International Journal of Research in Engineering, Science and Management
Volume-1, Issue-12, December-2018
www.ijresm.com | ISSN (Online): 2581-5792

Will Advances in Quantum Computing Render Present Cryptosystems Obsolete?

Kush Pandey
Student, Department of Mathematics, JPIS, Delhi, India

***Abstract*: This paper presents advances in quantum computing rendered present cryptosystems obsolete.**

***Keywords*: Quantum computing, cryptosystems**

1. Introduction

Advances in technology and the rise of quantum computers are prevalent subjects in today's day and age. However, this advancement comes with a cost. Quantum computers pose a threat to modern-day crypto systems and also threaten the integrity and confidentiality of data. This paper aims to tackle the question 'Will advances in Quantum Computing rendered present cryptosystems obsolete?'

Most commonly used cryptosystems are based on the integer factorization problem and the discrete log problem. Quantum computers are able to find solutions to these problems in polynomial time. Therefore, there is a need to find cryptosystems that are based on problems that are resistant to quantum computing algorithms.

This paper initially explains the basic ideas of cryptography, symmetric key encryption and asymmetric key encryption. Following that, an introduction to quantum computing is given and two algorithms – Shor's algorithm [3] and Grover's algorithm [2] – that pose a threat to modern-day cryptosystems are introduced. Finally, an overview of post-quantum cryptography is presented. The ideas explored in that section include lattice-based cryptography, multivariate-based cryptography, and code-based cryptography.

2. What is cryptography?

Mankind has always been fascinated with withholding information from people they do not trust; they have always had the urge to keep secrets. In order to facilitate this, cryptography was born.

The analogy of the box best describes cryptography. Suppose a message is kept in a box and locked by a lock whose combination is only known to the sender and the recipient. This is known as encryption. Then the locked message is sent to the recipient, who opens the lock with the code he knew in advance. This is called decryption. Cryptography starts when one ditches the box and uses ciphers (a disguised way of writing) instead. Think of it as scrambling and unscrambling letters [7].

Cryptography has been in use since before the birth of computers. It has taken many forms, such as letter substitution and shift ciphers, to rotor machines such as the enigma that was broken by the British and helped in defeating the Nazis in World War 2 [9]. Today, in this digital age, cryptography has become more crucial to information exchange than ever before. Cryptography forms the backbone of the all-digital communication that takes place. Data is encrypted with an encryption key, which converts plain text into ciphertext, an unreadable string of letters that can only be decrypted by using the decryption key. The ciphertext is what is transmitted, or communicated, between computers, as the risk of it being understood by an eavesdropper are minimal. After the recipient receives the ciphertext, it is converted back into plaintext. Modern cryptography revolves around key generation that makes encryption almost impossible to break without the private keys. Given the computational power of today's computers, encryption keys are irreversible for a well-designed cryptosystem in a reasonable amount of time (polynomial time). There are two main types of encryption, the symmetric key encryption and the public key encryption.

3. Symmetric Key (Private Key) Encryption

Symmetric Key encryption is a type of encryption where the plaintext is encrypted and decrypted using the same or closely related shared key, often called the private key. Though not widely used in today's day and age, symmetric key encryption was the preferred and safest method of encryption when transferring data during the majority of the twentieth century as the concept of public key systems was unknown. The fact that symmetric key encryptions are simple to adopt and fast to implement resulted in them being used wherever encryption was necessary. One problem that plagued the symmetric key encryption method is the "key distribution problem," which says that securely communicating the key to the recipient may not be much easier than communicating the ciphertext over that channel [9]. Another problem that is associated with the symmetric key system is the huge key size required for a high level of security. Symmetric Key encryption is used in places where large amounts of data need to be encrypted or if the data being encrypted will be decrypted shortly thereafter. An example of the same includes the Transport Layer Security, or TLS, which encrypts and decrypts information that is exchanged between the server and the computer. However, with

the advancements in technology, symmetric key encryption became easy to crack. This lead to the birth of an alternative method: asymmetric key encryption.

4. Asymmetric Key (Public Key) Encryption

Asymmetric Key Encryption was developed to overcome the "key distribution problem" that Symmetric Key encryption faced. It was introduced in 1976 in a paper written by Whitfield Diffie and Martin Helman titled 'New Directions in Cryptography' [1]. Asymmetric Key encryption is a type of encryption in which plaintext is encrypted using the public key and decrypted using the private key. The public key is shared with everyone whereas the private key is kept as a secret. In this form of encryption, anybody can encrypt a message with the recipient's public key. The secret private key owned by the recipient can only decrypt this ciphertext. Asymmetric key encryption relies on 'one-way trapdoor functions' i.e. functions that are easy compute in one direction, but it is difficult to compute their inverse. These include the integer factorization and the discrete log problem (DLP). The main idea behind the integer factorization problem is that it is easy to find to large primes p and q but difficult to find p and q given n=p·q. On the other hand, the idea DLP is that it is easy to compute h=g^x for a given g and x, but difficult to find x given h and g. The most well-known public key encryption is the RSA encryption [4]. It is based on the integer factorization problem. It uses the product of two very large prime numbers to generate keys, which are almost impossible to crack if the numbers are large (at least 2048 bits or more) and carefully chosen. Since algorithms based on classical computing are unable to offer the computational power required to crack this type of encryption, the RSA Public Key Encryption [4] remains a common encryption method.

5. What is Quantum Computing?

Richard Feynman first introduced quantum-computing concepts in 1982 in a paper titled 'Simulating Physics with Computers' [5]. These computers are based on the laws of quantum mechanics rather than classical algorithms.

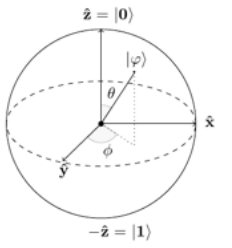

Fig. 1. The Bloch sphere

The smallest unit in a classical computer is a bit, which is either in the on state or the off state represented by 1 and 0 respectively. The speed of execution on a classical computer is determined by how quickly these bits can be accessed and changed. Therefore, the speed of encryption and decryption is also limited by a bit in a classical computer. Quantum computers, on the other hand, use quantum bits or 'qubits'. Qubits can exist in the 0 state, the 1 state, or a superposition of the two states at the same time [15]. Superposition does not mean that the qubit is divided half in this state and half in the other; it is in the 0 and 1 state at the same time. Qubits can be imagined as a sphere. While a classical bit can only be in two states - at either pole of the sphere - a qubit can be anywhere within that sphere. This representation is known as the Bloch sphere (see Fig. 1).

Mathematically, the vectors $|0\rangle$ and $|1\rangle$ form the basis of a qubit. The vector $|0\rangle$ represents [1,0] and $|1\rangle$ represents [0,1]. A qubit is a linear superposition of its basis vectors [9]. In other words, qubits can be represented as a linear combination of $|0\rangle$ and $|1\rangle$ with complex coefficients:

$$|v\rangle = a|0\rangle + b|1\rangle$$

Here, a and b are known as the amplitudes of the qubit and are complex numbers and since the absolute value of the qubit must measure up to be 1, $|a|^2 + |b|^2 = 1$

Quantum Computers take advantage of the fact that subatomic particles can co-exist in multiple states at a time. Because of this property, quantum computing operation requires less energy and can be performed more quickly than in classical computers. Any calculation using a qubit, acts on both values at the same time [15]. The parallelism of multiple qubits increases the number of states the computer is acting on simultaneously and therefore, quantum computers can perform tasks which may otherwise be impossible with modern computers.

A calculation on a qubit that is superimposed essentially acts on both values and can further be improved upon by using a system with multiple qubits in parallel. For example, a 4-qubit system would operate in 16 states. This improvement in performance because of parallelism grows exponentially as the number of qubits increases, allowing quantum computers to carry out tasks that may not be possible for classical computers.

Like modern computers, quantum computers also have gates. The most basic gate is the NOT gate that is used to invert the input. The following matrix is used to represent the NOT gate:

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

Another quantum gate that is commonly used in the quantum algorithms is the Hadamard gate. It creates a superposition of all the qubits.

For example, it will map $|0\rangle \rightarrow \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$ and $|1\rangle \rightarrow \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle$

6. The potential of quantum computers

The main advantage of using a quantum computer over a classical computer is the fact that quantum processors allow for

tasks to be performed simultaneously. This allows quantum algorithms such as Shor's Algorithm [3] and Grover's algorithm [2] to be more efficient than classical algorithms. Parallelism of qubits gives us a quantum computer more powerful than a classical computer [12]. The property of quantum parallelism is a result of the ability of a quantum memory register to exist in superposition of its base vectors. A number that is made up of n qubits can represent a quantum superposition of as many as 2^n states [15]. On a modern computer, any operation carried out will require 2^n steps whereas a quantum computer would complete these tasks in one step, as the actions will be carried out simultaneously. For example, a 50 qubit quantum computer can represent a quantum superposition of as many as 2^50 states. Each state is equivalent to a combination of 50 bits and is made up of 0s and 1s. This implies that any calculation performed through quantum gates would therefore operate on all 2^50 states all at the same time. An operation requiring 2^50 rounds of operation on a classical computer may be completed in 1 step on a quantum computer [15]. When a quantum computer is asked for a result, all the states would collapse into 1 state, which corresponds to the answer. However, one must note that not all classical computations are made faster by quantum computers.

One may draw parallels to multicore processing with quantum parallelism. However, rather than having separate processors, a quantum computer relies on the superposition of qubits to perform tasks simultaneously. Superposition also presents an effective solution to the problem of storage capacity as quantum computers can store much more information than a classical computer. Individual atoms can be used to store data that eliminates the possibility of running out of space on our hard drives.

Quantum computers also have various drawbacks. One drawback of quantum computers is that they are vulnerable to interferences, such as heat, noise and electromagnetic couplings, from the surroundings [9]. The base of quantum computing is the vibration in the atom, which can be disturbed by any external noise and lead to error in calculations. Also, the results obtained with the help of quantum computing are probabilistic. A quantum computer may yield several results from which only one is correct. Therefore, the process of verifying the correct answer undermines the advantage of increased calculation speed offered by a quantum computer [9]. Another drawback of quantum computing is that Qubits suffer from bit-flops. Like a bit can change is value from 0 to 1, a qubit can also change its value, leaving its superposition state.

7. Quantum Cracking

The study of determining the strengths and weaknesses of a cryptographic system is known as cryptanalysis [15]. Cryptanalysts employ various methods that reverse the cipher text into plaintext. While pen and paper were the only tools required to break the earliest forms of ciphers, today's encryption systems are based on hard math problems and employ powerful tools including computers that make cryptanalysis difficult. However, powerful computers help cryptanalysts as well.

8. Shor's Algorithm

The present public key encryption methods are secure because the encryption is a one-way function. One-way functions are described as functions that are easy to compute in one direction but difficult to find the inverse of the same function. Since no efficient solutions exist for these one-way functions, a classical computer is unable to reverse a cryptographic algorithm in polynomial time. Therefore, these algorithms are considered to be secure. One example of such a mathematical problem is factoring large semi primes, or the product of two prime numbers. No polynomial time algorithm for classical computers is known until today, making it a secure system. Many polynomial time algorithms for integer multiplication exist but there are none for factoring integers [11]. However, quantum computers using Shor's Algorithm could change all of this. Peter Shor came up with an algorithm in 1994 to factor large semi prime numbers, which is a difficult task for a classical computer [3]. This threatens the existence of various encryption algorithms, such as the RSA algorithm, which are based on the fact that classical computers find it nearly impossible to factor large semi-prime numbers in an efficient way [4]. The most efficient way to factor on a classical computer has a sub-exponential run-time, which implies that as the numbers get bigger, the time required to factor the number grows at an exponential rate.

Shor's algorithm focuses on finding the period of a function mod N. To give a bigger picture, Shor's algorithm is dependent on modular arithmetic, quantum parallelism and quantum Fourier transform. The algorithm has 2 parts to it. The first step is to convert the problem of finding factors of a number to finding its period. This is done using classical methods. Subsequently, finding the period is done using the Quantum Fourier Transform (QFT). The QFT is responsible for the quantum speedup in the algorithm [13]. In theory, Shor's algorithm creates a superposition of all the possible values of a function using a quantum gate and then transforms the amplitudes of the values in order to calculate the period of the function. This period is in turn used to factor the original integer.

In order to fully understand Shor's algorithm, one must be familiar with the following Theorem:

If N is a prime number and there exists a solution *x* such that $x^2 = 1 \; mod \; N$, where $x \neq \pm 1$; One of the greatest common divisor (gcd) of *(x-1, N)* and gcd of *(x+1, N)* is nontrivial factor of N

Proof:

Let $x \neq \pm 1 \; mod \; N$ and $x^2 = 1 \; mod \; N$. Then, $x^2 - 1 = 0 \; (mod \; N)$ so that $x^2 - 1$ *is* a multiple of N. Factoring this would yield that $N \mid (x+1)(x-1)$, but because

$x \neq \pm 1 \; (mod \; N)$, $N \nmid x \pm 1$.

Therefore, the gcd of (x-1, N) and gcd of (x+1, N) are factors of N and the gcd can be computed using Euclid's Algorithm

The basic Steps of Shor's Algorithm are as follows [3]:

Step 1: Choose a random integer $m < N$, such that m and N are co-prime. For this example, let $N = 15$ and $m = 2$.
Step 2: Use a quantum computer to find out the period P of the function $x \rightarrow m^x \ mod \ N$
Step 3: If P = Odd, go back to first step. Otherwise, if P = even, proceed ahead
Step 4: As P is even:

$$(m^{\frac{p}{2}} - 1)\ (m^{\frac{p}{2}} + 1) = (m^p - 1) = O \ mod \ N$$

Step 5: Compute d = gcd $((m^{\frac{p}{2}} - 1), n)$ using the Euclidean Algorithm. Since $(m^{\frac{p}{2}} + 1)$ cannot be $0 \ mod \ N$, d is a nontrivial factor of n.

If one wants to factorize the number 15, one needs a 4-qubit register. A 4-qubit register can be visualized as a traditional 4-bit register on a classical computer. The bit 1111 represents the number 15 in binary; therefore a 4-qubit register can calculate the prime factorization of this number. This step requires one to load the input register with equally weighted superposition of all integers from 0 to all possible states (16 states in this case) i.e. 0 to 15. This is done using the Hadamard quantum gate. Following that load the output register with the value zero. The calculations performed on the register are done parallel for each possible value of the qubits.

The initial state of the system will be represented as:

$$\frac{1}{\sqrt{16}} \sum_{a=0}^{15} |a, 000\rangle$$

*The comma denotes entangled states (the quantum state cannot be factored into its constituents)

Following this, the transformation $m^x mod \ N$ is applied to each number in the input register and the results are stored in the output register.

$$\frac{1}{\sqrt{16}} \sum_{a=0}^{15} |a, m^x mod \ N\rangle$$

After this, the QFT is applied

$$\frac{1}{\sqrt{16}} \sum_{a=0}^{15} QFT\ (|a\rangle) |m^x mod \ N\rangle$$

The QFT is used to find the period of the function in polynomial time on a quantum computer. This basic calculation can be modeled as:

$$Output \ Register = \ m^{Register\ 1} \ mod \ N$$

In the above equation, N represents the number one wants to factorize while m is any number greater than 1 but smaller than N that is chosen for the calculation. In the above calculation, m, the random number chosen for the calculation is raised to the power of the first qubit register and then computed mod N.

The Table (Table 1) below represents the result of the calculation.

Table 1
Result of the calculation

First Qubit Register	Calculation	Second Qubit Register
0	$2^0 \ mod \ 15$	1
1	$2^1 \ mod \ 15$	2
2	$2^2 \ mod \ 15$	4
3	$2^3 \ mod \ 15$	8
4	$2^4 \ mod \ 15$	1
5	$2^5 \ mod \ 15$	2
6	$2^6 \ mod \ 15$	4
7	$2^7 \ mod \ 15$	8
8	$2^8 \ mod \ 15$	1
9	$2^9 \ mod \ 15$	2
10	$2^{10} \ mod \ 15$	4
11	$2^{11} \ mod \ 15$	8
12	$2^{12} \ mod \ 15$	1
13	$2^{13} \ mod \ 15$	2
14	$2^{14} \ mod \ 15$	4
15	$2^{15} \ mod \ 15$	8

The second register contains the repeated output of 1,2,4,8. Since the pattern is made up of four numbers, the period or the frequency of the algorithm is 4. In this example, the frequency is easy to calculate but becomes difficult to perform once the numbers become larger.

The next step is to find the factor of N using the period one finds. The factor can be found out using the following equation-

$$\gcd\left(m^{\frac{p}{2}} - 1, N\right)$$

In this particular example, the calculation can be modeled as follows-

$$\gcd(3, 15) = 3$$

To confirm that 3 is a factor of 15, we can do this calculation: $15 \ mod \ 3 = 0$.

The above calculations show how Shor's algorithm can be used to solve the integer factorization problem. By starting at an equal superposition of qubits, and performing a QFT on it to find the period, one can calculate the unknown "exponent" in the integer factorization problem. This could have serious implications for public key encryption systems that rely on classical computers being unable to factor large semi-primes in a reasonable amount of time.

The average runtime for factoring integers according to the brute-force algorithm has an average time complexity of $O(2^{\log n})$ where $log \ n$ is the size of the input with n bits. This implies that as the number increases, the time required to factor it also increases as it is exponential. Factoring small numbers

within a reasonable amount of time is possible on classical computer, but doing the same for a large number is not possible. On the other hand, Shor's algorithm has a time complexity of $O((\log n)^3)$ which is a significantly smaller time when factoring large numbers even when compared to the best known integer factorization algorithms. The General Number Field Sieve is the most efficient classical algorithm for factoring integers larger than 10^{100}. It has a time complexity of $O(e^{\sqrt[3]{\frac{64}{9}}(\log n)^{\frac{1}{3}}(\log\log n)^{\frac{2}{3}}})$ where *n* is the number itself [18].

A comparison of the two runtimes is shown in the table (Table 2) below:

Table 2
Comparison of run times

The number being factored (n)	Average runtime on Shor's algorithm $O((\log n)^3)$	Average runtime of the General Number Field Sieve $O(e^{\sqrt[3]{\frac{64}{9}}(\log n)^{\frac{1}{3}}(\log\log n)^{\frac{2}{3}}})$
10^1	1	1
10^{100}	1,000,000	1,423,716
$10^{1,000}$	1,000,000,000	2.354×10^{17}
$10^{10,000}$	1,000,000,000,000	2.182×10^{45}

Shor's algorithm presents a very efficient way to factor large semi-prime numbers. The most widely used public key cryptosystem, the RSA, will be rendered useless by quantum computers as they will be able to factor large semi-primes in a reasonable amount of time.

9. Grover's algorithm

Searching an unsorted database takes a very long time on a classical computer. The only way is to go through each and every value and check. On the other hand, searching a sorted database is relatively easier. In the worst-case scenario, a search algorithm would require *n* steps (where is *n* is the number of entries in a database) when a database is randomly sorted [14]. On an average, searching an unsorted database would require $\frac{n}{2}$ steps. For example, in a database with 1,000,000 entries, the average time to find a particular value is 500,000. On the other hand, in the worst-case scenario, it may take 1,000,000 steps. This problem can be efficiently tackled by Lov Gorver's algorithm that uses quantum computers to search unsorted databases [1] [2]. Like every quantum algorithm, this algorithm is also dependent on quantum parallelism and superposition of the states. It aims at maximizing the amplitude of the right answer. It decreases search time from $\frac{n}{2}$ to $\sqrt{n}$ steps. This means that the number of steps falls from 500,000 to just 1,000.

Grover's algorithm has two steps [2]:

Step 1: Phase inversion
Step 2: Inversion about the mean

The first step in this algorithm is to create a superposition of states

$$|v\rangle = \sum_{x=0}^{N} \frac{1}{\sqrt{N}} |x\rangle$$

Following this, the amplitude of the number that is to be selected is inverted i.e. $\frac{1}{\sqrt{N}}|x\rangle \rightarrow -\frac{1}{\sqrt{N}}|x\rangle$. This is phase inversion.

After this, the average of the amplitudes is calculated. The average will be less than the highest amplitudes. Following this, the amplitudes are inverted about the mean. This implies that the amplitude of each state is transformed such that the amplitude is as far above the mean as it was below it before being inverted, and vice versa [16]. This results in the amplitude of the target increasing by a factor of 3. This is repeated $\sqrt{2^n}$ times following which the answer is found.

This can also be illustrated through an example. Let the following be the superposition of states and let the bolded super amplitude be the one that is to be enhanced:

$$|v\rangle = [\frac{1}{\sqrt{8}}, \frac{1}{\sqrt{8}}, \frac{1}{\sqrt{8}}, \frac{1}{\sqrt{8}}, \frac{1}{\sqrt{8}}, \frac{\mathbf{1}}{\sqrt{\mathbf{8}}}, \frac{1}{\sqrt{8}}, \frac{1}{\sqrt{8}}]$$

Then, the amplitude of the number one wants to enhance is inversed:

$$|v\rangle = [\frac{1}{\sqrt{8}}, \frac{1}{\sqrt{8}}, \frac{1}{\sqrt{8}}, \frac{1}{\sqrt{8}}, \frac{1}{\sqrt{8}}, \frac{-\mathbf{1}}{\sqrt{\mathbf{8}}}, \frac{1}{\sqrt{8}}, \frac{1}{\sqrt{8}}]$$

The average of these amplitudes is:

$$\frac{7 \cdot \frac{1}{\sqrt{8}} - \frac{1}{\sqrt{8}}}{8} = \frac{3}{4\sqrt{8}}$$

Inverting the amplitudes about the mean leads to the following answer:

$$|v\rangle = [\frac{1}{2\sqrt{8}}, \frac{1}{2\sqrt{8}}, \frac{1}{2\sqrt{8}}, \frac{1}{2\sqrt{8}}, \frac{1}{2\sqrt{8}}, \frac{\mathbf{5}}{\mathbf{2}\sqrt{\mathbf{8}}}, \frac{1}{2\sqrt{8}}, \frac{1}{2\sqrt{8}}]$$

Repeating this Grover's algorithm would yield:

$$|v\rangle = [\frac{-1}{4\sqrt{8}}, \frac{-1}{4\sqrt{8}}, \frac{-1}{4\sqrt{8}}, \frac{-1}{4\sqrt{8}}, \frac{-1}{4\sqrt{8}}, \frac{\mathbf{11}}{\mathbf{4}\sqrt{\mathbf{8}}}, \frac{-1}{4\sqrt{8}}, \frac{-1}{4\sqrt{8}}]$$

This would lead to a probability of $|\frac{11}{4\sqrt{8}}|^2 = 0.9543$ to find the correct answer that the user is looking for.

The average time to find a particular entry on a classical computer would take $\frac{n}{2}$ steps. This may be a reasonable amount of time for a small database but may not work for larger databases. Therefore, large databases such as the ones that store login credentials, such as username and password, or credit card numbers are not only encrypted but also stored in an unsorted

manner to enhance the security. However, all of this is under threat because of Grover's Algorithm. The following table (Table 3) compares the average runtime of Grover's algorithm with the classical algorithm for locating a particular entry in a database.

Table 3
Average steps required for sorting

Size of the database (n)	Average steps required on Grover's algorithm ($\sqrt{n}$)	Average steps required on a classical computer ($\frac{n}{2}$)
10	3.16	5
100	10	50
1,000	31.6	500
10,000	100	5,000
100,000	316	50,000
1,000,000	1,000	500,000
10,000,000	3,160	5,000,000

The difference is negligible for small databases (N=10). However, as N increases, Grover's algorithm becomes significantly more efficient. In cryptography, Grover's algorithm can be applied to brute-forcing algorithms [2]. When one uses Grover's algorithm to find out a password using the brute-force method, the algorithm will try each possible permutation until it finds the correct one [1]. Such applications threaten systems that use symmetric key as a standard for encryption (For example, the Advanced Encryption Standard (AES)).

10. Implications

Table 4
Impact of Shor's algorithm on cryptographic algorithms

Cryptosystem	Type	Purpose	Implication because of quantum computers
AES-256	Symmetric Key cryptography	Encryption	Relatively secure
SHA-256, SHA-3	-	Hash Functions	Relatively secure
RSA	Public Key cryptography	Signatures and key establishment	Not Secure
Elliptic curve cryptography (E.g. ECDSA, ECDH)	Public Key cryptography	Signatures and key exchange	Not secure
Finite Field Cryptography (E.g. DSA)	Public Key cryptography	Signatures and key exchange	Not secure

The integer factorization problem (e.g. RSA) and the discrete logarithm problem (e.g., DSA signatures and ElGamal encryption) form the basis of the public key cryptosystems used today. These encryption methods have come under threat because of Shor's algorithm as well as the progress made in building actual quantum computers. Recently developed algorithms that are based on elliptic curves (such as ECDSA) use a modification of the discrete logarithm problem and hence are also threatened by Shor's algorithm [9].

The table (Table 4) below illustrates the impact of Shor's algorithm on cryptographic algorithms.

Grover's algorithm, on the other hand, threatens symmetric key encryption. These cryptographic schemes (such as the Advanced Encryption Standard-AES) are safe even from quantum computers if they have sufficient key sizes. Since Grover's algorithm offers a square-root sped up to brute-force algorithms, it makes an n-bit cipher from 2^n to $\sqrt{2^n} = 2^{\frac{n}{2}}$ [9]. In reality, this implies that a private key encryption with a key length of 32 bits would provide security equivalent to a key length of 16 bits. Therefore, the Advanced Encryption Standard (AES) is considered to be resilient to quantum computation when its key size is 192 bits or 256 bits. Another fact that concretizes the fact that the AES is secure in the post-quantum era is that the NSA (The National Security Agency) uses the AES encryption method to protect classified documents and encryption them with key sizes of 192 or 256 bits [9]. This is because the NSA understands that public key cryptosystems will not be resilient to quantum computers and hence they want to store their files in such a manner that the files do not fall into the wrong hands ever.

Table 5
Comparison of the classical and quantum security levels for the most used cryptographic schemes

Cryptographic Algorithm	Key Size (in bits)	Effective Key strength	
		Classical computers	Quantum computers
RSA - 1024	1024	80	0
RSA - 2048	2048	112	0
ECC - 256	256	128	0
ECC - 384	384	256	0
AES - 128	***128***	***128***	***64***
AES - 256	***256***	***256***	***128***

The table (Table 5) below compares the classical and quantum security levels for the most used cryptographic schemes [9].

11. Post quantum cryptography

The question that becomes evident is: are there any cryptosystems that are safe to use in the age of quantum computing? The aim of post quantum cryptography is to develop encryption algorithms that are safe against both quantum and classical computers. The mathematical based solutions that have come up are the following:

- Lattice based cryptography
- Multivariate-based cryptography
- Code-based cryptography

Lattice-based cryptography is something that eliminates the weakness of the RSA system. It involves the multiplication of matrices rather than the multiplication of large prime numbers. This encryption method is based on the mathematical hardness of lattice problems, such as the short vector problem (SVP) [9]. It involves finding out the shortest non-zero vector where the input is a lattice represented by an arbitrary basis. The Ajtai-

Dwork (AD), Goldreich-Goldwasser-Halevi (GGH) and NTRU encryption schemes are examples of lattice-based cryptosystems [9].

The multivariate-based cryptographic algorithms are based on the difficulty of solving multivariate polynomials over a finite field. Examples of such cryptographic algorithms include the MI88, Pat96b, and KPG99. An MQ-problem involves finding the solution to $x \epsilon \mathbb{F}$ for a given system of quadratic polynomials and a given vector $y \epsilon \mathbb{F}^m$ [10].

Code based cryptography is a type of crypto scheme that uses error correcting codes. It is based on the fact that decoding linear codes is difficult. Code-based cryptosystems are considered to be resilient to quantum computers. Linear block codes are generally used in error-detection and error-correction codes. It involves a generator matrix *G* that transforms the message *m* into a codeword *c*. There is always a parity check matrix *H* that is derived from the generator matrix and is used to check the consistency of the codeword. Code-based cryptography relies on the fact that there exist no efficient algorithms to decode a general linear code [6].

In code-based cryptography, to encrypt a message, a random error vector is added. To decrypt the message, one must remove the errors that had been added. It is imperative to hide the algebraic structure of the code so that it is difficult for the adversary to decrypt it.

One of the most prominent code-based cryptosystems is the McEliece cryptosystem and was introduced in the 1970s by Robert McEliece [6]. It was not adopted during the 1970s mainly because of its large key sizes. The public key of the cryptosystem is a randomized generator matrix that is multiplied with a random permutation and a non-singular matrix. If the generator matrix *G* is $k \times n$, then the random non-singular matrix *S* is $k \times k$ and the random permutation matrix *G* is $n \times n$. For the generator matrix, there is an efficient algorithm *Dec* that can decode the codeword with up to T errors. The public key is $G^* = S \cdot G \cdot P$. The message *m* is multiplied by G^* and a random error *e*, a random n-bit vector, is added to create the codeword *C*. The codeword can have errors after it has been transmitted through a noisy channel and has been received by the recipient. Therefore, $e<T$, so that there is room for unwanted errors. To decrypt the message, the recipient can reverse the secret permutation *P*. Following this, the original message can be recovered by inverting the linear transformation given by S [8]. This can be written as:

1. $c^* \leftarrow c \cdot P^{-1}$
2. $m^* \leftarrow Dec(c^*)$
3. $m \leftarrow c^* \cdot S^{-1}$

Code-based cryptosystems cannot be broken by quantum computing algorithms yet. Therefore, they can be used in post-quantum cryptography.

12. Conclusion

Quantum computing is on the rise. However, making a quantum computer that can be used by a common man will still take a couple of years. The most powerful quantum computer built till date is by IBM and is a 50-qubit machine and can only remain in a quantum state for 90 microseconds [17]. Information plays an important role in today's society and the integrity and confidentiality of the same are very important. Quantum computing and quantum algorithms pose a threat to cryptosystems that are commonly used in information transmission in the modern era. This includes the public key algorithms (for example RSA, ElGamal, ECC and DSA) and symmetric key algorithms (such as 3DES, AES). A fully operational quantum computer would render public key encryption methods that are considered unbreakable as of now to be insecure [9]. Solutions that have come up for this include lattice- based cryptography, code-based cryptography, and multivariate-based cryptography.

References

[1] Diffie, Whitfield, and Martin E. Hellman. "New Directions in Cryptography." IEEE, 1976.
[2] Grover, Lov K. "A Fast Quantum Mechanical Algorithm for Database Search." Bell Labs, 1996.
[3] Shor, Peter. "Quantum Algorithm for Factoring, 1994; Shor." Springer, 1994.
[4] Rivest, R.L., et al. "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems." MIT Lab. for Computer Science and Department of Mathematics, Cambridge, MA, 1978.
[5] Feynman, Richard P. "Simulating Physics with Computers." International Journal for Theoretical Physics, vol. 21, nos. 6/7, 1982. people.eecs.berkeley.edu/~christos/classics/Feynman.pdf.
[6] Hans Christoph Hudde. "Development and Evaluation of a Code-Based Cryptography Library for Constrained Devices." Ruhr Universitat Bochum, 7 Feb. 2013.
[7] Cruise, Brit. "What Is Cryptography?" Khan Academy, Khan Academy, 2012. www.khanacademy.org/computing/computerscience/cryptography/crypt/v/intro-to-cryptography
[8] Marek Repka, and Pavol Zajac. "Overview of the Mceliece Cryptosystem and Its Security." Tatra Mountains Mathematical Publications, Slovak Academy of Sciences, 2014.
[9] Vasileios Mavroeidis, et al. "The Impact of Quantum Computing on Present Cryptography." (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 9, No. 3, Department of Informatics, University of Oslo, Norway, 2018.
[10] Christopher Wolf. "Introduction to Multivariate Quadratic Public Key Systems and Their Applications." E´cole Normale Sup´erieure, D´epartement d'Informatique, 22 Mar. 2006.
[11] Shah Muhammad Hamdi, Syed Tauhid Zuhori, Firoz Mahmud, Biprodip Pal. "A Compare between Shor's quantum factoring algorithm and General Number Field Sieve", 2014 International Conference on Electrical Engineering and Information & Communication Technology, 2014.
[12] Vlatko Vedral, Martin B. Plenio. "Basics of quantum computation", Progress in Quantum Electronics, 1998.
[13] Eleanor Rieffel. "An introduction to quantum computing for non-physicists", ACM Computing Surveys, 9/1/2000.
[14] Jeffrey K. Uhlmann. "Hybrid quantum-classical computing with applications to computer graphics", Acm Siggraph 2005 Courses on - Siggraph 05, 2005.