Post Quantum Hybrid Network Security and Architecture

By
Anthony Essaye

Faculty of Natural and Applied Sciences
Department of Computer Science
Notre Dame University - Louaize

A thesis submitted in partial fulfillment of the requirements
for the Master of Science in Computer Science

January 2021

Post Quantum Hybrid Network Security and Architecture

By

Anthony Essaye

Approved by:

Dr. Hicham Hage: Associate Professor of Computer Science
Thesis Advisor.

Dr. Hoda Maalouf: Associate Professor of Computer Science
Member of Committee.

Dr. Marie Khair: Associate Professor of Computer Science
Member of Committee.

January 20, 2021

Date of Thesis Defense

Declaration

I hereby declare that I am the sole author of this thesis. To the best of my knowledge, this piece of work does not contain any material that has been previously published by any other person unless stated and acknowledged in their respective places.

Abstract

Quantum Computers, the next wave of computing, and an emerging field in computer science have brought numerous advancements to problem solving and a quantum approach to programming a certain machine. This thesis tackles three main areas in the approach to quantum computers. First, the thesis provides a state-of-the-art review to Quantum computers/ Second it proposes a timeline for the progress of the technology and highlights the importance of early action. Finally, it focuses on the implications of such advancements on the current standards of network security. Specifically, it proposes a possible feasible approach to mitigate any shortcomings of the previously thought computationally secure RSA encryption system, by leveraging the strong points of both classical computers and quantum ones to build a hybrid network, which is not only robust, but also quantum resilient.

Keywords: Quantum Computers, Post Quantum Cryptography, RSA, Quantum Information, Quantum Theory, Shor's Algorithm, Grover's Algorithm, Qubits, Hashing, Code-Based Algorithms, Lattice Systems

Table of Contents

List of Figures

List of Tables

List of Code Listings

List of Abbreviations

QM	Quantum Machine
QC	Quantum Computer
NIST	National Institute of Standards and Technology
RSA	Public Key Encryption Algorithm
E()	Encryption Function
D()	Decryption Function
C	Cipher Text
PT	Plain Text
LCM	Least Common Multiple
GCD	Greatest Common Divisor
CNOT	Controlled Not Gate
$\|\Psi\rangle_A$	The state of a qubit A
CMOS	Battery powered chip / Usually used to store the real time clock
QFT	Quantum Fourier Transform
QMR	Quantum Memory Register
PK	Public Key
SHA	Hashing algorithm

Acknowledgments

This piece of work would not have been possible without the presence of many different groups of people in my life. First and foremost, I want to express a ton of gratitude to my parents and brother. They have always believed in me and made sure to provide me with whatever is needed to achieve my goals. Without them, this would not have been possible.

Secondly, I would want to thank Dr. Hicham Hage and Dr. Hoda Maalouf. Without their guidance and motivation, I would not have even pursued my Master's degree. By extension, I would also dedicate this to Dr. Elie Fares, Dr. Khalil Challita, Dr. Nazir Hawi and Dr. Pierre Akiki for their effort and time during my Master's degree. All the aforementioned professors have made my journey worth it. Their combined experience and knowledge have elevated my educational journey to the next level. Their love for NDU and their mission helped make NDU my second home. Moreover, I want to thank Dr. Bassem Sabra. Without his permission to attend his advanced physics courses, most of this thesis would have not been possible.

Thirdly, I should never forget about the people who were there during it all. Those who started out as my friends and became my family. I would like to thank Dhalia Nazha, Elena Njeim, Georgio Adwan, Mario Hayek, Mira Hashem, Mohamad Kazma, Stefano Fallaha, Tatianna Nicolakis, Veronica Rammouz and Ziad Adwan for their never-ending support even through the toughest of times. For all of that, I am extremely grateful.

This thesis should act as living proof, that no matter what happens around us, we can achieve great things. As long as we do not give up, everything is possible.

Gradatim Ferociter,

Anthony Essaye

Chapter 1: Introduction

Quantum Computers, recent famous buzzwords that are as widely used as Artificial Intelligence. They are considered the next wave of computing. Projected to have computational powers like non other. This chapter will introduce the big picture view of the problem and will discuss how this thesis will tackle the solution.

1.1 Introduction to the General Problem

Billions of devices are currently connected to the internet around the world. These devices all use agreed upon standards to establish security and trust between each other. These security concepts need to make sure that data remains untampered in every phase since its inception until its delivery. Moreover, standards were put in place for the verification of the data's creator and the secrecy of this data being only available to those that are meant to have access to it. Currently the encryptions used to establish such trusts vary. However, the most widely used cryptosystem is an implementation of a public-private key system called RSA. RSA is an algorithm that consists of lots of mathematical steps that ensure that a certain message is encrypted in a such a way that only the intended party would be able to access the original information. Encryptions could be considered either computationally secure or completely secure. Computationally secure refers to the fact that the lifespan of the encryption out lasts the validity of the data it secures or its value. Imagine securing data that is top secret for a company and this data might be something with political value. If this data is deemed to be useful for the coming decade then the encryption must be able to keep it secure for at least that long. A completely secure encryption is one that could never be broken no matter how much power an attacker has. RSA, mentioned above, is considered computationally secure, as it has the potential of keeping data secure for millions of years depending on the key size it uses. However, since the inception of quantum computers, it was prophesized that these computers would be able to break our cryptosystems in a matter of

minutes or hours instead of trillions of years. QCs will be able to outperform classical computers computationally and leverage weaknesses in the main algorithm that would drastically decrease the time required to brute force the encryption.

1.2 Problem Definition

Quantum computers were first mentioned in the late 50s. However, meaningful research into the field only started in the 80s. All the research done through-out the 80s up till the mid-90s was theoretical work. Up until Peter Shor published his paper in 1994 containing his latest study about how quantum phenomena could be leveraged to aid in finding the period to factorizing a certain number. This paper gained the attention of the scientific community and established lots of research groups with one aim, creating a quantum computer. Slowly, but steadily these machines are becoming more powerful. While these machines are not powerful enough for a useful task today, they have proven on a small scale that their respective algorithms are fast. Many hurdles still need to be addressed before any real progress can be achieved. Shor's algorithm focuses on leveraging the quantum ability of superposition, where the data stored in qubits is considered to be in 'all states at the same time' and these positions could be programmed to overlap in such a way that they could destructively cancel out the wrong answers while also amplifying the probability of the correct answer. Demonstrating this power in layman's terms would be to consider how a classical machine counts from 0 to 15. Classical computers serialize their data and go over them one by one. With QCs and superposition, it would only require 4 qubits in superposition to represent every number from 0 to 15 at the same time, requiring only 1 step to achieve all the values required.

1.3 Research Objectives

The objective behind this thesis is to first introduce what Quantum computers are and how they work, then this thesis will discuss a realistic timeline for early action against perceived actions, and finally devise some form of quantum resilient network. The thesis focuses on discussing current technologies, and the best proposals available for the next wave of

computing. While absurd ideas and theories could be envisioned, the end goal of this thesis is to design an architecture that is not only feasible but could also leverage current hardware and software to operate. This thesis would aim to discuss a type of hybrid network built from both quantum and classical computers. This hybrid network would have to satisfy a few main ideas such as:

1) Will need to be as fast as current networks
2) Must be quantum resilient in its security
3) Will consist of classical clients and quantum gateways
4) Must rely as much as possible on current infrastructure

1.4 Thesis Organization

First, this thesis will discuss the inner workings of RSA. Only when one knows how the algorithm works and its attack surfaces, that they can start to understand why this cryptosystem has withstood the test of time. Afterwards, Chapter 3 introduces quantum computers, how they work and why they are important. This section will be dense because of the weird innovations of quantum computers and the way they work. Chapter 4 discusses Shor's algorithm, the main driving force for the development of quantum computers. This will foreshadow the importance of early action. Since the field of Quantum Information is growing up and maturing really fast, NIST has already initiated a call for post quantum cryptosystems. Chapter 5 discusses some of the current proposals and how they would work. Chapter 6 will then discuss the BB84 protocol that suggests a method to distribute keys over quantum channels. Then this protocol would be leveraged to plan out possible approaches to build out quantum resilient hybrid networks. Such networks would include both quantum computers and classical computers working together in harmony without any noticeable change on the user's end. Final chapter will only recap the findings and present the conclusions based on all the data researched throughout the thesis.

Chapter 2: Deep Dive Into RSA

RSA, the encryption known and loved by everyone. The basis of modern-day computing and networking solutions founded originally R.L. Rivest, A. Shamir, and L. Adleman with extreme futurism in mind. RSA was developed with Public-Key cryptosystems concepts in mind. The requirements were simple, develop a system that has the following four properties: (A) Deciphering an enciphered form of a message yields the original message, (B) Both E (the encryption procedure) and D (the decryption procedure) are easy to compute, (C) There shall be no easy way to compute D if E was ever disclosed to the public except if the intended party has the equivalent key, and (D) If a plain text message was deciphered and then ciphered, then the plain text should be retrieved., this would be represented as E(D(PT)) = PT. These methods will not only ensure that a message would only look garbage to an unintended eavesdropper but could also be used in manners to ensure privacy and form a unique signature (Rivest, Shamir, and Adleman 2019).

2.1 How RSA Works

The message is firstly divided into blocks with each block represented as an integer between 0 and n-1. This step's purpose is just to get a numeric form of the message that will later be used to encrypt the message. Grab a pair of positive integers to represent the public encryption key, denoted here by e and n. Encrypt the message by raising it to the power e and modulo n. The remainder in this case will be the ciphertext C. Decrypting this message requires raising the ciphertext to another power d modulo n.

$$C \equiv E(M) \equiv M^e \ (mod\ n)$$

The size of the message is not affected, the plain text and cypher one will both have a size range from 0 to n-1. The method above used three numbers *e, d,* and *n*. These numbers can be considered as two pairs one for encryption and one for decryption denoted by (*e, n*) and (*d, n*) respectively. Each user will need to generate these pairs and shares the encryption function publicly. This leads to the question, how are these keys generated? First, generate *n* by calculating the product of two extremely large prime numbers denoted as *p* and *q*. Then, the value *n* is shared publicly. The strength of this being that factorizing *n* would probably take more than the entire lifespan of the universe to compute, while also hiding how *d* can be calculated from *e*. Now calculate *d* which should also be a large random integer which should be relatively prime with respect to the product of (*p-1*) and (*q-1*) satisfying the equation below

$$\gcd\big(d, (p-1)\cdot(q-1)\big) = 1$$

Then as mentioned by Rivest et al., derive *e* using the following formula:

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

2.2 Algorithm

Dissecting how the main algorithm operates would highlight why RSA has been highly successful and has stood the test of time. This section describes firstly the efficiency of this algorithm. This will also show how RSA leverages the use of prime numbers to construct large keys which are extremely hard to factorize, thus giving a computationally secure method of creating keys. Potential methods of attack are also discussed with the reasoning behind why it is not currently interesting to try and attack RSA with current technologies.

2.2.1 Efficiency of Encryption and Decryption

It is vitally important for RSA to be fast. Encryption and decryption by users who have the keys should be almost instant. And thus, the encryption process was divided into four steps as follows:

1) Create a binary representation of *e* as a decreasing series starting from *k*: $e_k e_{k-1} \ldots e_1 e_0$
2) Set C = 1
3) Keep repeating the following steps for *k, k-1, …, 0*

1. Divide C^2 by n and set its remainder to C
2. If $e_x = 1$, then set $C = \frac{(C \cdot M)}{n}$

4) Now C is the required ciphertext.

This method results, at most, in the complexity:

$$2 \cdot log_2 (e)$$

And this complexity would be the number of multiplications or divisions if decrypting which are usually trivial operations that cost $O(1)$ in time. This procedure while efficient is not half as efficient as the best solution using procedures by Knuth. What's good about this approach is that both encrypting and decrypting take exactly the same effort as they are the inverse of each other in terms of procedural calculations. While this algorithm by Rivest et al., was created in 1978, it was proven to encrypt a message of 200 digits in a few seconds with non-specialized hardware. However, data transmission capabilities have grown exponentially since then, so has processing power. All this process relies on how fast prime numbers of that size could be found.

2.2.2 Calculating Prime Numbers

The whole premise of RSA depends on the fact that $n = p \cdot q$ where p and q are the required prime numbers. The logic behind this being that the larger *p* and *q*, the harder our key would be to factorize. According to the standards of the time, it was recommended to use 100 digit sized prime numbers which would result in a 200-digit long key. Using the prime number theorem which dictates that:

$$(\ln 10^x)/2$$

Numbers must be generated before finding an actual prime number. If then a 100-digit number is given to that formula, it equates to 115 numbers must be tested before finding a prime number. To test for a prime number, generate a random number *b* from a uniform distribution and test whether the *gcd (x, b) = 1* (Solovay and Strassen 1977). If this holds for 100 random numbers, then the chosen number *x* is almost certainly a prime number. The probability of success using this method is 1/2 and if an error occurred it will be detected

later on when trying the encryption decryption pairs. It is also recommended that p and q differ in size by a few digits to increase the protection against factoring algorithms.

2.2.3 Choosing *d* and computing *e* from d and *Φ(n)*

Choosing d (the decryption key) is relatively easy. It must be relatively prime to $\Phi(n)$, and the easiest measure for this would be a prime number greater than the $max(p, q)$. Using Euclid's algorithm to find the $gcd(\Phi(n), d)$ and using the series $x_0, x_1, x_2, \ldots$ where x_0 would be equivalent to $\Phi(n)$, $x_1 = d$, and then x_{i+1} equivalent to $x_{i-1} (mod\ x_i)$, until a value $x_k = 0$ is found. This operation will not be hard since it is known that k would be found within our stated time complexity (on a very linear scale).

2.3 Approaches to Cracking RSA based encryption

An encryption's strength is only measured by its relative weakness. How easy is it to break? While there is no formal testing standard for encryptions, they are usually tested against strong computers over an extended period of time. This combined with a cryptanalysis of the algorithm have proven that RSA would be exceedingly difficult to break. An intruder would have a few options, they would have to either factorize n (which will be proven to be difficult in the next section), computing $\Phi(n)$ without factorizing n, or find the decryption key with some method.

2.3.1 Factorizing *n*

Factoring n is crucial and enables the attacker to calculate $\Phi(n)$ and then the decryption key. However, factoring a number by itself is extremely hard. This becomes only worse when the factors of said number are prime numbers instead of compound ones. Research into factorizing numbers led to a factorization time of $O(n^{1/4})$ (Pollard 1974). However, the fastest factoring algorithm described by (Rivest, Shamir, and Adleman 1978), at the time of them writing their paper is an algorithm derived by Richard Schroeppel (unpublished) which can factor n in

$$\exp\sqrt{\ln(n) \cdot \ln(\ln(n))} = n^{\sqrt{lnln(n)/\ln(n)}}$$

steps. If this formula was to be applied to the *n* that was suggested above with size of 200 digits, then it will take 1.2 x 10^{23} operations which would approximately take 3.8 x 10^{9} years of time.

2.3.2 Computing *Φ(n)* without factorizing *n*

$\Phi(n)$ represents the Euler totient function that gives numbers that are positive and less than n and are relatively prime to *n*. For prime numbers *p*, then $\Phi(p) = p - 1$. In this case $\Phi(n) = \Phi(p) \,.\, \Phi(q)$ Keeping in mind that

$$e \; mod \; \Phi(n) = d^{-1}$$

Calculating *Φ(n)* would be an interesting approach to an attacker. This method however is no easier than factorizing *n*. *Φ(n)* is defined as such:

$$\Phi(n) = n - (p + q) + 1$$

Then, derive that (*p*-*q*) is $\sqrt{(p+q)^2 - 4n}$, then *q* would be half the difference of ($p + q$) and ($p - q$). Without going further into the details of the mathematics of this calculation for the sake of conciseness, it gets increasingly obvious that this method would be even harder than factorizing n. The only other approach would be to try and find the decryption key.

2.3.3 Determining *d* in some other way

As mentioned previously *d* is chosen from a set the was deemed "large enough" or by a rule of thumb more than *max(p, q)*. This technique was used so a direct search for the key (by simple incrementation or other ways) would be extremely hard. The next step to finding such a value would be to find the

$$LCM((p-1),(q-1))$$

Finding such multiple, given that n is a large digit, would be extremely difficult. It is important to mention that according to (Rivest et al. 1978), "It may be possible to prove that any general method of breaking our scheme yields an efficient factoring algorithm" This specific statement will be important to the discussion of this thesis in later sections.

Chapter 3: Introduction to Quantum Computers and Their Advancements

Quantum Information and Quantum Computing have been touted as the next big step in digital information processing. It stands as a giant futuristic field next to artificial intelligence and others. This chapter will discuss what quantum computers are, how they work and how they differ from "classical" computers. The term classical in this chapter will always refer to the computers that are in use today in our daily lives.

3.1 Quantum Computers – The Basics

3.1.1 Bit vs. Qubits

The fundamental, smallest unit of data that is used in traditional computers is called a bit. A bit is a simple representation of 0 or 1 and is usually the state of a transistor. If the transistor is off, the bit would be a 0, and it would be 1 otherwise. Quantum computers on the other hand do not use bits, because they do not use transistors. Quantum computers use the spin of certain atoms (which could be a photon, an electron, etc…).

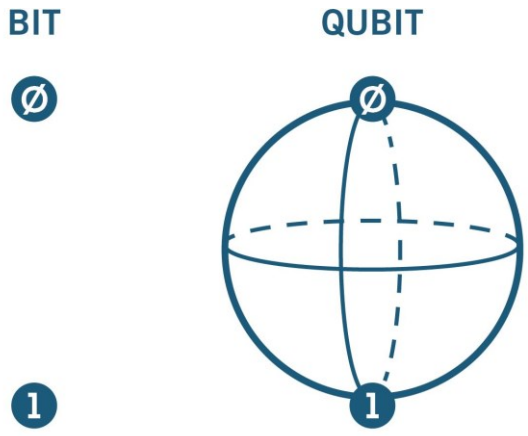


Figure 3.1: Bit to Qubit Comparison. Imported from (Anon 2019.)

The problem arises when the spin is measured. What does spin equate to in terms of bits? There is no solid standard, but scientists have decided to either read the spin vertically or horizontally in such a manner that spinning upwards for example equates to 1 and spinning

downwards equates to 0, equivalently so if the reading mechanism is flipped to the horizontal plane. That partially solves the problem. Scientists then looked into handling the natural fact that atoms spin with any vector and therefore gives the aspect of uncertainty while reading qubits. This is the first major difference between a bit and a qubit. While a bit is either a 0 or a 1, a qubit can be anywhere between 0 and 1 with a certain probability.

3.1.2 Deterministic vs Probabilistic

Classical computers are said to be deterministic in a sense that when a calculated operation is done it's simply done. The calculation, if possible, will end in a result. However, quantum computers are considered to be probabilistic, a certain operation might end up with many different results with different weighted probabilities. Also, for operations to be possible on quantum machines, then they would also have to be reversible. This quantum mechanics phenomena could be considered as traveling in time. A deterministic problem will be explained with the next example. Consider 6 bags of marbles, the marbles in one step will travel from one bag to another in such a manner: (Yanofsky and Mannucci 2008)

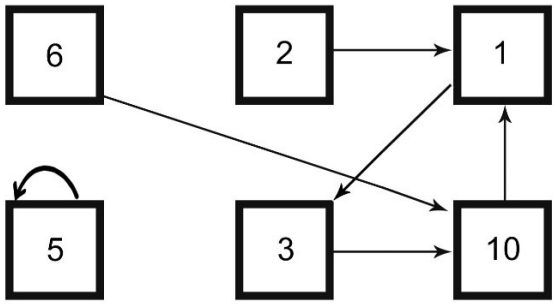


Figure 3.2: One swoop move marble problem (Yanofsky and Mannucci 2008)

The connections in the system could be easily represented in a matrix M

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

The moves in the system could be represented as $X = [6 \quad 2 \quad 1 \quad 5 \quad 3 \quad 10]^T$. The result of this system is just the multiplication of M with X

$$MX = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 6 \\ 2 \\ 1 \\ 5 \\ 3 \\ 10 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 12 \\ 5 \\ 1 \\ 9 \end{bmatrix}$$

The resulting vector $[0 \quad 0 \quad 12 \quad 5 \quad 1 \quad 9]^T$ is the end state of the given problem, and this is a deterministic solution, this problem would not be solvable on a quantum computer because M has no inverse and reversing this multiplication would not be possible and thus this violates the time traveling basis needed for quantum computing.

Consider another problem with the same concept just different parameters and a probabilistic angle

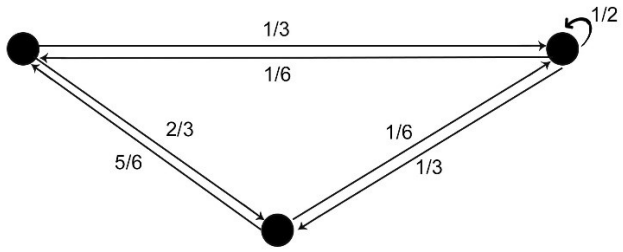


Figure 3.3: One marble move problem (Yanofsky and Mannucci 2008)

$$\mathrm{M} = \begin{bmatrix} 0 & 1/6 & 5/6 \\ 1/3 & 1/2 & 1/6 \\ 2/3 & 1/3 & 0 \end{bmatrix}$$

In this machine M would be considered the original state, with the fraction being a probability. The sum of each row and column would equal to 1. What will be injected next is actually another state $\mathrm{X} = [1/6 \quad 1/6 \quad 2/3]^T$. This second state is one of many different combinations. Then take the product of M and X.

$$MX = \begin{bmatrix} 0 & 1/6 & 5/6 \\ 1/3 & 1/2 & 1/6 \\ 2/3 & 1/3 & 0 \end{bmatrix} \begin{bmatrix} 1/6 \\ 1/6 \\ 2/3 \end{bmatrix} = \begin{bmatrix} 21/36 \\ 9/36 \\ 6/36 \end{bmatrix}$$

The end state now gives us a 21/36 chance that a marble would be on vertex 0 and so on. The initial state M is invertible and thus making the whole procedure reversible.

3.1.3 Superposition and Entanglement

The system described above is based on the spin of atoms and the probabilistic nature of this spin. However, in the world of quantum mechanics, the rabbit hole goes deep. It seems that the deeper scientists dig in the quantum realm the weirder the laws of physics become. To start with the second example above, the probability of a marble landing on vertex 0 in the end state was shown to be $\frac{21}{36}$, but in reality, the result is not known until the initial measurement. The spin of the atoms is obscure to us as third-party viewers, the marble can be on any vertex between 0,1, or 2. This phenomenon is known as superposition, or the fact that an atom could have any value at a certain moment. In quantum computing this is usually mistaken as a qubit being a 0 and a 1 at the same time, technically this is incorrect as a qubit could only be either, but the actual value is unknown. When a qubit is measured, the system essentially breaks down and whatever value is shown will be the value forever. This problem could be easily circumvented by creating clones of the system and tweaking the parameters and shift the probabilities to get the needed results (this will be discussed further in **Section 3.1.4**). Entanglement on the other hand represents the phenomena in which two or more atoms are connected to each other in a non-physical way. These atoms do not have to exist close to each other. They affect each other in such a way that flipping one atom in a certain way could flip the entangled atoms altogether. This will be used later on to exponentially increase the power of every calculation on a quantum machine.

3.1.4 Complex Numbers and Interference

The knowledge presented in previous sections discussed most of the basics of quantum computing. However, the one missing pillar is the operations that are applied on qubits. Many logical gates are available specifically for quantum computers, some that are especially designed to handle many qubits at the same time. These gates allow quantum computers to be probabilistic machines. What was not mentioned earlier however is that these probabilities could actually be complex bound. Moving from one state to the other could be either constructive or destructive. This could be easily represented in Young's double slit experiment.

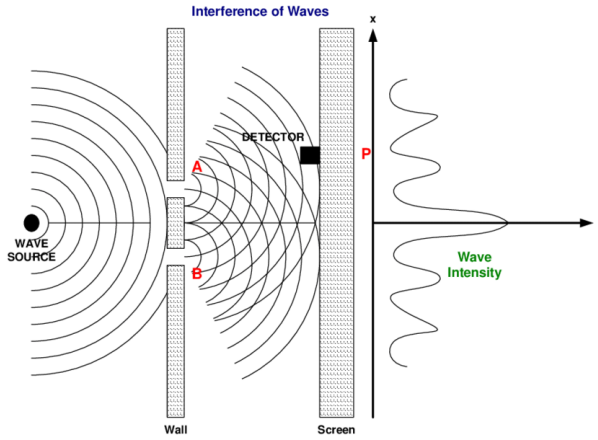


Figure 3.4: Double slit experiment (Carnal and Mlynek 1991)

Figure 3.4 represents the typical double slit experiment. At the middle of section C, the darker area represents the over lapping of waves. Figure 3.5 shown below represents a version zoomed into said area.

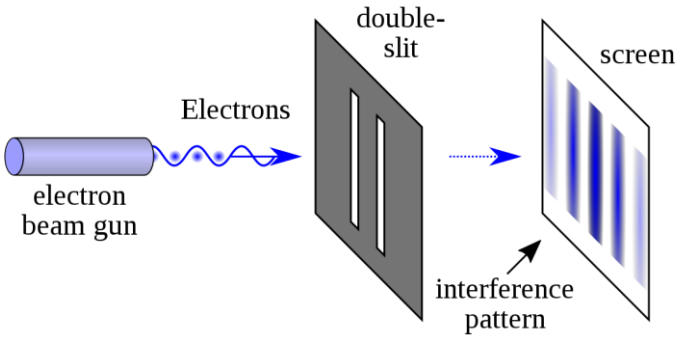


Figure 3.5: Separation of waves (Carnal and Mlynek 1991)

When looking into the magnified version of the double slit experiment, the separation between groups of photons that hit the screen on the other side could clearly be seen. This in a quantum mechanics system is seen as interference. Some waves will combine to amplify themselves, and others will simply destroy each other and disappear. The advantage of this phenomena is that tweaking a quantum algorithm in such a way to amplify (increase the probability) a correct answer while destroying the wrong ones is actually possible.

3.1.5 Filters

This section introduces a fundamental phenomenon in physics known as polarization filters. Polarization refers to the direction of which a certain atom spins. Polarization filters are responsible for filtering out atoms that spin in a certain direction. For instance, a horizontal filter it will only filter out atoms spinning on the horizontal plane. Other atoms that are spinning on the vertical plane will pass right through. Consider the vertical polarizing filter in the figure below

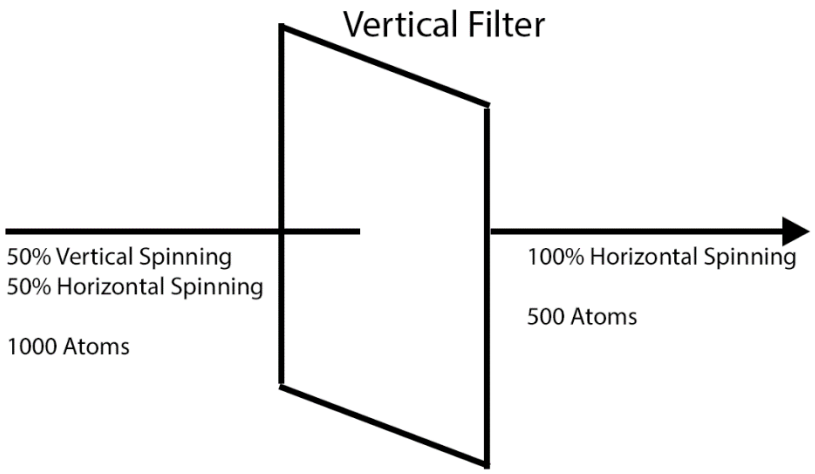


Figure 3.6: Vertical Filter polarizing 1000 atoms of equal divided spin

Figure 3.6 represents a ray of randomly spinning atoms that are equally divided between vertical spins and horizontal ones. After they pass through the filter, all the vertical spinning atoms were cancelled and only the horizontally spinning ones remained. Now, another filter that is horizontally polarized would be added after our first filter, represented in the figure below.

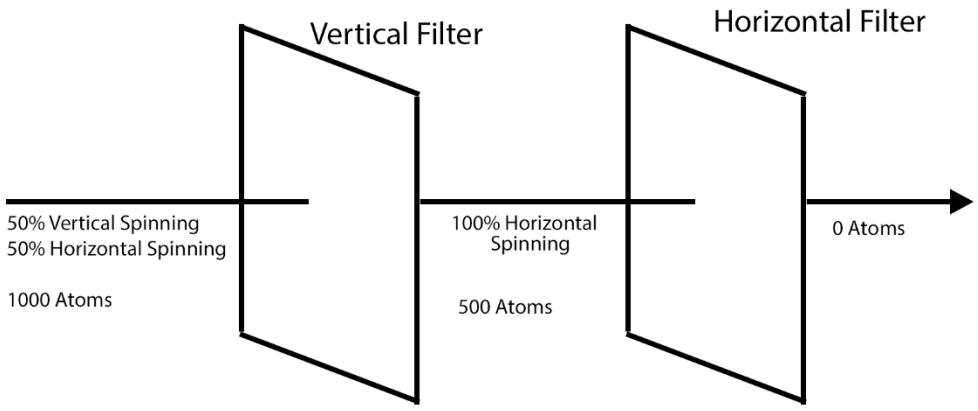


Figure 3.7: Horizonal Filter polarizing the remaining atoms

Obviously, the horizontal filter will block any remaining atoms, and with the result is 0 atoms at the end of our experiment. Quantum mechanics becomes more interesting when a 45º angled filter is introduced in the middle. Normally, more filters imply a higher filtration power resulting in less and less outcome at the end.

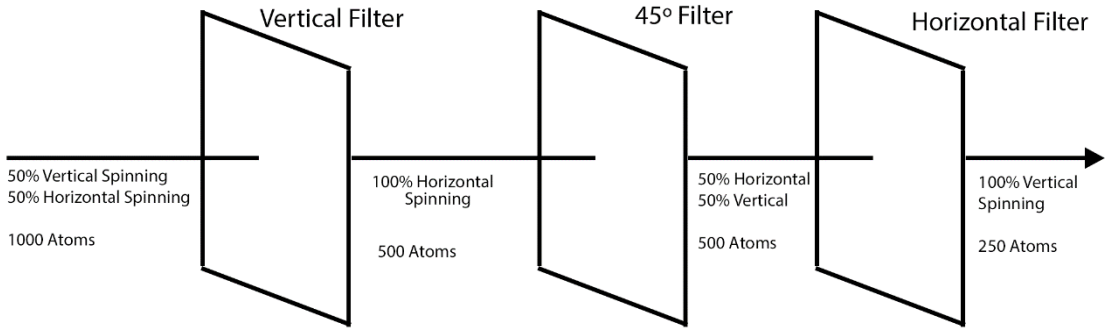


Figure 3.8: Effect of added 45º Filter

As Figure 3.8 demonstrates the addition of a 45º filter actually adds an uncertainty to the spin of atoms, where half of the atoms would spin horizontally and the other half would spin vertically. The end result in this case would be 25% of the original amount of atoms spinning vertically instead of no atoms at all.

3.1.6 Quantum Gates

Most computer scientists refer to their programs with the code they have written. We always think in terms of bits and bytes, and not the hardware level. On the hardware level, everything on a classical computer is changed to Boolean algebra which is handled by certain hardware

logic gates. This concept also applies to QCs (Quantum Computer), with the addition of specialized gates to handle quantum logic. These gates are then referred to as quantum gates. The hardware expects some kind of bit string input $\{0,1\}^n$. The most common classical gates are NOT, OR, AND and NAND. These gates are used to transform a certain bit string input to an output. Then the output is represented as a Boolean function $f: \{0,1\}^n \rightarrow \{0,1\}$ of the input bit (Akama 2015). Quantum gates were first proposed by (Deutsch 1985) and were designed in a manner to function quantum mechanically. Technically, these gates are different from classical logic gates, but they can simulate them. Quantum gates' operations are represented in unitary transformation on states. At the end of an operation, qubits are measured, and a real number is obtained from the operations carried out by quantum gates.

3.1.6.1 Abstract Design for Quantum Gates

Quantum gates have to abide to 3 operations and follow them no matter their function. These operations being:

1) Initializing qubits as a computational base
2) Apply unitary transformations to initialized qubits
3) Compute the probability amplitude

These quantum gates have to also adhere to the concept of runnable operations on quantum computers. The main points they have to handle would be:

1) Superposition
2) Parallelism
3) Reversibility

Using the unitary matrix U, and to satisfy reversibility then must adhere to the following $UU^* = U^*U = E$. This would allow us to recalculate the input by only looking at the output and knowing the operations. With this logic in mind, the AND gate is considered to be a non-reversible operation. By only checking the result of an AND gate the original given input could not be deduced. If the result of an AND gate is 0, all that is known is that one of the inputs must have been 0, but which input if not both is 0 remains unknown.

3.1.6.2 Common Quantum Gates

As discussed in the previous section, quantum gates should be designed to be reversible. Such that, the result would have enough information to calculate the input. This section will discuss a few of the most used gates.

Figure 3.9: Representation of a NOT gate

This is the representation of a simple NOT gate that flips a bit value. Another NOT gate was added to quantum gates and it was named the controlled NOT or CNOT for short, represented in the figure below.

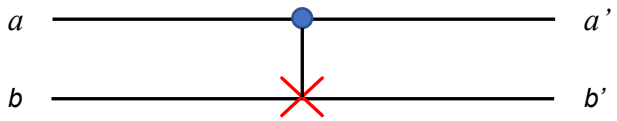


Figure 3.10: Representation of a CNOT gate

The CNOT gate handles two inputs and two outputs. The input line with the blue circle is a controlled line while the line below with the red X symbol is an input-output line. If these lines were exchanged, the resulting gate would be equal to the original one. This is gate's computation is controlled by the input to the control line. If the control line is 0 then the input does not trigger the NOT function and the opposite is true. It shows that $b' = a \oplus b$

a	0	0	1	1
b	0	1	0	1
a'	0	0	1	1
b'	0	1	1	0

Table 3.1 Truth table for CNOT

This is reversible in the sense that the identity gate is available when two CNOT gates are connected together. The resulting gate would be named the CCNOT gate or the Toffoli gate.

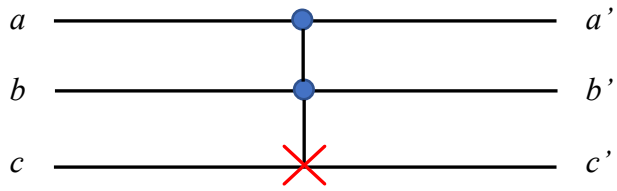


Figure 3.11: Representation of a Toffoli gate

This gate works in the same manner as CNOT whereas there are two control lines instead of one. The resulting equation would be $c' = (a \wedge b) \oplus c$ and its truth table as represented below

a	0	0	0	0	1	1	1	1
b	0	0	1	1	0	0	1	1
c	0	1	0	1	0	1	0	1
a'	0	0	0	0	1	1	1	1
b'	0	0	0	1	0	0	1	1
c'	0	1	0	1	0	1	1	0

Table 3.2 Truth table for CCNOT

The third gate I would like to mention is the SWAP gate. All the SWAP gate has to do is exchange two inputs. It is represented formally by Figure 3.9 below and it could also be represented using a CNOT gate as shown in Figure 3.10:

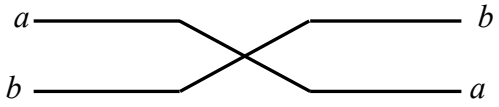


Figure 3.12: Representation of a SWAP gate

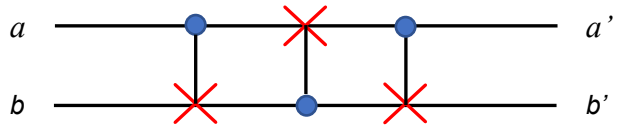


Figure 3.13: Representation of a SWAP using a CNOT gate

3.1.7 No-Cloning Theorem

The No-Cloning theorem focuses on the fact that a state of a certain quantum cannot be copied. It could be measured, but not copied to replace another quantum state (Wootters and Zurek 1982). This might seem inconvenient in a way when measured against a classical computer, but this plays an important role in quantum mechanics. Referring back to the unitary transformations mentioned in earlier sections, this theorem is a consequence of the transformation. Assume two quantum systems A and B, the state of quantum system A is defined as $|\Psi\rangle_A$. While system B has a basis state $|\mathrm{e}\rangle_B$, which is independent of the state of A. The composite of both systems would be denoted as $A \otimes B$ with the states:

$$|\Psi\rangle_A|\mathrm{e}\rangle_B$$

Assume a unitary U that would act as a cloning function then:

$$U(|\Psi\rangle_A|\mathrm{e}\rangle_B) = |\Psi\rangle_A|\Psi\rangle_B$$

$$U(|\emptyset\rangle_A|\mathrm{e}\rangle_B) = |\emptyset\rangle_A|\emptyset\rangle_B$$

For our assumption to be correct, then our unitary operation should apply for all values of $|\Psi\rangle$ and $|\emptyset\rangle$. Using scalar product with the Bra-Ket notation the following could be extrapolated,

$$\langle e|_B\langle\emptyset|_A\, U * U \mid \Psi\rangle_A|\mathrm{e}\rangle_B = \langle\emptyset|_B\langle\emptyset|_A|\Psi\rangle_A\Psi\rangle_B$$

$$\rightarrow \langle\emptyset|\Psi\rangle = \langle\emptyset|\Psi\rangle^2$$

This implies that Ψ and $\emptyset$ are orthogonal. But they do not hold in general and can be shown that no operation generates the clone of the qubits. The main reason why cloning a qubit is impossible is because of the most important rule of a quantum system. Once the system is measured the system breaks down, the measurement is performed probabilistically and destructively.

3.2 History of Quantum Computers

The earliest research into the theory of quantum computation is attributed to many different scientists in the late 70s and early 80s, however the earliest mention about quantum computers is attributed to physicist Richard Feynman in his 1960 lecture *"There's Plenty of*

Room at the Bottom" (Vasconcelos 2020). Physicists and computer scientists started researching and exploring devices that leveraged the power of quantum mechanics. At the time, scientists were adamant that they were on the fundamental limits of computation (Prashant 2005). Only a handful of scientists started working on an actual physical quantum device mainly, David Deustch of the University of Oxford, Charles Bennet of IBM. The first key came in 1982 when Feynman constructed an abstract model to show how a quantum system could be used in computations and explained the idea of physical simulations. After further analysis, he also discovered that such devices can solve problems that would just be impractical for classical computers, that relates to the fact that a classical computer would require exponential time to solve while a quantum device could leverage the "*weirdness*" of quantum mechanics to solve in polynomial time. In 1985, Deutsch extended on Feynman's work and asserted that Quantum machines could eventually become general-purpose computers. He proved this by designing a model that could perfectly simulate a Turing machine and thus solving the same problems a classical computer could. No traction or interest occurred for quantum computers between 1985 and 1994 when mathematician Peter Shor devised a quantum algorithm for factorizing numbers in polynomial time and suggested in his paper that this could be extrapolated in such a manner to crack security that depends on keys that were multiples of combinations of numbers. This was a huge advancement in both Cryptography and Number Theory. Chapter 4 details further Peter Shor's algorithm (famously known as Shor's algorithm). After Shor's publication and with more public interest, the first physical quantum computer was demonstrated by Isaac Chuang and Yoshihisa Yamamoto. The basis of their approach was to tackle Deutsch's problem by having Alice select a number x between 0 and $2n-1$ and mail this number to Bob. Then $f(x)$ a function of x is calculated and the result (either 0 or 1) is sent back to Alice (Chuang and Yamamoto 1995). The challenge is for Alice to know what $f(x)$ is by asking Bob the minimal amount of questions. How long will this operation take? Classically speaking Alice's message can only contain one value of x. In the worst-case scenario Alice will need to ask Bob $n + 1$ times at least using the best deterministic algorithm. In other words, if Bob's answer should be 1, he might send 0 n times before finally sending a 1.

In a quantum system this would behave extremely differently. Now Bob and Alice would exchange qubits instead of normal bits. Alice in this case would need to send Bob a qubit

that is in super position with all the values between [0, $2^n - 1$], thus covering all the possible values. When Alice receives the reply, she cannot simply check it, because if the system is measured then it would break down into one answer which might not be the correct answer. She should instead shift the spin of the atoms by π relative to x and send them back to Bob. After Bob's second calculation, all the values that are incorrect will be out of phase leaving the correct solution to be calculated by an interference experiment. The solution is now found in 2 steps instead of $n + 1$. The change from classical to quantum with the addition of unitary functions using phase shifts has changed the complexity of the problem from $O(exp\ n)$ to $O(n)$. This quantum computer was demonstrated in 1998 and had 2 qubits. At the time of writing this thesis, the most powerful quantum computer stands at 64 qubits and designed by American company Honeywell.

3.3 The Machine

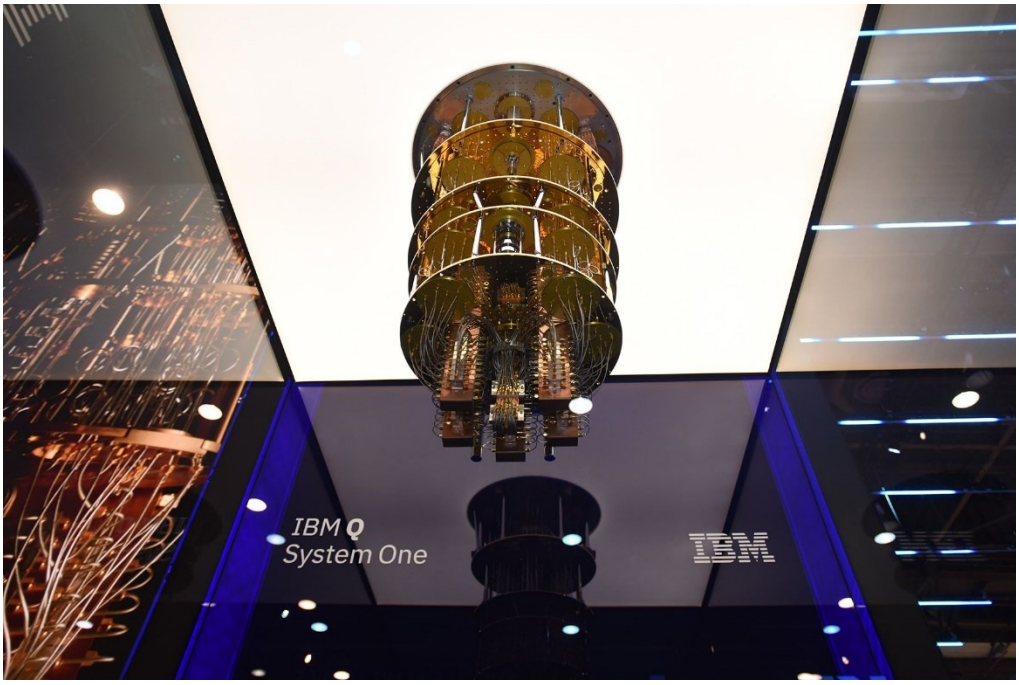


Figure 3.14: An IBM Quantum Computer (Nick Summers 2019)

It is important to talk about the hardware powering a QC. Knowing the hardware will make it easy to understand what the pros and cons of a QC are and would also help emphasize how new algorithms will take advantage of the available pros to become more efficient from their

classical counter parts. This section also discusses why current QCs cannot scale as much as scientists would like and what approaches are being tested out to solve this problem.

3.3.1 Quantum Data Plane

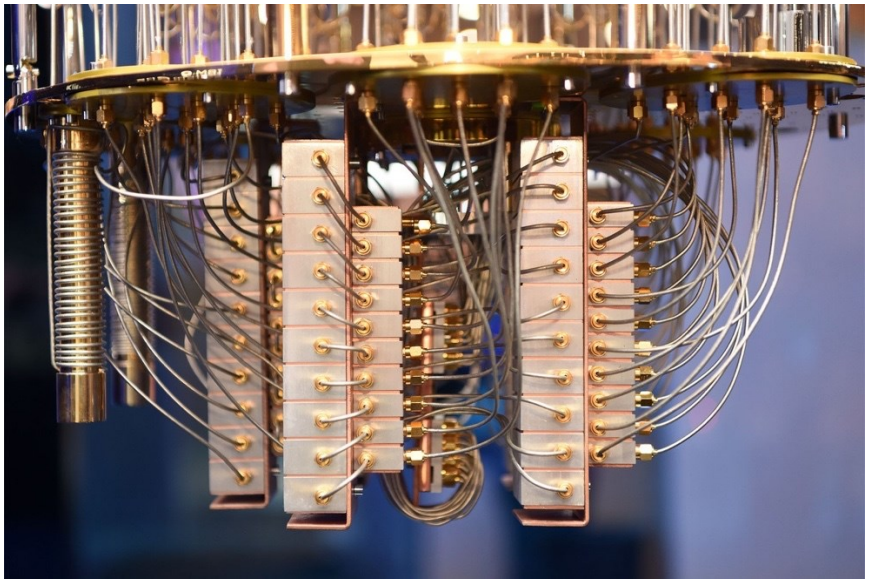


Figure 3.15: The Data Plane in a Quantum Computer (Nick Summers 2019)

The quantum data plane is the part of a QC that holds all the physical qubits and all the structures required to hold them. It is considered to be the 'heart' of the machine. Moreover, it usually contains hardware used to measure the state of the qubits and also to perform gate operations on them. For gates that compute more than one qubit at a time a programmable network is needed for the qubits to communicate together. It might not be possible for every qubit to interact with another directly because of some limitations and isolations. A workaround for this would be to design some architectural constraints from the ground up keeping in mind all these constraints. This shows that operation fidelity and connectivity are both important for the data layer. In a classical computer the control plane and data one's components are usually based on the same silicon technology and are integrated on the same device. However, this is not the case in a QC. The control of the quantum data plane requires technology vastly different than that of the data one. This control layer is usually built externally, analog in its nature, and needs to be able to communicate with the correct qubit. This could be achieved electrically using wires (where the wires would be considered part of the data plane) or transmitted via optical or microwave radiation. The second method should

be implemented with high specificity, such that it does not affect the wrong qubits. Using any of these techniques would offer an increasing difficulty as the number of qubits increase. This represents to us the concept of “errors” in a quantum machine and is usually referred to as the error rate of a certain system.

3.3.2 Control / Measurement Plane

This plane is responsible for converting the digital signal from the control processor to analog signals to perform the operations on qubits in the data plane. It is also responsible for converting the analog output to binary data for the control processor to handle. This process is tricky because quantum gates work in an analog manner. Any small error in this process would affect the final result and these errors would just accumulate as the machine runs. These errors could occur because of a bad isolation to these signals. These errors are considered systematic, and could be minimized through periodic calibrations, if they could be detected somehow (e.g. by solving problems with known outcomes). Every control signal is connected to every other control signal in the system which more than doubles the computation required to calibrate correctly as the number of qubits doubles. The technology used for the controlled signals relies heavily on the technology used for the qubits (will be discussed in a later section). Classical gates have been heavily developed over the past half century and they now have noise immunity and negligible error rates. Quantum operations on the other hand depend on the precision of which control signals are delivered. These usually contain the highest forms of error rates which could be affected by the environment of the machine. Currently, these gates are fast and are only hampered by the speed of the data plane and not the control and measurement plane.

3.3.3 Control Processor Plane

This plane is responsible for identifying and triggering the sequence of gate operations and measurements (Watson et al. 2018). These sequences execute as a program provided by the host processor. The most difficult task for the plane is to carry out an error correction algorithm. These algorithms require lots of preprocessing on a classical computer, and this significantly slows down the quantum computer. The only way to speed this up would be if the error correction algorithms is optimized to match the speedups of the quantum computers. Constructing such a plane that could handle error-correction proved to be a very challenging

area of research. One approach is to split the plane in two. The first part runs the quantum program while the second is a scalable hardware block that interfaces with the measurement plane. This plane operates in an abstract manner. It converts code to commands for the measurement layer. The user does not need to understand how the control plane works, rather the user will need to depend on a host computer. This will work in the same manner in which classical computers simply use different expansion cards to handle different tasks while reporting back to the central CPU. In this case the host computer is an actual classical computer, that will provide all the software development. Using this approach, researchers do not have to start entirely from scratch.

3.3.4 Approaches to Qubit Design

There is no one standard that fits all when it comes to qubit design. Lots of researchers are working on many different types of qubit designs with varying degrees of success. This section will talk more about two designs and the control planes needed for these designs.

3.3.4.1 Trapped Ion Qubits

This design is considered to be one of the oldest. It was actually demonstrated in 1995 with the help of technological advances of the time (Monroe et al. 1995). This allowed for testing on a small scale for a wide range of quantum algorithms. Scaling this technology has proven to be challenging for a few reasons. Mainly because of the varying technologies required such as vacuum, laser and optical systems. These areas and others must be addressed before scaling. The idea of a trapped ion quantum data plane contains ions that are considered our qubits and trapping these qubits at specific locations. The control plane consists of a laser that can be directed at a qubit to change its state. It also contains another laser and a set of detectors that enables the measurements of these qubits. A 20-qubit system that’s formed to act as a general-purpose quantum computer has error rates between 2 and 5 percent, and this takes into consideration the use of extremely high-fidelity components. As the number of qubits increases, error rates increase as well. However, the positive aspect of these qubits is their versatility that enabled quantum algorithms and tasks to be implemented on them. It is expected that this technology can optimized to scale to 50 qubits in the matter of only a few months (Kendon 2018).

3.3.4.2 Superconducting Qubits

This design is usually referred to as artificial atoms. They are created lithographically; they share design with current silicon integrated chips. They are cooled to milli Kelvin temperatures where they exhibit quantized energy levels. This qubit design is considered to be the current leading design because of its compatibility with microwave control electronics, operation at nanosecond time, continually improving coherence times and potential for scaling. Scaling to thousands of qubits is not beyond our reach using this design, however a few main factors would need to be checked first. This thesis will not go into the factors in details but to list a few:

1) Maintaining qubit quality while scaling up the number of qubits.
2) Refrigeration, wiring and packaging
3) Control and measurement

3.3.5 Scalability

As mentioned in earlier sections scalability is always an issue when it comes to quantum computers. First factor that quantum computers need to scale is error rate. Error rates of 2 to 5% are unacceptable and extremely non comparable to the 10^{-3} to 10^{-4} error rate of traditional circuits. Effectively qubits will need to reach these low error rates in order to prove viable. Secondly as QCs start to scale, researchers would need to redesign the process monitoring systems, the statistical process control. They will need to find new methods for reducing defects relevant to high coherence devices. Specially designed tools will need to be developed to target specific qubit-fabrication processes. Another thing to take into consideration is the wafer the contains the qubits. Current state-of-the-art today contains qubits with cell sizes of 50 microns. A large IC with 20mm x 20mm dimensions could fir 1,600 qubits. If an entire 300mm wafer were to be used, then it could potentially fit 250,000 qubits. While this seems like a huge number, this would need to be scaled even more, and for that to occur the size of qubit will have to decrease so that a 300mm wafer could be denser in qubits. Controlling more qubits will also require a new strategy for the control plane. Currently, each control signal is driven externally. Some control signals have to move closer to the qubits. Moving these signals closer to the qubits implies that our control plane will now also have to operate at milli Kelvin temperatures, so this plane will have to work

efficiently and not generate lots of heat. At present, the technology required to work at such cool temperatures such as cryogenic CMOS chips exists. 300mm wafers are also considered to be extremely large, in a perfect scenario, researchers would want to divide this wafer into multiple subsystems and this would require that these subsystems communicate with each other. Currently this is being worked on with two general approaches. The first option requires the assumption that there exists an interconnection between the modules at milli kelvin temperatures, such that photons can be used to communication. This involves creating channels for the photons, so they can communicate with the qubit and convert the quantum information back to another qubit. The second option is to couple the state of the qubit with a higher energy photon. This requires a high-fidelity microwave-to-optical conversion technique.

3.4 Quantum Supremacy

Quantum computers have always shown promise in their power of computation. They are predicted to be able to solve problems that classical computers simply cannot solve. This promise was named Quantum Supremacy. Quantum supremacy's formal definition is the ability of quantum computers to solve problems that would have not otherwise been solved. In 2019, Google engineers published a paper claiming and demonstrating a problem solved using a quantum computer in 200 seconds and calculated that it would have taken the largest supercomputer currently in production about 10,000 years to solve (Arute et al. 2019). As the engineers' set it out, they needed to figure out two questions: first, can a quantum system perform in a large Hilbert space with low error rates? And second, they asked themselves if they could actually find a problem that is near impossible for classical computers but relatively easy on quantum machines? As mentioned by their publication a number of technical advances were made that helped with error corrections. The problem that researchers created was designing circuits to entangle a set of qubits, then they took samples of outputs created by their circuit. Their output consisted of bit strings. Due to interference some of these strings had a higher probability of appearing than others. Classically computing these probabilities becomes exponentially harder by adding more qubits. These methods are verified again on their quantum processor. The important part about this experiment is

calculating the cost on a classical machine. The importance behind it is twofold. Firstly, it would verify the fidelity of the quantum processor and their benchmarking technique. Secondly, estimating the cost with minimal errors on a small scale would help build better estimations for harder circuits. The tests were conducted in such a manner that started with a very small scale, with a 2-qubit computer, and scaled up to the full 53 qubits that Google's quantum computer had. What the engineers found is they could test and simulate the full quantum state of 43 qubits using Schrödinger's algorithm and the Julich supercomputer which has 100,000 cores and 250 terabytes of RAM. For more than 43 qubits, engineers had to use Google's datacenters as the Julich supercomputer ran out of RAM. They also needed to tweak their algorithm and use a hybrid of Schrödinger-Feynman algorithm to break up the circuit into two patches of qubits and efficiently simulate each patch using the Schrödinger algorithm. Then, they connected the patches back up using Feynman's path integral algorithm. Calculating on a small scale and extrapolating, engineers at Google figured that their 53-bit quantum computer could solve their problem in 600 seconds, however it would take 50 trillion core hours and consume one petawatt hour of energy to achieve this result on a classical computer. If these numbers are matched to a 200-second-long problem on a quantum computer to the Summit supercomputer (the most powerful supercomputer), it would take 10,000 years to solve this problem, which implies that this problem is basically unsolvable on a classical computer.

Chapter 4: Quantum Computers and Their Effects on RSA

Peter Shor's paper about factorizing large numbers did act as a catalyst for advances in quantum computing. The main reason however is driven by the implications that factorizing large numbers would lead to the devastation of a few cryptographic algorithms. RSA, specifically, had the most to lose. RSA stood the test of time and the idea of even a slight chance at cracking it seemed like a great motivation for many researchers and engineers. This chapter will go through the theoretical and applicable part of approaching this problem.

4.1 Shor's Algorithm

Peter Shor went through the trouble of giving his paper the very long title of: "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer". His research went viral, and ever since it did, everyone started referring to the research as simply Shor's Algorithm. Before going into the details of his research, I would like to personally note out two things. First, this paper was initially published in 1994. Second, something very spectacular caught my attention in the opening page of Shor's research, he noted: "Even if no useful quantum computer is ever built, this research does illuminate the problem of simulating quantum mechanics on a classical computer" (Shor 1997). This statement is powerful in the sense that the research itself was based only on the mathematics and physics, with only the technology available at the time, proving once again, the progressiveness of Peter Shor's research.

Shor's algorithm heavily depends on results from number theory. The main result which is the function $F(a) = x^a \ mod \ n$ being a periodic function. *X* would be a coprime of *n* (*n* is the number to be factorized) where $gcd(x, n) = 1$. Considering an exponential number of values of *a*, this would take at least exponential time using classical machines. However, Shor's research took advantage of quantum parallelism to be able to calculate the value *a* in a single step. Knowing that *F(a)* is periodic implies that there exists some period *r*. Moreover, $x^0 mod \ n = 1$ and this extrapolates to $x^r mod \ n = 1, \ x^{2r} mod \ n = 1$, etc. Every time an

exponent is a multiple of *r* then the period has reset. $x^r \equiv 1 \, mod \, n$ is found using simple algebraic manipulation. Now take the square root of x^r such that $(x^{r/2})^2 = x^r$.

Then extrapolate that

$$(x^{r/2})^2 \equiv 1 \, mod \, n$$

$$\rightarrow \; (x^{r/2})^2 - 1 \equiv 0 \, mod \, n$$

$$\rightarrow \; (x^{r/2} - 1)(x^{r/2} + 1) \equiv 0 \, mod \, n \; (if \; r \; is \; even)$$

$(x^{r/2} - 1)(x^{r/2} + 1)$ in the case above is a multiple of *n*, the number that needs to be factored. The only condition in this case is $x^{r/2}$ should not be equal to 1 or -1. This would lead to the multiple to the answer being trivial. To find one of the factors of *n,* calculate the $gcd(x^{r/2} - 1, n)$ and the $gcd(x^{r/2} + 1, n)$. Shor focuses on finding *r,* by creating a quantum memory register (QMR) with two parts. The first part would include a super position of values of *a* that satisfy $x^a mod \, n$ where *a* would be between 0 and *q-1* and $n^2 \leq q \leq 2n^2$. The algorithm calculates $x^a mod \, n$ then places the results in the second part of the QMR. After that, the state of the second register is measured. Once it is measure it will collapse into ac certain value *k*. The first part of our QMR will now contain values $c, c + r, c + 2r$, ..., with c being the lowest value that satisfies

$$x^c \, mod \, n = k$$

After that the algorithm performs a Quantum Fourier Transform (QFT for short) to amplify the probability of the first part of the QMR. Measuring the first part of the QMR, the register will now contain a multiple of the inverse period. This number is given to a classical computer for some analysis, a guess is made on the actual value of r, and then the algorithm computes the possible factors of n. The improvements were clear to Shor as the best classical algorithm by Lenstra et al. at the time, had a running time (Lenstra and Verheul 1990)

$$\exp\left(c(\log n)^{1/3} (\log\log n)^{2/3}\right)$$

Although many algorithms were defined since then, not much improvement was made to the running time on a classical machine. Shor's algorithm leveraging the power of quantum computers has a running time of

$$O((\log n)^2 (\log \log n) (\log \log \log n))$$

This is a huge leap from the exponential realm to the polynomial one. No quantum computer is strong enough to be able to carry out this calculation yet, however the next section will talk about a potential approach on physical quantum computers.

4.2 Shor's Algorithm in Practice

As previously stated in Chapter 3, Shor's algorithm acted as a great catalyst for the creation of quantum computers. Shor's algorithm was successfully tested on a small scale and was able to factor tiny number such as 15 and 21 (Kendon 2018). This proves that the algorithm is valid and can result in a correct factorization. However, with recent research it became evident that Shor's algorithm would require a bit of tweaking and optimization to be viable (Gidney and Ekerå 2019). This section will focus on the latest effort devised by Gidney and Ekera in 2019.

4.2.1 General Algorithm

The general algorithm used is a slight variation of Shor's algorithm. From a quantum aspect there exists two changes. First, create two exponents e_1 and e_2, with lengths *2m* and *m* qubits respectively, *m* in this case would be positive and would have to satisfy the equation $p + q < 2^m$. Second, the period finding function is changed from $f(e) = g^e$ to $f(e_1, e_2) = g^{e_1} y^{-e_2}$ This would result in an improvement from the *2n* qubits required by Shor's algorithm to *1.5n + O(1)* qubits in the exponent length. This would also result in a reduction in the number of multiplications required on the quantum computer. Equivalently to Shor's main algorithm, the function is divided into two exponent registers of uniform superpositions of 2^m and 2^{2m} respectively. This would allow for all standard optimizations such as square-and-multiply and the recycling of qubits to be implemented. The part that differs the most is the 2nd part of Shor's algorithm that depends on post processing to be done on a classical computer. In this case *r*, the initial powers of *N* the number being factorized, is not required to be known. However, the algorithm uses a lattice-based technique to recover *d* (equaling to $p + q$) with an accuracy of 99%. With $N = p * q$ and $d = p + q$, p and q are recovered deterministically.

Comparing the two algorithms it is obvious they are similar from an implementation perspective with just minor tweaks.

4.2.2 Optimizations

Before any optimizations could be implemented, a circuit needs to be built that would effectively run Shor's algorithm. Quantum circuits are built in the same manner as transistor-based ones with their respective gates.

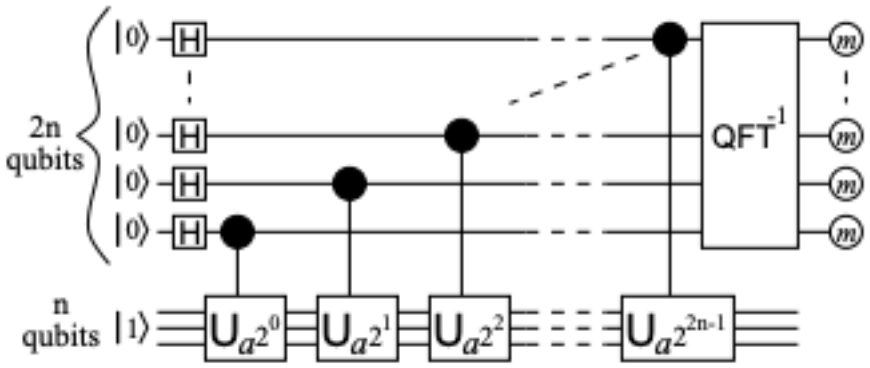


Figure 4.16: A Circuit Representation Of Shor's Algorithm (Beauregard 2003)

In Figure 4.16, *H* represents the Hadamard gate (the gate responsible for superposing qubits) and U_a implements $|x\rangle \rightarrow |(ax)\ mod\ N\rangle$ and the measurements followed by classical postprocessing yields the order r of a modulo N with good probability. The first improvement to the circuit would be to switch the integer representation into a coset representation. The default representation is visualized using a Bra-Ket representation, where the value k would be represented as $|k\rangle$. However, a coset representation would be a periodic superposition with period *N* and an offset *k*. Ideally, $k\ mod(N)$ would be represented as the following state

$$\sqrt{2^{-c_{pad}}} \sum_{j=0}^{2^{c_{pad}-1}} |\ jN + k\rangle$$

C_{pad} just represents the extra shifts added to the qubits at the end of each register. This optimization would show that the periodicity of the superposition causes a non-modular adder to perform approximate addition, while also decreasing the error rate by a huge factor by adding paddings to the register. This padding would be logarithmic in the number of operations. The second optimization is considered arithmetic windowing by using modular

exponentiation construction at two levels of the circuit. First window will include the controlled additions in the multiplication level of Shor's algorithm. These groups will be joined into lookup additions. The result of the lookup consists of the value to add into a register. In this case, the windows of qubits that would have normally been additions are now addresses in classically precomputed tables of values to add into the target. The second level, the level of exponentiation, the controlled multiplications are windowed. This is done by including exponent qubits in the addresses of the lookups being performed within the multiplications. The improvements of this optimizations range from marginal to reasonable depending on the size of the input and number of qubits. The final optimization is the use of oblivious carry runaways. In normal cases a carry signal is generated at the bottom of a register and must then propagate to the top of the register. Instead, our aim is to short-circuit the process and construct appropriate runways that allow large additions to be performed divided into pieces. Each piece will be run in parallel; this will lead to the termination of carries. This specific optimization will help in suppressing errors that were discussed earlier in chapter 3. Moreover, it has minimal overhead and could be introduced gradually through the process. The choice of these 3 optimizations is intentional, because they complement each other and do not conflict. Choosing the coset representation in the first optimization is only beneficial for offsets less than N and this could be enforced in the windowed arithmetic of the second optimization that precomputes the lookup tables to be less than N. It is important to note that there are two potential cases where all these optimizations do not mesh together perfectly. The first case is by using oblivious runways that reduce the depth of addition but not the lookups. This in turn affects the size of the windows and the relative costs. The second case occurs when iterating over qubits during a multiply-add. An issue would occur if the runway qubits and padding ones were not iterated over. This could easily be remedied by temporarily adding back the runways back into the register before performing any multiply-add. The runway sizes could be reduced to zero but that would require an extra step of propagating the carry qubit across the register, but this would render the optimization useless. In this case, space takes a hit for a runtime speed improvement.

4.2.3 Implications on RSA

According to recent publications (Gidney and Ekerå 2019), the assumption was that the error rate of the gates their quantum computer was using is 10^{-3}, which in itself implied that the

probability of an error rate in code cycle d was approximately $10^{-\left\lceil \frac{d}{2}+1 \right\rceil}$. The cost estimate of this calculation will be based on this assumption. This thesis will not discuss the exact mathematics behind the runtime but will rather extrapolate the data from the original publication (Check Appendix A for some analysis and further references). Assuming the key used was 2048 bits in size, Gidney et al. divided the runtime as such:

1) Search in a lookup table would take about 14 milliseconds
2) Addition would take an extra 22 milliseconds
3) Remaining operations such as un-computing the lookup, rearranging the rows, etc. will take an extra millisecond

All these operations summed up will take 37 milliseconds. The total would be

$$TotalRuntime(n) \approx LookupAdditionCount(n)\,.37\ milliseconds$$

This rough estimate leaves many aspects up for grabs and ignores details such as in some cases lookups become slower for larger problems, etc. That said the total estimate for the runtime would result in approximately 7 hours for a 2048 bit key, moreover this would need around 23 **million** qubits to cover the whole algorithm and its errors.

4.3 Importance of Early Action

Chapter 4 has been building hype about the importance of Shor's algorithm and the effects it might have on factorizing numbers and RSA. Admittedly, the ending of section 4.2 might seem anticlimactic based on the fact that the best solution researchers found so far would require 23 million qubits while the most powerful quantum machine currently available is powered by 64 qubits. I would like to emphasize a few key points that might explain my worries about the topic. 50 years ago, in the year 1970, the most powerful processor was the Intel 4004 chip running at 740 kHz, the most amount of ram was 1024 bits, the largest hard drive had a storage space of 100MB and weighed as much as a car. Personal computers did not exist. Companies such as Apple and Microsoft were not yet incorporated. The beloved C programming language was not yet released by Dennis Ritchie. However, 50 years later, phones that could fit in a pocket have millions of times more processing power than the

computers that put humankind on the moon. At the beginning of this thesis, I stated my amazement by quoting Peter Shor's remark about the possibility of quantum computers ever being created. This was only in 1994. Research in the field of quantum information has been ramping up over the last decade and countries are pouring billions of dollars into the development of such machines and their potential. The most powerful computer in current existence has 64 qubits; IBM already announced their 128-qubit quantum computer to be released somewhat at the end of 2020. Writing this thesis specifically has been tough, because the technology is moving extremely fast that facts might become outdated by the time this is finished. Last year, QCs could not run at temperatures higher than 0.015 Kelvins, yet at the end of 2019 new research allowed them to run at up to 1 Kelvin, that already is a 65x improvement. The trend, as seen by researchers, will lead to very capable QCs in the next 20 to 40 years. So, the question remains, why should this be an issue this early on? Chapter 5 will discuss that this issue is important enough that the National Institute of Standards and Technology (NIST) has already launched competitions to create the next cryptographic standards. NIST was also responsible for the move from DES to AES encryptions. They are the driving force for these standards and their implementations. To answer the question though, it is extremely important to treat this as a security threat simply because of government standards. According to a report signed by President Barack Obama in 2009, government agencies have to release top classified information no more than 75 years after the year of their collection (The White House 2009). Section 6.1 of NIST's Federal Government Security, states that any government agency must use 2048-bit RSA encryption while transferring data (Barker 2020). Possibly any eavesdropper can just collect encrypted data packets that are transmitted from or to government organizations and store them for later use. If someone stored this data until QCs are capable then this data will be available to the public before their original expiry date. This is exactly why governments are currently extremely invested in Quantum machines. Chapter 5 will focus more on current suggestions for the next wave of cryptographic systems.

Chapter 5: Post Quantum Cryptography

Previous chapters have discussed one of the most important PK cryptographic systems and discussed how QCs affect it. NIST already initiated a Post Quantum Cryptography project to facilitate the process of collecting new approaches and algorithms to combat the shortcomings of our current security. The entry period for proposals ended in November of 2017 (Kendon 2018). 70 proposals were received, and they are in now all being reviewed. As of 2020, NIST has narrowed down the list down to 26 algorithms (Alagic et al. 2019). A final result should occur by the year 2024. This thesis will discuss a few of the candidates and their different approaches.

5.1 Candidates

5.1.1 Candidate Group 1: Lattice Systems – 12 Algorithms

Lattice represents a discrete set of points in space that has the property that the sum of two points on the lattice is also on the lattice. The most well-known problem on lattices is to find a vector in a lattice. Solving this currently takes exponential time in every dimension of the lattice. Early research shows that this case will also hold up on quantum computers. Many systems are already developed to take advantage of the short vector problem, the most recent ones have been the implementation of the “New-Hope” system in the chrome browser by Google. Google reported that this system added less than 20 milliseconds per key exchange. While researchers try to avoid this overhead, this experiment only proves that post-quantum key exchange base on lattice systems are possible and there should be no main struggle implementing them. These systems are a good candidate because they rely on existing and proven technologies. The best way to describe these systems would be to first imagine having a lettuce (like the vegetable) field, and a snail. Figure 5.17 represents a two-dimensional lettuce field. A snail happens to be having its lunch at lettuce A.

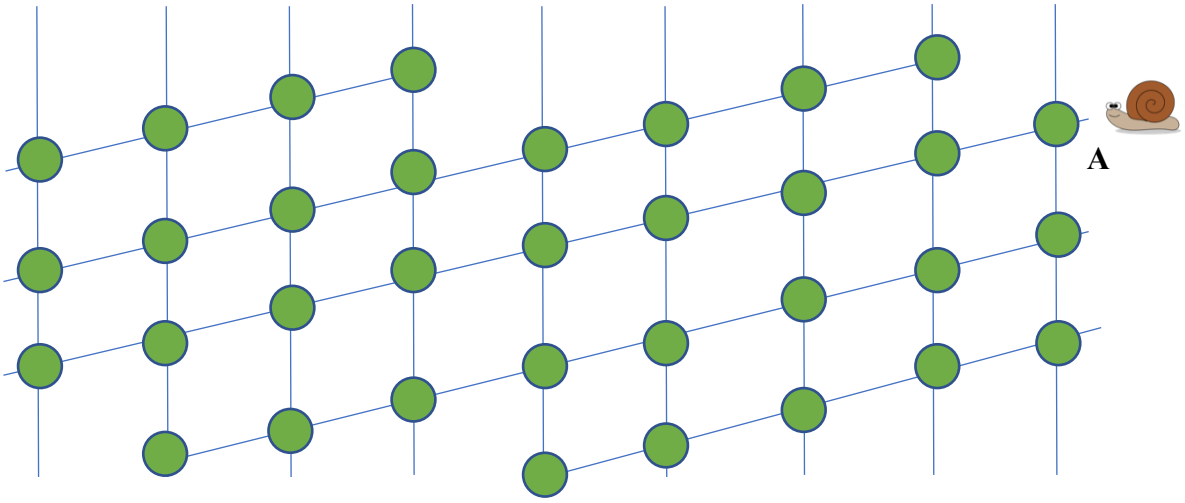


Figure 5.17: Two-Dimensional Lattice System represented by Green Circles and a Snail with starting position at A

The basis of a lattice system is having a vector to move from one lattice till the other. In this example, the snail finished eating at lettuce A, and needs to move to another lettuce or it will die of hunger. Looking at this in a two-dimensional system seems really easy, the snail has in this case 3 choices that would be relatively easy to achieve. Extrapolate this concept to higher dimensions. Imagine a 250-dimension lattice system, mathematically this increases the difficulty exponentially. One more concept to grasp is the concept of a "basis". A basis is achieved by choosing two lattices and using them to encrypt and decrypt. These bases are either good or bad. The basis in red, shown below, forms a shallow angle and that would be considered a 'bad' basis when compared to the basis represented in black, as the latter is closer to being orthogonal and thus forming a better basis

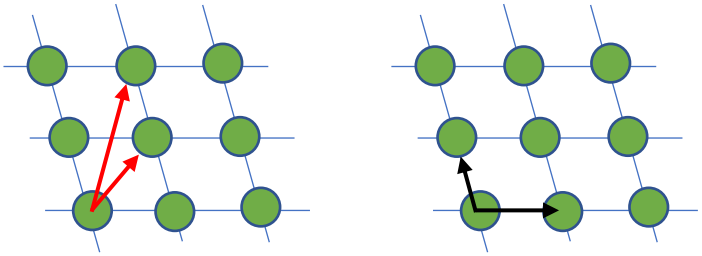


Figure 5.18: The Basis of a Lattice System Represented by Arrows.

5.1.2 Candidate Group 2: Coding-Based Systems – 7 Algorithms

Coding theory is defined as “the science of designing encoding schemes that let two parties communicate over a noisy channel” - (Kendon 2018), basically as long as two parties know what to look for, they will be able to securely converse even if the encrypted message was bounded with noise produced by the channel. Researchers figured that a lot of cryptographic systems that are code based could not be efficiently decoded. In some cases, this could take exponential time on both classical and quantum computers. These systems are already tried and tested and have been found to be really secure. A well-known system is called the McEliece cryptosystem.

Coding-Based System’s original purpose was as stated above error correction. Where a string of bits representing the original message is multiplied by a certain matrix, and thus the resulting message is error correctable. This certain matrix is predefined in such a manner where it would help in creating redundancies, error correction and is usually referred to as a “good” matrix. A good matrix could be changed into a bad one by multiplying it with other matrices in such a manner:

$$BadMatrix * GoodMatrix * BadMatrix = BadMatrix$$

This same concept is used in McEliece algorithm. The way it works is rather simple, the left side of the equation above would be Alice’s private key and the right side would be the public key. Assume a scenario where Bob wants to send a message to Alice. He would have to multiply his plain text bit value with Alice’s public key, then he would have to intentionally add errors to the encrypted message. Alice would receive the message with its errors, she would apply error correction with her good matrix and reverse the encryption and get the plain text. If an attacker intercepts this message, he/she would have to correct the errors using the bad matrix that is available publicly, but this is increasingly difficult and would take a very long time to do, rendering breaking the message improbable. Of course, a matrix multiplication would lead to a bit of overhead. For example, assume the plain text message is 5413 bits, after the matrix multiplication the encrypted message’s size would be 6960 bits, resulting in an overhead of 1547 bits. A key advantage here is that half of the bits in the overhead could be used for error correction. In this case 1547/2 would result in 773

correctable errors. The only downside of this method would be the key size, RSA's PK would result in a 2KB key length, however in this case it would result in 1MB key length.

5.1.3 Candidate 3: Hash-Based Signatures – 1 Algorithm

Hashing based algorithms have been around for a long time. They have been proven to be strong, and quantum computers seem to have no effect on them. The way to implement these algorithms would be in such a matter: Consider Alice and Bob, Alice leaves a message on her computer to Bob. In this case the public and private key would be implemented differently. Firstly, a hashing function such as SHA is needed. Alice's private key would consist of two numbers, while her public key would be the hashes of these random numbers. Moreover, she would need a signature and that would also be one of the random numbers generated above. For Bob to be able to read any of the messages on Alice's computer, she would need to send one of the random numbers, and that will be able to unlock one of her messages. This method has a few down sides such that, every time Alice sends one of the random numbers, she would have effectively sent out half of her private key, and this algorithm could only be used once. Moreover, this works on a bit level so it would introduce more overhead. The keys and signatures would also be really long and inefficient to send.

5.2 Deployment in Practice

Section 4.3 discussed the importance of early action; with the main issue being the time needed for the deployment of a new standard across the whole internet. Coming up with a new cryptographic system and migrating every system on the planet to it will take a long time. To deprecate an algorithm, most of the Internet's systems should be upgraded first. A good example of this would be SHA1, this hashing algorithm was considered to be insecure since 2004 (Kendon 2018). So far, SHA1 is still in use and many systems have not migrated or do not support SHA256 (the upgraded, more secure counterpart to SHA1). This provides a very good idea about the steps required to the transition to post-quantum cryptography. The first step would be to decide on a single standard, and to develop and ratify this standard. After a standard is set, the chosen algorithm must be implemented in many programming languages, software libraries and hardware (cryptographic chips). The next step would require the standards to be incorporated into an encryption format. After revision and

approval, vendors will need to implement the new algorithms. After that, no one could predict how many years it would take for the majority of Internet systems to upgrade and support the newer standards. During this transition phase, old vulnerable algorithms might not be disabled until the wide adoption of newer standards. Even after wide adoption, any date stored by companies and governments would need to be re-encrypted and old copies need to be destroyed. Some companies rely on just destroying the encryption keys and this malpractice would be useless against quantum computers. The importance of this transition and planning for it would rely on three main questions:

1) When will a sophisticated quantum computer powerful enough to run Shor's algorithm be available to the public?
2) How long will the transition phase be? And when will the old standard be deprecated?
3) What's the longest protection interval of concern?

To calculate this with a simple formula created by M. Mosca in 2005, 3 variables are taken into consideration:

- The first variable, X, would represent the longest protection time possible starting today
- The second variable, Y, would represent the migration time.
- The third variable, Z, would represent the time left until large enough quantum computers are built. This would be the biggest unknown at the moment and could only marginally be represented.

	Value
X	5 Years
Y	3 Years
Safety Margin	7 Years
Z	15 Years

Table 5.3 M. Mosca's Formula that defines an optimistic model

	Value
X	7 Years
Y	10 Years
Z	15 Years

Table 5.4 M. Mosca's Formula that defines a pessimistic model

Table 5.3 represents an optimistic model; it assumes that a quantum computer capable enough would not be designed for at least 15 years. It also assumes the time for collapse would also equal to 15 years. With the assumption that it would take 3 years to design and deploy a post-quantum system and 5 years for the longest shelf life, this would leave a 7-year safety margin. This optimistic calculation would allow for a bit of time for more optimizations. Table 5.4 represents a pessimistic model. It assumes the same time requirements for the creation of a quantum computer. However, the migration time has been set to 10 years, with an extra 3 years for implementation and deployment. There will be no time left for a safety margin. In this scenario, data that exists today is already at risk. Building on past experiences, the reality is bleaker than the pessimistic scenario. When DES was replaced by 3DES and when MD5 and SHA were replaced, it took over 10 years to transition to the new algorithm. If the shelf life remains at 7 years as assumed and if a quantum resilient standard was released by NIST in 2023, this would result in a transition deadline by 2040. This timeline gives enough interval for QCs to become powerful enough. Reality of the situation is that what might happen with the technology is unknown, as advancements in quantum theory are a daily artifact and it would be almost impossible to estimate where QCs would be by 2040.

Chapter 6: Hybrid Computing Networks

So far, this thesis has discussed the RSA cryptographic system. It introduced quantum computers, how they work, their progress and what they offer so far. Shor's algorithm and practical variations of it were also previously discussed. This research started over a year ago, with over 100 papers read and analyzed to be able to wrap this topic into one thesis. The issues found when it comes to quantum computers and classical computers is that all researchers are considering them separate devices. This chapter proposes a system where quantum computers and classical computers work hand in hand, taking into considerations the pros and cons of each device to build what would be known as a Hybrid Computing Network. This Hybrid Computing Network will have to achieve a few goals, the network:

1) Will need to be as fast as current networks
2) Must be quantum resilient in its security
3) Will consist of classical clients and quantum gateways
4) Must rely as much as possible on current infrastructure.

Much of what will be discussed will rely on theoretical algorithms, some of the data will be verified using simulations on a small scale. Looking at security, one can identify three key areas where security should be handled.

1) Security should be handled within a local storage
2) Security should be handled while transmitting data
3) Security should be handled while processing

In a hybrid network, it is very important to understand the strengths and weaknesses of each type of device. The security of data on a local device can either be guarded by the device itself or a quantum device with a certain encryption algorithm, this would be indifferent to us as long as the algorithm used is a quantum resilient one. Moving on to data transmission, security and bandwidth should be taken into consideration. Data transmission rate is currently extremely low on quantum networks but could be extremely secure. While the rate is high

on classical networks, however security currently should be considered compromised. In this case the power of QCs is harnessed to transmit keys securely instead of using open channels on a conventional network (this will be discussed later in this chapter). And lastly, security while processing. Processing is a clear winner for classical computers, just by the sheer power they currently present. The table below represents which devices or networks will be used in our proposed hybrid network

	Quantum Computers/Networks	Classical Computers/Networks
Local Storage Security	X	X
Data Transmission Security	X	
Processing Security		X

Table 6.5 Device Distribution on the Network

6.1 Key Distribution

Every cryptographic system depends on the right people having the right keys. Thus, key distribution is a big factor in a safe system. A lot of the classical algorithms either require a secure channel to transfer a key, or to create key pairs from a certain calculation in the public domain such as (Diffie, Diffie, and Hellman 1976). These methods are no longer secure, as the logarithms used by Diffie-Hellman inspired algorithms are weak against Shor's algorithm and secure channels would no longer be secure. Thus, a quantum approach is required to distribute the keys. This approach was proposed and achieved by an algorithm that is now famously known as BB84, named after its creators Gilles Brassard and Charles Bennett, proposed in the year 1984 and kept improving on the idea until the mid 2010s (Bennett and Brassard 2014). The way BB84 works is by leveraging the fundamental concepts of quantum mechanics, discussed in Chapter 3, to safely distribute keys. Consider the following example, Alice and Bob need to exchange keys using a quantum channel. A key, in a classical sense is just a string of bits. In our case this is implemented using a quantum channel. Each bit must be encoded with one of two orthogonal basis that will be denoted as such:

$$+ = \{|\rightarrow\rangle, |\uparrow\rangle = \{[1,0]^T, [0,1]^T\}$$

$$X = \{|\nwarrow\rangle, |\nearrow\rangle = \{\frac{1}{\sqrt{2}}[-1,1]^T, \frac{1}{\sqrt{2}}[1,1]^T\}$$

These basis represent the spin of a qubit. The first basis denotes that depending on the measurement technique, the qubit will denote 0 or 1, 100% of the times. The qubit in this case is either spinning perfectly vertical or perfectly horizontal. However, the second basis denotes that the qubit is spinning with either -45° or 45° angles, and this pattern represents a qubit. That said, when the qubit is measured it might break down into a different state because of its different orthogonal spin. Consider the conversion chart below:

State	$+$	$\boldsymbol{X}$
$\|0\rangle$	$\|\rightarrow\rangle$	$\|\nearrow\rangle$
$\|1\rangle$	$\|\uparrow\rangle$	$\|\nwarrow\rangle$

Table 6.6 Conversion from state to spin depending on basis

If a state was encoded using X with it originally being 0, and decoded using the + basis, then there is an equal chance of this state collapsing to either 0 or 1.

$$|\nearrow\rangle = \frac{1}{\sqrt{2}}|\uparrow\rangle + \frac{1}{\sqrt{2}}|\rightarrow\rangle$$

The probability would be measure by squaring the factor of the basis in this case squaring $\frac{1}{\sqrt{2}}$ would result in a 0.5 chance of it being 0 and 0.5 chance of it being 1. Now assume that Alice wants to send the bit string "011011101010"

Bit String	0	1	1	0	1	1	1	0	1	0	1	0
Random Bases	$+$	$+$	X	$+$	$+$	$+$	X	$+$	X	X	X	$+$
Final Qubit Spin	$\rightarrow$	$\uparrow$	$\nwarrow$	$\rightarrow$	$\uparrow$	$\uparrow$	$\nwarrow$	$\rightarrow$	$\nwarrow$	$\nearrow$	$\nwarrow$	$\rightarrow$

Table 6.7 Conversion from bit to qubit from Alice's side

The final qubit spin states are then sent over an open quantum channel to Bob, and Bob will have to decode them. However, Bob has no idea what bases were used and will have to use random bases to get a result.

Random Bases	X	$+$	X	X	$+$	X	$+$	$+$	X	X	X	$+$
Qubit Spin	↗	↑	↖	↖	↑	↗	↑	→	↖	↗	↖	→
Final Bit String	0	1	1	1	1	0	1	0	1	0	1	0

Table 6.8 Conversion from qubit spin to bit from Bob's side

As expected, Bob ended up with a vastly different bit string than the one that Alice sent. Since there are only two bases, Bob will end up with the at least half of the correct result. Bob will then send the whole bit string in the clear to Alice, and she would check which values are equal and drop the ones that are correct. The final step would force Alice to publish a part of her key for Bob to know the final answer. This procedure would be represented in the table below:

Random Bases	$+$	$+$	X	$+$	$+$	$+$	X	$+$	X	X	X	$+$
Bob's Random Bases	X	$+$	X	X	$+$	X	$+$	$+$	X	X	X	$+$
Equal?	F	T	T	F	T	F	F	T	T	T	T	T
Shared Key		1	1		1			0	1	0	1	0

Table 6.9 Final Result

As with any cryptographic system, one should assume that some malicious entity exists between the two communicating sides. In the case of using a quantum channel, an intruder Eve would have to listen in and use a basis of her own to read the spin of the qubits and then pass it on to Bob. This system would seem broken, but this is a quantum system and it adheres to the laws of quantum mechanics. Eve firstly, must guess each qubit correctly before passing it on to Bob. Bob can then detect this because in a quantum system, once a qubit is measured it collapses to a certain state that would then alert Bob. Moreover, because of the no cloning

theory discussed in Chapter 3, Eve does not have the luxury of saving a copy of the qubits to try again on the side without being detected. An added layer of security to BB84, would be the insertion of a qubit that is entangled with another. Alice would entangle two qubits and send one down the channel. If the qubit she has changes its state before it's expected to then Alice can deduce that someone has interfered with the transmission and drop the connection.

6.2 Transmission Security, Local Storage Security and Processing Security

Section 6.1 discussed the issue of key distribution over quantum channels, there remains two areas of focus for our network. To begin with a suitable algorithm would need to be used to encrypt the connection between two users. We refer to the previous chapter to deduce an answer. Currently, no standard for this solution is agreed upon and what would be stated in this section would be my own approach based on the knowledge currently present about quantum systems. The chosen approach would implement the hashing technique discussed in Chapter 5. In this case Alice would generate a number, and hash one of them. Consider a number X and its HASH(X). Alice would have to use the method devised in section to 6.1 to send the key **[HASH(X)]** to Bob. The hash key will be changed to a series of qubits and sent over an open channel to Bob. As mentioned above, Bob will read these qubits using his own bases. That will lead him to achieve the wrong hashed key. He will send this key back to Alice. She will match both keys and set aside any bits that are not equivalent. Following the BB84 protocol, Alice would have to reveal half of the bits she has so Bob can deduce the correct key (Keeping in mind that an eavesdropper has no way of snooping in on this procedure without breaking the whole system). Now when Alice wants to send a message to Bob, she will have to encrypt her message using HASH(X) as a key (depending on the algorithm, she might need more than 1 key) but Bob does not know HASH(X), so Alice would have to attach the incorrect bits that she set aside with their positions and Bob will be able to deduce the correct bit by trying this function:

$$DEDUCED\_HASH(X) = HASH(X) - Position\ and\ True\ Value\ of\ Incorrect\ Bits$$

On the receiving end Bob can deduce the message by decrypting using the complete DEDUCED_HASH function. To make this more secure Alice could generate more keys

exclusively between her and Bob and use different ones to hash. In this case Bob will have to try the different keys, since a hash function is a one-way function. The advantages of this system would be:

1) The DEDUCED_HASH would secure Alice and Bob's communications.
2) Eavesdroppers have no way of copying the qubits in transfer due to the no-cloning theorem.
3) This method takes advantage of quantum channels for key distribution and classical channels for encrypted data transmission.

However as with every system there exists a few down sides that might balance out the pros of such an approach:

1) The amount of keys would be a huge number of keys depending on the number of connected users, because each user would have a different key pair with every other user.
2) DEDUCED_HASH would have to evolve with time by using newer base hash techniques. If SHA-3 for example were ever to become weak, it could no longer be used as a base hash.
3) Keys might need to be changed regularly because of the revealed information

Local storage security could also be safe using this technique. The main argument is that the system should never be based the "security-by-obscurity" approach. A good encryption algorithm will have to be proven computationally secure, and that only the correct party with the correct keys can decrypt the data back to its original form. Local storage encryption techniques such as "Xor-encrypt-xor" are secure and would need two keys both of which would be hashes created randomly by Alice to secure her data. Since these keys are irreversible, she is fairly secure. Quantum computers show no signs of advantage when it comes to breaking local data encryptions. The only attack vector left for data would be while data is being processed. QCs are not powerful enough to process huge amounts of data, and even if that was not the case, the unpredictability of qubits and the error rates that are currently present impose another problem. However, QCs do not seem to impose any extra security issues to data in the processing phase, so in our hybrid system classical computers

would continue to handle data processing. The same applies to data that is currently in memory, current encryptions handled by the OS are computationally secure and quantum resilient.

6.3 Hybrid Network Abstract Architecture

Visualizing a hybrid network would not be difficult. The approach to describe a hybrid network would be to visualize quantum computers as gate keepers or key trustees. Clients would all remain classical computers that are only interconnected with the watch of a quantum computer on each network, acting as a gate keeper to each network. As a gate keeper a quantum computer would never be involved in touching users' data but it would only take care of key distribution. However such a network would be extremely costly and would need lots of infrastructure to implement. Consider a classical network, each user has their own machine and each machine is connected to a switch or router and then these routers are interconnected, forming the internet. The internet as a whole, is just a network of networks and it looks something as the figure below.

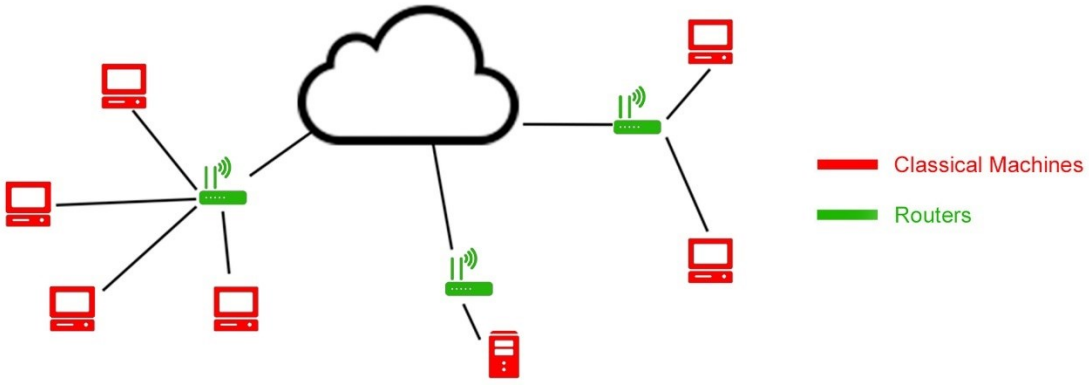


Figure 6.19: A Traditional Network Architecture

Looking at a traditional network, devices are connected to a router, with this router connecting to the "cloud". This cloud in itself being built from other machines connected to their respective routers and then to each other. Taking this infrastructure and trying to adopt a quantum aspect to it, would result in two network adaptations.

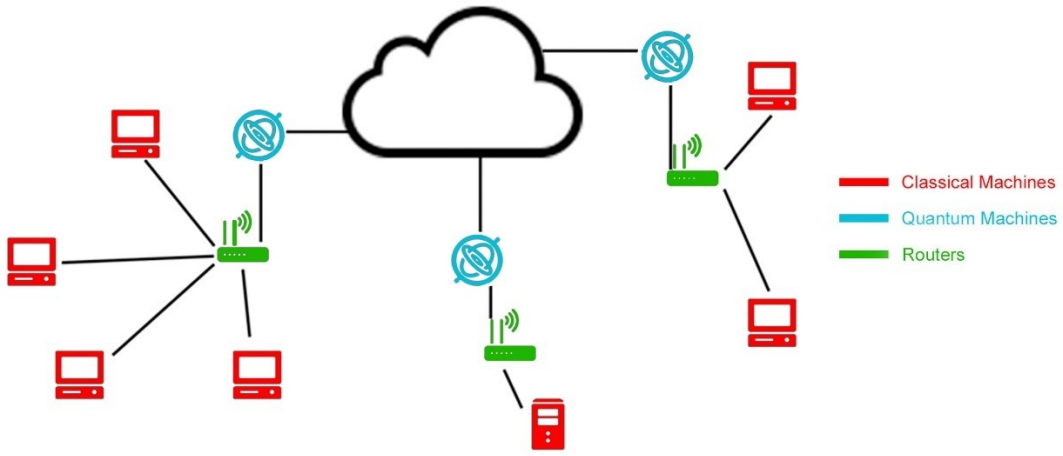


Figure 6.20: An Ideal Hybrid Network Architecture

Figure 6.20 represents what would be an ideal hybrid network, where each router would be mediated by a quantum computer, thus managing the quantum channels and spreading the network worldwide. Computers in this network will behave as a normal network would, but instead of initiating security requests between each other, a QC will handle the handshakes between different routers. QCs could also use readily available fiber optic networks to communicate. However, this network architecture represents a great burden on any network provider. Currently, QCs are huge machines that require lots of protection, maintenance and power to run. No matter how secure this network is, it is rendered financially non-viable to any user to use it. We can modify our model to create a more sustainable one. However, this would also require us to change our DEDUCED_HASH algorithm to accommodate a few factors of uncertainty. In the realistic model depicted in Figure 6.21, it is obvious that a QC cannot exist on every network, behind every firewall to watch and protect everyone. This model proposes an approach where a QC is an entity in the cloud. The QC would still act as a mediator, however our original DEDUCED_HASH function would have to change and accommodate this architecture. In this case, the connection between Alice, the QC and Bob is all considered insecure. As such, Alice would not be the user that creates a random hash, this will be delegated to the QC itself. The QC will have to create a hash and encode it as qubits and send it both to Alice and Bob for them to decode.

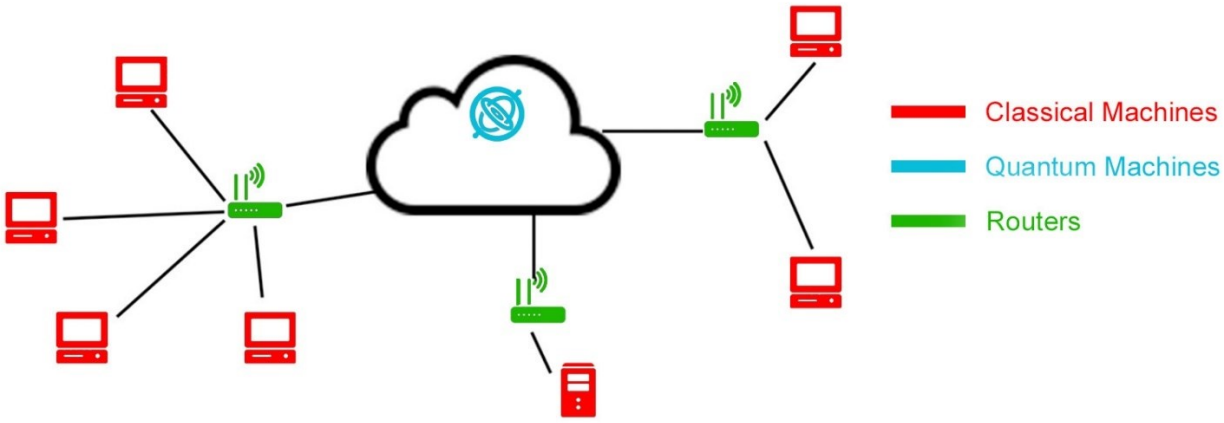


Figure 6.21: A Realistic Hybrid Architecture

Each of them will use different bases, leading to them having different keys, which they will send back to the QC. The QC then will follow the same technique used above and drop the non-matching bits and then reveal half of the correct bits to Alice and Bob. Alice and Bob could then deduce the hash and use it as a key for communications between them. This method has its own drawbacks and this is why it is considered insecure. The drawbacks mainly are:

1) Alice and Bob would have to trust the QC they are communicating with.
2) Keys will have to be renewed more often because of the length of the remaining key and the amount of information that is made public about them.
3) QCs can become the bottleneck if lots of requests depend on a low number of machines.

6.4 Example of Two Users Communicating Over a Hybrid Network Using The Realistic Model

As discussed in the previous section, a realistic model would contain one quantum computer running on some network between the two users. This example serves to demonstrate the goals of the hybrid network and give a more in depth look to the inner workings of such system. Assume the first user Alice would like to converse with a second user Bob. Alice

would want to start her conversation by saying “Hello”. No one other than Bob should read the message, and Bob should be able to verify the owner of the message.

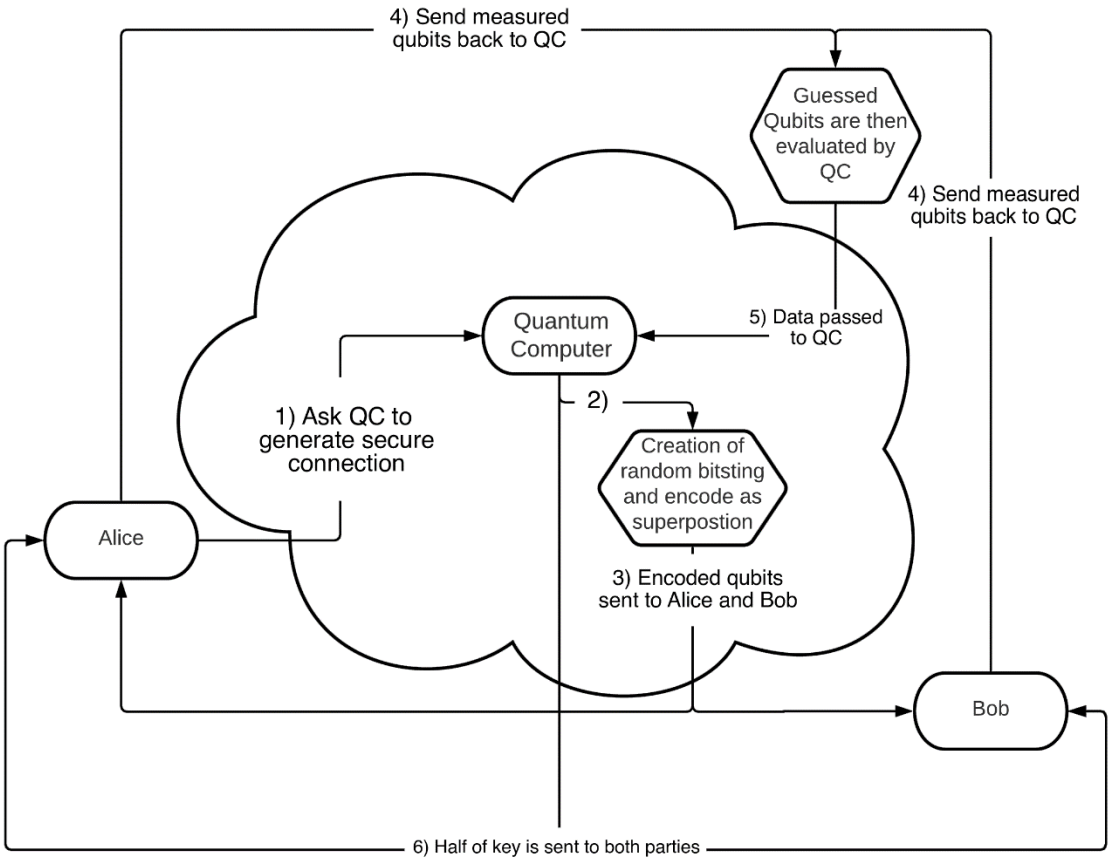


Figure 6.22: Flow in the Realistic Model

Converting “Hello” to binary would result in the bit string “01001000 01100101 01101100 01101100 01101111”, this is still represented in its original form and readable by anyone. The next step would require Alice to inform a quantum computer about her intent and the parties included in the conversation. The quantum computer would then set about to create a random formation of qubits with their own spin and send these qubits to both Alice and Bob. These qubits could be transmitted over fiber as has been demonstrated a recent study (Krutyanskiy et al. 2019). This meets one of the aforementioned goals of using current infrastructure as much as possible to help ease the deployment of this network. For extra security in the realistic model, the quantum computer would need to generate a larger sized key, since parts of this key would be discarded. For this example, assume that the quantum

computer generated a bit string “01101110” as a key and used a certain base to encode it into spin as listed in the table below. This key would be the result of the hash of a randomly generated number.

Bit	0	1	1	0	1	1	1	0
Base	+	+	X	+	+	+	X	+
Spin	→	↑	↖	→	↑	↑	↖	→

Table 6.10 Conversion from bit to qubit

The result in the third row is the spin of the qubits that would be transmitted to Alice and Bob. They will have to choose a random base (+ or X) to decode the bit string and send it back to the quantum computer

Spin	→	↑	↖	→	↑	↑	↖	→
Alice’s Base	+	X	X	+	X	+	X	+
Alice’s Bit String	0	0	1	0	0	1	1	0
Bob’s Base	X	+	X	X	+	+	+	+
Bob’s Bit String	0	1	1	0	1	1	1	0

Table 6.11 Conversion from qubit to bit by Alice and Bob

These bit strings will then be sent back to the quantum computer to evaluate the common key to be shared. This will vary from the original BB84 protocol in the sense that instead of equating the key with the original bit string only, it will also be equated with the third key generated by the intended user in this case Bob.

$$Final\ Key = (OS)\ AND\ (AS)\ AND\ (BS)$$

Original Bit String	0	1	1	0	1	1	1	0
Alice's Bit String	0	0	1	0	0	1	1	0
Bob's Bit String	0	1	1	0	1	1	1	0
Equal?	T	F	T	T	F	T	T	T
Final Key	0		1	0		1	1	0

Table 6.12 Key equivalency and final key generation

The final key result would be "010110" which in this example is not much shorter than the original key. In the worst case scenario the resulting key would be 25% the size of the original key (0.5 probability for Alice to be correct and 0.5 probability for Bob to be correct, resulting in a 0.25 probability in both being correct). Alice and Bob would receive half of the key from the quantum computer in plain text, using this information, they would be able to decode the rest of the key. The next step would be the quantum computer sharing half of this key with both Alice and Bob. Now that the key (referred to as DEDUCED_HASH) is distributed Alice and Bob could use AES to securely communicate. First step in this implementation would be to create an empty package such that P = ∉. Secondly the message M is created and inserted into the package P, now P = M. The second step would require M to be hashed. This hash is then appended to P, such that at the end of the second step P = M + Hash(M). Third step would require the encryption of the whole package using AES, with the DEDUCED_HASH as the key for the encryption. After this is complete P = AES(M+HASH(M)) with the DEDUCED_HASH used as the common key. This package P is sent over the network and could only be unlocked by the intended user using the same key. Alice in this case would send the final cypher text to Bob and he then would be able to decrypt the message with the common shared key and verify the message with the hash. This method will achieve all the goals stated for this network. It would be as fast or almost as fast as current networks since it heavily relies on the current infrastructure. It will be quantum resilient because of the fact that AES is not affected by quantum computers and the key is quantum resilient by nature. This network relies on both classical computers and quantum ones working cohesively to achieve its main goal. This network however suffers from the original downsides of the AES

encryption which is the generation of many keys, the added overhead is that keys could not be used many times, because of the fact that the process of creating and distributing the keys consists of sharing half of the key publicly for the clients to deduce their key, which would result in more key creations.

Chapter 7: Conclusion

This thesis discussed a whole spectrum of scientific areas spanning computer science, math, and physics. All of which have been crucial to lead us to designing a hybrid network, that is theoretically viable and one that checks all the goals stated in Chapter 1. This thesis has shown without a shadow of a doubt that the future of our privacy and security is at risk. RSA stood the test of time but its strongest point against classical computers turned out to be its weakest point again quantum ones. Quantum computers were heavily introduced in chapter 3. The reason for this focus being that quantum computers leveraging quantum mechanics demonstrated weird behavior. Researchers are continuously evolving techniques to use these behaviors to create interesting algorithms that would be beneficial to humans in the future, with the adverse effect that some of these algorithms strike some of the number theory basics that a lot of crypto systems depend on. With the introduction of some of the quantum resilient algorithms and some quantum protocols, better solutions could be built.

7.1 Summary and Main Contribution

This thesis focused on three main contributions, the first being a review of the state-of-the-art of current Quantum Computers. Secondly, an action timeline was devised based on historic events in similar situations and the importance of early action was highlighted. The third contribution of the thesis is the proposed architecture of a hybrid network and how such a network could operate. The chosen method of distributing keys was over quantum channels using the BB84 protocol. The network that was built relied heavily on symmetric encryption using the hash generated by a user. This hash is then transmitted to another user over a quantum channel. Then the resulting hash after performing the BB84 is used as a key along with the positions of incorrect bits. This method offers not just security but also trust as the hash could be used in the encryption process and as a certificate for trust. The provided use

case mentioned in the previous section, discussed how such a network would operate and demonstrated how each of the provided goals were met. The discussed network had to:

1) Be as fast as current networks. This was proven to be mostly correct since the suggested network relied heavily on the current infrastructure.
2) Be quantum resilient in its security. This goal was met because of the use of quantum resilient symmetric encryption such as AES and key distribution using quantum channels which in itself is quantum resilient due to the No-Cloning theorem discussed earlier.
3) Consist of classical clients and quantum gateways. The discussed method described quantum computers as gateways in such a way that they handled the distribution of keys over the network, while classical clients were used as they would normally be in today's networks.
4) Rely as much as possible on current infrastructure. This goal was met because the hybrid network was designed around this ideology instead of building it from scratch.

We also discussed that security for data in the process phase and the local stage are not affected by the introduction of quantum computers to any network.

7.2 Possible Extensions and Future Work

The work presented in this thesis has mostly been theoretical. Quantum computers are only powered by a few qubits which are only sufficient for small scale applications. This work can be extended by a full network stack built on the ideology of a hybrid network. An implementation of the realistic model defined in Chapter 6, would require to keep a few things in check:

1) Trusted connection between a router and a quantum machine
2) Error rates of quantum machines to decrease heavily
3) Cost of operation for quantum machines need to decrease

These 3 hurdles are the reasons why the realistic model is not implementable at the moment. A good estimate to have these 3 points checked out would be in 5 to 10 years and this would

ultimately undercut the doom and gloom of a possible “general-purpose” quantum computer to be built by that time.

Bibliography

Akama, Seiki. 2015. *Elements of Quantum Computing: History, Theories and Engineering Applications*.

Alagic, Gorjan, Jacob Alperin-Sheriff, Daniel Apon, David Cooper, Quynh Dang, Yi-Kai Liu, Carl Miller, Dustin Moody, Rene Peralta, Ray Perlner, Angela Robinson, and Daniel Smith-Tone. 2019. "Status Report on the First Round of the NIST Post-Quantum Cryptography Standardization Process."

Arute, Frank, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C. Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando G. S. L. Brandao, David A. Buell, Brian Burkett, Yu Chen, Zijun Chen, Ben Chiaro, Roberto Collins, William Courtney, Andrew Dunsworth, Edward Farhi, Brooks Foxen, Austin Fowler, Craig Gidney, Marissa Giustina, Rob Graff, Keith Guerin, Steve Habegger, Matthew P. Harrigan, Michael J. Hartmann, Alan Ho, Markus Hoffmann, Trent Huang, Travis S. Humble, Sergei V Isakov, Evan Jeffrey, Zhang Jiang, Dvir Kafri, Kostyantyn Kechedzhi, Julian Kelly, Paul V Klimov, Sergey Knysh, Alexander Korotkov, Fedor Kostritsa, David Landhuis, Mike Lindmark, Erik Lucero, Dmitry Lyakh, Salvatore Mandrà, Jarrod R. Mcclean, Matthew Mcewen, Anthony Megrant, Xiao Mi, Kristel Michielsen, Masoud Mohseni, Josh Mutus, Ofer Naaman, Matthew Neeley, Charles Neill, Murphy Yuezhen Niu, Eric Ostby, Andre Petukhov, John C. Platt, Chris Quintana, Eleanor G. Rieffel, Pedram Roushan, Nicholas C. Rubin, Daniel Sank, Kevin J. Satzinger, Vadim Smelyanskiy, Kevin J. Sung, Matthew D. Trevithick, Amit Vainsencher, Benjamin Villalonga, Theodore White, Z. Jamie Yao, Ping Yeh, Adam Zalcman, Hartmut Neven, and John M. Martinis. 2019. "Quantum Supremacy Using a Programmable Superconducting Processor." *Nature* 574:505.

Barker, Elaine. 2020. "NIST Special Publication 800-175B Revision 1 Guideline for Using Cryptographic Standards in the Federal Government: Cryptographic Mechanisms."

Beauregard, Stéphane. 2003. *Circuit for Shor's Algorithm Using 2n+3 Qubits*.

Bennett, Charles H., and Gilles Brassard. 2014. “Quantum Cryptography: Public Key Distribution and Coin Tossing.” *Theoretical Computer Science*.

Carnal, O., and Jo Mlynek. 1991. “Youngs Double-Slit Experiment with Atoms: A Simple Atom Interferometer.” *Physical Review Letters* 66(21):2689–92.

Chuang, Isaac L., and Yoshihisa Yamamoto. 1995. *A Simple Quantum Computer*.

Deutsch, D. 1985. “Quantum Theory, The Church-Turing Principle and the Universal Quantum Computer.” *Proceedings of The Royal Society of London, Series A: Mathematical and Physical Sciences*.

Diffie, Whitfield, Whitfield Diffie, and Martin E. Hellman. 1976. “New Directions in Cryptography.” *IEEE Transactions on Information Theory*.

Gidney, Craig, and Martin Ekerå. 2019. *How to Factor 2048 Bit RSA Integers in 8 Hours Using 20 Million Noisy Qubits*.

ITIF. 2019. *What Is Quantum Computing? ITIF Technology Explainer Series • Www.Itif.Org Why Now?* ITIF.

Kendon, Viv. 2018. *Quantum Computing: Progress and Prospects (2018)*. Vol. 9781461418.

Krutyanskiy, V., M. Meraner, J. Schupp, V. Krcmarsky, H. Hainzer, and B. P. Lanyon. 2019. “Light-Matter Entanglement over 50 Km of Optical Fibre.” *Npj Quantum Information*.

Lenstra, Arjen K., and Eric R. Verheul. 1990. *Selecting Cryptographic Key Sizes □ Extended Abstract*.

Monroe, C., D. M. Meekhof, B. E. King, W. M. Itano, and D. J. Wineland. 1995. “Demonstration of a Fundamental Quantum Logic Gate.” *Physical Review Letters*.

Nick Summers. 2019. “IBM’s Latest Quantum Computer Is a 20-Qubit Work of Art.” Retrieved (https://www.engadget.com/2019-01-08-ibm-q-system-one-quantum-computer.html#gallery=88820acc-a551-35ca-bec6-a1547b4f7174&slide=7a6d08e5-47c7-320e-83d1-f49a02a33761&index=1).

Pollard, J. M. 1974. “Theorems on Factorization and Primality Testing.” *Mathematical*

Proceedings of the Cambridge Philosophical Society.

Prashant. 2005. *A Study on the Basics of Quantum Computing*.

Rivest, R. L., A. Shamir, and L. Adleman. 1978. "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems."

Rivest, Ronald L., Adi Shamir, and Leonard M. Adleman. 2019. "A Method for Obtaining Digital Signatures and Public Key Cryptosystems." Pp. 217–39 in *Secure Communications and Asymmetric Cryptosystems*. Taylor and Francis.

Shor, Peter W. 1997. "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer." *SIAM Journal on Computing* 26(5):1484–1509.

Solovay, R., and V. Strassen. 1977. "A Fast Monte-Carlo Test for Primality." *SIAM Journal on Computing*.

The White House. 2009. *Title 3-The President EO 13526 Classified National Security Information*.

Vasconcelos, Francisca. 2020. *Quantum Computing @ MIT: The Past, Present, and Future of the Second Revolution in Computing*.

Watson, T. F., S. G. J. Philips, E. Kawakami, D. R. Ward, P. Scarlino, M. Veldhorst, D. E. Savage, M. G. Lagally, Mark Friesen, S. N. Coppersmith, M. A. Eriksson, and L. M. K. Vandersypen. 2018. "A Programmable Two-Qubit Quantum Processor in Silicon." *Nature*.

Wootters, W. K., and W. H. Zurek. 1982. "A Single Quantum Cannot Be Cloned." *Nature*.

Yanofsky, Noson S., and Mirco A. Mannucci. 2008. *Quantum Computing for Computer Scientists*.

Appendix A: Qiskit Implementation of Shor's Algorithm

The goal of Shor's algorithm as discussed in this thesis was to convert a factoring problem into a period finding one. Peter Shor also represented this problem in polynomial time and this is why it gained interest in the computer science community. This appendix serves to implement Shor's algorithm in the Qiskit framework developed by IBM to run code on physical quantum computers.

Firstly, recall the periodic function

$$f(x) = a^x mod\ N$$

In this case both a and N are positive integers where a is less than N. Moreover, there are no common factors between them. The period is considered the smallest integer, greater than zero, that satisfies

$$a^r mod\ N = 1$$

In the example, set below a will be equal to 7 and N equal to 15. The code below will set the range for the x-axis to be 15 and will then plot the period up till x ≅ 15.

Listing 7.1 Periodic Function in Shor's Algorithm

```
N = 15
a = 7

# Calculate the plotting data
xvals = np.arange(15)
yvals = [np.mod(a**x, N) for x in xvals]

# Use matplotlib to display it nicely
fig, ax = plt.subplots()
ax.plot(xvals, yvals, linewidth=1, linestyle='dotted', marker='x')
ax.set(xlabel='$x$', ylabel='$%i^x$ mod $%i$' % (a, N),
       title="Periodic Function in Shor's Algorithm")
try: # plot r on the graph
```

```
    r = yvals[1:].index(1) +1
    plt.annotate(text='', xy=(0,1), xytext=(r,1), arrowprops=dict(arrowstyle='<->'))
    plt.annotate(text='$r=%i$' % r, xy=(r/3,1.5))

except:
    print('Could not find period.')
```

The resulting plot would show the period and the frequency of repetition on one graph for the period function using the given a and N.

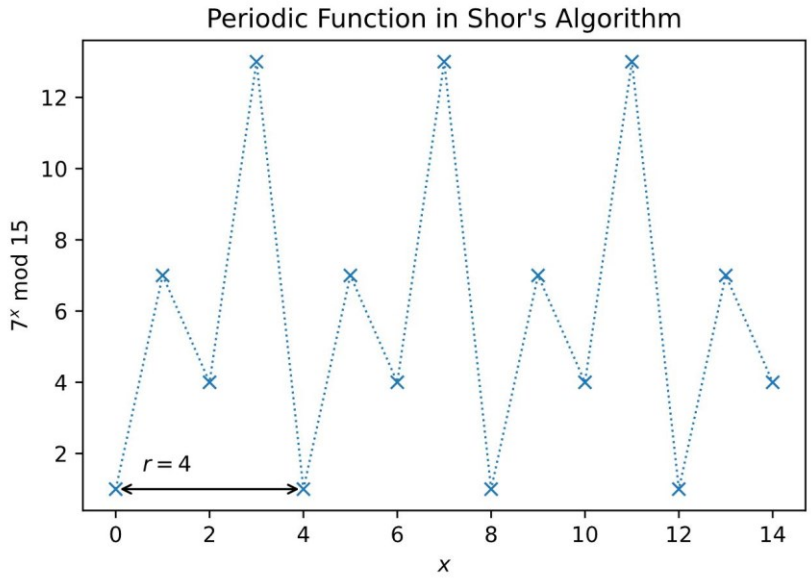


Figure A.23: Periodic Function in Shor's Algorithm

The solution for this problem according to Shor is to apply a quantum phase estimation on the unitary operator

$$U|y\rangle \equiv |ay \bmod N\rangle$$

Consider the starting state $|1\rangle$, working out the eigenstate of U would reveal that each successive application of U will multiply the state of the register by $a \ (mod\ N)$ and after r iterations the state $|1\rangle$ would be achieved again.

$$U|1\rangle = |7\rangle, U^2|1\rangle = |4\rangle, U^3|1\rangle = |13\rangle, \dots, , U^{r-1}|1\rangle = |13\rangle\,, U^r|1\rangle = |1\rangle$$

The eigenstate of *U* would result in the superposition of the states in a given cycle, in this case $|u_0\rangle$. An interesting eigenstate would be the one in which the phase is different for each basis states. As an example, a good case would be when the phase of the k[th] state is proportional to k, as shown below.

$$|u_1\rangle = \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} e^{-\frac{2\pi ik}{r}} |a^k \ mod \ N\rangle$$

where

$$U|u_1\rangle = e^{\frac{2\pi i}{r}}|u_1\rangle$$

This eigenvalue is interesting because it contains *r*. Moreover, *r* causes the phase differences between the computational basis to be equal. This could be generalized further as it is not the only eigenstate by multiplying an integer, *s*, to the phase difference.

$$|u_s\rangle = \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} e^{-\frac{2\pi isk}{r}} |a^k \ mod \ N\rangle$$

where

$$U|u_s\rangle = e^{\frac{2\pi is}{r}}|u_s\rangle$$

such that

$$0 \leq s \leq r-1$$

This case is special because the sum of all the eigenstates will result in the destruction of most signals with opposite phases, ending up with one basis state $|1\rangle$.

$$\frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} |u_s\rangle = |1\rangle$$

Since the basis state is a superposition of these eigenstates, a quantum phase estimation on *U* of the state by measuring a phase using the following equation:

$$\emptyset = \frac{s}{r}$$

Using the continued fractions algorithm on ∅, r could be derived. To implement this in code, U^x would have to be calculated by repeating the circuit x times. The function c_amod15 returns the controlled-U gate for a, repeated $power$ times, and qft_dagger function will recreate the inverse quantum fourier transform procedure.

Listing 7.2 c_amod15 and qft_dagger function representations in Qiskit

```
def c_amod15(a, power):
    """Controlled multiplication by a mod 15"""
    if a not in [2,7,8,11,13]:
        raise ValueError("'a' must be 2,7,8,11 or 13")
    U = QuantumCircuit(4)
    for iteration in range(power):
        if a in [2,13]:
            U.swap(0,1)
            U.swap(1,2)
            U.swap(2,3)
        if a in [7,8]:
            U.swap(2,3)
            U.swap(1,2)
            U.swap(0,1)
        if a == 11:
            U.swap(1,3)
            U.swap(0,2)
        if a in [7,11,13]:
            for q in range(4):
                U.x(q)
    U = U.to_gate()
    U.name = "%i^%i mod 15" % (a, power)
    c_U = U.control()
    return c_U

def qft_dagger(n):
    """n-qubit QFTdagger the first n qubits in circ"""
    qc = QuantumCircuit(n)
    # Don't forget the Swaps!
    for qubit in range(n//2):
        qc.swap(qubit, n-qubit-1)
```

```
    for j in range(n):
        for m in range(j):
            qc.cu1(-np.pi/float(2**(j-m)), m, j)
        qc.h(j)
qc.name = "QFT†"
return qc
```

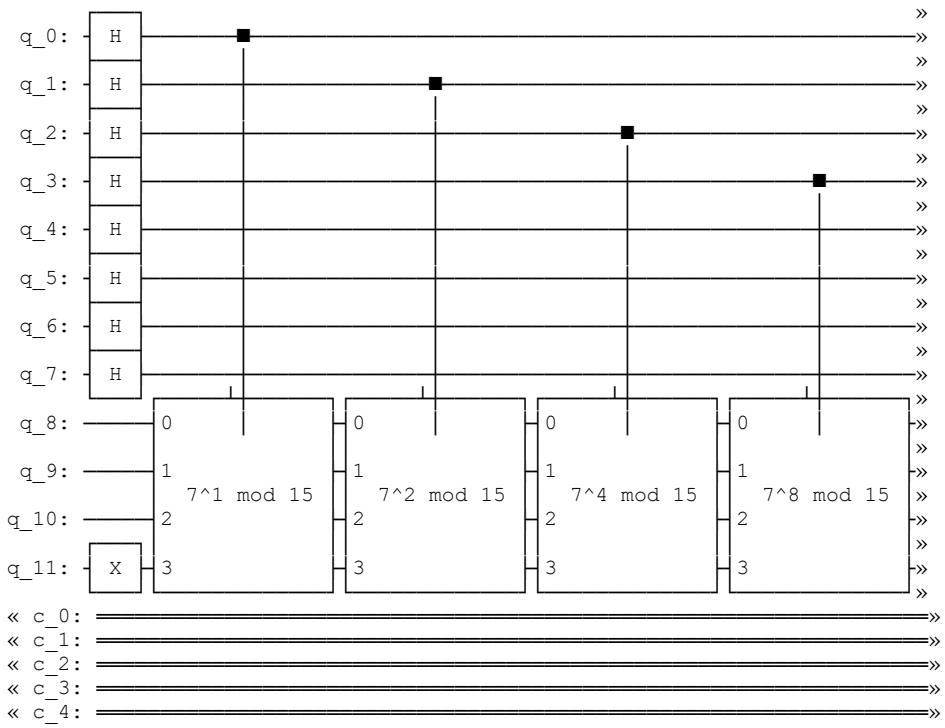
After this construction phase, leverage the Qiskit framework to construct the circuit for Shor's algorithm.

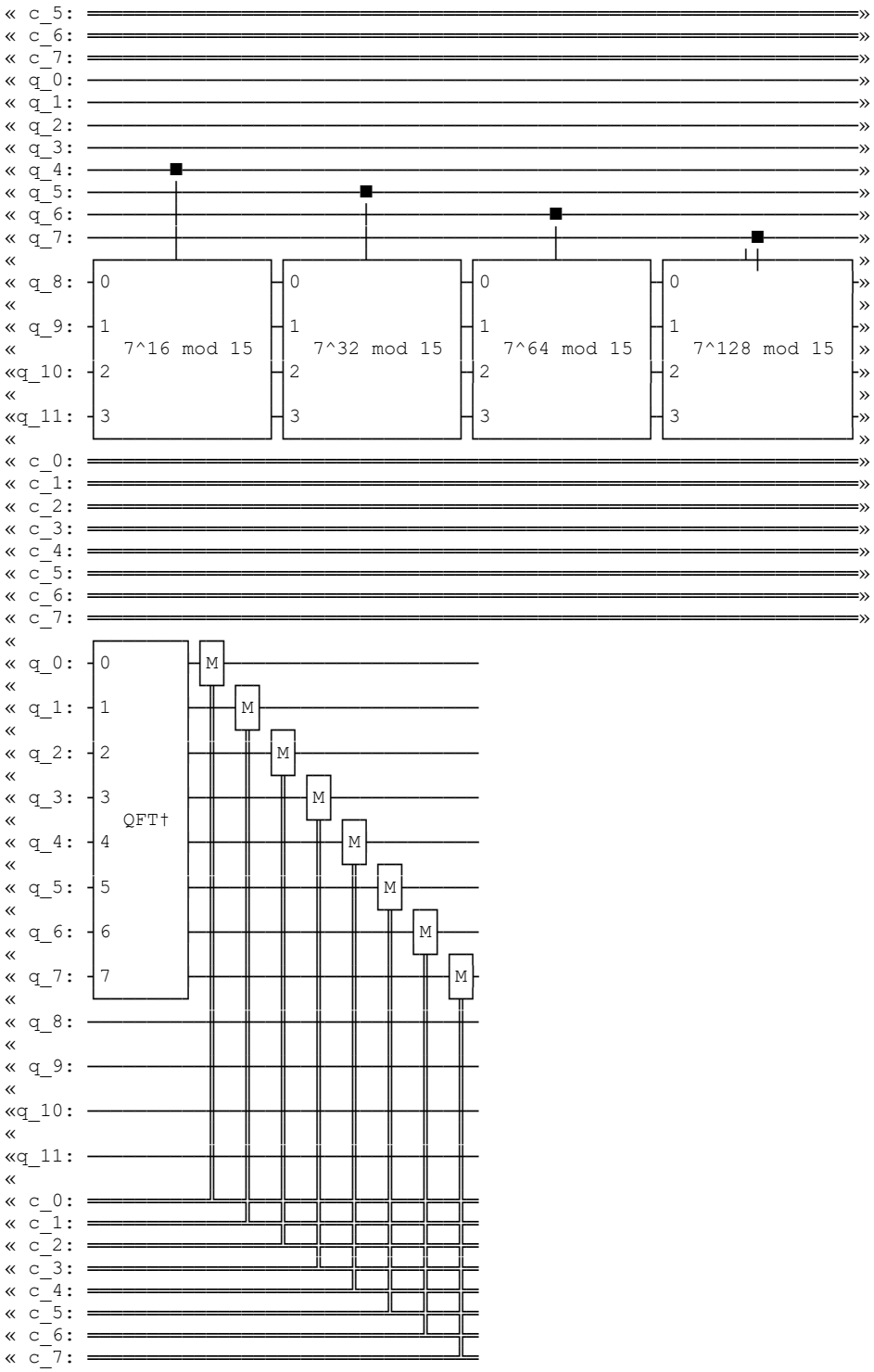
Listing 7.3 Qiskit implementation in Shor's Algorithm

```
# Specify variables
n_count = 8 # number of counting qubits
a = 7
# Create QuantumCircuit with n_count counting qubits
# plus 4 qubits for U to act on
qc = QuantumCircuit(n_count + 4, n_count)

# Initialise counting qubits
# in state |+>
for q in range(n_count):
    qc.h(q)

# And ancilla register in state |1>
qc.x(3+n_count)

# Do controlled-U operations
for q in range(n_count):
    qc.append(c_amod15(a, 2**q),
              [q] + [i+n_count for i in range(4)])

# Do inverse-QFT
qc.append(qft_dagger(n_count), range(n_count))

```

```
# Measure circuit
qc.measure(range(n_count), range(n_count))
qc.draw('text')

# Second block to plot the histogram
backend = Aer.get_backend('qasm_simulator')
results = execute(qc, backend, shots=2048).result()
counts = results.get_counts()
plot_histogram(counts)

rows = []
for phase in measured_phases:
    frac = Fraction(phase).limit_denominator(15)
    rows.append([phase, "%i/%i" % (frac.numerator, frac.denominator), frac.denominator])
# Print as a table
headers=["Phase", "Fraction", "Guess for r"]
df = pd.DataFrame(rows, columns=headers)
print(df)
```

The resulting circuit is represented below:




c_5:
c_6:
c_7:
q_0:
q_1:
q_2:
q_3:
q_4:
q_5:
q_6:
q_7:
q_8:
q_9:
q_10:
q_11:
0
1
2
3
7^16 mod 15
7^32 mod 15
7^64 mod 15
7^128 mod 15
c_0:
c_1:
c_2:
c_3:
c_4:
c_5:
c_6:
c_7:
QFT†
M

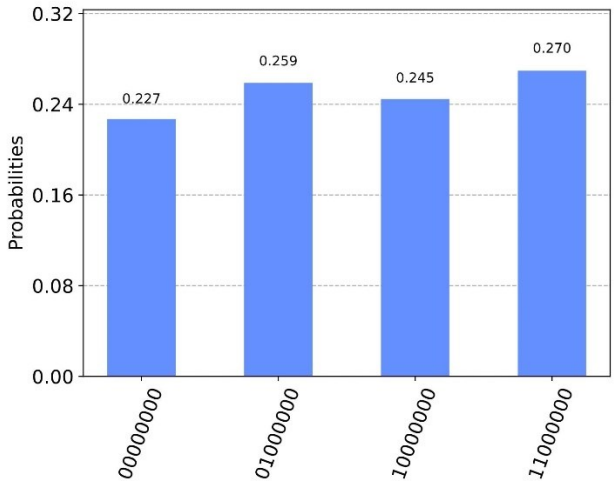


Figure A.24: Probability of values of *r* using Shor's algorithm

Applying the circuit that was built results in the probabilities shown in the figure above. Converting these base 2 values into decimals and performing continued fractions with a denominator limit of 15 results with the possible values of *r* as shown below.

	Phase	Fraction	Guess for r
0	0.00	0/1	1
1	0.25	1/4	4
2	0.50	1/2	2
3	0.75	3/4	4

Two of the eigenvalues generated were equal to 4 which in itself is equal to *r*. This example shows that Shor's algorithm could generate wrong values and that depends on the value of *s*. These could be generated if $s = 0$ or if *s* and *r* are not coprime, this would result in a factor of *r* instead of *r* itself. This could be mitigated if the code is run again to get a satisfying result for *r*. This code in itself is not running in a polynomial order as Shor's algorithm suggests. However, further optimizations could be made to convert it into a polynomial solution by performing modular exponentiation. These optimizations will not be discussed in this appendix.