Advantage of Quantum Computing in Breaking RSA Encryption

Srinivas Rao Tammireddy
Roll No: 217Y1A05C0
Guide: Dr. S. Prathap Singh

March 24, 2025

Certificate

This is to certify that the seminar report titled **Advantage of Quantum Computing in Breaking RSA Encryption** has been successfully completed by **Your Name**, Roll No: **Your Roll Number**, under the guidance of **Guide Name**. This report is submitted in partial fulfillment of the requirements for the seminar presentation.

Signature of Guide
Signature of Student

Contents

List of Figures

Abstract

Quantum computing has emerged as a revolutionary paradigm in computation, offering significant advantages over classical computing for certain types of problems. One of the most notable implications of quantum computing is its potential to break widely used cryptographic systems, such as RSA encryption. This report explores the principles of RSA encryption, the capabilities of quantum computing, and how quantum algorithms, particularly Shor's algorithm, pose a threat to the security of RSA. The report also discusses the implications of this threat and the need for quantum-resistant cryptographic systems.

Chapter 1

Introduction to RSA Encryption and Quantum Computing

RSA encryption is a cornerstone of modern cryptography, widely used to secure digital communications. Its security is based on the computational difficulty of factoring large integers, a problem that classical computers find infeasible to solve within a reasonable timeframe. However, the advent of quantum computing introduces a paradigm shift in computational capabilities. Quantum computers, leveraging principles such as superposition and entanglement, can solve certain problems exponentially faster than classical computers. This chapter provides an overview of RSA encryption, its working principles, its critical role in ensuring secure communication, and the potential impact of quantum computing on its security.

1.1 Overview of RSA Encryption

RSA encryption, named after its inventors Rivest, Shamir, and Adleman, is a public-key cryptographic system that relies on two keys: a public key for encryption and a private key for decryption. The security of RSA is rooted in the mathematical challenge of factoring the product of two large prime numbers, which forms the basis of the private key.

1.2 Working Principles of RSA

The RSA algorithm involves three main steps:

1. **Key Generation:** Two large prime numbers are chosen, and their product forms the modulus. The public and private keys are derived using modular arithmetic.

2. **Encryption:** The sender uses the recipient's public key to encrypt the message, ensuring that only the recipient can decrypt it.

3. **Decryption:** The recipient uses their private key to decrypt the message, recovering the original plaintext.

1.3 Role of RSA in Secure Communication

RSA encryption is widely used in various applications, including:

- Securing online transactions and communications.
- Authenticating digital signatures to verify the integrity of messages.
- Protecting sensitive data in email and file encryption.

1.4 Introduction to Quantum Computing

Quantum computing leverages the principles of quantum mechanics, such as superposition and entanglement, to perform computations. Unlike classical computers, which process information in binary bits, quantum computers use quantum bits (qubits) that can represent multiple states simultaneously.

1.5 Impact of Quantum Computing on RSA

The computational power of quantum computers poses a significant threat to RSA encryption. Quantum algorithms, such as Shor's algorithm, can factorize large integers exponentially faster than classical algorithms, rendering RSA insecure. This has profound implications for digital security and the need to develop quantum-resistant cryptographic systems.

This report will delve into the mathematical foundation of RSA encryption, the vulnerabilities of RSA to quantum attacks, and the implications of quantum computing on digital security. It will also explore the existing methods for breaking RSA encryption, the drawbacks of these methods, and the advantages of transitioning to quantum-resistant cryptographic systems. The report will conclude with a discussion on the future enhancements and applications of quantum-resistant cryptography.

Chapter 2

The Vulnerability of RSA Encryption to Quantum Computing: Literature Review

2.1 Introduction to RSA Vulnerability

Rivest–Shamir–Adleman (RSA) encryption, a cornerstone of modern digital communication, provides security by transforming readable data into an unintelligible format that can only be reversed with a specific key [2]. This method relies on the computational difficulty of solving certain mathematical problems using classical computers [3]. However, as discussed in Chapter 1, the emergence of quantum computing presents a potential paradigm shift in computational power, threatening the security of widely used encryption algorithms like RSA [7]. This chapter examines the fundamental principles of RSA encryption, the theoretical capability of quantum computers utilizing Shor's algorithm to compromise its security, the current state of quantum computing technology, the potential impact on cybersecurity, and the ongoing development of countermeasures in the form of post-quantum cryptography.

2.2 RSA Encryption Fundamentals Review

RSA is an asymmetric encryption algorithm that employs a pair of mathematically linked keys: a public key and a private key [3]. The public key is freely available and used for encrypting messages, while the private key is kept confidential by the owner and is necessary for decryption [3]. This system eliminates the need for prior key sharing, a significant advantage over traditional symmetric encryption methods [10].

The security of RSA hinges on the generation of these keys, which involves selecting two large prime numbers, denoted as p and q [3]. These primes are multiplied to produce a modulus n, which forms part of both the public and private keys [3]. The public key consists of the pair (n, e), where e is the encryption exponent. The private key includes n and a decryption exponent d, which is the modular multiplicative inverse of e modulo $\phi(n)$, where $\phi(n)$ is Euler's totient function calculated as $(p-1) \cdot (q-1)$ [9].

Encryption involves raising the plaintext message m to the power of e and taking the modulus of n to obtain the ciphertext C ($C = m^e \bmod n$) [9]. Decryption reverses this process by raising the ciphertext C to the power of d and taking the modulus of n to retrieve the original message m ($m = C^d \bmod n$) [9]. This process is further illustrated in Chapter 3.2.

2.3 Mathematical Foundation of RSA Security Analysis

The security of RSA encryption is fundamentally based on the computational difficulty of factoring large integers, specifically the modulus n, back into its original prime factors p and q [3]. While multiplying two large prime numbers is a computationally straightforward task, reversing this process to find the prime factors of their product becomes exponentially harder as the size of the numbers increases [3].

The RSA problem is defined as the task of recovering the original message given only the public key (n, e) and the ciphertext [5]. The most efficient known classical method to solve this problem involves factoring the modulus n. The length of the RSA key, typically 1024 or 2048 bits, determines the size of the modulus n. Larger key lengths result in significantly larger numbers, making factorization practically infeasible for classical computers within a reasonable timeframe [3]. For instance, factoring a 2048-bit number using classical methods would take an estimated 300 trillion years [6]. This computational barrier is the cornerstone of RSA's security [4], as also discussed in Chapter 5.

2.4 Shor's Algorithm: A Quantum Approach

In 1994, Peter Shor developed a quantum algorithm that possesses the theoretical capability to factorize large integers exponentially faster than the best-known classical algorithms [8]. Shor's algorithm leverages the principles of quantum mechanics, such as superposition and quantum parallelism, to perform computations in a fundamentally different way than classical computers [7]. The principles of quantum mechanics underlying this algorithm are detailed in Chapter 4.

The algorithm consists of two main stages: a classical reduction of the factorization problem to the problem of finding the period of a modular function, followed by the use of the Quantum Fourier Transform (QFT) on a quantum computer to efficiently determine this period [8]. Once the period is found, the prime factors of the original number can be derived through classical post-processing using techniques like Euclid's algorithm to find the greatest common divisor [11].

The ability of Shor's algorithm to solve the integer factorization problem in polynomial time poses a direct threat to the security of RSA encryption, as it could potentially allow an attacker with a sufficiently powerful quantum computer to efficiently derive the private key from the public key [8]. This threat is further elaborated in Chapter 3.2.

Quantum computers are not yet advanced enough to break RSA-2048 encryption due to limitations in stable qubits and error correction [8, 13]. Estimates suggest 4,000 to 20 million logical qubits are needed [14]. The largest number factored so far is 21 [8], and hybrid approaches have only tackled very small keys [15]. This aligns with the challenges in Chapter 11.1.

Chapter 3

Background on RSA Encryption and Its Vulnerabilities

The existing cryptographic systems, including RSA, rely on the computational difficulty of certain mathematical problems, such as integer factorization. Classical computers are unable to solve these problems efficiently, which forms the basis of RSA's security. This chapter discusses the existing system and its reliance on classical computational limitations.

3.1 Mathematical Foundation of RSA

The RSA cryptosystem, introduced by Rivest, Shamir, and Adleman in 1978 [1], is based on the mathematical principles of number theory and modular arithmetic.

3.2 Key Generation Process

The RSA key generation process involves the following steps:

1. Select two large prime numbers, p and q (typically 1024 bits or more).
2. Compute their product, $n = p \times q$, which serves as the modulus.
3. Calculate the totient, $\phi(n) = (p-1)(q-1)$.
4. Choose a public exponent, e, such that $1 < e < \phi(n)$ and $\gcd(e, \phi(n)) = 1$.
5. Compute the private exponent, d, as the modular multiplicative inverse of e modulo $\phi(n)$, satisfying $e \cdot d \equiv 1 \pmod{\phi(n)}$.

As shown in Figure 3.1, this process ensures the generation of secure keys for encryption and decryption.

3.3 Encryption and Decryption Methods

To encrypt a message M using a public encryption key (e, n), proceed as follows:

1. Represent the message as an integer between 0 and $n-1$. For long messages, break them into blocks and represent each block as such an integer.

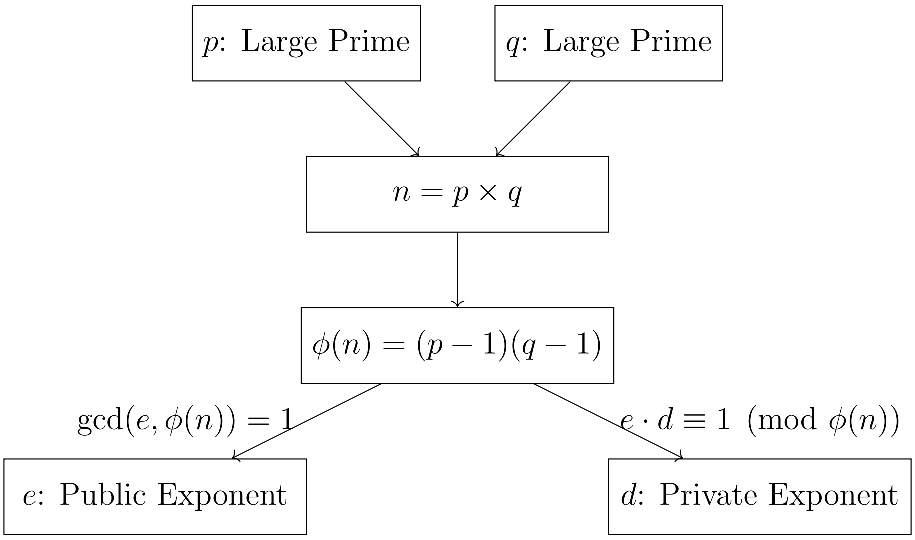


Figure 3.1: Diagram illustrating the RSA key generation process.

2. Encrypt the message by raising it to the e-th power modulo n:

$$C \equiv M^e \pmod{n},$$

where C is the ciphertext.

To decrypt the ciphertext C, use the private decryption key (d, n) and compute:

$$M \equiv C^d \pmod{n}.$$

Figure 3.2 illustrates the encryption and decryption process.

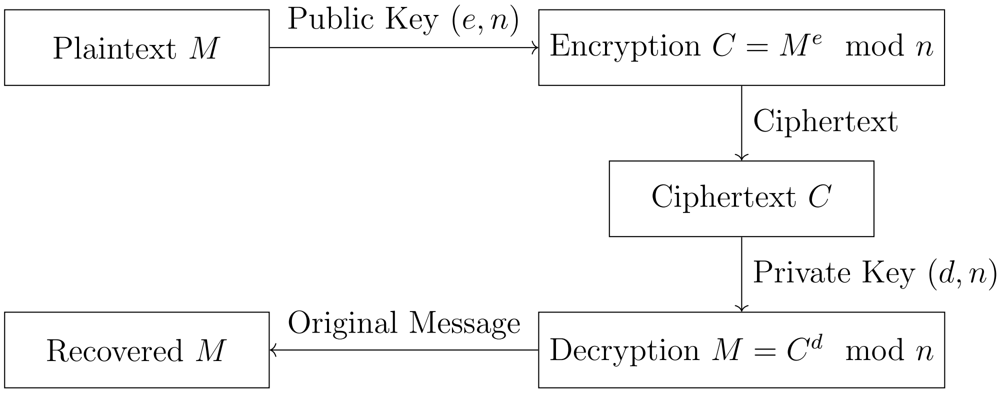


Figure 3.2: Diagram illustrating the RSA encryption and decryption process.

3.4 Example: Encrypting a Message Using RSA

To illustrate the RSA encryption process, consider the following example:

3.4.1 Key Generation

- Choose two prime numbers: $p = 17$ and $q = 19$.
- Compute $n = p \times q = 17 \times 19 = 323$.
- Calculate the totient: $\phi(n) = (p-1)(q-1) = 16 \times 18 = 288$.
- Select a public exponent $e = 5$, such that $1 < e < \phi(n)$ and $\gcd(e, \phi(n)) = 1$.
- Compute the private exponent d, satisfying $e \cdot d \equiv 1 \pmod{\phi(n)}$. Using the extended Euclidean algorithm, $d = 173$.

The public key is $(e, n) = (5, 323)$, and the private key is $(d, n) = (173, 323)$.

3.4.2 Encryption of the Message

To encrypt the message `Hello`, convert each character to its ASCII equivalent:

- `H` = 72, `e` = 101, `l` = 108, `o` = 111.

Message is represented as: `[72, 101, 108, 108, 111]`.

3.4.3 Encryption of the Message

To encrypt the message `Hello`, convert each character to its ASCII equivalent:

- `H` = 72, `e` = 101, `l` = 108, `o` = 111.

Encrypt each character M using the formula:

$$C \equiv M^e \pmod{n}.$$

- For `H`: $C \equiv 72^5 \pmod{323} = 21$.
- For `e`: $C \equiv 101^5 \pmod{323} = 271$.
- For `l`: $C \equiv 108^5 \pmod{323} = 109$.
- For `o`: $C \equiv 111^5 \pmod{323} = 42$.

The encrypted message is represented as:

Ciphertext: `[21, 271, 109, 109, 42]`.

The corresponding encrypted text is:

Encrypted Message: `§ďmm*`.

3.4.4 Decryption of the Message

To decrypt the ciphertext, use the private key $(d, n) = (173, 323)$ and the formula:

$$M \equiv C^d \pmod{n}.$$

- For $C = 21$: $M \equiv 21^{173} \pmod{323} = 72$ (`H`).
- For $C = 271$: $M \equiv 271^{173} \pmod{323} = 101$ (`e`).
- For $C = 109$: $M \equiv 109^{173} \pmod{323} = 108$ (`l`).
- For $C = 42$: $M \equiv 42^{173} \pmod{323} = 111$ (`o`).

The decrypted message is `Hello`, demonstrating the correctness of the RSA encryption and decryption process.

3.5 Mathematical Proof of Correctness

The correctness of the decryption process is guaranteed by Euler's theorem, which states:

$$M^{\phi(n)} \equiv 1 \pmod{n},$$

for any integer M coprime to n. Here, $\phi(n)$ is the Euler totient function, which gives the number of positive integers less than n that are relatively prime to n. For prime numbers p, we have:

$$\phi(p) = p - 1.$$

In the case of RSA, where $n = p \cdot q$ (the product of two primes), the totient function is given by:

$$\phi(n) = \phi(p) \cdot \phi(q) = (p - 1) \cdot (q - 1).$$

Since d is chosen such that it is relatively prime to $\phi(n)$, it has a multiplicative inverse e modulo $\phi(n)$, satisfying:

$$e \cdot d \equiv 1 \pmod{\phi(n)}.$$

To prove the correctness of decryption, consider the encryption and decryption processes:

$$D(E(M)) \equiv (E(M))^d \equiv (M^e)^d \pmod{n} = M^{e \cdot d} \pmod{n}.$$

Using the property $e \cdot d = k \cdot \phi(n) + 1$ for some integer k, we can rewrite:

$$M^{e \cdot d} = M^{k \cdot \phi(n) + 1} \equiv M \pmod{n}.$$

This follows from Euler's theorem, which ensures that $M^{\phi(n)} \equiv 1 \pmod{n}$ for any M coprime to n. Thus, the decryption process correctly recovers the original message M.

Furthermore, for all M, including cases where M is divisible by p or q, the above equality holds due to the Chinese Remainder Theorem. Therefore, the encryption and decryption functions E and D are inverse permutations, ensuring the correctness of RSA [1].

3.6 Very Large Primes in RSA

The security of RSA encryption heavily relies on the use of very large prime numbers. These primes are chosen to ensure that the modulus $n = p \times q$ is sufficiently large, making factorization computationally infeasible for classical computers.

3.6.1 Properties of Large Primes

Very large primes used in RSA have the following properties:

- **Size:** Typically, primes are hundreds of digits long (e.g., 1024-bit, 2048-bit, or even larger primes in modern implementations). For example, a 200-digit n provides a strong margin of safety against classical attacks, while shorter lengths like 80 digits offer only moderate security.

- **Randomness:** Primes are generated randomly to avoid predictability, ensuring that attackers cannot guess or precompute potential primes.

- **Certainty:** Probabilistic primality tests, such as the Miller-Rabin test or the Baillie-PSW primality test, are used to ensure the numbers are prime with high confidence. Deterministic tests like the AKS primality test may also be used for absolute certainty, though they are computationally more expensive.

3.6.2 Generation of Large Primes

The process of generating large primes involves the following steps:

1. **Random Selection:** Randomly select a large odd number within the desired range (e.g., 1024-bit or 2048-bit range).

2. **Primality Testing:** Test the number for primality using efficient algorithms:

    - **Miller-Rabin Test:** A probabilistic test that can quickly identify composite numbers. Multiple iterations increase the confidence level.
    - **Lucas-Lehmer Test:** Often used for specific types of primes, such as Mersenne primes.
    - **AKS Primality Test:** A deterministic test that guarantees correctness but is slower than probabilistic methods.

3. **Iteration:** If the number fails the primality test, repeat the process with a new candidate until a prime is found.

3.6.3 Challenges with Large Primes

While the use of very large primes is essential for RSA security, it introduces several challenges:

- **Computational Cost:** Generating and verifying large primes is computationally expensive, especially for key sizes of 4096 bits or more.

- **Storage and Transmission:** Large primes require more storage space and bandwidth for secure transmission, increasing the overhead in cryptographic systems.
- **Quantum Threats:** Despite the use of very large primes, RSA remains vulnerable to quantum algorithms like Shor's algorithm, which can efficiently factorize the modulus n and compromise the encryption.

3.6.4 Importance of Large Primes in RSA

The use of very large primes is a cornerstone of RSA encryption, ensuring its robustness against classical attacks. The difficulty of factoring the product of two large primes underpins the security of RSA, making it a reliable cryptographic standard for decades. However, the emergence of quantum computing necessitates the development of quantum-resistant alternatives to address the vulnerabilities posed by quantum algorithms.

3.7 Python Implementation of RSA Encryption and Decryption

The following Python code demonstrates the RSA encryption and decryption process using two different approaches:

3.7.1 Generic RSA Implementation

Listing 3.1 provides a generic implementation of RSA encryption and decryption in Python. It includes functions for key generation, encryption, and decryption, showcasing the mathematical principles of RSA.

- The `generate_keys` function generates public and private keys using two prime numbers.
- The `encrypt` function encrypts a plaintext message using the public key.
- The `decrypt` function decrypts the ciphertext using the private key.

This implementation demonstrates the core concepts of RSA and is suitable for educational purposes.

```python
import random  # Random module for key generation
from math import gcd  # GCD function for coprime check
from sympy import mod_inverse  # Modular inverse calculation
# Function to compute modular exponentiation
def mod_exp(base, exp, mod):
    result = 1
    while exp > 0:
        if exp % 2 == 1:  # If exponent is odd
            result = (result * base) % mod
        base = (base * base) % mod
        exp //= 2
    return result
# RSA Key Generation
def generate_keys(p, q):
    n, phi = p * q, (p - 1) * (q - 1)
    e = random.randrange(2, phi)
    while gcd(e, phi) != 1:
        e = random.randrange(2, phi)
    d = mod_inverse(e, phi)
    return (e, n), (d, n)  # Public key, Private key

# RSA Encryption
def encrypt(message, public_key):
    e, n = public_key
    return [mod_exp(ord(char), e, n) for char in message]

# RSA Decryption
def decrypt(ciphertext, private_key):
    d, n = private_key
    return ''.join([chr(mod_exp(char, d, n)) for char in ciphertext])
\newpage
# Example Usage
p = int(input("Enter a prime number p: "))
q = int(input("Enter a prime number q: "))
public_key, private_key = generate_keys(p, q)

message = "HELLO"
print("Original Message:", message)
print("Encrypted Message:", encrypt(message, public_key))
print("Decrypted Message:", decrypt(ciphertext, private_key))
```

Listing 3.1: RSA Encryption and Decryption Example (Python 3.8+)

3.7.2 Cryptography Library-Based Implementation

Listing 3.2 demonstrates an RSA implementation using the `cryptography` library in Python. This approach leverages built-in functions for key generation, encryption, and decryption, providing a more secure and efficient solution. For more details, refer to the official documentation at cryptography.

- The `generate_rsa_keys` function generates RSA key pairs and serializes them in PEM format.
- The `encrypt_message` function encrypts a message using the public key and OAEP padding.
- The `decrypt_message` function decrypts the ciphertext using the private key and OAEP padding.

This implementation is suitable for real-world applications, as it adheres to modern cryptographic standards.

Library Installation

To use the `cryptography` library, install it using the following command:

```
pip install cryptography
```

Advantages of Using the `cryptography` Library

- **Security:** The library adheres to modern cryptographic standards, ensuring robust security for encryption and decryption.
- **Ease of Use:** Provides high-level APIs for common cryptographic operations, reducing the complexity of implementation.
- **Performance:** Optimized for performance, making it suitable for real-world applications.
- **Community Support:** Actively maintained and widely used, ensuring reliability and access to community resources.

```python
from cryptography.hazmat.primitives.asymmetric import rsa, padding
from cryptography.hazmat.primitives import serialization, hashes

def generate_rsa_keys() -> tuple[bytes, bytes]:
    '''
    Generate RSA public and private keys.

    Returns:
        tuple[bytes, bytes]: A tuple containing the private key and pub
    '''
    private_key = rsa.generate_private_key(
        public_exponent=65537,
        key_size=2048
    )
    public_key = private_key.public_key()
    return (
        private_key.private_bytes(
            encoding=serialization.Encoding.PEM,
            format=serialization.PrivateFormat.PKCS8,
            encryption_algorithm=serialization.NoEncryption()
        ),
        public_key.public_bytes(
            encoding=serialization.Encoding.PEM,
            format=serialization.PublicFormat.SubjectPublicKeyInfo
        )
    )
def encrypt_message(public_key_pem: bytes, message: bytes) -> bytes:
    '''
    Encrypt a message using an RSA public key.

    Args:
        public_key_pem (bytes): The public key in PEM format.
        message (bytes): The plaintext message to encrypt.

    Returns:
        bytes: The encrypted ciphertext.
    '''
    public_key = serialization.load_pem_public_key(public_key_pem)
    return public_key.encrypt(
        message,
        padding.OAEP(
            mgf=padding.MGF1(algorithm=hashes.SHA256()),
            algorithm=hashes.SHA256(),
            label=None
        )
    )

def decrypt_message(private_key_pem: bytes, ciphertext: bytes) -> bytes
    '''
    Decrypt a ciphertext using an RSA private key.

    Args:
        private_key_pem (bytes): The private key in PEM format.
        ciphertext (bytes): The encrypted ciphertext to decrypt.

    Returns:
        bytes: The decrypted plaintext message.
    '''
    private_key = serialization.load_pem_private_key(private_key_pem, p
    return private_key.decrypt(
        ciphertext,
        padding.OAEP(
            mgf=padding.MGF1(algorithm=hashes.SHA256()),
            algorithm=hashes.SHA256(),
            label=None
        )
    )

# Example usage
private_key_pem, public_key_pem = generate_rsa_keys()
message = b"Hello, Quantum World!"
ciphertext = encrypt_message(public_key_pem, message)
decrypted_message = decrypt_message(private_key_pem, ciphertext)

# Print results
print("Original Message:", message)
print("Encrypted Message:", ciphertext)
```

```python
print("Decrypted Message:", decrypted_message)
```

Listing 3.2: Python code for RSA encryption and decryption

3.8 Vulnerabilities of RSA

While RSA is secure against classical attacks with sufficiently large key sizes, it has inherent vulnerabilities:

- **Small Key Sizes:** RSA keys smaller than 2048 bits are susceptible to factorization using advanced classical algorithms.
- **Implementation Flaws:** Poor implementation practices, such as weak random number generation, can compromise RSA security.
- **Side-Channel Attacks:** Physical attacks exploiting timing, power, or electromagnetic emissions can reveal private keys.
- **Quantum Threats:** The advent of quantum computing introduces algorithms, such as Shor's algorithm, that can efficiently factorize large integers, rendering RSA insecure.

3.9 Mitigation Strategies

To address these vulnerabilities, several strategies have been proposed, as depicted in Figure 3.3:

- **Increasing Key Sizes:** Using larger key sizes (e.g., 4096 bits) to enhance resistance against classical attacks.
- **Implementing Countermeasures:** Employing constant-time algorithms and secure random number generators to prevent side-channel attacks.
- **Transitioning to Quantum-Resistant Cryptography:** Developing and adopting cryptographic systems that are secure against quantum attacks.

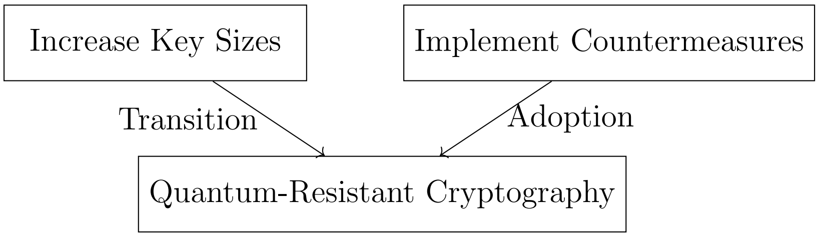


Figure 3.3: Diagram illustrating mitigation strategies for RSA vulnerabilities.

Chapter 4

Quantum Mechanics and Its Role in Quantum Computing

Quantum mechanics is the branch of physics that describes the behavior of matter and energy at the smallest scales, such as atoms and subatomic particles. It forms the theoretical foundation of quantum computing, enabling the development of algorithms that outperform classical counterparts for specific problems.

4.1 Principles of Quantum Mechanics

Quantum mechanics is governed by several key principles that distinguish it from classical physics:

4.1.1 Quantization of Physical Properties

Quantization is a fundamental concept in quantum mechanics, where certain physical properties, such as energy, angular momentum, and charge, can only take on discrete values. This is in contrast to classical mechanics, where these properties are continuous.

Energy Quantization

In quantum systems, energy levels are quantized, meaning that a particle can only occupy specific energy states. For example, the energy levels of an electron in a hydrogen atom are given by:

$$E_n = -\frac{13.6\,\text{eV}}{n^2},$$

where n is a positive integer representing the principal quantum number.

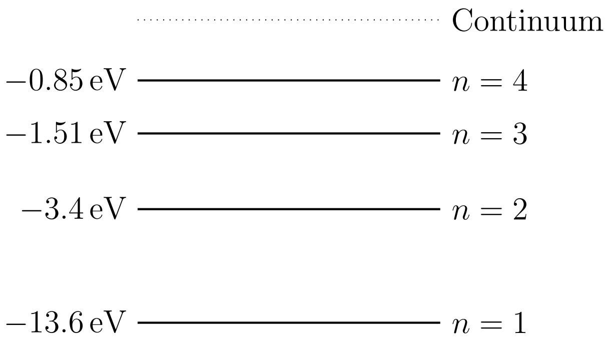


Figure 4.1: Discrete energy levels of an electron in a hydrogen atom.

Angular Momentum Quantization

The angular momentum of a particle is also quantized. For example, the magnitude of the angular momentum of an electron in an atom is given by:

$$L = \sqrt{\ell(\ell + 1)}\hbar,$$

where ℓ is the orbital quantum number, and $\hbar$ is the reduced Planck's constant.

Charge Quantization

Electric charge is quantized in units of the elementary charge e. For example, the charge of an electron is $-e$, and the charge of a proton is $+e$.

Quantization is a cornerstone of quantum mechanics, providing the framework for understanding the discrete nature of physical phenomena at microscopic scales.

4.1.2 Wave-Particle Duality

Wave-particle duality is a fundamental concept in quantum mechanics, stating that quantum entities, such as electrons and photons, exhibit both wave-like and particle-like behavior depending on the experimental setup. For example, in the double-slit experiment, particles like electrons create an interference pattern when not observed, demonstrating wave-like behavior. However, when observed, they behave like particles, striking specific locations on a detector.

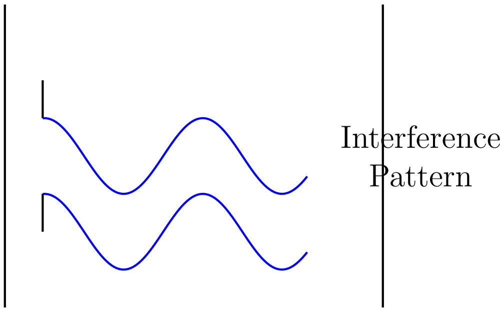


Figure 4.2: Wave-particle duality demonstrated in the double-slit experiment.

4.1.3 Uncertainty Principle (Heisenberg's Principle)

The uncertainty principle, formulated by Werner Heisenberg, is a fundamental concept in quantum mechanics. It states that certain pairs of physical properties, such as position (x) and momentum (p), cannot be simultaneously measured with arbitrary precision. Mathematically, the principle is expressed as:

$$\Delta x \cdot \Delta p \geq \frac{\hbar}{2},$$

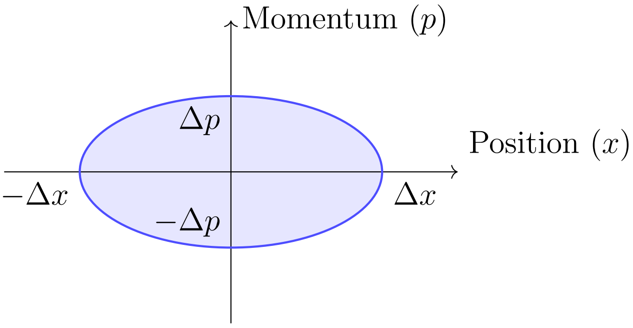


Figure 4.3: Graphical representation of the uncertainty principle. The softly shaded ellipse illustrates the trade-off between uncertainties in position (x) and momentum (p).

Implications of the Uncertainty Principle

The uncertainty principle has profound implications for quantum mechanics:

- **Wave-Particle Duality:** It highlights the wave-like nature of particles, as the position and momentum cannot be precisely defined simultaneously.
- **Quantum Systems:** It imposes fundamental limits on the precision of measurements in quantum systems, influencing the design of experiments and technologies like quantum computing.
- **Non-Deterministic Nature:** It underscores the probabilistic nature of quantum mechanics, where outcomes are described by probabilities rather than certainties.

4.1.4 Wavefunction and Schrödinger's Equation

The wavefunction $|\psi\rangle$ describes the quantum state of a system and contains all the information about the system. The time evolution of the wavefunction is governed by the Schrödinger equation:

$$i\hbar\frac{\partial}{\partial t}|\psi(t)\rangle = \hat{H}|\psi(t)\rangle,$$

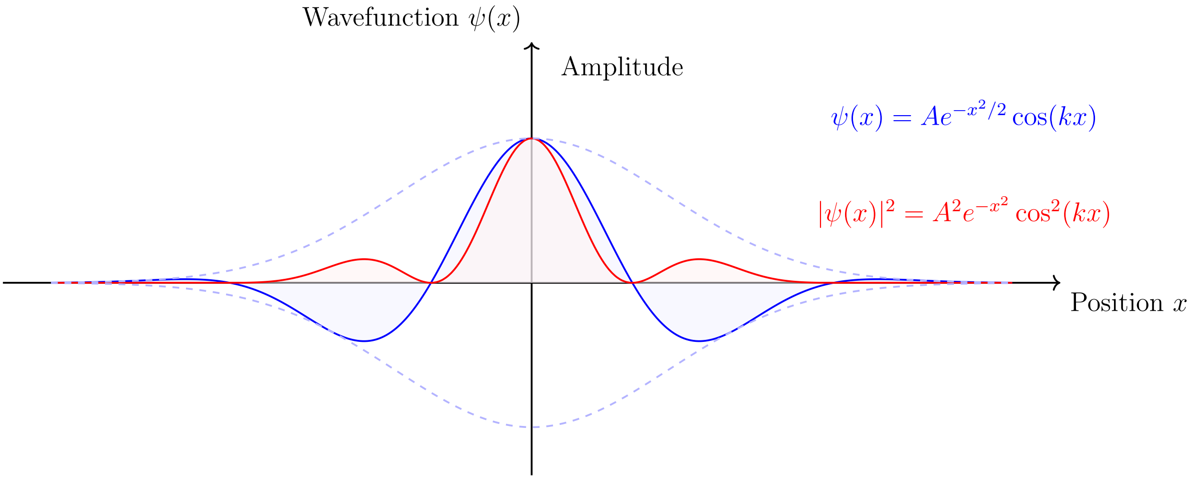


Figure 4.4: High-resolution wavefunction $\psi(x)$ with a Gaussian envelope modulated by a cosine wave, and its probability density $|\psi(x)|^2$ filled.

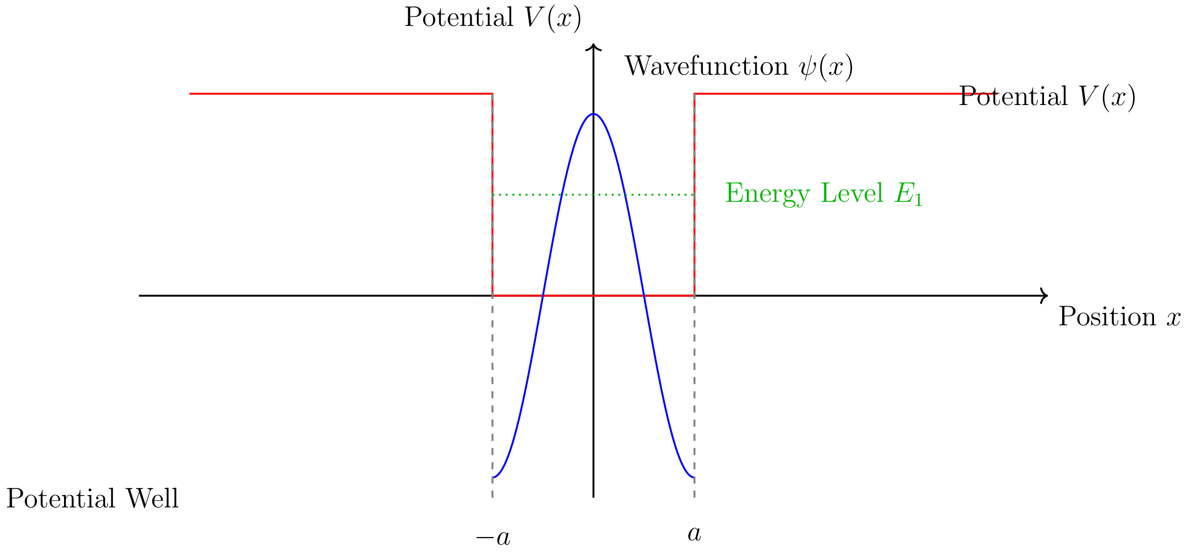


Figure 4.5: High-resolution wavefunction $\psi(x)$ in a potential well $V(x)$.

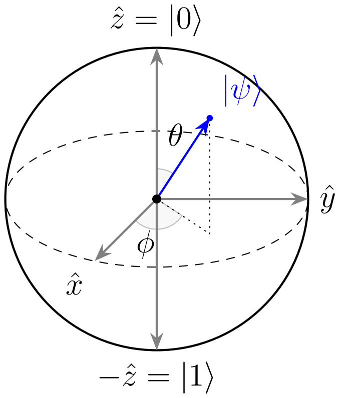
Solving the Schrödinger equation provides the wavefunction $|\psi(t)\rangle$ at any given time, enabling the prediction of quantum behavior.

4.1.5 Superposition Principle

Superposition is a fundamental principle of quantum mechanics that allows quantum systems to exist in multiple states simultaneously. In the context of quantum computing, a quantum bit (qubit) can represent both $|0\rangle$ and $|1\rangle$ at the same time, unlike classical bits that can only represent one state at a time. Mathematically, a qubit in superposition is represented as:

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

where α and β are complex numbers satisfying $|\alpha|^2 + |\beta|^2 = 1$. This property enables quantum computers to process a vast number of possibilities simultaneously, leading to exponential speedups for certain types of problems.



$$|\psi\rangle = \cos\left(\frac{\theta}{2}\right)|0\rangle + e^{i\phi}\sin\left(\frac{\theta}{2}\right)|1\rangle$$

Figure 4.6: Representation of a qubit on the Bloch sphere, showing the state $|\psi\rangle = \cos\left(\frac{\theta}{2}\right)|0\rangle + e^{i\phi}\sin\left(\frac{\theta}{2}\right)|1\rangle$ with angles θ and ϕ. The vector $|\psi\rangle$ is shown in blue, illustrating that the atom is in a superposition of states $|0\rangle$ and $|1\rangle$ simultaneously.

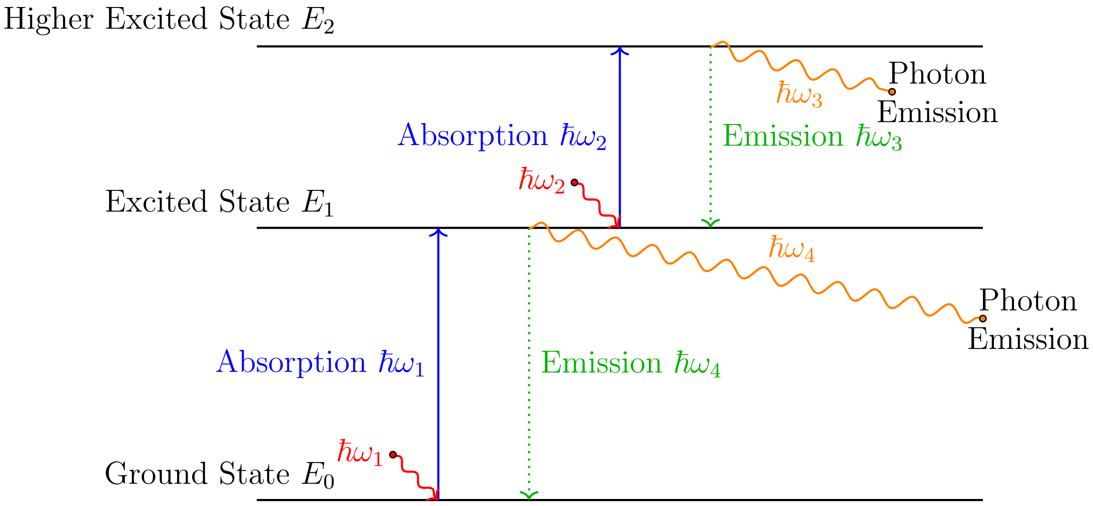


Figure 4.7: Electron excitation and relaxation: The electron transitions from the ground state E_0 to the excited state E_1 upon absorbing a photon of energy $\hbar\omega_1$, and further to a higher excited state E_2 upon absorbing another photon of energy $\hbar\omega_2$. The relaxation process involves the emission of photons $\hbar\omega_3$ and $\hbar\omega_4$.

4.1.6 Quantum Entanglement

Entanglement is a unique quantum phenomenon where two or more quantum particles become interconnected in such a way that the state of one particle is directly related to the state of the other, regardless of the distance between them. For example, the entangled state of two qubits can be represented as:

$$|\psi\rangle = \frac{1}{\sqrt{2}}\left(|00\rangle + |11\rangle\right).$$

This correlation persists even if the particles are separated by vast distances. In quantum computing, entanglement is used to create highly efficient communication and computation protocols.

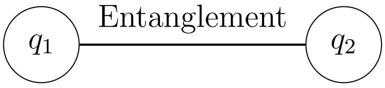


Figure 4.8: Illustration of entangled qubits q_1 and q_2.

4.1.7 Quantum Interference

Quantum interference arises from the wave-like nature of quantum particles. When quantum states overlap, their probability amplitudes can interfere constructively or destructively. Constructive interference amplifies the probability of correct solutions, while destructive interference cancels out incorrect ones. For example, in a quantum algorithm, the interference can be represented as:

$$|\psi\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) + \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) = |0\rangle.$$

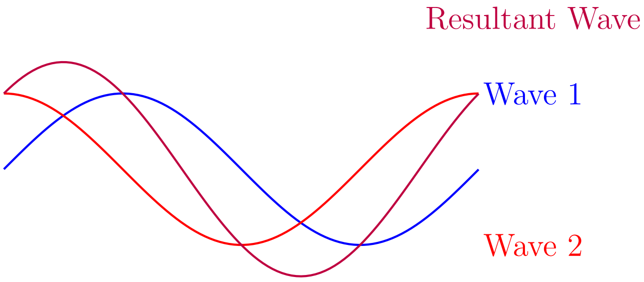


Figure 4.9: Constructive interference of quantum states.

4.1.8 Measurement and Wavefunction Collapse

Measurement in quantum mechanics is the process of observing a quantum system, which causes the system to collapse into a definite state. Before measurement, a quantum system exists in a superposition of all possible states, each with a certain probability amplitude. Upon measurement, the system randomly collapses into one of these states. For example, measuring the state:

$$|\psi\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle,$$

will yield $|0\rangle$ or $|1\rangle$ with equal probability $\frac{1}{2}$.

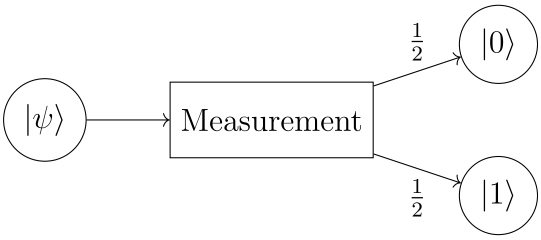


Figure 4.10: Measurement collapsing a qubit into a definite state.

4.1.9 Quantum Tunneling

Quantum tunneling is a phenomenon where a quantum particle can pass through a potential energy barrier, even if it does not have enough energy to overcome it classically. This is due to the wave-like nature of quantum particles, which allows their wave functions to extend into regions that are classically forbidden. For example, the probability of tunneling through a barrier can be represented as:

$$T \approx e^{-2\kappa L},$$

where T is the transmission probability, κ is a constant related to the barrier height, and L is the barrier width.

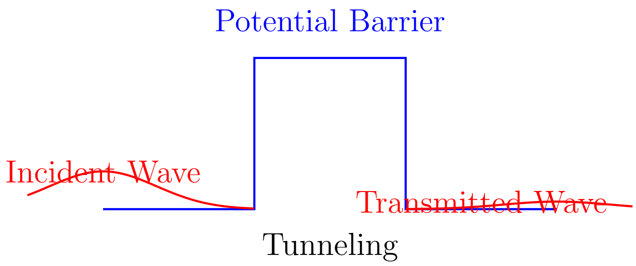


Figure 4.11: Quantum tunneling through a potential barrier.

4.2 Mathematical Framework of Quantum Mechanics

The mathematical framework of quantum mechanics is based on linear algebra and complex vector spaces. The key notations and concepts include:

- **Schrödinger Equation:** The time evolution of a quantum system is governed by the Schrödinger equation:
$$i\hbar\frac{\partial}{\partial t}|\psi(t)\rangle = \hat{H}|\psi(t)\rangle,$$
where $\hat{H}$ is the Hamiltonian operator representing the total energy of the system.

- **Probability Amplitudes:** The probability of measuring a particular outcome is given by the square of the amplitude of the corresponding state vector component. For example, the probability of measuring $|0\rangle$ in the state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ is $|\alpha|^2$.

- **State Vectors:** The state of a quantum system is represented by a vector in a complex Hilbert space, denoted as $|\psi\rangle$. For example, a single qubit state can be written as:
$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle,$$
where $\alpha, \beta \in \mathbb{C}$ and $|\alpha|^2 + |\beta|^2 = 1$.

- **Bra-Ket Notation:** Quantum states are expressed using Dirac's bra-ket notation:
  - $|\psi\rangle$: A column vector (ket) representing the state.
  - $\langle\psi|$: A row vector (bra), the Hermitian conjugate of $|\psi\rangle$.

The inner product of two states $|\psi\rangle$ and $|\phi\rangle$ is written as $\langle\psi|\phi\rangle$, and the outer product is $|\psi\rangle\langle\phi|$.

- **Operators:** Physical observables, such as position and momentum, are represented by Hermitian operators acting on state vectors. For example, the Pauli-X operator is:
$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}.$$
- **Tensor Products:** Multi-qubit systems are represented using tensor products. For example, the state of two qubits $|\psi_1\rangle$ and $|\psi_2\rangle$ is:
$$|\psi\rangle = |\psi_1\rangle \otimes |\psi_2\rangle.$$
- **Unitary Operators:** Quantum gates are represented by unitary matrices U, satisfying $U^\dagger U = I$, where $U^\dagger$ is the conjugate transpose of U and I is the identity matrix.

4.3 Quantum Gates and Circuits

Quantum gates are the building blocks of quantum circuits, analogous to classical logic gates. They manipulate qubits and perform operations based on the principles of quantum mechanics:

4.3.1 Single-Qubit Gates

Single-qubit gates operate on a single qubit and are represented by 2×2 unitary matrices. Examples include:

- **Pauli-X Gate:** Acts as a quantum NOT gate, flipping the state of a qubit. Its matrix representation is:
$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}.$$
- **Hadamard Gate:** Creates a superposition of states. Its matrix representation is:
$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}.$$
- **Phase Gate:** Introduces a phase shift. For example, the S-gate is:
$$S = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}.$$

4.3.2 Multi-Qubit Gates

Multi-qubit gates operate on multiple qubits and are essential for creating entanglement and performing complex quantum operations:

- **CNOT Gate:** A controlled-NOT gate flips the target qubit if the control qubit is $|1\rangle$. Its matrix representation is:

$$\text{CNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$

- **Toffoli Gate:** A controlled-controlled-NOT gate, which flips the target qubit if both control qubits are $|1\rangle$.

4.3.3 Quantum Circuits

A quantum circuit is a sequence of quantum gates applied to qubits to perform a computation. Mathematically, the overall operation of a quantum circuit is represented by the product of the unitary matrices of its gates. The output is obtained by measuring the qubits, collapsing their states into classical bits.

4.4 Relevance to Quantum Computing

Quantum mechanics provides the theoretical underpinnings for quantum computing, enabling the development of powerful algorithms. For example:

- **Shor's Algorithm:** Efficiently factors integers by leveraging quantum Fourier transforms.
- **Grover's Algorithm:** Provides a quadratic speedup for unstructured search problems.

By leveraging principles such as superposition, entanglement, and interference, quantum computers can solve problems that are intractable for classical computers.

4.5 Challenges in Quantum Mechanics for Computing

Despite its potential, quantum mechanics introduces challenges for quantum computing:

- **Decoherence:** Quantum states are fragile and can lose coherence due to interactions with the environment, leading to errors in computation.
- **Error Correction:** Quantum error correction codes, such as the surface code, are required to mitigate the effects of noise and decoherence. These codes rely on encoding logical qubits into multiple physical qubits.
- **Scalability:** Building large-scale quantum computers with many qubits while maintaining coherence and low error rates is a complex engineering problem.

This chapter establishes the connection between quantum mechanics and quantum computing, highlighting the transformative potential of this interdisciplinary field and the challenges that must be overcome to realize its full potential.

Chapter 5

Existing Methods for Breaking RSA Before Quantum Computing

Before the advent of quantum computing, several classical methods were explored to break RSA encryption. These methods, while not as efficient as quantum algorithms, posed potential threats under certain conditions. This chapter discusses the existing systems and techniques used to attack RSA encryption prior to the emergence of quantum computing.

5.1 Brute Force Attacks

Brute force attacks involve systematically trying all possible private keys to decrypt a message. However, due to the large key sizes used in RSA (typically 2048 bits or more), brute force attacks are computationally infeasible with classical computers.

5.2 Mathematical Factorization Algorithms

The security of RSA relies on the difficulty of factoring large integers. Several classical algorithms have been developed to factorize numbers, including:

- **Trial Division:** A basic method that tests divisibility by smaller numbers, but it is highly inefficient for large integers.
- **Pollard's Rho Algorithm:** A probabilistic algorithm that works well for smaller numbers but struggles with large RSA keys.
- **Quadratic Sieve:** One of the fastest classical algorithms for factoring integers up to 100 digits.
- **General Number Field Sieve (GNFS):** The most efficient classical algorithm for factoring large integers, used in practical attacks on RSA with smaller key sizes.

5.3 Side-Channel Attacks

Side-channel attacks exploit physical implementations of RSA rather than its mathematical foundation. Examples include:

- **Timing Attacks:** Measuring the time taken for cryptographic operations to infer private keys.
- **Power Analysis:** Observing power consumption patterns during encryption or decryption.
- **Electromagnetic Analysis:** Capturing electromagnetic emissions to extract sensitive information.

5.4 Distributed Computing Efforts

Projects like the RSA Factoring Challenge encouraged distributed computing efforts to factorize RSA keys. While these efforts demonstrated the vulnerability of smaller key sizes, they were not practical for breaking modern RSA implementations with sufficiently large keys.

5.5 Limitations of Classical Methods

Despite these methods, breaking RSA encryption with classical computers remains infeasible for adequately large key sizes. The computational resources and time required grow exponentially with key size, ensuring the security of RSA against classical attacks.

While RSA encryption has been a reliable cryptographic standard, it is not without limitations. The primary challenge lies in its reliance on the difficulty of integer factorization. With the advent of quantum computing, this foundational assumption is under threat, as quantum algorithms like Shor's algorithm can efficiently solve the integer factorization problem.

Chapter 6

Drawbacks of Existing Methods

While existing methods for breaking RSA encryption have been extensively studied, they come with significant drawbacks that limit their practicality and effectiveness. This chapter discusses the key limitations of these methods:

6.1 Brute Force Attacks

- **Exponential Time Complexity:** The time required to test all possible keys grows exponentially with key size, making brute force attacks infeasible for modern RSA implementations.
- **Resource Intensive:** Brute force attacks require substantial computational resources, which are often unavailable or impractical to deploy.

6.2 Mathematical Factorization Algorithms

- **Inefficiency for Large Keys:** Classical factorization algorithms, such as the General Number Field Sieve (GNFS), become increasingly inefficient as the size of the RSA key increases.
- **High Computational Cost:** These algorithms demand significant computational power and time, limiting their applicability to smaller key sizes.

6.3 Side-Channel Attacks

- **Dependency on Physical Access:** Side-channel attacks often require physical access to the cryptographic device, which is not always feasible.
- **Mitigation Techniques:** Modern cryptographic implementations include countermeasures to reduce the effectiveness of side-channel attacks, such as constant-time algorithms and noise injection.

6.4 Distributed Computing Efforts

- **Scalability Issues:** Distributed computing efforts face challenges in scaling to factorize larger RSA keys due to the exponential growth in computational requirements.
- **Coordination Overhead:** Managing and coordinating large-scale distributed systems introduces additional complexity and overhead.

6.5 General Limitations

- **Reliance on Classical Computing:** All existing methods are constrained by the limitations of classical computing, which cannot efficiently solve the integer factorization problem for large key sizes.
- **Inability to Address Quantum Threats:** These methods do not account for the advancements in quantum computing, which pose a more significant threat to RSA encryption.

These drawbacks highlight the need for more advanced approaches, such as quantum-resistant cryptographic systems, to address the vulnerabilities of RSA encryption in the face of evolving computational capabilities.

Chapter 7

Shor's Algorithm and Its Impact on RSA

Shor's algorithm is a quantum algorithm that can factorize large integers exponentially faster than classical algorithms. This chapter delves into the workings of Shor's algorithm and explains how it directly undermines the security of RSA encryption, making it vulnerable to quantum attacks.

Chapter 8

Implications of Quantum Computing on Cryptography

The ability of quantum computers to break RSA encryption has far-reaching implications for cryptography. This chapter discusses the potential risks to digital security, including compromised financial transactions, data breaches, and threats to national security, emphasizing the urgency of addressing these challenges.

Chapter 9

Advantages of Proposed System

The proposed quantum-resistant cryptographic systems offer several advantages:

- Enhanced security against quantum attacks, ensuring the confidentiality of sensitive information.
- Compatibility with existing communication protocols, allowing for a smoother transition to quantum-resistant systems.
- Scalability for future cryptographic needs, addressing the growing demand for secure communication in the quantum era.
- Reduced risk of data breaches and financial fraud caused by quantum-enabled attacks.

Chapter 10

Application

Quantum-resistant cryptographic systems have a wide range of applications, including:

- Securing financial transactions and online banking systems.
- Protecting sensitive government and military communications.
- Ensuring the privacy of personal data in healthcare and other industries.
- Safeguarding intellectual property and trade secrets in the corporate sector.
- Enabling secure communication in emerging technologies such as the Internet of Things (IoT) and autonomous vehicles.

Chapter 11

Future Enhancement and Research Directions in Quantum Computing

The field of quantum computing is rapidly evolving, with ongoing research and development efforts to overcome existing challenges and enhance the capabilities of quantum systems.

11.1 Current State of Quantum Computing Technology

The threat posed by quantum computing to RSA encryption is based on Shor's algorithm, which can efficiently factorize large integers. However, the practical realization of this threat depends on the current state of quantum computing technology. Several challenges limit the immediate impact of quantum computing on RSA encryption:

- **Qubit Stability:** Quantum computers require stable qubits to perform reliable computations. Current quantum systems face challenges in maintaining qubit coherence and stability over extended periods.

- **Error Rates:** Quantum operations are susceptible to errors due to noise and environmental factors. Error correction techniques are essential to mitigate these errors and ensure the accuracy of quantum computations.

- **Scalability:** Building large-scale quantum computers with thousands of qubits is a complex engineering feat. Scaling quantum systems while maintaining coherence and low error rates remains a significant challenge.

- **Error Correction:** Quantum error correction codes are essential to protect quantum information from errors. Implementing error correction at scale is a non-trivial task that requires significant computational resources.

- **Algorithmic Improvements:** While Shor's algorithm demonstrates the theoretical threat to RSA encryption, practical implementations require further algorithmic improvements and optimizations to factorize large integers efficiently.

While the theoretical threat posed by Shor's algorithm is significant, the current state of quantum computing technology is not yet advanced enough to break RSA encryption with commonly used key lengths like 2048 bits [8]. Current quantum computers are still in the early stages of development and face limitations in the number of stable and error-corrected qubits (quantum bits) they possess [13].

Estimates for the number of logical qubits required to break RSA-2048 vary, ranging from around 4,000 to 20 million, depending on the specific algorithm implementation and assumptions about error correction [14]. The largest number factored using Shor's algorithm on a quantum computer to date is 21 [8]. While there have been claims of breaking larger RSA keys using hybrid quantum-classical approaches with fewer qubits, these have generally involved significantly smaller key sizes (e.g., 22 bits) and do not yet pose a threat to standard RSA implementations [15].

11.2 Quantum-Resistant Cryptography

Quantum-resistant cryptography, also known as post-quantum cryptography, aims to develop cryptographic algorithms that are secure against both classical and quantum computers. These algorithms are based on mathematical problems that are believed to be resistant to quantum attacks, such as lattice-based cryptography, hash-based cryptography, code-based cryptography, and multivariate polynomial cryptography.

11.2.1 Standardization Efforts

Organizations like NIST (National Institute of Standards and Technology) are actively working on standardizing quantum-resistant algorithms. The NIST Post-Quantum Cryptography Standardization project has identified several candidate algorithms for public-key encryption, digital signatures, and key exchange protocols. These efforts aim to ensure global adoption and interoperability of quantum-resistant cryptographic systems.

11.2.2 Challenges and Research Directions

While quantum-resistant cryptography holds promise, several challenges remain:

- **Performance Overhead:** Quantum-resistant algorithms often require larger key sizes and computational resources compared to traditional cryptographic systems.
- **Security Analysis:** Extensive research is needed to analyze the security of proposed algorithms against both classical and quantum attacks.
- **Integration with Existing Systems:** Transitioning to quantum-resistant cryptography requires compatibility with existing communication protocols and infrastructure.

11.2.3 Future Enhancements

The advent of quantum computing poses a significant challenge to traditional cryptographic systems like RSA encryption. However, it also drives innovation in the field of cryptography, leading to the development of quantum-resistant algorithms. Future enhancements in this area include:

- Standardizing quantum-resistant algorithms to ensure global adoption and interoperability.
- Improving the efficiency and performance of quantum-resistant cryptographic systems to meet real-world demands.
- Conducting extensive research to identify and mitigate potential vulnerabilities in proposed systems.

Chapter 12

Conclusion

In conclusion, the transition to quantum-resistant cryptography is essential to mitigate the risks posed by quantum computing. By adopting these advanced systems, we can ensure the continued security of sensitive information and maintain trust in digital communication in the quantum era.

Bibliography

[1] Rivest, R., Shamir, A., & Adleman, L. (1978). *A Method for Obtaining Digital Signatures and Public-Key Cryptosystems*. Communications of the ACM, 21(2), 120–126. doi:10.1145/359340.359342. CiteSeerX: 10.1.1.607.2677. S2CID: 2873616. Archived from the original (PDF) on 2023–01–27. Available at: `http://people.csail.mit.edu/rivest/Rsapaper.pdf`.

[2] Okta. (2025). *RSA Encryption: Definition, Architecture, Benefits & Use*. Retrieved March 23, 2025, from `https://www.okta.com/identity-101/rsa-encryption/`.

[3] SecureW2. (2025). *Understanding RSA Asymmetric Encryption: How It Works*. Retrieved March 23, 2025, from `https://www.securew2.com/blog/what-is-rsa-asymmetric-encryption`.

[4] SoftwareDominos. (2025). *Understanding RSA: The Mathematics Behind Secure Encryption*. Retrieved March 23, 2025, from `https://softwaredominos.com/home/software-engineering-and-computer-science/understanding-rsa-the-mathematics-behind-secure-encryption/`.

[5] Wikipedia. (2025). *RSA problem*. Retrieved March 23, 2025, from `https://en.wikipedia.org/wiki/RSA_problem`.

[6] ODU Digital Commons. (2025). *The Vulnerabilities to the RSA Algorithm and Future Alternative Algorithms to Improve Security*. Retrieved March 23, 2025, from `https://digitalcommons.odu.edu/cgi/viewcontent.cgi?article=1081&context=covacci-undergraduateresearch`.

[7] Authorea. (2025). *Implementation and Analysis of Shor's Algorithm to Break RSA Cryptosystem Security*. Retrieved March 23, 2025, from `https://www.authorea.com/users/708580/articles/693052-implementation-and-analysis-of-shor-s-algorithm-to-break-rsa-cryptosystem-`

[8] SpinQ. (2025). *How Shor's Algorithm Breaks RSA: A Quantum Computing Guide*. Retrieved March 23, 2025, from `https://www.spinquanta.com/news-detail/shors-algorithm`.

[9] GeeksforGeeks. (2025). *RSA Algorithm in Cryptography*. Retrieved March 23, 2025, from `https://www.geeksforgeeks.org/rsa-algorithm-cryptography/`.

[10] Splunk. (2025). *RSA Algorithm in Cryptography: Rivest Shamir Adleman Explained*. Retrieved March 23, 2025, from `https://www.splunk.com/en_us/blog/learn/rsa-algorithm-cryptography.html`.

[11] TechRxiv. (2025). *Implementation and Analysis of Shor's Algorithm to Break RSA Cryptosystem Security*. Retrieved March 23, 2025, from https://www.techrxiv.org/users/708580/articles/693052/master/file/data/RESERCH%20FINAL/RESERCH%20FINAL.pdf.

[12] ResearchGate. (2025). *Implementation and Analysis of Shor's Algorithm to Break RSA Cryptosystem Security*. Retrieved March 23, 2025, from https://www.researchgate.net/publication/377245624_Implementation_and_Analysis_of_Shor's_Algorithm_to_Break_RSA_Cryptosystem_Security.

[13] Cato Networks. (2025). *Is the Recent Quantum Hype by Google Willow's Chip a Threat to RSA Algorithm?* Retrieved March 23, 2025, from https://www.catonetworks.com/blog/is-recent-quantum-hype-by-google-willows-chip-a-threat-to-rsa-algorithm/.

[14] BTQ. (2025). *How Far Away Is The Quantum Threat?* Retrieved March 23, 2025, from https://www.btq.com/blog/how-far-away-is-the-quantum-threat.

[15] Freemindtronic. (2025). *Quantum computing RSA encryption: a threat and a solution*. Retrieved March 23, 2025, from https://freemindtronic.com/quantum-computing-rsa-encryption-freemindtronic-nfc-technology/.

[16] Palo Alto Networks. (2025). *What Is Quantum Computing's Threat to Cybersecurity?* Retrieved March 23, 2025, from https://www.paloaltonetworks.com/cyberpedia/what-is-quantum-computings-threat-to-cybersecurity.

[17] IBM Community. (2025). *How many Number of qubits required for decryption*. Retrieved March 23, 2025, from https://community.ibm.com/community/user/ai-datascience/discussion/how-many-number-of-qubits-required-for-decryption.

[18] Sectigo. (2025). *Quantum computing concerns & positive impacts*. Retrieved March 23, 2025, from https://www.sectigo.com/resource-library/quantum-computing-concerns-positive-impacts.

[19] NIST Computer Security Resource Center. (2025). *Post-Quantum Cryptography*. Retrieved March 23, 2025, from https://csrc.nist.gov/projects/post-quantum-cryptography.

[20] Wikipedia. (2025). *NIST Post-Quantum Cryptography Standardization*. Retrieved March 23, 2025, from https://en.wikipedia.org/wiki/NIST_Post-Quantum_Cryptography_Standardization.

[21] Google Online Security Blog. (2024). *Post-Quantum Cryptography: Standards and Progress*. Retrieved March 23, 2025, from https://security.googleblog.com/2024/08/post-quantum-cryptography-standards.html.