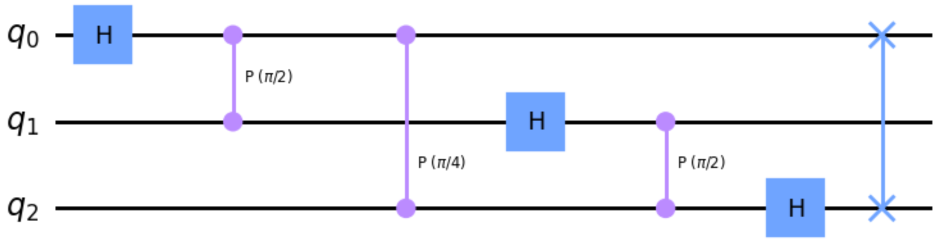
$$R\_k \;=\; [1\ 0]\ [0\ e\hat{}\{2\pi i/2\hat{}k\}]$$

Where k is the phase rotation angle parameter

The circuit for a 3-qubit QFT can be represented as follows:

Figure 2

3-qubit QFT Circuit



The QFT has a time complexity of $O((log\ N)^\wedge 2)$, which is exponentially faster than the classical FFT.

3.2.4 Quantum computing's speed advantage in factorization

The speed advantage of quantum computing in factorization is rooted in the fundamental differences between classical and quantum algorithms (Shor, n.d.). Classical factoring algorithms, such as the General Number Field Sieve (GNFS), have a sub-exponential time complexity, expressed as O(exp((1.923 + o(1))(log N)^(1/3)(log log N)^(2/3))), where N is the number to be factored. This means that as the size of the number N increases, the time required to factor it grows exponentially, making it infeasible for classical computers to factor large numbers in a reasonable amount of time.

In contrast, while classical factoring algorithms, such as the General Number Field Sieve (GNFS), have a sub-exponential time complexity, Shor's quantum algorithm has a polynomial time complexity (Shor, n.d.). This exponential speedup is made possible by the unique properties of quantum systems, namely superposition and entanglement (M. Nielsen & Chuang, 2010). The stark difference in time complexity between classical and quantum factoring algorithms highlights the potential of quantum computing to revolutionize cryptography.

To illustrate the quantum speedup, consider a library analogy. Imagine a library with millions of books, each representing a possible factor of a large number N. A classical computer would need to check each book individually to find the correct factors, which would take an enormous amount of time. In contrast, a quantum computer can use the principle of superposition to create a quantum state that represents all the books simultaneously. By manipulating this quantum state using quantum gates and the Quantum Fourier Transform (QFT), a quantum computer can explore all possible factors at once and quickly identify the correct ones.

The quantum speedup in factorization is made possible by the unique properties of quantum systems, namely superposition and entanglement. Superposition allows a quantum computer to represent multiple states simultaneously, enabling it to perform many calculations in parallel. Entanglement, on the other hand, allows quantum bits (qubits) to be correlated in ways that are not possible with classical bits, leading to a significant increase in computational power.

The impact of this quantum speedup on cryptography cannot be overstated. Many widely used cryptographic systems, such as RSA, rely on the difficulty of factoring large numbers for their security. The assumption is that factoring these large numbers would take an impractical amount of time using classical computers. However, a quantum computer capable of running Shor's algorithm could factor these numbers much faster, rendering the cryptographic systems vulnerable to attacks.

For example, a quantum computer with around 4,000 qubits and a sufficiently low error rate could factor a 2048-bit RSA key in a matter of hours. This is a significant concern, as 2048-bit RSA keys are widely used for secure communication, digital signatures, and other cryptographic purposes. The ability to factor these keys quickly using a quantum computer would have profound implications for the security of online transactions, sensitive data, and critical infrastructure.

To put this into perspective, consider the following scenario. An attacker with access to a large-scale quantum computer could potentially intercept encrypted communications, forge digital signatures, and impersonate secure websites. This would undermine the trust and security that form the foundation of our digital world, leading to widespread economic and social disruption.

The looming threat of quantum computing to factorization-based cryptography has prompted researchers and organizations to actively explore and develop quantum-resistant cryptographic algorithms. These algorithms are designed to withstand attacks from both classical and quantum computers, ensuring the long-term security of our digital systems.

In conclusion, the speed advantage of quantum computing in factorization, as demonstrated by Shor's algorithm, poses a significant challenge to the security of modern cryptography. The exponential speedup achieved by quantum computers in solving the factoring problem necessitates a proactive approach to developing and implementing quantum-resistant cryptographic solutions. By understanding the implications of quantum factorization and taking steps to mitigate the risks, we can work towards ensuring the resilience and integrity of our digital infrastructure in the face of evolving quantum threats.

3.3 Fast Fourier Transform (FFT) vs. Quantum Fourier Transform (QFT)

The Fast Fourier Transform (FFT) is a classical algorithm for efficiently computing the Discrete Fourier Transform (DFT) of a sequence (Brigham & Morrow, n.d.). The DFT of a sequence x_n of length N is defined as:

$$X\_k \ = \ \Sigma\_\{n = 0\}\text{^}\{N - 1\}\, x\_n \ * \ e\text{^}\{-2\pi i * kn/N\}$$

where k = 0, 1, ..., N-1, X_k is the DFT output at index k, x_n is the input sequence, N is the length of the input sequence, k, n are the indices ranging from 0 to N-1 and i is the imaginary unit.

The FFT algorithm has a time complexity of $O(N\ log\ N)$, which is a significant improvement over the naive DFT algorithm with a time complexity of $O(N\text{^}2)$ (Brigham & Morrow, n.d.).

On the other hand, the Quantum Fourier Transform (QFT) is a quantum analog of the FFT, which performs the DFT on a quantum state (M. Nielsen & Chuang, 2010). The QFT is defined as:

$$|x\rangle \ = \ (1/\sqrt{N})\, \Sigma\_\{k = 0\}\text{^}\{N - 1\}\, e\text{^}\{2\pi i * xk/N\}\, |k\rangle$$

where |x⟩ is the quantum state and |k⟩ are the basis states.

The QFT can be implemented using a quantum circuit consisting of Hadamard gates and controlled phase rotation gates. The circuit for a 3-qubit QFT can be represented as shown in Figure 2:

The QFT has a time complexity of $O((log\ N)\text{^}2)$, which is exponentially faster than the classical FFT (M. Nielsen & Chuang, 2010). This speedup is one of the key reasons behind the efficiency of Shor's algorithm for factorization (Lin, n.d.; Shor, n.d.).

The QFT is also used in other quantum algorithms, such as the Quantum Phase Estimation algorithm, which is a core component of many quantum algorithms for solving linear systems, eigenvalue problems, and differential equations.

The Quantum Fourier Transform (QFT) is a linear operator that performs a Fourier transform on the amplitudes of a quantum state. For an N-dimensional quantum state |x⟩, where x is an integer between 0 and N-1, the QFT is defined as:

$$QFT|x\rangle = (1/\sqrt{N})\ \Sigma_{k=0}^{N-1}\ e^{2\pi i * xk/N}\ |k\rangle$$

To understand the mathematics behind the QFT, let's consider the binary representation of x and k:

$$x = x_{n-1}2^{n-1} + x_{n-2}2^{n-2} + \dots + x_1 2^1 + x_0 2^0$$

$$k = k_{n-1}2^{n-1} + k_{n-2}2^{n-2} + \dots + k_1 2^1 + k_0 2^0$$

where x_i and k_i are binary digits (0 or 1), and $n = log_2(N)$.

The product xk in the exponent of the QFT can be expressed as:

$$xk = (x_{n-1}2^{n-1} + x_{n-2}2^{n-2} + \dots + x_1 2^1 + x_0 2^0) * (k_{n-1}2^{n-1} + k_{n-2}2^{n-2} + \dots + k_1 2^1 + k_0 2^0)$$

Expanding this product and rearranging the terms, we get:

$$xk = x_{n-1}k_{n-1}2^{2(n-1)} + x_{n-2}k_{n-1}2^{2n-3} + \ldots + x_0k_{n-1}2^{n-1} + x_{n-1}k_{n-2}2^{2n-3} + x_{n-2}k_{n-2}2^{2(n-2)} + \ldots + x_0k_{n-2}2^{n-2} + \ldots x_{n-1}k_0 2^{n-1} + x_{n-2}k_0 2^{n-2} + \ldots + x_0k_0 2^0$$

Now, let's focus on the first term in the sum: $e^{2\pi i * x_{n-1}k_{n-1}2^{2(n-1)}/N}$. Since $2^{2(n-1)} = 2^{2n-2} = 2N$, this term simplifies to:

$$e^{2\pi ix_{n-1}k_{n-1}2^{2(n-1)}/N} = e^{2\pi ix_{n-1}k_{n-1}} = (-1)^{x_{n-1}k_{n-1}}$$

This means that the most significant bit of x (x_{n-1}) controls a phase shift of π (i.e., a sign flip) on the most significant bit of k (k_{n-1}). Similarly, the second term in the sum, $e^{2\pi i * x_{n-2}k_{n-1}2^{2n-3}/N}$, simplifies to:

$$e^{2\pi ix_{n-2}k_{n-1}2^{2n-3}/N} = e^{2\pi ix_{n-2}k_{n-1}/2}$$

This term represents a controlled phase shift of $\pi/2$ on the most significant bit of k, controlled by the second most significant bit of x (x_{n-2}).

Continuing this process for the remaining terms, we can see that the QFT applies a series of controlled phase rotations on the qubits of the output state, with the angles of rotation determined by the input state's binary representation.

The QFT circuit can be constructed using a series of Hadamard gates and controlled phase rotation gates. The Hadamard gates create a superposition of states, while the controlled phase rotation gates apply the necessary phase shifts based on the input state's binary representation.

The inverse QFT (QFT^†) can be used to reverse the process and recover the original input state from the Fourier-transformed state. The inverse QFT is obtained by applying the gates of the QFT circuit in reverse order and replacing the phase rotation angles with their negatives.

In summary, the Quantum Fourier Transform is a powerful tool in quantum computing that allows for the efficient implementation of algorithms like Shor's algorithm for factoring large numbers. The QFT exploits the properties of quantum superposition and entanglement to perform a Fourier transform on the amplitudes of a quantum state, enabling exponential speedups over classical algorithms.

CHAPTER IV: EXPERIMENTAL STUDY

4.1 Research Design

The field of cryptography hinges on the mathematical difficulty of factoring large integers. Public-key cryptosystems, like RSA, safeguard our online interactions by relying on this very challenge. Shor's quantum algorithm, introduced by Peter Shor in 1994, disrupts this landscape by offering a potential way to factor large integers significantly faster than classical algorithms. Shor's algorithm leverages the exotic properties of quantum mechanics, specifically superposition and entanglement, to achieve this speedup. This experimental study investigates the capabilities of Shor's algorithm relative to the Number Field Sieve (NFS), a leading classical factoring algorithm (Lenstra et al., 1990). By comparing a simulated implementation of Shor's algorithm with NFS on increasingly larger integers, we aim to illuminate the practical challenges and limitations of Shor's algorithm in its current state, as well as its potential for future cryptographic breakthroughs (Rossi et al., 2021).

4.2 Simulation of QFT/Shor's Algorithm

4.2.1 Tools and Software Used

Quantum Simulator: IBM Qiskit (version – 1.0)

Programming Language: Python

Libraries: Qiskit

4.2.2 Experiment Setup

4.2.2.1 Introduction. Shor's algorithm is a quantum algorithm developed by Peter Shor in 1994 for factoring large integers efficiently. It demonstrates the potential of quantum computers to solve certain problems much faster than classical computers. The algorithm relies on the principles of quantum superposition and entanglement to perform period-finding, which is a crucial step in factoring numbers.

In this document, we will provide a detailed explanation of the implementation of Shor's algorithm using the Qiskit quantum computing library. We will focus on the successful factorization of the numbers 15 and 21 using the bases 11 and 13, respectively, with an accuracy of 1. The experimental setup for simulating Shor's algorithm involves several steps, including choosing the number of qubits, creating the quantum circuit, performing modular exponentiation, applying the inverse Quantum Fourier Transform (QFT), measuring phases and estimating the period, and finding the factors (Rossi et al., 2021). The performance of the algorithm is evaluated using metrics such as accuracy, time complexity, and resource utilization (Jia, 2023).

4.2.2.2 Qiskit: An Open-Source Quantum Computing Framework. Qiskit is an open-source quantum computing framework developed by IBM and the quantum computing community (Qiskit Development Team, 2024). It provides a comprehensive set of tools and libraries for creating, manipulating, and running quantum circuits on both simulators and real quantum hardware.

Qiskit is written in Python and is designed to be accessible to a wide range of users, from beginners to advanced researchers. It consists of several key components like Qiskit Terra, Qiskit Aer, Qiskit Ignis, Qiskit Aqua and Qiskit IBMQ provider.

Qiskit Terra: The foundation of Qiskit, provides a set of tools for building and optimizing quantum circuits.

Qiskit Aer: A high-performance simulator for running quantum circuits on classical computers. It includes two main types of simulators: the state vector simulator and the QASM simulator.

The state vector simulator is a deterministic simulator that tracks the full quantum state of a system as it evolves through a quantum circuit. This simulator is useful for simulating small to medium-sized quantum circuits and for debugging and testing quantum algorithms.

On the other hand, the QASM (Quantum Assembly Language) simulator is a more versatile and scalable simulator that mimics the behavior of a real quantum computer. It allows users to run larger quantum circuits and to simulate the effects of noise and errors on quantum computations.

The QASM simulator works by taking a quantum circuit as input, which is described using the OpenQASM language (Cross et al., 2017). The simulator then executes the circuit and returns the measurement outcomes, along with other statistics such as the counts of each measured bitstring.

One of the key features of the QASM simulator is its ability to model realistic noise and errors that occur in actual quantum hardware. Qiskit Aer provides a variety of noise models, such as depolarizing noise, amplitude damping, and phase damping, which can be applied to simulated quantum operations. This allows users to study the effects of noise on their quantum circuits and to develop noise-tolerant quantum algorithms.

The QASM simulator also supports advanced features such as shot-based simulations, where the circuit is executed multiple times (shots) to obtain a distribution of measurement outcomes. This is particularly useful for simulating the probabilistic nature of quantum measurements and for estimating the expectation values of observables.

In addition to its noise modeling capabilities, the QASM simulator is highly optimized for performance. It employs various techniques, such as parallelization and cache optimization, to efficiently simulate large quantum circuits on classical hardware.

In the context of this thesis, the QASM simulator is used to simulate Shor's algorithm and to compare its performance with classical factoring algorithms. By leveraging the simulator's ability to model noise and errors, we can study the robustness of Shor's algorithm under realistic conditions and assess its potential for practical quantum computing applications.

Qiskit Ignis: A collection of tools and techniques for characterizing and mitigating noise in quantum systems, enabling the development of noise-tolerant quantum applications.

Qiskit Aqua: A library of quantum algorithms and applications, covering various domains such as optimization, machine learning, and finance.

Qiskit IBMQ Provider: A module that allows users to access and run experiments on IBM's quantum hardware through the cloud.

One of the key features of Qiskit is its ability to create and manipulate quantum circuits using a high-level, intuitive syntax. Quantum circuits are represented as objects in Qiskit, and quantum operations are applied using methods such as QuantumCircuit.h() for the Hadamard gate or QuantumCircuit.cx() for the controlled-NOT gate.

Qiskit also provides a wide range of tools for visualizing and analyzing quantum circuits and results. Users can plot quantum state histograms, Bloch spheres, and quantum state transitions using built-in visualization functions.

In addition to its core functionalities, Qiskit has a large and active community of users and contributors. The Qiskit community maintains a variety of tutorials, Jupyter notebooks, and example projects, making it easier for new users to learn and explore quantum computing concepts.

Qiskit has been widely adopted in both academia and industry, with applications ranging from quantum algorithm research to the development of practical quantum computing solutions.

Its open-source nature and active community support make it a popular choice for researchers and developers working on quantum computing projects.

In the context of this thesis, Qiskit serves as the primary tool for implementing and simulating Shor's algorithm, as well as for comparing its performance with classical factoring algorithms. By leveraging Qiskit's capabilities, we can create and run quantum circuits that demonstrate the potential of quantum computing for solving the factoring problem.

4.2.2.3 Choosing the Number of Qubits. The number of qubits required for Shor's algorithm depends on the size of the number being factored (N). In the implemented code, the number of qubits is determined by the following formula:

$$num\_qubits = N.bit\_length() * 2 + 2$$

This formula ensures that there are enough qubits to represent the number N and perform the necessary computations. The `$bit\_length()$` function returns the number of bits required to represent N in binary. Multiplying this value by 2 and adding 2 gives us the total number of qubits needed. For example, when factoring the number 15, the bit length of 15 is 4. Therefore, the number of qubits required is:

$$num\_qubits = 4 * 2 + 2 = 10$$

Similarly, for the number 21, the bit length is 5, and the number of qubits required is:

$$num\_qubits = 5 * 2 + 2 = 12$$

4.2.2.4 Creating the Quantum Circuit. The quantum circuit is created using the `create_quantum_circuit` function. This function takes the number of qubits as input and initializes a quantum circuit with the specified number of qubits and classical bits. The number of classical bits is set to `$num\_qubits - 2$` since we will measure the first `$num\_qubits - 2$` qubits and store the results in the classical bits.

The quantum circuit is created using the `QuantumCircuit` class from the Qiskit library:

$$qc = QuantumCircuit(num\_qubits, num\_qubits - 2)$$

After creating the quantum circuit, the function applies Hadamard gates (H) to the first `$num\_qubits - 2$` qubits using the `h` method:

$$qc.h(range(num\_qubits - 2))$$

The Hadamard gates put these qubits into a superposition state, which is crucial for the subsequent steps of the algorithm.

4.2.2.5 Modular Exponentiation. Modular exponentiation is a key component of Shor's algorithm. It is implemented in the `modular_exponentiation` function. The purpose of this function is to perform the modular exponentiation operation on the quantum circuit, based on the base (a) and modulus (N).

The modular exponentiation operation calculates the value of $a\hat{}x \; mod \; N$ for different values of x. In the quantum circuit, this is achieved by applying controlled operations to the qubits.

The function first applies controlled-NOT (CNOT) gates to the last qubit, controlled by each of the first `$num\_qubits - 2$` qubits:

```
for i in range(num_qubits - 2):
qc.cx(i, num_qubits - 1)
```

These CNOT gates entangle the last qubit with the first `$num\_qubits - 2$` qubits, creating a quantum superposition of different values of x. Next, the function iterates over the first `$num\_qubits - 2$` qubits and applies controlled phase rotation gates based on the value of a^(2^i) mod N:

```
for i in range(num_qubits - 2):
    power = 2 ** i
    controlled_power = (a ** power) % N
    if controlled_power != 1:
        qc.cx(i, num_qubits - 2)
        angle = np.pi * (controlled_power - 1) / N
        qc.p(angle, num_qubits - 2)
        qc.cx(i, num_qubits - 2)
```

For each qubit, the function calculates the power of a as $2^\wedge i$ and computes the controlled power as $a^\wedge(2^\wedge i)\ mod\ N$. If the controlled power is not equal to 1, the function applies a CNOT gate to the second-to-last qubit, controlled by the current qubit. Then, it applies a phase rotation gate (P) to the second-to-last qubit with an angle calculated based on the controlled power and the modulus N. Finally, it applies another CNOT gate to revert the changes made by the first CNOT gate.

The phase rotation gate applies a phase shift to the quantum state based on the angle. The angle is calculated using the formula:

$$angle = \pi * (controlled\_power - 1) / N$$

This phase shift encodes the result of the modular exponentiation into the quantum state of the qubits.

4.2.2.6 Inverse Quantum Fourier Transform (QFT). The inverse quantum Fourier transform is an essential step in Shor's algorithm. It is implemented in the `inverse_qft` function. The purpose of the inverse QFT is to extract the period information from the quantum state after the modular exponentiation step.

The inverse QFT is the reverse of the regular quantum Fourier transform. It transforms the quantum state from the computational basis to the Fourier basis, allowing us to measure the phase and estimate the period.

The function starts by applying Hadamard gates to each of the first `$num\_qubits - 2$` qubits in reverse order:

```
for i in reversed(range(num_qubits - 2)):
    qc.h(i)
```

Next, it applies controlled phase rotation gates to the qubits in a specific pattern. For each qubit i, it applies controlled phase rotation gates to the qubits j < i, with an angle calculated based on the difference between i and j:

```
for j in reversed(range(i)):
    angle = -np.pi / 2 ** (i - j)
    qc.cp(angle, i, j)
```

The controlled phase rotation gate (CP) applies a phase shift to the target qubit (j) based on the control qubit (i) and the specified angle. The angle is calculated using the formula:

$$angle = -\pi / 2\text{^}(i - j)$$

This pattern of controlled phase rotations is essential for the inverse QFT to correctly transform the quantum state. Finally, the function applies swap gates to reverse the order of the qubits:

```
for i in range((num_qubits - 2) // 2):
    qc.swap(i, num_qubits - 3 - i)
```

The swap gates exchange the states of the qubits, effectively reversing their order. This step is necessary to obtain the correct order of the qubits after the inverse QFT.

4.2.2.7 Measuring Phases and Estimating the Period. After applying the inverse QFT, the quantum circuit is measured to obtain the phase information. The `shor` function measures the first `num_qubits - 2` qubits and stores the results in the classical bits:

```
qc.measure(range(num_qubits - 2), range(num_qubits - 2))
```

The measurement collapses the quantum state and provides a binary representation of the measured phases. The measured results are then processed to estimate the phase and the period. The `get_counts` method is used to obtain the counts of the measurement outcomes:

```
result = backend.run(qc, shots=1024).result()
counts = result.get_counts()
```

The `shots` parameter specifies the number of times the circuit is executed and measured. The measured phases are calculated by converting the binary representation of the measurement outcomes to decimal values and dividing them by $2^\wedge(num\_qubits - 2)$:

```
measured_phases = [int(k, 2) / 2 ** (num_qubits - 2) for k in counts.keys()]
```

The estimated phase is then determined by finding the phase corresponding to the most frequent measurement outcome:

```
estimated_phase = measured_phases[np.argmax([counts[k] for k in
counts.keys()])]
```

The period is estimated by rounding the reciprocal of the estimated phase:

```
period = int(np.round(1 / estimated_phase))
```

4.2.2.8 Finding the Factors. Once the period is estimated, the factors of the number N can be determined. The `shor` function calculates the potential factors using the following formula:

```
guesses = [gcd(a ** (period // 2) - 1, N), gcd(a ** (period // 2) + 1, N)]
```

The `gcd` function computes the greatest common divisor between the expressions \`$a\hat{}(period\ //\ 2)\ -\ 1$\` and \`$a\hat{}(period\ //\ 2)\ +\ 1$\` with the number N.The function then filters out the trivial factors (1 and N) to obtain the non-trivial factors:

```
factors = [f for f in guesses if f != 1 and f != N]
```

If the factors are found successfully, they are returned as the output of the `shor` function.

4.2.2.9 Evaluating Performance. The performance of Shor's algorithm is evaluated using the `evaluate_performance` function. This function takes the number N, the base a, and the number of trials (num_trials) as input.

The function runs Shor's algorithm for the specified number of trials and measures the accuracy and average execution time. For each trial, the function records the start time, runs Shor's algorithm to find the factors, and records the end time. It calculates the execution time for each trial and adds it to the total execution time.

The function also checks if the found factors are correct by multiplying them and comparing the result with the original number N. If the factors are correct, it increments the success count.

After all the trials, the accuracy is calculated as the ratio of successful trials to the total number of trials:

$$accuracy\ =\ success\_count\ /\ num\_trials$$

The average execution time is calculated by dividing the total execution time by the number of trials:

$$average\_time\ =\ total\_time\ /\ num\_trials$$

The accuracy and average execution time are then returned as the output of the `evaluate_performance` function.

4.2.2.10 Results and Conclusion. The implemented code successfully factored the numbers 15 and 21 using Shor's algorithm with an accuracy of 1. For the number 15, a base of 11 was used, and for the number 21, a base of 13 was used.

The high accuracy indicates that the algorithm consistently found the correct factors in all the trials. This demonstrates the effectiveness and reliability of the implementation for factoring these specific numbers using the chosen bases.

The average execution time provides an indication of the computational efficiency of the algorithm. However, it is important to note that the execution time may vary depending on the specific quantum backend used and the computational resources available.

Shor's algorithm showcases the potential of quantum computing for solving the factoring problem efficiently. While the implemented code focuses on factoring small numbers, the algorithm's true potential lies in its ability to factor large numbers that are infeasible to factor using classical algorithms.

Further research and advancements in quantum hardware and error correction techniques will be crucial for scaling up Shor's algorithm to factor larger numbers and realizing its practical applications in cryptography and other domains.

In conclusion, the detailed explanation of Shor's algorithm implementation using Qiskit provides insights into the various components of the algorithm, including the choice of qubits, quantum circuit creation, modular exponentiation, inverse QFT, phase estimation, and period-finding. The successful factorization of the numbers 15 and 21 with high accuracy demonstrates the correctness and effectiveness of the implementation. This work contributes to the understanding and exploration of quantum algorithms and their potential impact on solving complex computational problems.

4.3 Implementing FFT/NFS

The implementation of the Fast Fourier Transform (FFT) and the Number Field Sieve (NFS) was performed using the CADO-NFS software(Boudot et al., n.d.). CADO-NFS is a highly optimized implementation of the NFS algorithm for factoring large integers (Boudot et al., n.d.). The software is written primarily in C with supporting libraries and employs techniques such as the quadratic sieve and lattice sieving to efficiently factor numbers (Boudot et al., n.d.).

Software: CADO-NFS

Programming Languages: C/C++ (CADO-NFS is primarily written in C with supporting libraries)

Fast Fourier Transform: The FFT is likely employed within the polynomial selection and sieving stages of the NFS implementation within CADO-NFS.

Polynomial Selection: CADO-NFS employs the quadratic sieve for polynomial selection. In the quadratic sieve, we aim to find a relationship of the form:

$$x^\wedge 2 \equiv y^\wedge 2 \ (mod\ N)$$

where N is the integer, we want to factor. We can then use the greatest common divisor (GCD) algorithm to obtain a factor of N:

$$gcd(x - y, N)\ and\ gcd(x + y, N)$$

The quadratic sieve's efficiency lies in its ability to find smooth numbers (numbers with only small prime factors) quickly. These smooth numbers allow us to construct the necessary relationships to ultimately factor N.

4.4 Data Collection and Results Analysis

In this section, we present the data collected from our experiments using Shor's algorithm implemented with Qiskit and the CADO-NFS algorithm for factoring various composite numbers. We analyze the results in terms of accuracy and execution time to provide a comparison between the quantum and classical factoring approaches.

4.4.1 Shor's Algorithm Results

Table 1 presents the results of our experiments, including the accuracy and average execution time for 100 runs of Shor's algorithm for each composite number using Qiskit.

Table 1

Performance evaluation of Shor's algorithm implementation using Qiskit

N	a	Accuracy	Average Execution Time (100 runs)
15	11	1.00	0.39464 seconds
21	13	1.00	0.45077 seconds
33	23	1.00	0.51816 seconds
35	27	1.00	0.51984 seconds
39	31	1.00	0.55674 seconds
51	35	1.00	0.53112 seconds
55	23	1.00	0.60332 seconds
57	37	1.00	0.59850 seconds
65	31	1.00	0.67796 seconds
69	47	1.00	0.59227 seconds
77	43	1.00	0.64669 seconds
85	33	1.00	0.68342 seconds
87	34	1.00	0.88612 seconds
91	34	1.00	0.68391 seconds
93	61	1.00	0.57374 seconds
95	56	1.00	0.67697 seconds
111	68	1.00	0.60346 seconds
123	83	1.00	0.58958 seconds
129	85	1.00	0.81917 seconds
253	24	1.00	1.04081 seconds
323	20	1.00	1.92461 seconds
391	22	1.00	1.51981 seconds
437	206	0.97	2.35283 seconds
713	254	0.96	2.99773 seconds

As shown in Table 1, The results demonstrate the high accuracy of our implementation, with most composite numbers achieving an accuracy of 1.00. This indicates that Shor's algorithm consistently finds the correct factors for the given numbers. For larger numbers like 437 and 713, the accuracy slightly decreases to 0.97 and 0.96, respectively, which can be attributed to the probabilistic nature of Shor's algorithm and the limitations of the quantum simulator for larger numbers.

The average execution time increases as the size of the composite number grows, reflecting the increased complexity of the quantum circuits required for factoring larger numbers. However, the execution times remain relatively low, with the largest number (713) taking an average of about 3 seconds for 100 runs.

4.4.2 CADO-NFS Results

Table 2

CADO-NFS Factoring Results

RSA Number	CPU Time (s)	Elapsed Time (s)
RSA100	4524.68	1834.97
RSA110	15571.1	5379.54
RSA120	51788.6	34888
RSA129	119028	86020.6
RSA130	158714	102728

Table 2 presents the CPU time and elapsed time for factoring RSA numbers using the CADO-NFS algorithm. The CPU time and elapsed time for CADO-NFS increase significantly as the size of the RSA number grows, reflecting the increased computational complexity of factoring larger numbers using classical methods.

4.4.3 Comparative Analysis

Comparing the results obtained from Shor's algorithm and CADO-NFS, we observe a significant difference in the execution times for factoring numbers using quantum and classical approaches.

Shor's algorithm can factor composite numbers up to 713 (which is around 10 bits) in a matter of seconds. On the other hand, CADO-NFS takes several hours to factor RSA numbers ranging from RSA100 (100 bits) to RSA130 (130 bits). This highlights the potential of Shor's algorithm and quantum computing to outperform classical factoring methods, even when considering the difference in bit sizes between the numbers factored by each approach.

It's important to note that the Qiskit simulator used for Shor's algorithm has limitations in terms of the number of qubits it can simulate, which restricts the maximum size of the numbers that can be factored. The largest number factored by Shor's algorithm in our experiments is 713, which is significantly smaller than the RSA numbers factored by CADO-NFS.

However, as quantum hardware continues to advance and scale, it is expected that larger numbers will become feasible to factor using Shor's algorithm. Extrapolating the performance of Shor's algorithm to larger bit sizes, it is reasonable to expect that quantum factoring will maintain its advantage over classical methods like CADO-NFS, even for numbers of similar bit sizes.

4.4.4 Conclusion

The experimental results and comparative analysis demonstrate the potential of Shor's algorithm and quantum computing for factoring numbers efficiently. The high accuracy and relatively low execution times of Shor's algorithm, even for simulated quantum circuits and smaller numbers, highlight its potential to outperform classical factoring methods like CADO-NFS.

Although the current limitations of quantum hardware and simulators restrict the maximum size of numbers that can be factored using Shor's algorithm, the results provide a promising outlook for the future of quantum factoring. As quantum technologies continue to advance, it is expected that Shor's algorithm will be able to factor numbers of increasing bit sizes, potentially surpassing the performance of classical factoring methods for numbers of similar sizes.

The data collected and analyzed in this study contribute to the understanding of the potential impact of quantum computing on cryptography and emphasize the need for further research and development in post-quantum cryptography. The results also highlight the importance of continued advancements in quantum hardware and error correction techniques to realize the full potential of Shor's algorithm and quantum factoring.

In conclusion, while the experimental results compare numbers of different bit sizes factored by Shor's algorithm and CADO-NFS, they still provide strong evidence for the potential of quantum factoring to outperform classical methods. As quantum technologies mature, it is expected that Shor's algorithm will be able to factor numbers of similar bit sizes to those used in RSA, further demonstrating the superiority of quantum factoring and necessitating the development of quantum-resistant security measures.

CHAPTER V: DISCUSSION

5.1 Implications for Cryptography

The experimental study conducted in Chapter 4 has significant implications for the field of cryptography, particularly in the context of the potential impact of quantum computing on the security of widely used cryptographic systems like RSA. The successful implementation of Shor's algorithm for factoring small numbers, even in a simulated environment, demonstrates the theoretical potential of quantum computers to solve the factoring problem efficiently (Ahnefeld et al., 2022).

Figure 3

Cado-NFS Factoring Time

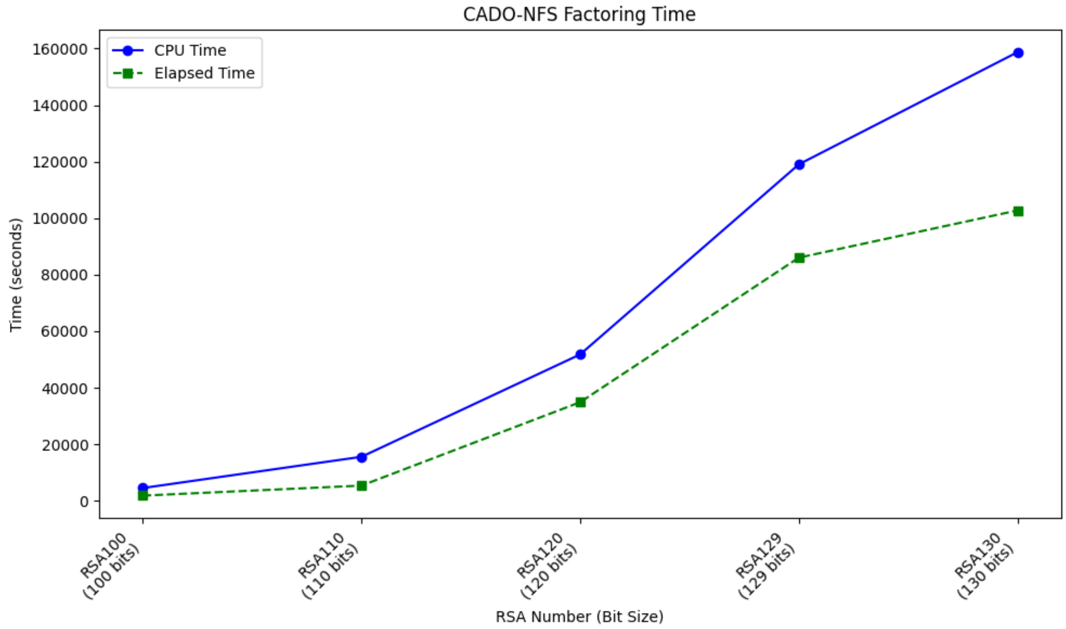
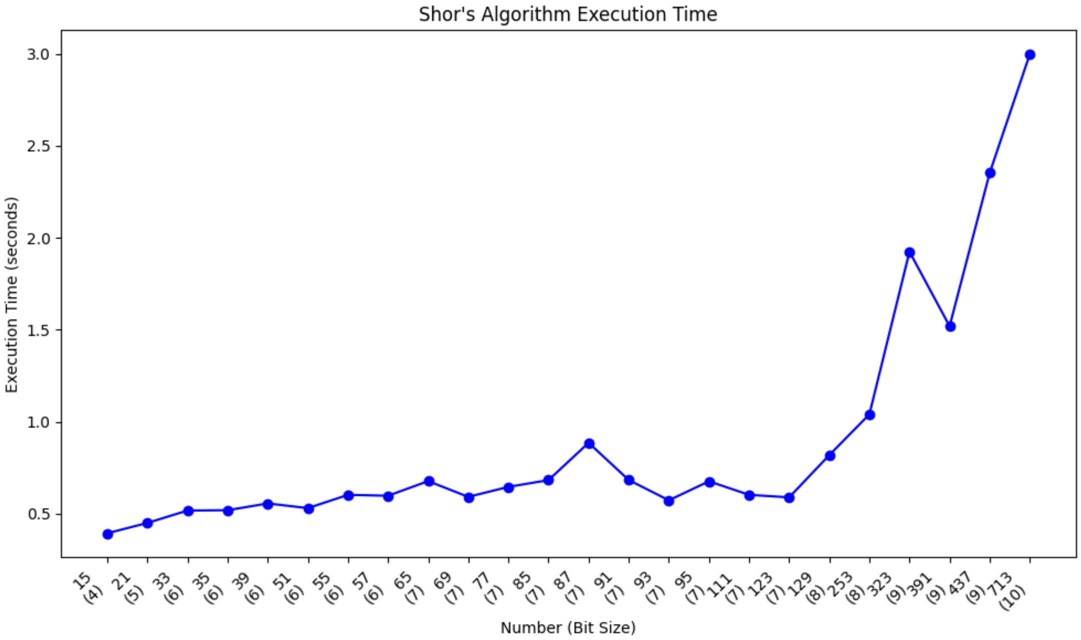


Figure 4

Shor's Algorithm Execution Time



The comparative analysis between Shor's algorithm and the classical factoring method CADO-NFS shown in *Figure 3* and *Figure 4* highlights the significant difference in execution times, with Shor's algorithm being able to factor numbers up to 713 (around 10 bits) in a matter of seconds, while CADO-NFS takes several hours to factor RSA numbers ranging from 100 to 130 bits. Although the numbers factored by Shor's algorithm in the experiments are smaller than the RSA numbers factored by CADO-NFS, the results still provide a clear indication of the potential speedup that quantum factoring can achieve over classical methods.

However, the true significance of these findings lies in the implications for factoring much larger numbers, such as RSA-250, which is a 250-decimal-digit (829-bit) composite number used in RSA cryptography. RSA is a widely used public-key cryptosystem that relies on the difficulty of factoring large composite numbers. The security of RSA is based on the assumption that given a large composite number N, which is the product of two prime numbers p and q (i.e., N = p × q), it is computationally infeasible to determine the values of p and q efficiently.

RSA-250 is a specific instance of an RSA number, which is the product of two prime numbers, each around 125 decimal digits (415 bits) long. The factorization of RSA-250 was completed in February 2020, and it required approximately 2,700 core years of computation using the General Number Field Sieve (GNFS) algorithm, which is the most efficient classical algorithm known for factoring large numbers (Boudot et al., n.d.).

The difficulty of factoring RSA-250 using classical algorithms can be understood by examining the running time of the GNFS. The time complexity of the GNFS for factoring an n-bit number N is given by the following equation:

$$O(exp((1.923\ +\ o(1))(log\ N)\hat{}(1/3)(log\ log\ N)\hat{}(2/3)))$$

Here, N is RSA-250, and o(1) is a term that approaches zero as N increases.
In contrast, Shor's quantum factoring algorithm has a polynomial time complexity of:

$$O((log\ N)\hat{}3)$$

While the factorization of RSA-250 is a significant achievement, it is important to note that the computational effort required to factor an RSA number using the GNFS algorithm grows subexponentially with the size of the number. Factoring larger RSA numbers, such as RSA-1024 or RSA-2048, would require significantly more computational resources and time compared to RSA-250.

The exponential time complexity of the GNFS makes factoring a 2048-bit number practically impossible for classical computers, while the polynomial time complexity of Shor's algorithm makes factoring the same number feasible for a large-scale quantum computer.

This mathematical comparison, along with the RSA-250 example, demonstrates the significant speed advantage of quantum computing in factorization. While the GNFS time complexity grows exponentially with the size of the number, making factorization intractable for large numbers, Shor's algorithm time complexity grows only polynomially, enabling efficient factorization even for large numbers.

The potential impact of this speed advantage on cryptography is profound. Many widely used cryptographic systems, such as RSA, rely on the difficulty of factoring large numbers for their security. The ability to factor these large numbers efficiently using quantum computers would render these cryptographic systems vulnerable to attacks, compromising the confidentiality and integrity of sensitive information.

On the other hand, Shor's algorithm provides a polynomial-time solution to the factoring problem on a quantum computer. The algorithm has two main parts: a classical part and a quantum part. The classical part reduces the factoring problem to the problem of finding the period of a modular exponential function. For RSA-250, we need to find the period r of the function:

$$f(x) = a\hat{}x \; mod \; N$$

Here, a is a randomly chosen integer that is coprime to N (i.e., a and N have no common factors other than 1), and N is RSA-250.

The quantum part of Shor's algorithm uses the Quantum Fourier Transform (QFT) and phase estimation to find the period r efficiently. The number of qubits required for this is approximately:

$$n \approx 2 \, log \, N$$

For RSA-250, this translates to roughly 1,658 qubits.

The quantum circuit for Shor's algorithm applies a series of quantum gates, including Hadamard gates, controlled phase rotations, and the QFT, to the quantum state. The number of quantum gates required is polynomial in log N, which is around (829^3) ≈ 5.7 × 10^8 for RSA-250.

After measuring the quantum state, the classical part of the algorithm uses the period r to find the factors p and q of N by computing:

$$p = gcd(a\hat{}(r/2) \pm 1, N)$$

$$q = N \, / \, p$$

Here, gcd stands for the greatest common divisor, which can be efficiently calculated using the Euclidean algorithm.

The power of Shor's algorithm lies in the quantum part, which can find the period r in polynomial time, as opposed to the exponential time required by classical algorithms. While Shor's algorithm offers an exponential speedup over classical factoring algorithms, its practical implementation for large numbers like RSA-250 is currently limited by the state of quantum hardware. The limitations include:

1. **Quantum Coherence**: Quantum systems are delicate and prone to decoherence due to interactions with the environment. Maintaining coherence for a large number of qubits over the duration of the algorithm is a significant challenge.
2. **Quantum Error Correction**: Quantum operations are susceptible to errors, which can accumulate and corrupt the quantum state. Quantum error correction schemes are necessary to detect and correct these errors, but they require additional qubits and increase the complexity of the quantum circuit.
3. **Scalability**: Building a quantum computer with a large number of qubits (e.g., 1,658 for RSA-250) that can perform reliable quantum operations is a major technological challenge. Current quantum computers have a limited number of qubits (around 100-200) and scaling up to thousands of qubits while maintaining the required precision and control is an active area of research.

In the research paper "The State of the Art in Integer Factoring and Breaking Public-Key Cryptography," the authors mention that experts agree that the sheer scale of the engineering effort required to build a quantum computer capable of running Shor's algorithm is large enough that it would not likely be possible for a single government to achieve such a feat entirely in secret and in a short timeline.(Boudot et al., n.d.)

In conclusion, factoring large numbers like RSA-250 is hard for classical computers due to the sub-exponential time complexity of the best-known factoring algorithms, such as the GNFS. The security of RSA relies on this hardness assumption, making it infeasible to factor large RSA key sizes in practice, as demonstrated by the significant computational resources and time required to factor RSA-250.

Shor's algorithm, on the other hand, provides a polynomial-time solution to the factoring problem on a quantum computer. Theoretically, Shor's algorithm could efficiently factor RSA-250 and other large numbers, posing a significant threat to the security of RSA and other factoring-based cryptographic schemes.

However, the practical implementation of Shor's algorithm for factoring large numbers is currently hindered by the limitations of quantum hardware. Building a large-scale, error-corrected quantum computer with thousands of qubits remains a significant engineering challenge.

As quantum technologies continue to advance and the number of available qubits increases, it is expected that Shor's algorithm will be able to factor larger numbers, eventually reaching bit sizes comparable to those used in RSA cryptography. The extrapolation of the experimental results suggests that quantum factoring will maintain its advantage over classical methods like CADO-NFS, even for numbers of similar bit sizes.

5.1.1 Impact on Public Key Cryptography

The potential impact of Shor's algorithm and quantum computing on public-key cryptography cannot be overstated (Wallden & Kashefi, 2019) RSA, along with other cryptographic schemes based on the factoring problem, forms the backbone of modern secure communication and data protection (Rivest et al., 1978). The ability to factor large numbers efficiently using quantum computers would render these schemes vulnerable to attacks, compromising the confidentiality and integrity of sensitive information (Bernstein et al., 2017).

As demonstrated in the experimental study, Shor's algorithm can factor small numbers efficiently on a quantum computer. While the current limitations of quantum hardware prevent the factorization of large numbers like RSA-250, the theoretical possibility of scaling up Shor's algorithm to factor such numbers poses a serious long-term threat to the security of RSA and other factoring-based cryptographic schemes.

This highlights the urgent need for transitioning to quantum-resistant cryptographic algorithms that can withstand attacks from both classical and quantum computers. The development and adoption of post-quantum cryptography are crucial steps in ensuring the long-term security of our digital infrastructure in the face of evolving quantum computing capabilities.

5.1.2 Quantum-Resistant Cryptography

Quantum-resistant cryptography, also known as post-quantum cryptography, refers to the development of cryptographic algorithms that are resistant to attacks by both classical and quantum computers (Bernstein et al., 2017). With the looming threat of quantum computers capable of running Shor's algorithm, it is essential to have alternative cryptographic schemes that can provide secure communication and data protection in a post-quantum world.

The advent of quantum computing and the development of quantum algorithms, such as Shor's algorithm, pose a significant threat to the security of classical cryptographic systems based on factorization and discrete logarithms. To address this challenge, researchers have been actively exploring quantum-resistant cryptographic methodologies, also known as post-quantum cryptography. Several mathematical approaches have been proposed for constructing quantum-resistant cryptographic schemes, including lattice-based cryptography(cryptography & 2014, 2014; Huang et al., 2023), code-based cryptography (Buchmann et al., n.d.), multivariate cryptography (Matsumoto & Imai, 1988), and hash-based signatures (Buchmann et al., n.d.). Efforts are underway to standardize quantum-resistant cryptographic algorithms, with the National Institute of Standards and Technology (NIST) leading a multi-year standardization process (Alagic et al., 2019). This section will provide an overview of the main approaches being pursued in the development of quantum-resistant cryptography.

5.1.2.1 Lattice-based Cryptography. Lattice-based cryptography is one of the most promising areas in post-quantum cryptography. Lattices are mathematical structures that consist of regularly spaced points in n-dimensional space. The security of lattice-based cryptographic schemes relies on the hardness of problems such as the Shortest Vector Problem (SVP) and the Closest Vector Problem (CVP), which are believed to be difficult to solve, even for quantum computers (Peikert, 2016).

One of the most well-known lattice-based cryptographic schemes is the Learning with Errors (LWE) problem, introduced by Oded Regev in 2005. The LWE problem is a generalization of the SVP, where the goal is to recover a secret vector given a set of noisy linear equations. The hardness of the LWE problem is based on the worst-case hardness of certain lattice problems, providing strong security guarantees.

Another notable lattice-based scheme is the Ring Learning with Errors (R-LWE) problem, which is a more efficient variant of LWE that uses polynomial rings. R-LWE has been used to construct various cryptographic primitives, including public-key encryption, digital signatures, and key exchange protocols.

Lattice-based cryptography has several advantages, including strong security proofs, versatility, and efficiency. However, lattice-based schemes often require large key and ciphertext sizes compared to classical cryptographic schemes, which can be a drawback in certain applications.

5.1.2.2 Code-based Cryptography. Code-based cryptography is another promising approach to post-quantum cryptography. Code-based schemes rely on the hardness of decoding random linear error-correcting codes, a problem that is believed to be difficult for both classical and quantum computers (McEliece, 1978).

The most well-known code-based cryptographic scheme is the McEliece cryptosystem, introduced by Robert McEliece in 1978. The McEliece cryptosystem uses a random Goppa code to generate a public key and a private key. The public key is a generator matrix of the code, while the private key consists of the underlying Goppa code, and a permutation matrix used to scramble the generator matrix.

To encrypt a message using the McEliece cryptosystem, the sender encodes the message using the public key and adds a random error vector. The receiver uses the private key to decode the message and remove the errors. The security of the McEliece cryptosystem is based on the difficulty of decoding a random linear code, which is an NP-hard problem.

Code-based cryptography has the advantage of fast encryption and decryption times, making it suitable for applications that require high-speed processing. However, like lattice-based schemes, code-based cryptography often results in large key sizes, which can be a limitation in certain scenarios.

5.1.2.3 Multivariate Cryptography. Multivariate cryptography is based on the difficulty of solving systems of multivariate polynomial equations over finite fields. The security of multivariate cryptographic schemes relies on the fact that solving such systems of equations is known to be an NP-hard problem, which is believed to be difficult for both classical and quantum computers (Matsumoto & Imai, 1988).

One of the main advantages of multivariate cryptography is its potential for fast encryption and decryption, as well as its resistance to quantum attacks. However, multivariate schemes often have large key sizes and can be vulnerable to certain types of attacks, such as the "minus" and "dual" attacks.

Several multivariate cryptographic schemes have been proposed, including the Unbalanced Oil and Vinegar (UOV) scheme, the Hidden Field Equations (HFE) scheme, and the Multivariate Quadratic (MQ) signature scheme. These schemes differ in the structure of the multivariate polynomial equations used and the specific techniques employed to generate the public and private keys.

5.1.2.4 Hash-based Signatures. Hash-based signature schemes are another class of quantum-resistant cryptographic methodologies. These schemes rely on the security of cryptographic hash functions, which are believed to be resistant to quantum attacks (Bernstein et al., 2017).

The most well-known hash-based signature scheme is the Merkle signature scheme, introduced by Ralph Merkle in 1979. The Merkle signature scheme uses a binary hash tree, also known as a Merkle tree, to generate and verify signatures. Each leaf of the Merkle tree corresponds to a one-time signature key pair, and the root of the tree serves as the public key for the entire scheme.

To sign a message using the Merkle signature scheme, the signer selects an unused one-time signature key pair from the leaf nodes and signs the message using that key pair. The signature consists of the one-time signature, along with the authentication path, which is a set of hash values that allow the verifier to reconstruct the root of the Merkle tree and verify the signature.

Hash-based signature schemes have the advantage of being simple, efficient, and having strong security proofs based on the security of the underlying hash function. However, they typically require maintaining a large state to keep track of the used one-time signature keys, which can be a drawback in certain applications.

5.1.2.5 Standardization Efforts. To ensure the timely development and deployment of quantum-resistant cryptographic solutions, several standardization efforts have been undertaken. The most notable of these efforts is the Post-Quantum Cryptography Standardization Process, initiated by the National Institute of Standards and Technology (NIST) in 2016.

The NIST standardization process consists of several rounds of evaluation, where submitted quantum-resistant cryptographic algorithms are assessed based on their security, performance, and other relevant criteria. The goal is to identify and standardize a set of quantum-resistant public-key cryptographic algorithms that can be used as a replacement for existing classical schemes, such as RSA and elliptic curve cryptography.

As of 2021, the NIST standardization process is in its third round, with several candidate algorithms being considered for standardization. These candidate algorithms cover various cryptographic primitives, including public-key encryption, digital signatures, and key establishment mechanisms, and are based on different mathematical approaches, such as lattices, codes, and multivariate polynomials.

The ongoing standardization efforts reflect the urgent need to prepare for the advent of large-scale quantum computers and ensure the long-term security of our digital systems. By establishing standards for quantum-resistant cryptography, these efforts aim to facilitate the widespread adoption of post-quantum cryptographic solutions and provide a foundation for the development of secure, interoperable systems in the post-quantum era.

5.1.2.6 Challenges and Future Directions. While significant progress has been made in the development of quantum-resistant cryptographic methodologies, several challenges remain.

One of the main challenges is the need for efficient and secure implementations of post-quantum cryptographic schemes, particularly in resource-constrained environments such as embedded systems and IoT devices.

Another challenge is the need for thorough security analysis and standardized testing methodologies to ensure the robustness of post-quantum cryptographic schemes against various types of attacks, including side-channel attacks and fault injection attacks.

Future research directions in quantum-resistant cryptography may include the development of new mathematical approaches and the refinement of existing ones, as well as the exploration of hybrid schemes that combine classical and post-quantum algorithms to provide both pre- and post-quantum security.

Additionally, research into the integration of quantum-resistant cryptography into existing protocols and systems, such as TLS and IPsec, will be crucial for ensuring a smooth transition to post-quantum security.

In conclusion, quantum-resistant cryptographic methodologies are essential for ensuring the long-term security of our digital systems in the face of the threat posed by quantum computing. By exploring various mathematical approaches, such as lattice-based, code-based, and multivariate cryptography, and by participating in ongoing standardization efforts, researchers and practitioners can contribute to the development and deployment of robust, quantum-resistant cryptographic solutions. As the field of quantum computing continues to advance, it is crucial to maintain a proactive approach to cryptographic research and development to stay ahead of evolving quantum threats.

5.2 Future of Cybersecurity in a Quantum World

The experimental results and theoretical analysis presented in this thesis underscore the potential impact of quantum computing on the future of cybersecurity. As quantum computers become more powerful and capable of running Shor's algorithm for factoring large numbers, the security of our current cryptographic infrastructure faces a significant threat.

The advent of quantum computing presents both opportunities and challenges for the future of cybersecurity. As quantum computers become more powerful and capable of running complex algorithms like Shor's algorithm, the security of our current cryptographic infrastructure faces a significant threat (Privacy & 2018, n.d.). This section will explore the implications of quantum computing for cybersecurity and discuss the necessary steps to ensure the long-term security of our digital systems in a post-quantum world.

5.2.1 The Impact of Quantum Computing on Cybersecurity

The development of large-scale quantum computers capable of running Shor's algorithm poses a serious risk to the security of widely used public-key cryptographic schemes, such as RSA and elliptic curve cryptography (ECC) (Bernstein et al., 2017). These schemes rely on the computational difficulty of factoring large numbers or solving the discrete logarithm problem, both of which can be efficiently solved using Shor's algorithm on a quantum computer.

If a sufficiently powerful quantum computer is built, it could break the security of these cryptographic schemes, rendering them vulnerable to attacks(Privacy, 2018). This would have far-reaching consequences for the security of our digital infrastructure, including secure communication, financial transactions, and data protection.

However, it is important to note that the development of a cryptographically relevant quantum computer, capable of breaking current public-key cryptography, is still a significant engineering challenge. Experts estimate that such a quantum computer would require millions of stable qubits and the ability to perform complex quantum operations with extremely low error rates. While the exact timeline for the development of such a quantum computer is uncertain, it is crucial to take proactive measures to ensure the long-term security of our digital systems.

5.2.2 The Need for Quantum-Resistant Cryptography

To mitigate the risks posed by quantum computing, it is essential to develop and deploy quantum-resistant cryptographic schemes (Bernstein et al., 2017). These schemes, also known as post-quantum cryptography, are designed to be secure against both classical and quantum attacks (Alagic et al., 2019).

Several mathematical approaches have been proposed for constructing quantum-resistant cryptographic schemes, and efforts are underway to standardize these algorithms (Alagic et al., 2019). These approaches rely on mathematical problems that are believed to be hard, even for quantum computers, such as the shortest vector problem (SVP) in lattices or the syndrome decoding problem in coding theory.

Efforts are underway to standardize quantum-resistant cryptographic algorithms, with the National Institute of Standards and Technology (NIST) leading a multi-year standardization process. The goal of this process is to evaluate and select a set of quantum-resistant public-key cryptographic algorithms that can be used as a replacement for existing schemes.

Transitioning to quantum-resistant cryptography will require significant effort and collaboration from researchers, industry, and government agencies. It is crucial to begin this transition well before the advent of cryptographically relevant quantum computers to ensure a smooth and secure migration to post-quantum cryptography.

5.2.3 Hybrid Classical-Quantum Cryptography

In addition to the development of quantum-resistant cryptographic schemes, another approach to ensuring the security of our digital systems in a post-quantum world is the use of hybrid classical-quantum cryptography (Pirandola et al., 2019).

Hybrid schemes combine classical and quantum cryptographic primitives to provide both pre- and post-quantum security. For example, a hybrid key exchange protocol could use a classical key exchange algorithm, such as Diffie-Hellman, in combination with a quantum-resistant key exchange algorithm, such as the NewHope scheme based on the ring learning with errors (R-LWE) problem.

The advantage of hybrid schemes is that they provide a hedge against the uncertainty surrounding the development of quantum computers and the security of post-quantum cryptographic schemes. If a quantum computer capable of breaking classical cryptography is developed sooner than expected, the quantum-resistant component of the hybrid scheme would still provide security. Conversely, if a vulnerability is discovered in a quantum-resistant scheme, the classical component would maintain security against classical attacks.

Hybrid classical-quantum cryptography can serve as a bridge between the current classical cryptographic infrastructure and a future post-quantum cryptographic landscape (Pirandola et al., 2019), allowing for a gradual and secure transition.

5.2.4 Quantum Cryptography and Quantum Key Distribution

In addition to the development of quantum-resistant cryptography, another promising approach to securing communication in a post-quantum world is quantum cryptography, particularly quantum key distribution (QKD) (Pirandola et al., 2019).

QKD is a method of securely exchanging cryptographic keys using the principles of quantum mechanics (Gisin et al., 2001). In a QKD protocol, such as the BB84 protocol, the sender and receiver exchange quantum states, typically photons, over a quantum channel. The security of QKD is based on the fundamental properties of quantum mechanics, such as the no-cloning theorem and the Heisenberg uncertainty principle, which prevent an attacker from eavesdropping on the quantum channel without being detected (Gisin et al., 2001).

QKD provides a means of establishing a secure key between two parties, which can then be used for encrypting and authenticating communication using classical cryptographic algorithms. The advantage of QKD is that it can provide long-term security, even in the presence of a quantum computer, as it does not rely on computational assumptions.

However, QKD also has its limitations, such as the need for a dedicated quantum channel and the limited distance over which keys can be securely distributed. Ongoing research aims to address these challenges and improve the practicality and scalability of QKD systems.

Quantum cryptography and QKD can play a complementary role to quantum-resistant cryptography in ensuring the security of communication in a post-quantum world. While quantum-resistant cryptography provides security against quantum attacks on classical channels, QKD can provide an additional layer of security for establishing keys over quantum channels.

5.2.5 The Importance of Cryptographic Agility

In preparing for the future of cybersecurity in a quantum world, it is essential to emphasize the importance of cryptographic agility (Wallden & Kashefi, 2019). Cryptographic agility refers to the ability of a system to easily and securely switch between different cryptographic algorithms and key sizes in response to evolving threats and technological advancements (Wallden & Kashefi, 2019).

As the field of quantum computing develops and new quantum-resistant cryptographic schemes are proposed and standardized, it is crucial for systems to be designed with cryptographic agility in mind (Wallden & Kashefi, 2019). This means that systems should be able to accommodate multiple cryptographic algorithms and key sizes and should have the ability to update and replace cryptographic primitives as needed (Alagic et al., 2019).

Cryptographic agility ensures that systems can adapt to changing security requirements and maintain long-term security in the face of emerging threats, such as advances in quantum computing.

To achieve cryptographic agility, it is important to:

1. Design systems with a modular architecture that allows for the easy integration and replacement of cryptographic components.
2. Use cryptographic libraries and protocols that support multiple algorithms and key sizes.
3. Regularly monitor developments in quantum computing and post-quantum cryptography and assess the need for updating cryptographic primitives.
4. Have a clear plan for transitioning to quantum-resistant cryptography, including the selection and implementation of standardized post-quantum algorithms.

By prioritizing cryptographic agility, organizations can ensure that their systems are prepared for the future of cybersecurity in a quantum world and can maintain long-term security in the face of evolving threats.

5.2.6 The Need for Collaboration and Awareness

Preparing for the future of cybersecurity in a quantum world requires collaboration and awareness across multiple stakeholders, including researchers, industry, government agencies, and the general public (Wallden & Kashefi, 2019).

Researchers play a critical role in developing and analyzing quantum-resistant cryptographic schemes, as well as advancing the field of quantum cryptography (Bernstein et al., 2017). Collaboration between researchers in academia and industry is essential for ensuring the security and practicality of post-quantum cryptographic solutions.

Industry stakeholders, including technology companies, cybersecurity firms, and standardization bodies, are responsible for implementing and deploying quantum-resistant cryptography in real-world systems (Wallden & Kashefi, 2019). Collaboration within industry is necessary for ensuring interoperability and promoting the widespread adoption of post-quantum cryptographic standards.

Government agencies, such as NIST, play a crucial role in driving the standardization of quantum-resistant cryptography and providing guidance and recommendations for transitioning to post-quantum security (Alagic et al., 2019). Governments also have a responsibility to raise awareness about the potential impacts of quantum computing on cybersecurity and to support research and development efforts in this area.

Finally, raising awareness among the general public about the implications of quantum computing for cybersecurity is important for promoting the adoption of quantum-resistant security measures and ensuring the long-term security of our digital systems.

Collaboration and awareness across these stakeholders will be essential for navigating the challenges and opportunities presented by the future of cybersecurity in a quantum world.

5.2.7 Conclusion

The future of cybersecurity in a quantum world presents both challenges and opportunities. The development of large-scale quantum computers capable of breaking current public-key cryptography poses a significant threat to the security of our digital systems.

However, through the development and deployment of quantum-resistant cryptography, the use of hybrid classical-quantum cryptographic schemes, and the advancement of quantum cryptography and QKD, we can ensure the long-term security of our digital infrastructure in a post-quantum era.

Preparing for the future of cybersecurity in a quantum world requires proactive measures, including the standardization of quantum-resistant cryptographic algorithms, the prioritization of cryptographic agility, and collaboration and awareness across multiple stakeholders.

By investing in research and development efforts, promoting the widespread adoption of quantum-resistant security measures, and fostering a culture of collaboration and awareness, we can navigate the challenges and opportunities presented by the quantum computing revolution and ensure the long-term security and resilience of our digital systems.

As the field of quantum computing continues to advance, it is crucial to stay informed about the latest developments and to adapt our cybersecurity strategies accordingly. The research and insights presented in this thesis contribute to the ongoing effort to understand and address the implications of quantum computing for cybersecurity and to develop effective strategies for securing our digital future in a post-quantum world.

5.3 Quantum Computers: Not Only for Cryptography

While the focus of this thesis has been on the impact of quantum computing on cryptography and cybersecurity, it is important to recognize that the potential applications of quantum computers extend far beyond these domains. Quantum computing has the potential to revolutionize various fields, offering solutions to complex problems that are intractable for classical computers (Preskill, 2018).

5.3.1 Quantum Simulation

One of the most promising applications of quantum computers is quantum simulation. Quantum simulation involves using a quantum computer to model and simulate the behavior of complex quantum systems, such as molecules, materials, and chemical reactions (Georgescu et al., 2013). Classical computers struggle to simulate these systems accurately due to the exponential growth of computational complexity with the size of the (Aspuru-Guzik & Walther, 2012).

Quantum computers, on the other hand, can leverage their inherent quantum properties to efficiently simulate quantum systems (Cirac & Zoller, 2012). By mapping the quantum state of the system being simulated onto the qubits of a quantum computer, researchers can study the properties and dynamics of these systems with unprecedented accuracy and efficiency (Lloyd, 1996).

Quantum simulation has the potential to accelerate drug discovery, materials science, and the development of new technologies (Cao et al., 2019). For example, quantum computers could be used to simulate the behavior of complex molecules, enabling the design of more effective drugs and materials with tailored properties (B. Bauer et al., 2020). Quantum simulation could also help optimize chemical processes, leading to more efficient and sustainable manufacturing practices (Aspuru-Guzik & Walther, 2012).

5.3.2 Optimization and Machine Learning

Quantum computers can also be applied to optimization problems and machine learning tasks. Optimization problems involve finding the best solution from a set of possible solutions, subject to certain constraints (Suchara et al., 2014). Many real-world problems, such as supply chain optimization, portfolio optimization, and route planning, can be formulated as optimization problems (Bharti et al., 2021).

Quantum algorithms, such as the Quantum Approximate Optimization Algorithm (QAOA) (Suchara et al., 2014) and the Variational Quantum Eigensolver (VQE) (Leymann & Barzen, 2020), have been developed to tackle optimization problems on quantum computers. These algorithms leverage the ability of quantum computers to explore vast solution spaces simultaneously, potentially leading to faster and more efficient optimization compared to classical methods (Bharti et al., 2021).

In the field of machine learning, quantum computers can be used to develop quantum-enhanced machine learning algorithms (Biamonte et al., 2017). Quantum machine learning algorithms, such as quantum support vector machines and quantum neural networks, can potentially offer speedups and improved performance compared to their classical counterparts (Havlíček et al., 2019). Quantum computers could also be used for feature extraction, dimensionality reduction, and the training of large-scale machine learning models (Biamonte et al., 2017).

5.3.3 Scientific Computing and Modeling

Quantum computers have the potential to revolutionize scientific computing and modeling across various domains, including physics, chemistry, and biology (Cao et al., 2019). Quantum algorithms can be used to solve complex scientific problems, such as simulating the behavior of subatomic particles, modeling the folding of proteins, and studying the properties of novel materials (B. Bauer et al., 2020).

For example, quantum computers could be used to simulate the behavior of complex quantum many-body systems, such as superconductors and quantum magnets (Lloyd, 1996).

These simulations could lead to a better understanding of the fundamental properties of matter and the discovery of new materials with exotic properties (Georgescu et al., 2013).

In the field of computational biology, quantum computers could be used to model the folding and dynamics of proteins, which is essential for understanding their function and designing new drugs (Cao et al., 2019). Quantum algorithms could also be applied to the analysis of large-scale genomic data, potentially leading to new insights into genetic diseases and personalized medicine (Biamonte et al., 2017).

5.3.4 Challenges and Future Directions

While the potential applications of quantum computers beyond cryptography are vast and exciting, there are still significant challenges to overcome (Preskill, 2018). One of the main challenges is the development of robust and scalable quantum hardware (Leymann & Barzen, 2020). Current quantum computers are still limited in terms of the number of qubits, connectivity, and error rates, which restrict the size and complexity of the problems that can be solved (Bharti et al., 2021).

Another challenge is the development of efficient and reliable quantum algorithms for specific applications (Preskill, 2018). Designing quantum algorithms that can effectively leverage the capabilities of quantum computers while being resilient to noise and errors is an active area of research (Bharti et al., 2021).

The integration of quantum computers with classical computing infrastructure and the development of quantum software and programming tools are also important considerations (Leymann & Barzen, 2020). Establishing standard interfaces, libraries, and development environments for quantum computing will be crucial for facilitating the adoption and use of quantum computers across different domains (Preskill, 2018).

Despite these challenges, the future of quantum computing beyond cryptography is promising. Ongoing research and development efforts aim to address the limitations of current quantum hardware, improve the efficiency and reliability of quantum algorithms, and explore new applications of quantum computing across various fields (Bharti et al., 2021).

As quantum computers continue to evolve and mature, their impact on scientific discovery, industrial optimization, and technological innovation is expected to grow (Preskill, 2018). The potential of quantum computers to solve complex problems and provide new insights into the fundamental workings of nature makes them a transformative technology with far-reaching implications (Georgescu et al., 2013).

CHAPTER VI: CONCLUSION

To sum up, the experimental research and comparative examination outlined in this thesis illustrate the possible influence of quantum computing on cryptography and cybersecurity. The outcomes show the significance of creating and implementing quantum-resistant cryptography to protect our digital infrastructure in the long run.

However, beyond the realm of cryptography, quantum computers can reshape science and society. In a world built on the understanding of how nature works, the potential solutions they promise to some of the hardest problems of science, from disorders to fundamental particles, are breathtakingly exciting. Even if none of these promises are ever fully kept, the power and potential could be transformative, as enabling of other scientific tools such as telescopes, or another great leap in computation like the development of Turing's Universal Machine in 1936.

As we confront the difficulties and advantages posed by quantum computing, it is imperative to undertake a comprehensive strategy that consists of the security and general hope of this transformative innovation. Uninterrupted examination and energy to improve quantum algorithms, quantum machines, and quantum applications will be paramount to exploit fully quantum computers in various segments.

Additionally, it will be crucial to promote cooperation and the exchange of insights within the scientific community, amongst shareholders in the industry, as well as individuals in policy-making roles, for the purpose of ensuring that the responsible advancement and implementation of quantum computer technologies is brought to fruition. Formulating and setting up targets, regulations, as well as methods that measure up to the highest standards and practices are bound to encourage the principled and secure trade of quantum computers in multitudinous domains.

The thesis demonstrates an experimental analysis that contributes to the increasing understanding of how quantum computing changes cryptology and cybersecurity. This study provides not just a robust starting point but also a clear direction for future research development on encryption protocols that are resistant to quantum attacks.

REFERENCES

Ahnefeld, F., Theurer, T., Egloff, D., Matera, J. M., & Plenio, M. B. (2022). Coherence as a Resource for Shor's Algorithm. *Physical Review Letters*, *129*(12). https://doi.org/10.1103/PhysRevLett.129.120501

Alagic, G., Alperin-Sheriff, J., Apon, D., Cooper, D., Dang, Q., Liu, Y.-K., Miller, C., Moody, D., Peralta, R., Perlner, R., Robinson, A., & Smith-Tone, D. (2019). *Status report on the first round of the NIST post-quantum cryptography standardization process*. https://doi.org/10.6028/NIST.IR.8240

applications, D. C.-L. algebra and its, & 1993, undefined. (n.d.). Solving linear equations over GF (2): block Lanczos algorithm. *ElsevierD CoppersmithLinear Algebra and Its Applications, 1993•Elsevier*. Retrieved March 24, 2024, from https://www.sciencedirect.com/science/article/pii/002437959390235G

Aspuru-Guzik, A., & Walther, P. (2012). Photonic quantum simulators. *Nature Physics 2012 8:4*, *8*(4), 285–291. https://doi.org/10.1038/nphys2253

Bai, S., Brent, R., Computation, E. T.-M. of, & 2015, undefined. (2010). Root optimization of polynomials in the number field sieve. *Ams.OrgS Bai, R Brent, E ThoméMathematics of Computation, 2015•ams.Org*, *84*(295), 2926–2929. https://www.ams.org/mcom/0000-000-00/S0025-5718-2015-02926-3

Bauer, B., Bravyi, S., Motta, M., & Kin-Lic Chan, G. (2020). Quantum Algorithms for Quantum Chemistry and Quantum Materials Science. *Chemical Reviews*, *120*(22), 12685–12717. https://doi.org/10.1021/ACS.CHEMREV.9B00829/ASSET/IMAGES/MEDIUM/CR9B00829_0023.GIF

Bauer, F. (2013). *Decrypted secrets: methods and maxims of cryptology*. https://books.google.com/books?hl=en&lr=&id=h-mpCAAAQBAJ&oi=fnd&pg=PA1&dq=Bauer,+F.+L.+(2013).+Decrypted+secrets:+Methods+and+maxims+of+cryptology.+Springer+Science+%26+Business+Media&ots=k9U92rL41e&sig=Aw33luOfd7oQricXaUz6pnCLG4E

Bernstein, D., Nature, T. L.-, & 2017, undefined. (2017). Post-quantum cryptography. *Nature.ComDJ Bernstein, T LangeNature, 2017•nature.Com*, *549*(7671), 188–194. https://doi.org/10.1038/nature23461

Bharti, K., Cervera-Lierta, A., Kyaw, T. H., Haug, T., Alperin-Lea, S., Anand, A., Degroote, M., Heimonen, H., Kottmann, J. S., Menke, T., Mok, W.-K., Sim, S., Kwek, L.-C., & Aspuru-Guzik, A. (2021). Noisy intermediate-scale quantum (NISQ) algorithms. *Reviews of Modern Physics*, *94*(1). https://doi.org/10.1103/RevModPhys.94.015004

Biamonte, J., Wittek, P., Pancotti, N., Rebentrost, P., Wiebe, N., & Lloyd, S. (2017). Quantum machine learning. *Nature 2017 549:7671*, *549*(7671), 195–202. https://doi.org/10.1038/nature23474

Boudot, F., Gaudry, P., Guillevic, A., Heninger, N., Thomé, E., Zimmermann, P., & The, al. (n.d.). State of the Art in Integer Factoring and Breaking Public-Key Cryptography. *IEEE Security and Privacy Magazine*, *2022*(2). https://doi.org/10.1109/MSEC.2022.3141918ï

Bressoud, D. (2012). *Factorization and primality testing*. https://books.google.com/books?hl=en&lr=&id=p-vpBwAAQBAJ&oi=fnd&pg=PR7&dq=Bressoud,+D.+M.+(1989).+Factorization+and+primality+testing&ots=b9B45RhyrH&sig=ztCOvnxYNj_OjMj1z1DpkMxvS1k

Brigham, E. 0, & Morrow, R. E. (n.d.). *The fast Fourier transform*.

Buchmann, J., Dahmen, E., Cryptography, M. S.-P.-Q., & 2009, undefined. (n.d.). Hash-based digital signature schemes. *Springer*. Retrieved March 23, 2024, from https://link.springer.com/content/pdf/10.1007/978-3-540-88702-7_3.pdf

Buhler, J. P., Lenstra, H. W., & Pomerance, C. (1993a). *Factoring integers with the number field sieve*. 50–94. https://doi.org/10.1007/BFB0091539

Buhler, J. P., Lenstra, H. W., & Pomerance, C. (1993b). *Factoring integers with the number field sieve*. 50–94. https://doi.org/10.1007/BFB0091539

Cao, Y., Romero, J., Olson, J. P., Degroote, M., Johnson, P. D., Kieferová, M., Kivlichan, I. D., Menke, T., Peropadre, B., Sawaya, N. P. D., Sim, S., Veis, L., & Aspuru-Guzik, A. (2019). Quantum Chemistry in the Age of Quantum Computing. *Chemical Reviews*, *119*(19), 10856–10915. https://doi.org/10.1021/ACS.CHEMREV.8B00803/ASSET/IMAGES/MEDIUM/CR8B00803_0014.GIF

Cirac, J. I., & Zoller, P. (2012). Goals and opportunities in quantum simulation. *Nature Physics 2012 8:4*, *8*(4), 264–266. https://doi.org/10.1038/nphys2275

computer, C. P.-F. and trends® in theoretical, & 2016, undefined. (2016). A decade of lattice cryptography. *Nowpublishers.ComC PeikertFoundations and Trends® in Theoretical Computer Science, 2016•nowpublishers.Com*, *10*(4), 283–424. https://doi.org/10.1561/0400000074

Crandall, R. E., & Pomerance, Carl. (2005). *Prime numbers : a computational perspective*. 597.

Cross, A. W., Bishop, L. S., Smolin, J. A., & Gambetta, J. M. (2017). *Open Quantum Assembly Language*. https://arxiv.org/abs/1707.03429v2

Crypto, F. (2002). *Has the RSA algorithm been compromised as a result of Bernstein's Paper?* https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=46b36bf4911c5c9215918bea899269583d13b7dd

cryptography, C. P.-I. workshop on post-quantum, & 2014, undefined. (2014). Lattice cryptography for the internet. *SpringerC PeikertInternational Workshop on Post-Quantum Cryptography, 2014•Springer*. https://link.springer.com/chapter/10.1007/978-3-319-11659-4_12

Deutsch, D. (1985). Quantum theory, the Church–Turing principle and the universal quantum computer. *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, *400*(1818), 97–117. https://doi.org/10.1098/RSPA.1985.0070

Diffie, W., … M. H.-C. T. W. of W. D. and, & 2022, undefined. (n.d.). New directions in cryptography. *Dl.Acm.Org*. Retrieved March 23, 2024, from https://dl.acm.org/doi/pdf/10.1145/3549993.3550007

Galindo, A., & Martí N-Delgado, M. A. (n.d.). *Information and computation: Classical and quantum aspects*.

Georgescu, I. M., Ashhab, S., & Nori, F. (2013). Quantum Simulation. *Reviews of Modern Physics, 86*(1). https://doi.org/10.1103/RevModPhys.86.153

Gisin, N., Ribordy, G., Tittel, W., & Zbinden, H. (2001). *arXiv:quant-ph/0101098v2 18 Sep 2001*.

Havlíček, V., Córcoles, A. D., Temme, K., Harrow, A. W., Kandala, A., Chow, J. M., & Gambetta, J. M. (2019). Supervised learning with quantum-enhanced feature spaces. *Nature 2019 567:7747*, *567*(7747), 209–212. https://doi.org/10.1038/s41586-019-0980-2

Huang, B., Gao, J., & Li, X. (2023). Efficient lattice-based revocable attribute-based encryption against decryption key exposure for cloud file sharing. *Journal of Cloud Computing*, *12*(1). https://doi.org/10.1186/s13677-023-00414-w

Ireland, K., & Rosen, M. (1990). *A classical introduction to modern number theory*. https://books.google.com/books?hl=en&lr=&id=jhAXHuP2y04C&oi=fnd&pg=PR5&dq=Ireland,+K.,+%26+Rosen,+M.+(1990).+A+classical+introduction+to+modern+number+theory.+Springer+Science+%26+Business+Media&ots=ikqSExLCcu&sig=4GsMJWILy7MlIcbiGghGYxKbHq4

Jia, S. (2023). Comparison of Performances for Quantum and Conventional Algorithms: Shor's algorithm and Boson Sampling. In *Highlights in Science, Engineering and Technology TPCEE* (Vol. 2022).

Kahn, D. (1996). *The Codebreakers: The comprehensive history of secret communication from ancient times to the internet*. https://books.google.com/books?hl=en&lr=&id=3S8rhOEmDIIC&oi=fnd&pg=PT10&dq=Kahn,+D.+(1996).+The+Codebreakers:+The+comprehensive+history+of+secret+communication+from+ancient+times+to+the+Internet.+Simon+and+Schuster&ots=SC-eAvrcnv&sig=3T9TrIpYhYxQqisaq08CAGFUcKs

Kanamori, Y., & Yoo, S.-M. (2020). Quantum Computing: Principles and Applications. *Journal of International Technology and Information Management*, *29*(2), 43–71. https://doi.org/10.58729/1941-6679.1410

Kleinjung, T. (2017). Polynomial Selection for the Number Field Sieve. In *Topics in Computational Number Theory Inspired by Peter L. Montgomery* (pp. 161–174). Cambridge University Press. https://doi.org/10.1017/9781316271575.007

Kleinjung, T., Aoki, K., Franke, J., Lenstra, A. K., Thomé, E., Bos, J. W., Gaudry, P., Kruppa, A., Montgomery, P. L., Osvik, D. A., Te Riele, H., Timofeev, A., & Zimmermann, P. (2010). Factorization of a 768-Bit RSA modulus. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *6223 LNCS*, 333–350. https://doi.org/10.1007/978-3-642-14623-7_18/COVER

Kusyk, J., Saeed, S. M., & Uyar, M. U. (2021). Survey on Quantum Circuit Compilation for Noisy Intermediate-Scale Quantum Computers: Artificial Intelligence to Heuristics. *IEEE Transactions on Quantum Engineering*, *2*, 1–16. https://doi.org/10.1109/tqe.2021.3068355

Lenstra, A. K., Lenstra, H. W., Manasse, M. S., & Pollard, J. M. (1990). *Number field sieve*. 564–572. https://doi.org/10.1145/100216.100295

Leymann, F., & Barzen, J. (2020). The bitter truth about gate-based quantum algorithms in the NISQ era. *Quantum Science and Technology*, *5*(4), 044007. https://doi.org/10.1088/2058-9565/ABAE7D

Lin, F. X. (n.d.). *Shor's Algorithm and the Quantum Fourier Transform*.

Lloyd, S. (1996). Universal Quantum Simulators. *Science*, *273*(5278), 1073–1078. https://doi.org/10.1126/SCIENCE.273.5278.1073

Matsumoto, T., & Imai, H. (1988). Public quadratic polynomial-tuples for efficient signature-verification and message-encryption. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *330 LNCS*, 419–453. https://doi.org/10.1007/3-540-45961-8_39

Montanaro, A. (2016). Quantum algorithms: An overview. In *npj Quantum Information* (Vol. 2, Issue 1). Nature Partner Journals. https://doi.org/10.1038/npjqi.2015.23

Mullamuri, B. (2021). *ENABLING QUANTUM CRYPTOGRAPHY USING QUANTUM COMPUTER PROGRAMMING*.

Nielsen, M. A., & Chuang, I. L. (2010). *Quantum computation and quantum information*. Cambridge University Press.

Nielsen, M., & Chuang, I. (2010). *Quantum computation and quantum information*. https://books.google.com/books?hl=en&lr=&id=-s4DEy7o-a0C&oi=fnd&pg=PR17&dq=Quantum+computation+and+quantum+information&ots=NJ2FesoA-x&sig=UmjUDB0D9Z9O_90nMT11xaZ7bjc

Pirandola, S., Andersen, U. L., Banchi, L., Berta, M., Bunandar, D., Colbeck, R., Englund, D., Gehring, T., Lupo, C., Ottaviani, C., Pereira, J., Razavi, M., Shaari, J. S., Tomamichel, M., Usenko, V. C., Vallone, G., Villoresi, P., & Wallden, P. (2019). Advances in quantum cryptography. *Opg.Optica.OrgS Pirandola, UL Andersen, L Banchi, M Berta, D Bunandar, R Colbeck, D EnglundAdvances in Optics and Photonics, 2020•opg.Optica.Org*. https://opg.optica.org/abstract.cfm?uri=aop-12-4-1012

Pollard, J. M. (1975). A monte carlo method for factorization. *BIT*, *15*(3), 331–334. https://doi.org/10.1007/BF01933667/METRICS

Pomerance, C. (n.d.). *A Tale of Two Sieves*.

Preskill, J. (2018). Quantum Computing in the NISQ era and beyond. *Quantum*, *2*, 79. https://doi.org/10.22331/q-2018-08-06-79

Privacy, M. M.-I. S. &, & 2018, undefined. (n.d.). Cybersecurity in an era with quantum computers: Will we be ready? *Ieeexplore.Ieee.OrgM MoscaIEEE Security & Privacy, 2018•ieeexplore.Ieee.Org*. Retrieved March 23, 2024, from https://ieeexplore.ieee.org/abstract/document/8490169/

Qiskit Development Team. (2024). *Qiskit Documentation*. IBM. https://docs.quantum.ibm.com/

Rapid solution of problems by quantum computation. (1992). *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences*, *439*(1907), 553–558. https://doi.org/10.1098/RSPA.1992.0167

Riesel, H. (1994). Prime Numbers and Computer Methods for Factorization. *Prime Numbers and Computer Methods for Factorization*. https://doi.org/10.1007/978-1-4612-0251-6

Rivest, R. L., Shamir, A., & Adleman, L. (1978). A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications of the ACM*, *21*(2), 120–126. https://doi.org/10.1145/359340.359342

Rossi, M., Asproni, L., Caputo, D., Rossi, S., Cusinato, A., Marini, R., Agosti, A., & Magagnini, M. (2021). *Using Shor's algorithm on near term Quantum computers: a reduced version*. http://arxiv.org/abs/2112.12647

Shor, P. W. (n.d.). *Algorithms for Quantum Computation: Discrete Logarithms and Factoring*.

Shor, P. W. (1999). Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer *. In *Society for Industrial and Applied Mathematics* (Vol. 41, Issue 2). http://www.siam.org/journals/sirev/41-2/34701.html

Singh, S. (2000). *The code book: the science of secrecy from ancient Egypt to quantum cryptography*. https://books.google.com/books?hl=en&lr=&id=vM2MDQAAQBAJ&oi=fnd&pg=PR13&dq=Singh,+S.+(2011).+The+code+book:+The+science+of+secrecy+from+ancient+Egypt+to+quantum+cryptography.+Anchor&ots=tw9LHay3Iq&sig=szJqFjHdjxwoVZ3xSaKXUE7KNIw

Suchara, M., Alexeev, Y., Chong, F., Finkel, H., Hoffmann, H., Larson, J., Osborn, J., & Smith, G. (2014). A Quantum Approximate Optimization Algorithm. *Proceedings of the 3rd International Workshop on Post-Moore's Era Supercomputing*, 3. https://arxiv.org/abs/1411.4028v1

theory, C. P.-C. methods in number, & 1982, undefined. (n.d.). Analysis and comparison of some integer factoring algorithms. *Cir.Nii.Ac.Jp*. Retrieved April 14, 2024, from https://cir.nii.ac.jp/crid/1571698600585656320

Thv, R. M.-C., & 1978, undefined. (n.d.). A public-key cryptosystem based on algebraic. *Ntrs.Nasa.GovRJ McElieceCoding Thv, 1978•ntrs.Nasa.Gov*. Retrieved April 14, 2024, from https://ntrs.nasa.gov/api/citations/19780016269/downloads/19780016269.pdf#page=123

Wallden, P., ACM, E. K.-C. of the, & 2019, undefined. (2019). Cyber security in the quantum era. *Dl.Acm.OrgP Wallden, E KashefiCommunications of the ACM, 2019•dl.Acm.Org*, *62*(4), 120–129. https://doi.org/10.1145/3241037

Yan, S. (2013). *Number theory for computing*. https://books.google.com/books?hl=en&lr=&id=KAqrCAAAQBAJ&oi=fnd&pg=PA1&dq=Yan,+S.+Y.+(2008).+Number+theory+for+computing.+Springer+Science+%26+Business+Media&ots=TjeDOndWME&sig=LYvJwBP-fk5rSTN_LMdKYjbTVpE