The Effect of Quantum Computing on Cryptography

Mehaan Sibal

December 2023

1 Introduction to Cryptography

A basic definition of Crytogrpahy is a data based system of keys working on encrypting and decrytping data.

2 Asymmetric and Symmetric Algorithms

2.1 Asymmetric Algorithms

In terms of Asymmetric Algorithms a public key cryptography system is used. When sending a message the unencrypted message is called the plaintext. The plaintext is combined with a key which is imputed into an algorithm, encrypting the plain text giving the result of the ciphertext. The public key is made available to anyone who wants to communicate securely with the owner of the key pair. It can be freely distributed and shared with others. On the other hand, the private key is kept secret and known only to the owner of the key pair. The private key is used for decrypting messages that have been encrypted using the corresponding public key.

2.1.1 Asymmetric Key Algorithm Example

Looking at an example of an asymmetric key algorithms: To start, we need to generate two large prime numbers, p and q. These primes should be of roughly equal length and their product should be much larger than the message we want to encrypt. We can generate the primes using any primality testing algorithm, such as the Miller-Rabin test. Once we have the two primes, we can compute their product $n = p * q$, which will be the modulus for our system. Next, we need to choose an integer e such that

$$1 < e < \phi(n) \text{ and } gcd(e, \phi(n)) = 1$$

$$\text{where } \phi(n) = (p-1) * (q-1) \text{ is Euler's totient function.}$$

This value of e will be the public key exponent. To compute the private key exponent d, we need to find an integer d, such that

$$d * e \equiv 1 \mod \phi(n)$$

. This can be done using the extended Euclidean algorithm. Our public key is (n, e) and our private key is (n, d). Next we need Encryption. To encrypt a message m, we need to convert it to an integer between 0 and $n-1$. This can be done using a reversible encoding scheme, such as ASCII or UTF-8. Once we have the integer representation of the message, we compute the ciphertext c as

$$c \equiv m^e \mod n$$

This can be done efficiently using modular exponentiation algorithms, such as binary exponentiation.
Finally Decryption: To decrypt the ciphertext c, we compute the plaintext m as

$$m \equiv c^d \mod n$$

2.1.2 Mathematical Securities

The security of asymmetric key algorithms also relies on several mathematical properties:
One-Way Functions: These functions are easy to compute in one direction but computationally difficult to reverse. For example, it should be easy to calculate $y = f(x)$, but difficult to calculate x given y.
Trapdoor Functions: These are one-way functions that become easy to reverse if certain additional information, known as the trapdoor, is available. The private key acts as the trapdoor in asymmetric key algorithms.
Mathematical Infeasibility: The security of asymmetric key algorithms is

based on the mathematical infeasibility of certain problems, such as factoring large prime numbers or solving discrete logarithm problems. There are also several advantages to Asymmetric key algorithms over symmetric key algorithms. Asymmetric key algorithms enable secure communication between parties who have never met before and do not share a common secret key. Additionally, they facilitate secure key exchange protocols and digital signatures.

2.2 Symmetric Key Algorithms

Now talking about Symmetric algorithms, Symmetric key algorithms are a type of cryptographic technique that use the same secret key for both encryption and decryption. These algorithms are widely used in various applications, including secure communication protocols, digital signatures, and message authentication. In this answer, we will discuss the basics of symmetric key algorithms, how they work, and how they differ from asymmetric key algorithms.

2.2.1 How Symmetric Key Algorithms Work

How Symmetric Key Algorithms Work: A symmetric key algorithm consists of two main components: a key generator and a key distributor. The key generator creates a secret key, which is then distributed to the intended recipient using a secure channel. Once the recipient receives the secret key, they can use it to encrypt and decrypt messages. The encryption process using a symmetric key algorithm works as follows:

- The sender and the recipient agree on a symmetric key.
- The sender encrypts the message using the symmetric key.
- The recipient decrypts the message using the same symmetric key.
- The decryption process is the reverse of the encryption process, and it works as follows:
- The recipient encrypts the message using the symmetric key.
- The sender decrypts the message using the same symmetric key.

2.2.2 Symmetric Key Algorithms vs Asymmetric Key Algorithms

Asymmetric key algorithms, use a pair of keys: a public key for encryption and a private key for decryption. In contrast, symmetric key algorithms use a single secret key for both encryption and decryption. Other Key Differences:

1. Key Management: Symmetric key algorithms require secure key management, as the secret key must be kept confidential. Asymmetric key algorithms, on the other hand, use a public key for encryption and a private key for decryption, making key management easier.

2. Key Length: Symmetric key algorithms typically require longer keys than asymmetric key algorithms. This is because symmetric key algorithms rely on the strength of the key to secure the communication, whereas asymmetric key algorithms use the public key to encrypt and the private key to decrypt.

3. Performance: Symmetric key algorithms are generally faster and more efficient than asymmetric key algorithms, as they require fewer computational resources.

4. Key Exchange: Symmetric key algorithms require a secure key exchange mechanism to distribute the secret key between the sender and the recipient. Asymmetric key algorithms use a public key infrastructure (PKI) to exchange public keys.

2.2.3 Key Lengths importance to Symmetric and Asymmetric Algorithms

Key lengths are a fundamental aspect of cryptographic algorithms in both symmetric and asymmetric. They determine the size of the secret keys used in encryption and decryption processes. In cryptography, a key is a piece of information that is used to transform plaintext into ciphertext or vice versa. The length of a key refers to the number of bits it contains. The longer the key, the more possible combinations there are, making it harder for an attacker to guess or brute-force the key. With a longer key length, the search space for potential keys increases exponentially, making it computationally infeasible to break the encryption by trying all possible keys. For example, in the Advanced Encryption Standard (AES) algorithm, key lengths of 128, 192, and 256 bits are supported, we will talk more about this later. The security of asymmetric algorithms relies on mathematical problems that are difficult to solve without knowledge of certain secret parameters. The length of the keys used in asymmetric cryptography is typically much larger than those used in symmetric cryptography. This is because asymmetric algorithms are generally slower than symmetric ones and require more computational resources. Longer keys provide increased security against attacks such as brute-force or exhaustive search. For instance, in the RSA (Rivest-Shamir-Adleman) algorithm, a typical key length ranges from 1024 to 4096 bits. The larger the RSA key size, the stronger the encryption and the more difficult it becomes to factorize the modulus, which is the basis of RSA's security. Key lengths are crucial in cryptographic algorithms as they directly impact the security of encrypted data. A shorter key length increases the risk of successful attacks, while a longer key length enhances security but may also introduce performance trade-offs. Therefore, finding the right balance between security and efficiency is essential when selecting key lengths for cryptographic algorithms.

3 RSA vs AES

First looking at RSA. The RSA encryption algorithm is a widely used asymmetric encryption algorithm that was developed by Ron Rivest, Adi Shamir, and Leonard Adleman in 1977. It is named after their initials. RSA is based on the mathematical properties of prime numbers and modular arithmetic.

3.1 RSA Key Genration

Select two large prime numbers, p and q. Using $n = p * q$, where n is the modulus. Calculate:

$$\phi(n) = (p-1) * (q-1) \text{ ,where } \phi(n) \text{ is Euler's totient function}$$

Choose an integer e, such that

$$1 < e < \phi(n) \text{ and} gcd(e, \phi(n)) = 1$$

This value e becomes the public key exponent. Calculate d as the modular multiplicative inverse of

$$e \ modulo \ \phi(n)$$

This value d becomes the private key exponent. Encryption: To encrypt a message M, convert it into a numerical representation. Use the recipient's public key (n, e) to compute

$$C \equiv M^e (\mathrm{mod} n) \text{ ,where C is the ciphertext}$$

Decryption: To decrypt the ciphertext C, use the recipient's private key (n, d). Compute

$$M \equiv C^d \mod n$$

Where M is the original message

3.1.1 Pros of RSA

Security: RSA encryption provides a high level of security due to its reliance on prime factorization, which is computationally expensive to break. Asymmetric Nature: The use of separate keys for encryption and decryption allows for secure communication without sharing sensitive information. Digital Signatures: RSA can also be used for digital signatures, providing authentication and integrity to messages.

3.1.2 Cons of RSA

Performance: RSA encryption is computationally intensive, especially for large key sizes. This can impact performance in resource-constrained environments. Key Management: RSA requires the secure management of keys, including key generation, distribution, and storage.
Vulnerability to Quantum Computers: RSA encryption is vulnerable to attacks by quantum computers, which could potentially break the underlying mathematical problem upon which RSA relies.
Note: these are some of the main Pros and Cons of RSA, there are many more that are not listed here.

3.2 AES Encryption Algorithm

The Advanced Encryption Standard (AES) is a symmetric encryption algorithm that is widely used to secure sensitive data. It was selected by the National Institute of Standards and Technology (NIST) in 2001 as the successor to the Data Encryption Standard (DES). AES has become the standard for encryption and has various applications, including communications, protecting rest data, and ensuring the integrity of digital signatures.

3.2.1 How AES Works

AES operates on fixed-size blocks of data, with a block size of 128 bits. As stated previously, AES supports three key sizes: 128 bits, 192 bits, and 256 bits. The algorithm consists of several rounds, with the number of rounds depending on the key size. For AES-128, there are 10 rounds; for AES-192, there are 12 rounds; and for AES-256, there are 14 rounds. During each round, AES performs four operations: SubBytes, ShiftRows, MixColumns, and AddRoundKey. These operations involve substitution, permutation, and linear transformations on the input data using a round key derived from the original encryption key. The SubBytes operation substitutes each byte of the input with a corresponding byte from an S-box lookup table. This non-linear substitution provides confusion in the cipher. The ShiftRows operation cyclically shifts the bytes in each row of the state matrix. This diffusion ensures that each byte influences multiple output bytes. The MixColumns operation applies a linear transformation to each column of the state matrix. This operation provides further diffusion and ensures that each byte affects multiple output bytes. The AddRoundKey operation XORs each byte of the state matrix with a corresponding byte from the round key. The round key is derived from the original encryption key using a key expansion algorithm. By repeating these operations for the specified number of rounds, AES achieves a high level of security by combining confusion and diffusion properties.

3.2.2 Pros of AES

Security: Regarded as a highly secure encryption algorithm. It has undergone extensive analysis and scrutiny by cryptographic experts worldwide, and no practical attacks have been found against the full AES algorithm.
Efficiency: Designed to be efficient in both software and hardware implementations. It can encrypt and decrypt data quickly, making it suitable for a wide range of applications.
Standardization: Standardized encryption algorithm adopted by governments, organizations, and industries globally. Its widespread use ensures interoperability and compatibility across different systems and platforms.

3.2.3 Cons of AES

Key Management: As with any encryption algorithm, the security of AES relies heavily on proper key management practices. If the encryption keys are weak or compromised, the effectiveness of AES can be undermined.
Side-Channel Attacks: Although AES itself is resistant to various cryptographic attacks, side-channel attacks can exploit implementation vulnerabilities such as timing information, power consumption, or electromagnetic radiation leakage. Quantum Computing Threat: While not specific to AES, the advent of quantum computers poses a potential threat to all existing symmetric encryption algorithms. Quantum computers could potentially break the underlying mathematical problems that provide security for AES.

4 Quantum Computing

4.1 Definition of Quantum Computing

Quantum computing is technology that uses quantum mechanics to perform complex calculations and simulations that are beyond the capabilities of classical computers. In classical computing, information is processed using bits that can only be in one of two states, 0 or 1. However, in quantum computing, information is processed using quantum bits that can exist in multiple states simultaneously, allowing for exponentially faster processing times and more accurate results.

4.2 How Quantum Computing works

Quantum computing works by using quantum-mechanical phenomena, such as superposition and entanglement, to perform calculations that are not possible with classical computers.
Superposition: In quantum computing, qubits can exist in multiple states simultaneously, which allows for the processing of multiple possibilities at once.
Entanglement: When two or more qubits are entangled, their properties become intertwined, allowing for the manipulation of one qubit to affect the state of the other.
Quantum Gates: Quantum gates are the quantum equivalent of logic gates

in classical computing. They are the basic building blocks of quantum algorithms and are used to manipulate qubits and perform quantum computations.
Quantum Algorithms: Quantum algorithms are designed to take advantage of the unique properties of qubits and quantum gates to solve specific problems. Examples of quantum algorithms include Shor's algorithm for factoring large numbers and Grover's algorithm for searching an unsorted database.

4.3 Super Computers vs Quantum Computers

One of the key differences between supercomputers and quantum computers lies in their underlying hardware architecture. Supercomputers typically consist of a large number of interconnected processors working together in parallel to solve computational problems. These processors can be either general-purpose CPUs or specialized accelerators like GPUs or FPGAs. The performance of a supercomputer is measured in terms of its floating-point operations per second (FLOPS). In contrast, quantum computers rely on quantum bits and quantum gates to perform computations. Qubits can be implemented using various physical systems such as superconducting circuits, trapped ions, or topological states. Quantum gates manipulate the state of qubits through operations like superposition, entanglement, and measurement. The performance of a quantum computer is measured using metrics like the number of qubits, gate fidelity, and quantum volume. Another significant difference between supercomputers and quantum computers is the nature of the problems they are suited for. Supercomputers for solving computationally intensive tasks that can be divided into smaller independent parts and executed in parallel. These include simulations, data analysis, and optimization problems. Quantum computers, on the other hand, are still in the early developmental stages and are primarily focused on solving specific types of problems that are difficult for classical computers. These include factoring large numbers (which is the basis for many encryption algorithms), simulating quantum systems, and solving optimization problems with a large number of variables. Quantum computers have the potential to revolutionize fields such as cryptography, drug discovery, material science, and machine learning. It is important to note that while quantum computers have the potential to outperform classical supercomputers in certain areas, they are not expected to replace them entirely. Classical supercomputers will continue to be essential for a wide range of applications that do not benefit from quantum computing's unique capabilities. Additionally, the development of practical and scalable quantum computers still faces significant technical challenges, such as improving qubit stability, reducing errors, and increasing the number of qubits.

4.4 Quantum Computing breaking modern Cryptography

Quantum computing has the potential to break modern cryptography by exploiting the computational power and unique properties of quantum systems. Traditional cryptographic algorithms, such as RSA, AES, and ECC (Elliptic Curve Cryptography), rely on the difficulty of certain mathematical problems for their security. However, quantum computers can solve these problems much more efficiently than classical computers, rendering many of the current cryptographic techniques vulnerable. One of the most significant threats posed by quantum computing to modern cryptography is its ability to factor large numbers quickly. Factoring large numbers is the basis of several widely used encryption algorithms, including RSA, and these algorithms rely on the assumption that factoring large numbers is infeasible with classical computers. However, Shor's algorithm has the goal of factoring large numbers exponentially faster than the best known classical algorithms, such as the general number field sieve. Developed by Peter Shor in 1994, demonstrated that a sufficiently powerful quantum computer could factor large numbers exponentially faster than classical computers. Shor's algorithm exploits the quantum phenomenon of superposition and entanglement to perform parallel computations on multiple inputs simultaneously. By utilizing qubits, a quantum computer can explore a vast number of possibilities in parallel. This allows Shor's algorithm to factorize large numbers efficiently, breaking the security of RSA and other similar encryption schemes. Although Shor's algorithm won't be as lethal towards Symmetric key algorithms like AES and ECC, Grover's algorithm is there to break them as well. Looking at a quick summary of Grovers: This algorithm can search an unsorted database using a quantum computer with a complexity of $O(\sqrt{N})$, where N is the number of items in the database. This means that symmetric key algorithms with a key length of n bits would be as secure against a quantum computer as $n/2$ bits against a classical computer. Quantum computing powers towards modern cryptography have led to interest in post-quantum cryptography, which aims to develop new cryptographic algorithms that are resistant to attacks from quantum computers.

4.5 ECC (Elliptic Curve Cryptography) Algorithm

It is necessary to prove quantum computers break both symmetric and asymmetric algorithms. So let's first look at a very secure symmetric algorithm, ECC (Elliptic Curve Cryptography), and then see how Grover's Algorithm works and breaks it. The ECC (Elliptic Curve Cryptography) algorithm is based on the mathematics of elliptic curves over finite fields. It relies on the difficulty of the elliptic curve discrete logarithm problem (ECDLP) for its security. Elliptic curves are defined by equations of the form:

$$y^2 = x^3 + ax + b$$

Here, x and y are variables, and a, b are constants that define the shape of the curve. The curve is defined over a finite field, which is a finite set of numbers with operations like addition and multiplication defined.
The core operation in ECC is called point addition. Given two points p and q on the curve, it allows calculating a third point R, which lies on the curve and is the result of adding p and q together. Point addition is defined geometrically but can also be expressed mathematically using algebraic formulas derived from the curve equation. ECC relies on the fact that, given a point P on the curve and an integer k, it is computationally infeasible to calculate the point kP (multiplication of the point P by a scalar k). This forms the basis of the ECDLP, which provides the security of ECC. To use ECC for cryptographic purposes, a group of points on the curve is defined, with a specific base point G. The private key is a random integer d, and the public key is the point $Q = dG$ (multiplication of the base point by the private key). The security of ECC lies in most computers inability to calculate d given Q and G. For encryption, the sender chooses a random integer k, calculates the point $R = kG$, and sends the coordinates of R. The receiver, who knows the private key d, can then calculate the shared secret point $S = dR$ and derive the shared secret value using the x-coordinate of S. Taking a look at simple example: Suppose we have an elliptic curve defined with the equation

$$y^2 = x^3 + 2x + 2 \text{ modulo } 17$$

Assume base point $P(5, 1)$ on this curve.

1. Point Addition: Let's perform the operation $P+P$ using point addition. We draw a line through P that intersects the elliptic curve at two points P, $P1$, and find the reflection point $P2$ across the x-axis. The slope of the line passing through $P(5,1)$ and $P1(9,16)$ is $\frac{(16-1)}{(9-5)} = \frac{15}{4}$. The equation of the line is $y = \frac{15}{4}(x-5)+1$. Substituting the equation into the elliptic curve equation, we get:

   $$(\frac{15}{4}(x-5)+1)^2 = x^3 + 2x + 2 \text{ modulo } 17$$

   Simplifying the equation leads to

   $$x^3 + 3x + 3 \equiv 0 \text{ modulo } 17$$

   Solving this equation modulo 17, we find $x = 7$ and $x = 12$ as the possible x-coordinates of the resulting reflection point P2. We can choose either $x = 7$ or $x = 12$. Let's take $x = 12$. The reflection point $P2$ is $(12,6)$.

2. Scalar Multiplication: Suppose we want to compute 3P, where the base point P is still $(5,1)$. We perform scalar multiplication by adding P to itself two more times,ex. $P + P + P$. Using point addition, we get $(5,1) + (12,6) = (1,1)$. We have obtained the point $(1,1)$ as the result of scalar multiplication.

Looking at this it is easy to tell how complex these equations can get, almost impossible to the point for humans to calculate, and on a larger scale allows the ECC algorithm to be so efficient.

4.6 Grover's Algorithm and how it breaks ECC

Grover's algorithm is a quantum algorithm that provides a quadratic speedup for unstructured search problems. It was developed by Lov Grover in 1996 and is one of the most well-known quantum algorithms due to its potential impact on cryptography and database search. The algorithm can be used to search an unsorted database of N items in $O(\sqrt{N})$ time, compared to the classical $O(N)$ time complexity. Grover's algorithm consists of initialization, oracle application, and amplitude amplification. Initialization: In this step, the quantum computer initializes a superposition of all possible states. If we have N items in the database, this step creates an equal superposition of all

N states. Oracle Application: The oracle is a black box function that marks the desired state or states in the superposition. It performs a phase inversion on the marked states, effectively flipping their sign.
Amplitude Amplification: This step involves applying a sequence of operations to amplify the amplitude of the marked states while suppressing the amplitudes of the unmarked states. This amplification process increases the probability of measuring the marked states when performing a measurement at the end of the algorithm. The amplitude amplification steps are based on repetition of two operations: the inversion about the mean and the oracle reflection. These operations are applied $\sqrt{N}$ times, leading to the quadratic speedup compared to classical search algorithms.
Mathematical Representation: The algorithm can be represented mathematically using quantum gates and linear algebra. Let's consider a simple case where we have an unsorted database with N items and we want to find a specific item marked by an oracle function. We start with an equal superposition of all possible states:

$|\psi> = 1/\sqrt{N}\Sigma|x>$,where —x⟩ represents each possible state in the database

The oracle function marks the desired state by applying a phase inversion:

$O|x> = (-1)^{f(x)}|x>$,where $f(x)$is 1 if is the desired state and 0 otherwise

The amplitude amplification involves applying a sequence of operations involving the inversion about the mean (Hadamard gate) and the oracle reflection. After $\sqrt{N}$ iterations, we obtain a state that has a high probability of measuring the marked item. The final state after amplitude amplification can be represented as:

$$|\psi_f> = 1/\sqrt{K}\Sigma|x>$$

Where K represents the number of iterations required for successful amplification. Breaking ECC Cryptography relies on the difficulty of solving discrete logarithm problems over elliptic curves for its security. In classical computing, solving this problem has exponential time complexity, making it computationally infeasible for large key sizes. However, this algorithm provides a quadratic speedup for this type of problem, reducing its time complexity from $O(\frac{2^n}{2})$ to $O(2^{\frac{n}{2}})$. Using quantum amplitude amplification to find the desired term, the algorithm is able to create a superposition of all the states of the data, and then amplify using the quantum operator. Looking specifically into the mathematics behind the time complexity reduction from $O(\frac{2^n}{2})$ to $O(2^{\frac{n}{2}})$. The time complexity of the algorithm can be analyzed by considering the number of operations required to apply the operator O to the state $|\psi>$. Since each application of O requires a single quantum operation, the total number of operations can be written as: $O(\frac{2^n}{2}) = O(\frac{2^n}{2})^t$. By substituting $t = \frac{n}{2}$, $O(\frac{2^n}{2}) = (\frac{2^n}{2})^{\frac{n}{2}} = 2^{\frac{n}{2}}$. Therefore, the time complexity of Grover's algorithm is $O(2^{\frac{n}{2}})$, which is a quadratic speedup over the classical algorithm.

5 How Quantum Computing breaks the RSA encryption algorithm

As explained previously, quantum computers can break through the RSA algorithm using the Shor's algorithm. Let's look at the steps taken by Shor's algorithm to break RSA: Quantum Fourier Transform: Shor's algorithm begins by applying a quantum Fourier transform to convert the problem of factoring into a periodicity problem. This transform allows for efficient computation of periodic functions on a quantum computer.
Quantum Period Finding: After applying the quantum Fourier transform, Shor's algorithm uses a technique called quantum period finding to determine the period of a modular exponentiation function. This step is crucial as it provides information about the factors of the modulus.
Classical Post-processing: Once the period is found, classical post-processing is performed to extract the prime factors from the obtained information. This step involves applying classical algorithms to find the greatest common divisor between the period and the modulus, which reveals the prime factors.
Looking more into depth on the mathematical structure behind Shor's Algorithm it is understood that the algorithm first uses a quantum computer to find the period of a function that is related to the factors of the number being factored. The function is defined as follows: Let n be the number being factored, and let d be the smallest positive integer such that n is divisible by d. Then, the function $f(x)$ is defined as:

$$f(x) = x^d \text{ mod— } n$$

The period of this function is the smallest positive integer r such that

$$f(x + r) = f(x)$$

for all x. We are able to find the period r of $f(x)$ efficiently using a quantum computer. First, The algorithm starts with a quantum register of n qubits, where each qubit represents a possible value of the function $f(x)$. The initial state is a superposition of all possible values of $f(x)$, which can be represented as:

$$|\psi> = \Sigma x = 0^{n-1}|x>|f(x)>$$

Where $|x>$ is the basis state corresponding to the value x, and $|f(x)>$ is the basis state corresponding to the value $f(x)$. Using the quantum oracle, which is basically a black box that takes as input a quantum state and returns as output the value of the function $f(x)$evaluated at the input state. The quantum oracle is implemented using a quantum circuit that computes the function $f(x)$using the given input state. The quantum oracle is applied to the initial state $|\psi>$, which results in the state:

$$|\psi> = \Sigma x = 0^{n-1}|x>|f(x)>ß\Sigma x = 0^{n-1}|x>|f(x)> + \Sigma_{x=1}^{n-1}|x+1>|f(x+1)>$$

Where $|x+1>$ is the basis state corresponding to the value x + 1

Measurement of the period: The final step is to measure the state $|\psi>$ to obtain the period r. The period r can be obtained by measuring the state $|\psi>$ in the basis $|x>|x$ is an integer between 0 and $n-10$. The measurement outcome is a sequence of n integers, which can be used to compute the period r. The probability of obtaining a particular sequence of integers is given by:

$$P(r) = (\frac{1}{n}) * (\frac{d}{n})^{r-1} * (\frac{d}{n-1})^{n-r-1}$$

Where d is the smallest positive integer such that n is divisible by d.

The period r can be computed from the measurement outcome using the following formula:

$$r = \frac{n-1}{2} + \frac{n-1}{4} + \ldots + \frac{n-1}{2^k}$$

Where k is the number of times the measurement outcome is repeated.

Shor's algorithm can be used to break the RSA encryption algorithm by finding the factors of the modulus n. The RSA algorithm uses the difficulty of factoring large numbers to ensure the security of the encryption. However, Shor's algorithm can factor large numbers exponentially fast, allowing it to break RSA. Shor's algorithm, employs the modulus n, which is the product of two large prime numbers p and q. The period r of the function $f(x)$is found by applying the algorithm to the modulus n. Once the period r is known, the factors can be computed using the following formula:

$$p = r * d, q = n/d$$

Where d is the smallest positive integer such that n is divisible by d.

It is important to note that while Shor's algorithm has the potential to break RSA encryption, it can only do so at scale, error-corrected quantum computer with a sufficient number of qubits to be practical. Currently, quantum computers are still in their early stages of development and are not yet capable of breaking RSA encryption for large key sizes used in practice.

6 Quantum safe algorithm

6.1 What can be considered Quantum safe

There are some key features that can help reduce the voliatliy of quatum attacks. Key Space Size: The size of the key space should be very big to resist attacks from quantum computers. Computational Complexity: The algorithm should require computational steps that are possibly infeasible to perform in an amount of time. Also, even though this may seem obvious, reliance on computationally hard base problems. Looking at the Learning with Errors problem, LWE, for example, is said to be on the path of quatnam resistance in combination with other cryptographic algorithms.

6.2 New Hope

Post-quantum cryptography explores alternative mathematical problems that are believed to be hard even for quantum computers. Some of the proposed post-quantum cryptographic algorithms include lattice-based cryptography, code-based cryptography, multivariate cryptography, and hash-based cryptography. These algorithms are designed to withstand attacks from both classical and quantum computers, ensuring long-term security in the face of advancing technology.

7 Bibliography

[fta19] [JIA] [NC00] [Gil19] [WOH] [Sho] [mod23] [NIS22] [Sci] [qua22] [Ara18] [Sch18] [Eke99] [fG23]

References

[Ara18] Scott Araonson. Introduction to quantum information science lecture notes, Fall 2018.

[Eke99] Artur Ekert. Quantum computation and shor's factoring algorithm, January 1999.

[fG23] Geeks for Geeks. How to solve rsa algorithm problems? *Geeks for Geeks*, November 8, 2023.

[fta19] Emerging Technology from the arXiv. How a quantum computer could break 2048-bit rsa encryption in 8 hours. *MIT Technology Review*, May 30, 2019.

[Gil19] Martin Giles. Explainer: What is a quantum computer? *MIT Technology Review*, January 29, 2019.

[JIA] ZIHAO JIANG. Applications of number theory in cryptography.

[mod23] Diving deep into quantum computing: Modern cryptography. *TrendMicro*, September 12, 2023.

[NC00] Michael Nielsen and Isaac Chuang. *Quantum Computation and Quantum Information.* Cambridge University Press, October 2000.

[NIS22] NIST. Nist announces first four quantum-resistant cryptographic algorithms. *NIST*, July 5, 2022.

[qua22] https://courses.csail.mit.edu/6.857/2022/projects/su-zhang-zhu.pdf, May 2022.

[Sch18] Sebastian Schönnenbeck. Number theory in quantum computing, December 2018.

[Sci] Stanford Computer Science. Problems with public key.

[Sho] Petert W Shor. Publications list.

[WOH] JEREMY WOHLWEND. Elliptic curve cryptography: Pre and post quantum.