RV College of Engineering®

Autonomous institution affiliated to Visvesvaraya Technological University, Belagavi | Approved by AICTE, New Delhi

Go, change the world

Simulation of RSA Algorithm on a Quantum Computer

A Technical Seminar Report

Submitted by,

Kumar Shashank **1RV17EC063**

Nishant Agrawal **1RV17EC189**

Under the guidance of

Mr. Subramanya K N

Assistant Professor

Dept. of ECE

RV College of Engineering

In partial fulfillment of the requirements for the degree of

Bachelor of Engineering in

Electronics and Communication Engineering

2020-2021

RV College of Engineering®, Bengaluru

(*Autonomous institution affiliated to VTU, Belagavi*)

Department of Electronics and Communication Engineering

.

CERTIFICATE

Certified that the Technical Seminar titled ***Simulation of RSA Algorithm on a Quantum Computer*** is carried out by **Kumar Shashank** (**1RV17EC063**) and **Nishant Agrawal** (**1RV17EC189**) who are bonafide students of RV College of Engineering, Bengaluru, in partial fulfillment of the requirements for the degree of **Bachelor of Engineering** in **Electronics and Communication Engineering** of the Visvesvaraya Technological University, Belagavi during the year 2020-2021. It is certified that all corrections/suggestions indicated for the Internal Assessment have been incorporated in the Technical Seminar report deposited in the departmental library. The Technical Seminar report has been approved as it satisfies the academic requirements in respect of Technical Seminar work prescribed by the institution for the said degree.

Signature of Guide	Signature of Head of the Department	Signature of Principal
Mr. Subramanya K N	Dr. K S Geetha	Dr. K. N. Subramanya

External Viva

Name of Examiners	Signature with Date
1.	
2.	

DECLARATION

We, **Kumar Shashank** and **Nishant Agrawal** students of eighth semester B.E., Department of Electronics and Communication Engineering, RV College of Engineering, Bengaluru, hereby declare that the Technical Seminar titled '**Simulation of RSA Algorithm on a Quantum Computer**' has been carried out by us and submitted in partial fulfilment for the award of degree of **Bachelor of Engineering** in **Electronics and Communication Engineering** during the year 2020-2021.

Further we declare that the content of the dissertation has not been submitted previously by anybody for the award of any degree or diploma to any other university.

We also declare that any Intellectual Property Rights generated out of this project carried out at RVCE will be the property of RV College of Engineering, Bengaluru and we will be one of the authors of the same.

Place: Bengaluru

Date:

Name	Signature
1. Kumar Shashank(1RV17EC063)	
2. Nishant Agrawal(1RV17EC189)	

ACKNOWLEDGEMENT

We are indebted to our guide, **Mr. Subramanya K N**, Assistant Professor, RV College of Engineering for the wholehearted support, suggestions and invaluable advice throughout our Technical Seminar and also helped in the preparation of this thesis.

We also express our gratitude to our examiner **Dr. Kariyappa B.S.**, Professor , Department of Electronics and Communication Engineering for their valuable comments and suggestions.

Our sincere thanks to **Dr. K S Geetha**, Professor and Head, Department of Electronics and Communication Engineering, RVCE for the support and encouragement.

We express sincere gratitude to our beloved Principal, **Dr. K. N. Subramanya** for the appreciation towards this technical seminar work.

We thank all the teaching staff and technical staff of Electronics and Communication Engineering department, RVCE for their help.

Lastly, we take this opportunity to thank our family members and friends who provided all the backup support throughout the project work.

ABSTRACT

Cryptography is an art of transforming information into something unintelligible for anyone but the intended recipient. This process of transformation is called encryption, and the reverse process is called decryption. Quantum computing is a new and fascinating field at the intersection of Computer Science, Mathematics and Quantum Physics. It utilises quantum phenomena of superposition i.e. existence of multiple states at the same time and entanglement i.e. using the correlation between Qubits to deal with complex problems. Using laws of physics in cryptography guarantee that any attempt of eavesdropping will result in errors in the transmission and hence allows to distribute secret key over open communication channel.

Last few decades have witnessed major developments in cryptographic algorithm which can help in secure transmission but these algorithms can be affected by certain quantum algorithms. This project proposes a method to implement the encryption and decryption of the most widely used asymmetric algorithm Rivest-Shamir-Adleman (RSA) on Quantum computer and sharing the prime factor number on an open communication channel using Quantum Key Distribution (QKD). QKD is a process of using quantum communication to establish a shared key between two trusted parties so that the untrusted eavesdropper cannot learn anything about the key. The algebraic computation is carried out using Quantum Fourier Transform (QFT) which is quantum equivalent of Discrete Fourier Transform (DFT) and changes the basis from computational to Fourier basis. The decryption has been carried out using Shor's Algorithm which is used to find the prime factors of any number.

The asymmetric algorithm RSA using quantum computation is implemented on IBM Quantum Experience an open source tool to access a set of IBM's prototype quantum processors. The python program for the algorithm has been run on the processor. Initially, the user inputs the value for generation of public key and then the encryption is carried out on the input data. The value of coprime is then shared on an open communication channel to the receiver which takes this value and generates the factors using Shor's Algorithm which in turn generates the private key to decipher the data. QKD determines whether the interference by a third party has occurred or not with a high probability. Successful Encryption and Decryption of the message is carried out with a 97% probability that the eavesdropping by a third untrusted party is detected. The primality test implemented using Shor's Algorithm had a run time of 89ms on a quantum server which is fast as compared to any of the classical algorithms.

CONTENTS

LIST OF FIGURES

ABBREVIATIONS

DFT Discrete Fourier Transform

IBM International Business Machines Corporation

NMR Nuclear Magnetic Response

QASM Quantum Assembly

QFT Quantum Fourier Transform

QISKIT Quantum Information Science Kit

QKD Quantum Key Distribution

QPE Quantum Phase Estimation

RSA Rivest-Shamir-Adleman

Chapter 1

Introduction to Quantum Computing

CHAPTER 1

INTRODUCTION TO QUANTUM COMPUTING

1.1 Introduction

The quantum computer was originally conceived as a method to simulate the system of quantum mechanics, which cannot be effectively simulated on a classical computer. Since then, has been developed in the fields of physics and computer science to overcome the main technical challenges of implementing reliable quantum computers on a large scale, although still has many challenges. Furthermore, Moore's law results in the IC feature size being less than 30nm, in which quantum effects begin to prevail. Another challenge faces is the development of quantum algorithms, which are currently very few. Shor's factoring algorithm is one of the most famous quantum algorithms. Since the advantage of public key encryption technology used to protect data on computer networks is based on the difficulty of breaking down integers, the development of large-scale reliable quantum computers may allow crypt analysts to crack key ciphers public.

Figure 1.1: Quantum computer []

Quantum computing is founded on the premise that, unlike ordinary bits based on transistors, a single qubit can have both zero and one values at the same time. The notion is known as superposition: the value of a qubit is undefined and obeys the law of probability before it is measured. There is a chance of measuring zero and a chance of measuring one, but either way,

the quantum system will collapse, leaving only zero or one. Because one qubit may represent two values at once (zero and one), N qubits can represent 2N values at once, as opposed to the traditional N bit, which can only represent one value at a time.

For to realize large-scale and reliable quantum computers, it will be necessary to evaluate various quantum error correction schemes and their parameter trade-offs. Designers will be able to manually add quantum gates using the mouse. The application will also allow designers to save the quantum circuit to a file. This file can be used as input to a separate application for simulating quantum circuits.

1. Healthcare:
    - The Quantum computer will allow the simulation of larger molecules and allows researchers to model and simulate the interaction between drugs and more than 20,000 proteins encoded in the human genome to further promote pharmacological research.
    - Quantum technology can be used to provide faster and more accurate diagnosis in a variety of applications.
2. Finance:
    - One potential application of quantum technology is algorithmic trading: the use of complex algorithms to automatically trigger stock trading based on various market variables.
    - Quantum computers can greatly improve machine learning capabilities; greatly reduce the time required to train neural networks and increase the detection rate.
3. Marketing:
    - Quantum Computer will be able to aggregate and analyze large amounts of consumer data from various sources.
    - Big data analytics will allow government and business departments to precisely target individual consumers or voters based on their personal preferences and will help influence consumer spending and election results.
4. Meteorology:

- Machine learning using quantum computers will result in improved pattern recognition, making it easier to predict extreme weather events and potentially saving thousands of lives a year.
- Climatologists will also be able to generate and analyse more detailed climate models, proving greater insight into climate change and how we can mitigate its negative impact.

1.2 Motivation

1. Quantum computing applications include secure computing, reliable data storage and exponential acceleration.
2. Quantum computers are believed to greatly help solve optimization-related problems, which play a key role in everything from national defense to financial transactions.

1.3 Problem statement

Implementation of classical encryption standard as a Quantum circuit.

1.4 Objectives

The objectives of the project are

1. To elucidate the implications of quantum computing in present cryptographic systems.
2. To implement a Novel Quantum cryptography scheme when the users are fully classical.
3. To implement classical encryption standard as Quantum circuits.

1.5 Literature Review

[1] Secure transmission of message between the sender and receiver is carried out through a tool called as cryptography. Traditional cryptographical methods use either public key which is known to everyone or on the private key. In either case the eavesdroppers are able to detect the key and hence find the message transmitted without the knowledge of the sender and receiver. Quantum cryptography is a special form of cryptography which uses the Quantum mechanics to ensure unconditional security towards the transmitted message. Quantum cryptography uses the distribution of random binary key known as the Quantum Key Distribution (QKD) and

hence enables the communicating parties to detect the presence of potential eavesdropper. This paper also analyses few application areas of quantum cryptography and its limitation.Smart cards, PINs, password authentication, etc., use currentcryptographic techniques and they are performing well in keeping data secure. However, the overall security of an encryption system relies in the ability to keep cipher keys secret, but a typical human behavior is to write down passwords either in their inbox of mobile phone or e-mail id, this behavior makes security very vulnerable to attack. The concept of biometric-based keys seems to be one possible solution to this insecurity.

[2] Security is an important aspect of any network, but in particular to mobile adhoc networks. The wireless networks are potential for hacking using mobile devices. There is no clear line of defense for protecting the mobile neworks. The development of the Mobile Application Security System which uses a layered security approach and strong cryptographic techniques is seen as a feasible and low-cost solution to protect these application-based wireless networks.Cryptographers make efforts to build more and more sophisticated means to obscure the sensitive information which is to be transmitted. But the hackers, code breakers, eavesdroppers work furiously to crack the systems. If either cryptographers achieve security or the code breakers decipher the security the success will be temporary. This process of securing the message using ciphering and breaking the system using deciphering is an endless process. Once, this scenario is a chase in the race.

[3] Although written about fifteen years ago, Wiesner's seminal paper, to which the origin of quantum cryptography must be traced back, did not appear in print until the spring of 1983 . The first published account of these ideas thus appeard in the proceedings of the second annual CRYPTO conference. However, the concepts presented there were mostly of theoretical interest, because the technology involved in implementing them would have been far beyond the reach of our current knowledge. In particular, single polarized photons had to be trapped, bounding back and forth between perfectly reflecting mirrors, and perfect efficiency in photon detection was required. To make up for this inconvenience, we could prove that no technology whatsoever, as well as no amount of computing power, could break some of our schemes, as long as some of the most fundamental principles of quantum physics hold true.

[4] introduces role of spectroscopic methods in the development of the most impressive quantum algorithms (such as Shor's polynomial time number decomposition algorithm). Although the fast transformation algorithm reduces the number of operations required for the transforma-

tion from O(2/sup2n/) to O(n/sup2n/), in contrast, quantum transformation is super fast. The quantum Fourier transform can be performed in O(n/sup2/) time, while the specific case of Walsh-Hadamard and Chrestenson transform only requires O(n) operations. They have used Chrestenson gates to derive the quantum Fourier transform on non-binary quantum numbers, which can be used as the basic building block of quantum transforms. We review the role of quantum transformation in the design of efficient quantum algorithms. The circuit for fast quantum transformation is, for example, multi-domain transformation. More work can be done on the design of quantum transformation circuits in binary and ternary quantum gates. It is worth studying whether the same acceleration technology is applicable to the ternary quantum circuit structure. Finally, as quantum machines are physically implemented to verify the Shor and Grover algorithms, and as some current promising technologies may become a reality, it is necessary to reconsider quantum transformation circuits in consideration of this implementation.

[5] When elementary quantum systems, such as polarized photons, are used to transmit digital information, the uncertainty principle gives rise to novel cryptographic phenomena unachievable with traditional transmission media, e.g. a communications channel on which it is impossible in principle to eavesdrop without a high probability of disturbing the transmission in such a way as to be detected. Such a quantum channel can be used in conjunction with ordinary insecure classical channels to distribute random key information between two users with the assurance that it remains unknown to anyone else, even when the users share no secret information initially. We also present a protocol for coin-tossing by exchange of quantum messages, which is secure against traditional kinds of cheating, even by an opponent with unlimited computing power, but ironically can be subverted by use of a still subtler quantum phenomenon, the Einstein-Podolsky-Rosen paradox.

[6] The needs of the business, finance, security and logistics sectors are driving the continued growth of IT. Current research focuses on using destructive methods to provide new computing functions. Quantum computing represents a promising strategy that can provide new functions in secure computing, reliable data storage, and efficient applications. This article focuses on the growing understanding that quantum computing will affect a variety of consumer concerns and applications.

Paper [7] provides a complete review of the numerical simulation and optimization of shot peening found in the existing literature in the last 10 years. The review found that the devel-

oped numerical models combining the finite element and the discrete element are becoming increasingly mature and show their advantages by combining flow behavior and projectile randomness. Great importance must be attached to the constitutive equation of the target material. It is recommended to incorporate its sensitivity of strain rate, cyclical behavior and Bauschinger effect in the numerical model of material at the same time, since considering one of them in isolation can lead to an unreliable residual stress distribution. In addition, the hardening of the material is a key advantage of shot blasting

simulation or optimization in Paper [8], it has not received the attention of existing research. The study found that strength and coverage are two key control parameters, and it is recommended to use them as constraints to optimize shot peening. Finally, this work also found that recently developed heuristic algorithms (such as genetic algorithms) have shown their advantages and can find the best combination of shot peening parameters. In the near future, the synergy of combining these algorithms with approximate models is expected to attract more attention from researchers.

Paper [9] introduces role of spectroscopic methods in the development of the most impressive quantum algorithms. Although the fast transformation algorithm reduces the number of operations required for the transformation from O(2^{2n}) to O(n^{2n}), in contrast, quantum transformation is super fast. The quantum Fourier transform can be performed in O(n^2) time, while the specific case of Walsh-Hadamard and Chrestenson transform only requires O(n) operations. They have used Chrestenson gates to derive the quantum Fourier transform on non-binary quantum numbers, which can be used as the basic building block of quantum transforms. The role of quantum transformation in the design of efficient quantum algorithms is reviewed. The circuit for fast quantum transformation is, for example, multi-domain transformation. More work can be done on the design of quantum transformation circuits in binary and ternary quantum gates. It is worth studying whether the same acceleration technology is applicable to the ternary quantum circuit structure. Finally, as quantum machines are physically implemented to verify the Shor and Grover algorithms, and as some current promising technologies may become a reality, it is necessary to reconsider quantum transformation circuits in consideration of this implementation.

This paper[10] proposes a general method for building fault-tolerant quantum logic gates with simple primitives, which are simulations of quantum teleportation. This technique extends previous results based on traditional quantum teleportation, and leads to direct system con-

struction of many fault-tolerant encoding operations. This technology can also be applied to the construction of remote quantum operations that cannot be executed directly. Here it is proposed that a system technology uses the original 1-bit teleportation scheme to construct various quantum operations. Such a solution reduces the problem of constructing a quantum logic gate to prepare the auxiliary state generated by the same gate applied to a known state. For two types of applications, the practicality of this technology is particularly obvious: fault-tolerant quantum computing and remote quantum computing, our $\pi/8$ gate, phase control and Toffoli and the construction of the remote junction have been proved. With the recursive application of one-bit recursive scheme, it can also build infinite layers of fault-tolerant gates. This two-bit transfer scheme allows all C3 gates to be fault-tolerantly transferred fault-tolerant, and all Ck gates are transferred through the recursive application of this scheme. However, for a bit transfer,

Paper [11] provides sufficient conditions for the gate in C3 to be transferable, that is, any C3 gate node that can write into the product of the C3 gate and the single switched C3 gate. I don't know if this includes the C3 gate. It is difficult to describe a set of accurate one-bit transmission C3 gates. The difficulty is caused by the need for C2C1 gates in a one-bit transmission circuit. This kind of gate C2C1 can be conjugated from C2 through gate C3, so cannot be directly executed with fault tolerance. Due to our current lack of understanding of the general structure and nature of the Ck gate, the difference between the maximum transmission capacity of 1444 and 2 bits mode is still an interesting and difficult open question. However, as already shown, in the construction of quantum logic gates, one-bit transmission can provide a simpler protocol than two-bit transmission. This is because one-bit teleportation requires only Z projection measurement and as many auxiliary qubits as the state to be converted however, two-bit transmission requires Bell measurement, and the number of auxiliary qubits is two of the original state.
Paper[12] discusses that in recent years, due to the possible application of quantum computing to the difficult problems of classical calculus, it has received great attention. Although there are experimental problems with realizing quantum devices, theoretical physicists still try to design some implementations for quantum algorithms. Here some explicit schemes are introduced for performing basic arithmetic operations. Analysis shows how to build circuits for quantum addition and multiplication. It should be noted that in both cases, the input is the vector on which the calculation is based, and therefore the output vector. An important result is that although our process uses the superposition of the sum of state and phase, the final state does not have

the typical ambiguity of quantum mechanics. The observable value measurement allows to obtain one of the eigenvectors of the Hermitian operator associated with it. In our example, the output is already calculating the basis vector, so the measurement result is deterministic. The circuit implemented by can be used to construct a more complex schematic that can perform more complex operations. As shown in the upcoming article, the circuit in question is the basis of the ALU of early quantum CPUs. In the process of system evolution, the problems of decoherence and control was completely ignored. The experimental work shows how difficult it is to manage real qubits. As a result, it is difficult to obtain our circuit through the prior art method.

The author in paper[13] discusses about the FONMAN architecture of the classic computer comprises a central processing unit, instruction and memory retention data. It is demonstrated that the quantum random access memory integrated into the chip and demonstrate central processing unit are integrated into the chip. Try our quantum machine by executing the code that runs two Queen Superconductors coupled through two quantum buses, two quantum memories and two records to zero. Two important algorithms are shown for quantum calculus, the Fourier quantum transform, 66% fidelity and three-elbows toffee class or phase doors have a phase drawer of 98%. Our results show a potentially executable approach to factoring fractions, especially in combination with a particularly long consistency of the quit particularly long, and implements a simple quantum error correction signal. Quantum processors based on nuclear magnetic resonance, captured ions and semiconductor devices are used to perform Shor's quantum decomposition algorithm and quantum error correction. The quantum operations on which these algorithms are based include two-qubit gates, Fourier quantum transform and three-qubit Tofli gates. In addition to quantum processors, the second key element for quantum machines is quantum memory. Circuit superconductor has reached multiple milestones, including demonstrations of two-qubit gates and advanced control of qubits and photon quantum states. Here a superconducting integrated circuit that combines a processor that performs a quantum Fourier transform and a Toffoli-like three-bit OR gate, and has a memory and a clear register in a single device is demonstrated. The combination of quantum central processing unit (quCPU) and quantum random access memory (quRAM) constitutes two key elements of the classical von Neumann architecture and defines our quantum von Neumann architecture.

Paper[14] discusses the needs of the business, finance, security and logistics sectors are driving the continued growth of IT. Current research focuses on using destructive methods to provide

new computing functions. Quantum computing represents a promising strategy that can provide new functions in secure computing, reliable data storage, and efficient applications. This article focuses on the growing understanding that quantum computing will affect a variety of consumer concerns and applications.

Paper[15] tells about how Quantum computing provides great speed when performing tasks, such as data encryption and searching. Quantum algorithms can be modeled using classical computing devices, but classical computer simulations can not be addressed efficiently with the parallel processing that exists in the quantum algorithm. Quantum circuit models of the quantum algorithm are sufficient to describe known quantum algorithms. Simulators between quantum circuits and digital circuits are used to design an emulator of quantum algorithms in FPGA and allow efficient experiments with new quantum algorithms. This document concentrates important technologies in the modeling of quantum circuits, including the implementation of entanglement and probability calculations, as well as important issues in the required computing accuracy. Here the problems in the design and operation of the FPGA-based quantum circuit simulator, and developed the platform for the development of quantum circuits is introduced. The simulator allows the construction of fairly complex quantum circuits from the component library in a simple way. At the same time, simulates the parallelism existing in the quantum computer by constructing a parallel evolution path for each qubit in the FPGA. FPGA simulation has the advantages of because it is difficult to effectively simulate the parallel evolution of quantum systems in software. The simulator is also extensible and has the potential to simulate complex quantum circuits. The simulator can also incorporate other quantum computing concepts, such as quantum error correction, fault-tolerant quantum computing and quantum measurement technology. These are particularly useful in the development of practical systems for quantum computers. Other uses of the quantum simulator may be the analysis, optimization and approximation of the quantum Fourier transform, which is essential for a better quantum algorithm. It is planned further to explore the development of the reversible quantum gate library and the dedicated architecture for simulating quantum algorithms. Finally, a plan to use this simulator to study and optimize the quantum measurement algorithm, which is currently not practical in software simulation.

Here[16] the gate model quantum computer provides an experimentally achievable architecture for short-term quantum computing. In order to design a simplified quantum circuit that can simulate a highly complex quantum reference circuit, the number of input quantum states, the unit

operation of the quantum circuit, and the number of output measurement rounds must be optimized. In addition to optimizing the physical design of the hardware layer, quantum computers should also solve difficult problems very effectively. In order to produce the desired output system, the specific objective function associated with the computational problem introduced into the quantum computer must be maximized. The simplified gate structure should be able to generate the maximum value of the objective function. These parallel requirements must be met at the same time, making optimization difficult. A method for designing quantum circuits for gate model quantum computers and define the quantum triple annealing minimization (QTAM) algorithm. The goal of QTAM is to determine the best simplified topology for the quantum circuit in the hardware layer while maximizing the objective function of any computational problem. The algorithm and method presented in this document provide a framework for the design of quantum circuits for quantum computers short-term door model. As our purpose is to define a plan for current and future quantum computers, the developed algorithm and method were adjusted for any quantum dimension system and any quantum hardware limitation. The results through the architecture of the quantum computer of the GATE model. However, for the flexibility of the scheme, any implementation and input restrictions can be integrated into the minimization of quantum circuits. The target function that is going to be maximized in this way can also be arbitrarily selected. This allows flexible implementation to solve any computational problem for experimental quantum computers with any hardware restriction and development restrictions.

Paper[17] discusses about Machine learning has developed rapidly, has made many breakthroughs in theory, and has been widely used in various fields. As an important part of machine learning, optimization has attracted widespread attention from researchers. With the exponential growth of data volume and the increase of model complexity, optimization methods in machine learning are facing more and more challenges. In order to solve optimization problems or improve optimization methods in machine learning, many works have been proposed one after another. It is very important to review the system afterwards and summarize the optimization methods from the perspective of machine learning, which can provide guidance for optimization development and machine learning research. In this article, first the optimization problem in machine learning is described. Then the principle and progress of commonly used optimization methods are presented. Finally, it is explored and proposed some challenges and unsolved problems to optimize machine learning. This document introduces and summarizes

optimization methods from the perspective of machine learning, and studies the application of methods in various machine learning fields. First, it describes the theoretical basis of the optimization method based on the first-order, high-order and derivative-free aspects, and introduce the research progress in recent years. Then, it describes the application of the optimization method in different machine learning scenarios and the methods to improve its performance. Finally, it is discussed that some challenges and unresolved problems in the machine learning optimization method.

Paper[18] has proposed an efficient method to optimize the parameter values of the structure and the quantum circuit at the same time with very little computational overhead. The performance of the shallow circuit optimized by the structure is significantly better than that of the circuit using only parameter updates, which makes this method particularly suitable for noise medium-scale quantum computers. The method of optimizing is demonstrated using the variable quantum eigen-solver to find the ground state of lithium hydride and Heisenberg model in the simulation, and find the ground state of hydrogen in the IBM Melbourne quantum computer. In the case of circuit optimization, Rotosolve can be generalized to find K rotation angles and rotation angle minimizers at the same time, but 3K circuits need to be evaluated. Although this is an exponential cost, for small K, the method is computationally feasible and may provide advantages. This method belongs to the algorithm class and is called coordinate block minimization. In the case of circuit structure optimization, Rotoselect can be generalized to start from scratch instead of gradually increasing the growth circuit from a random initial structure. This is similar in spirit to Adapt-VQE , but the advantage is that each new gate effectively optimizes in a closed way, rather than using gradient-based optimization. Also, it can effectively eliminate gate by evaluating whether the gate contributes to the solution. Another interesting extension of is the use of Rotoselect to learn the best connection design for tangled doors. For example, consider an ion computer that can implement a connecting layer of 4,444 Mølmer-Sørensen gates. This choice provides greater expressive power in shallow circuits, but the potential slow clock speed of these entangled gates must be weighed against potential gate errors. Our algorithm can find this best position. Since n qubits are used, it can evaluate $\frac{n(n-1)}{2}$ options for non-directional two qubit gates in a layer, so this method is also valid.

Paper[19] discusses the Approximate Quantum Optimization Algorithm (QAOA) is a promising classical quantum hybrid technology for solving the combined optimization problem in noisy short-term gate-based quantum devices. In QAOA, the target is a function of the quantum

state, and the quantum state itself is a function of the gate parameters of the multilevel parameterized quantum circuit (PQC). The classical optimization procedure changes the continuous gate parameter to generate a distribution that has significant support for the best solution. Even at the lowest circuit depth, QAOA can still provide demonstrably important performance guarantees, and is expected to increase by with the depth of the circuit. However, the existing analysis does not take into account the non-ideality in qubit quality, that is, the short lifespan and imperfect operation of gate in real quantum hardware. In this article, the impact of various noise sources on QAOA performance in simulations and in real IBM quantum computers is studied. Our analysis shows that the optimal number of stages (p-value) of any instance of QAOA is limited by the noise characteristics of the target hardware, which is different from the current view, that is, QAOA of greater depth will provide better monotonous performance for a Issue given compared to shallow deployments. The performance of QAOA-MaxCut using the real noise properties of superconducting qubits is analyzed. Our results show that for any single-instance QAOA optimization process, the best p-value will be limited by the qubit quality metric of any target hardware. For the test case considered in this work, the maximum value of the best p is 2. For most actual size problems with existing error rates, any value higher than beyond p = 1 does not guarantee performance improvement. Although the optimal p-value limit is instinctive due to the limited coherence time, the similar limit due to gate error is important. The results show that even if it can achieve qubits with an infinite coherence time of the optimal depth of any QAOA single instance optimization process may be limited by the gate error rate of the hardware destination.

Paper[20] describes the gate model quantum computer with too many qubits is about to be simulated by the available classical computer. They have proposed a strategy to program these devices without bug fixes or compilation. This means that the number of logical qubits is the same as the number of qubits in the device. The hardware determines which pair of qubits can be addressed by a single operator. The goal is to establish a quantum state to solve computational problems, such as maximizing the combined objective function or minimizing the Hamiltonian. These problems may not naturally fit the physical design of qubits. Our algorithm uses parameterized unit sequences in the qubit design, and quantum states are generated based on these parameters. The measurement of the target guides the selection of new parameters. The goal is to move the target function upward. For example, if considered finding the approximate solution of MaxCut on 3 regular charts, and the hardware is physical qubits arranged in a

rectangular grid. It is shown that in all sufficiently large cases, the lowest depth version of the quantum approximation optimization algorithm will achieve an approximation ratio of at least 0.5293 of which exceeds random guesses. It turns on the algorithm to have different parameters to accommodate each rotation of the qubit X and each ZZ interaction related to the closest neighbor interaction in the grid. Small numerical experiments show that the classic envelope algorithm can be used to find the parameters, which can be found on the grid to optimize the objective function with different connectivity. They have discussed strategies for finding suitable parameters, but still have not provided evidence that the proposed method can beat the best classical algorithm. Ultimately, the strength of this method will be determined by running on actual hardware.

1.6 Brief Methodology of the project

The following steps have been carried out to implement the quantum circuit for the classical encryption standard:

1. Understanding classical encryption standards and steps involved to cipher the data.
2. The classical states in encryption standard is replaced using qubits in Quantum circuits.
3. Classical encryption standards uses multiple linear transformations in which all needed calculations can be expressed with NOT and CNOT gates.
4. The non linear transformations in the encryption standard which is a substitution table cannot be only expressed using NOT and CNOT gates hence those transformation are expressed as NCT gates (NOT, CNOT and Toffoli gates).

1.7 Organization of the report

This report is organized as follows. Write the discussions in each chapter. A sample is as follows.

- Chapter 2 discusses the fundamentals of Quantum Cryptography.
- Chapter 3 discusses the design of RSA Algorithm of Quantum Computers .
- Chapter 4 discusses the implementation of RSA Algorithm of Quantum Computers in IBM quantum experience.

- Chapter 5 discusses results of the algorithm and its analysis .
- Chapter 6 discusses conclusion and future scope of the project .

.

Chapter 2

Fundamentals of Quantum cryptography

CHAPTER 2

FUNDAMENTALS OF QUANTUM CRYPTOGRAPHY

Quantum cryptography is a science that applies quantum mechanics principles to data encryption and data transmission so that data cannot be accessed by hackers – even by those malicious actors that have quantum computing of their own.The first section discusses about the present cryptography. This chapter also describes the comparison between classical and quantum computing.This chapter also discusses the vulnerablity of cryptosystem to quantum algorithms.

2.1 Present cryptography

In this section the role of symmetric algorithms, asymmetric algorithms and hash functions in modern cryptography are briefly explained . The difficulty of factorizing large numbers, as well as the discrete logarithm problem which is the basis of strong asymmetric ciphers is discussed .

2.1.1 Symmetric Cryptography

In symmetric cryptography, the sender and the receiver use the same secret key and the same cryptographic algorithm to encrypt and decrypt data. For example, Alice can encrypt a plaintext message using her shared secret key and Bob can decrypt the message using the same cryptographic algorithm Alice used and the same shared secret key. The key needs to be kept secret, meaning that only Alice and Bob should know it; therefore, an efficient way for exchanging secret keys over public networks is demanded. Asymmetric cryptography was introduced to solve the problem of key distribution in symmetric cryptography. Popular symmetric algorithms include the advanced encryption standard (AES) and the data encryption standard (3DES).

2.1.2 Asymmetric Cryptography

Asymmetric cryptography or public key cryptography (PKC) is a form of encryption where the keys come in pairs. Each party should have its own private and public key. For instance, if Bob wants to encrypt a message, Alice would send her public key to Bob and then Bob can encrypt the message with Alice's public key. Next, Bob would transmit the encrypted message to Alice who is able to decrypt the message with her private key. Thus, we encrypt the message with a public key and only the person who owns the private key can decrypt the message.

2.2 Quantum Computing vs Classical Computing

In 1982, Richard Feynman came up with the idea of quantum computer, a computer that uses the effects of quantum mechanics to its advantage. In a sense, qubits are particles that can exist not only in the 0 and 1 state but in both simultaneously, known as superposition. A particle collapses into one of these states when it is inspected. Quantum computers take advantage of this property mentioned to solve complex problems. An operation on a qubit in superposition acts on both values at the same time. Another physical phenomenon used in quantum computing is quantum entanglement. When two qubits are entangled their quantum state can no longer be described independently of each other, but as a single object with four different states. In addition, if one of the two qubits state change the entangled qubit will change too regardless of the distance between them. This leads to true parallel processing power. The combination of the aforementioned phenomena result in exponential increase in the number of values that can be processed in one operation, when the number of entanglement qubits increase. Therefore, a n-qubit quantum computer can process 2 n operations in parallel. Two kinds of quantum computers exists; universal and non-universal. The main difference between the two is that universal quantum computers are developed to perform any given task, whereas non-universal quantum computers are developed for a given purpose (e.g., optimization of machine learning algorithms). Examples are, D-Wave's 2000+ qubits non-universal quantum computer and IBM's 17 qubits universal quantum computer with proper error correction. IBM's quantum computer is currently the state of the art of universal quantum computers. Both D-Wave and IBM have quantum computers accessible online for research purposes. Additionally, in October 2017, Intel in collaboration with QuTech announced their 17-qubits universal quantum computer.

Bone and Castro stated that a quantum computer is completely different in design than a classical computer that uses the traditional transistors and diodes. Researchers have experimented with many different designs such as quantum dots which are basically electrons being in a superposition state, and computing liquids. Besides, they remarked that quantum computers can show their superiority over the classical computers only when used with algorithms that exploit the power of quantum parallelism. For example, a quantum computer would not be any faster than a traditional computer in multiplication.

2.3 Cryptosystems vulnerable to Quantum Algorithms

This section discusses the impact of quantum algorithms on present cryptography and gives an introduction to Shor's algorithm and Grover's algorithm. Note that Shor's algorithm explained in the following subsection makes the algorithms that rely on the difficulty of factorizing or computing discrete logarithms vulnerable. Cryptography plays an important role in every electronic communication system today. For example the security of emails, passwords, financial transactions, or even electronic voting systems require the same security objectives such as confidentiality and integrity [12]. Cryptography makes sure that only parties that have exchanged keys can read the encrypted message (also called authentic parties). Quantum computers threaten the main goal of every secure and authentic communication because they are able to do computations that classical (conventional) computers cannot. Consequently, quantum computers can break the cryptographic keys quickly by calculating or searching exhaustively all secret keys, allowing an eavesdropper to intercept the communication channel between authentic parties (sender/receiver). This task is considered to be computational infeasible by a conventional computer.

2.3.1 Shor's Algorithm in Asymmetric Cryptography

In 1994, the mathematician Peter Shor in his paper "Algorithms for Quantum Computation: Discrete Logarithms and Factoring" [15], proved that factorizing large integers would change fundamentally with a quantum computer. Shor's algorithm can make modern asymmetric cryptography collapse since is it based on large prime integer factorization or the discrete logarithm problem. To understand how Shor's algorithm factorizes large prime numbers we use the following example. We want to find the prime factors of number 15. To do so, we need a 4-qubit register. We can visualize a 4-qubit register as a normal 4-bit register of a traditional computer. Number 15 in binary is 1111, so a 4- qubit register is enough to accommodate (calculate) the prime factorization of this number. According to Bone and Castro, a calculation performed on the register can be thought as computations done in parallel for every possible value that the register can take (0-15). This is also the only step needed to be performed on a quantum computer.

This algorithm was proposed by Peter Shor which was used to factor number into its prime components . It is a polynomial time quantum computer algorithm for integer factorization. The most popular classical factoring algorithm requires super polynomial time to factor the product of two primes, the widely used cryptographic system Rivest-Shamir-Adleman (RSA)

depends on considering being unthinkable to factor large numbers. Shor's Algorithm uses combination of Quantum Fourier Transform (QFT) and Quantum Phase Estimation (QPE) to find the factor of an integer on quantum computer. The efficiency of the QFT and modular exponentiation by repeated squaring helps in increasing the efficiency of Shor's Algorithm. Shor's Algorithm was first demonstrated by group at International Business Machines Corporation (IBM). The group factored the number 15 using a quantum computer implemented using Nuclear Magnetic Response (NMR) with 7 qubits. Figure 2.1 denotes a basic flowchart showing an overview of Shor's Algorithm. The algorithm does the following:

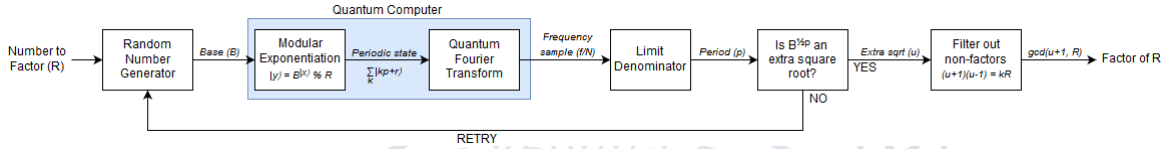


Figure 2.1: Flow Chart of Shor's Algorithm

1. n = 15, is the number to factorize
2. x = random number such as $1 < x < n - 1$
3. x is raised to the power contained in the register (every possible state) and then divided by n The remainder from this operation is stored in a second 4-qubit register. The second register now contains the superposition results. Let's assume that x = 2 which is larger than 1 and smaller than 14.
4. If the powers of the 4-qubit register is raised to x which is a maximum of 15 and divide by 15 the results is a repeating sequence of 4 numbers (1,2,4,8). It can confidently said then that f = 4 which is the sequence when x = 2 and n = 15. The value f can be used to calculate a possible factor with the following equation: Possible factor: $P = x^{f/2} - 1$

Shor's algorithm can be used additionally for computing discrete logarithm problems. Vazirani [16] explored in detail the methodology of Shor's algorithm and showed that by starting from a random superposition state of two integers, and by performing a series of Fourier transformations, a new superposition can be set-up to give us with high probability two integers that satisfy an equation. By using this equation we can calculate the value r which is the unknown "exponent" in the DLP.

2.3.2 Grover's algorithm in symmetric cryptography

Lov Grover created an algorithm that uses quantum computers to search unsorted databases. The algorithm can find a specific entry in an unsorted database of N entries in N searches. In comparison, a conventional computer would need N/2 searches to find the same entry. Bone and Castro [8] remarked the impact of a possible application of Grover's algorithm to crack Data Encryption Standard (DES), which relies its security on a 56-bit key. The authors remarked that the algorithm needs only 185 searches to find the key. Currently, to prevent password cracking we increase the number of key bits (larger key space); as a result, the number of searches needed to crack a password increases exponentially. Buchmann et al. [18] stated that Grover's algorithm have some applications to symmetric cryptosystems but it is not as fast as Shor's algorithm

2.4 Quantum Key Distribution

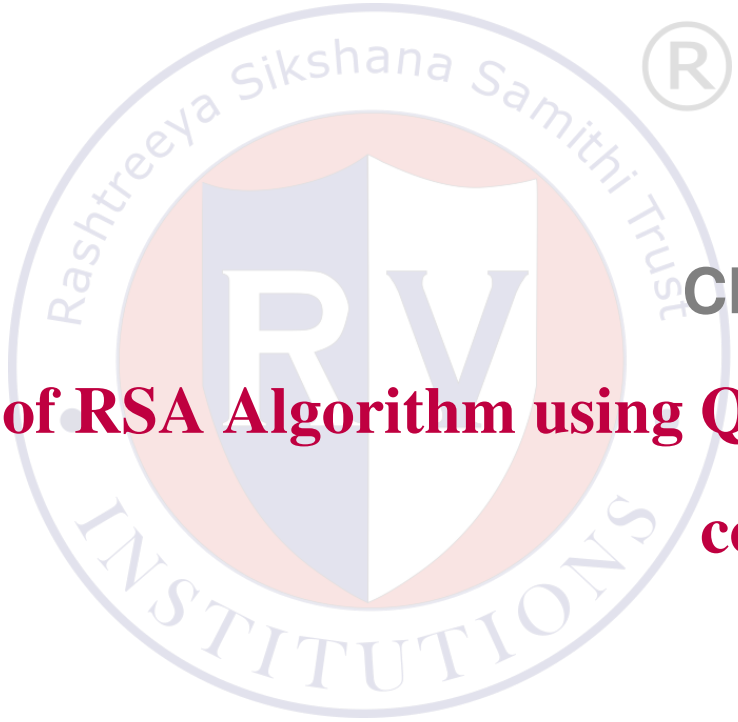
Quantum Key Distribution (QKD) addresses the challenge of securely exchanging a cryptographic key between two parties over an insecure channel. QKD relies on the fundamental characteristics of quantum mechanics which are invulnerable to increasing computational power, and may be performed by using the quantum properties of light, lasers, fibre-optics as well as free space transmission technology. QKD was first introduced in 1984 when Charles Bennett and Gilles Brassard developed their BB84 protocol. Research has led to the development of many new QKD protocols exploiting mainly two different properties that are described right below. Prepare-and-measure protocols use the Heisenberg Uncertainty principle stating that the measuring act of a quantum state changes that state in some way. This makes it difficult for an attacker to eavesdrop on a communication channel without leaving any trace. In case of eavesdropping the legitimate exchange parties are able to discard the corrupted information as well as to calculate the amount of information that has been intercepted. This property was exploited in BB84.

Entanglement based (EB) protocols use pairs of entangled objects which are shared between two parties. As explained in III, entanglement is a quantum physical phenomenon which links two or more objects together in such a way that afterwards they have to be considered as one object. Additionally, measuring one of the objects would affect the other as well. In practice when an entangled pair of objects is shared between two legitimate exchange parties anyone intercepting either object would alter the overall system. This would reveal the presence of an attacker along with the amount of information that the attacker retrieved. This property was

exploited in E91 protocol.

Both of the above-mentioned approaches are additionally divided into three families; discrete variable coding, continuous variable coding and distributed phase reference coding. The main difference between these families is the type of detecting system used. Both discrete variable coding and distributed phase reference coding use photon counting and post-select the events in which a detection has effectively taken place. Continuous variable coding uses homodyne detection which is a comparison of modulation of a single frequency of an oscillating signal with a standard oscillation.

In this chapter a detailed introduction to Quantum cryptography has been discussed . This chapter also discusses comparison between quantum and classical computing. Various quantum algorithms have been discussed and its implementation on Quantum computers.Quantum key distribution is also described in this chapter.



Chapter 3

Design of RSA Algorithm using Quantum computer

CHAPTER 3

DESIGN OF RSA ALGORITHM USING QUANTUM COMPUTER

RSA is a special asymmetric cryptosystem which enables one group to both encrypt and decrypt the data while restricting the other to only decrypting. This chapter discusses various step involved in designing of components of RSA algorithm on quantum computer. This chapter also describes the Quantum Key Distribution (QKD) method to transmit the message over an open communication channel. The chapter concludes with using Shor's Algorithm to decrypt the data.

3.1 RSA Algorithm

RSA is a public key cryptosystem which is widely used for secured transmission. It is one of the oldest public-key cryptosystem which enables one group to encrypt and decrypt data while restricting another to only decrypting. An RSA user creates and publishes a public key based on two large prime numbers, along with an auxiliary value. The prime numbers are kept secret. Messages can be encrypted by anyone, via the public key, but can only be decoded by someone who knows the prime numbers [21]. The mathematics required to encrypt the data is straightforward for a computer, but decrypting the data takes an unreasonably large amount of computing resources. Two distinct pieces of information are required to obtain the full range of the RSA function, a public and a private key. RSA's public key derives from the two product of two large prime numbers, which is available to anyone publicly for encrypting data. However, only people with the actual prime numbers themselves can decrypt the data; this is called the private key. Figure 3.1 describes a basic approach of working of RSA algorithm.

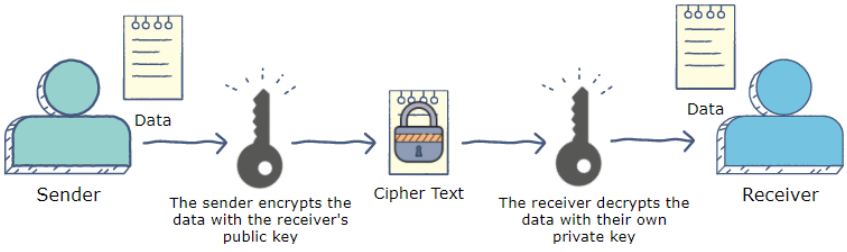


Figure 3.1: Working of RSA Algorithm

This algorithm involves three steps described as follows:

3.1.1 Key Generation

This is the first step involved in ciphering the data using RSA algorithm. Two distinct prime numbers P and Q is taken as an input from user or can be generated randomly. These two numbers are kept secret. Primality test is run on these numbers to verify whether they are prime or not. This test is run on quantum computer using Shor's Algorithm. This algorithm was used to factor number into its prime components and performs the computation in polynomial time whereas the best known Quantum algorithm requires super polynomial time to factor numbers. The Shor's algorithm is called for both the numbers. If the factor list is NULL then the input number is Prime else the different number is considered. Once the numbers are finalised, the value of N is calculated using the Equation 3.1.

$$N = P * Q \tag{3.1}$$

The multiplication of both the prime numbers is computed using multiplication function based on QFT. QFT provides an alternative way to perform arithmetic operations on a quantum computer. The QFT of a state $|x\rangle$ is described as in Equation 3.2.

$$QFT_N|x\rangle = \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} \omega_N^{xy} |y\rangle \tag{3.2}$$

The QFT allows to encode a number x in the relative phases of the states of uniform superpositions consisting in the sum of all the states. The ADD operation can be performed using QFT using one of the basic gate i.e. Controlled Phase Gate, CZ. These quantum operator acts on distributed phases of the quantum states. These gates are controlled by the n qubits that represent the represent the number. The combined effect of all the gates introduce a total phase for each state so that the QFT register stores the sum of two numbers. The quantum circuit to multiply is carried out by performing n consecutive controlled QFT addition. The result will be $2n$ qubit register encoding the number $a * b$. Hence, QFT provides a more versatile way to perform arithmetic operations.

Next, $\phi(n)$ is calculated as per the Equation 3.3.

$$\phi(n) = (P - 1) * (Q - 1) \tag{3.3}$$

Then an integer e is chosen such that $1 < e < \phi(n)$ and $gcd(e, \psi(n)) = 1$. The value of d is

determined where d is modular multiplicative inverse of e modulo $\phi(n)$ as given by the Equation 3.4.

$$d = e^{-1} mod \phi(n) \tag{3.4}$$

3.1.2 Key Distribution

Once the key for a particular case of transmission is done and public key is generated then key distribution takes place. Suppose two people Alice and Bob wants to set up a transmission between them using this algorithm. The public key is used to encrypt the message. Hence it has to be transmitted over a secure channel. In this project the public key is shared over an open communication channel using QKD method. Quantum key distribution is a cryptographic technique which allows two remote users to establish a secure key (random bit string) between them which can be used further for secure communication, using the principles of quantum mechanics. In 1984, Charles Bennett and Giles Brassard put forward the first quantum cryptography protocol named as BB84 protocol which uses polarisation states of photon as the physical entity to generate a secure bit string among two users. This protocol makes use of a fact that measuring a qubit can change its state. If Alice sends Bob a qubit, and an eavesdropper (Eve) tries to measure it before Bob does, there is a chance that Eve's measurement will change the state of the qubit and Bob will not receive the qubit state Alice sent.

The Quantum key distribution protocol involves the following steps:

1. Alice chooses a string of random bits and also chooses bases for each bits and keeps this data private to him.

2. Alice then encodes the message which is a string of random bits using the bases for that bit and the encoded message is sent to Bob.

3. Bob generates a random string of bases and measures the encoded message using those bases. The measurement is kept private by Bob.

4. Bob and Alice then publicly share which basis they used for each qubit. If Bob measured a qubit in the same basis Alice prepared it in, they use this to form part of their shared secret key, otherwise they discard the information for that bit.

5. Finally, Bob and Alice share a random sample of their keys, and if the samples match, they can be sure (to a small margin of error) that their transmission is successful.

This method ensures a successful and secured transmission of public key over an open communication channel. Even if a third untrusted party tries to eavesdrop, the measurement will cause the states to change changing the measurement at the Bob side and since the sample will not be same can identify interference and retransmission will take place.

3.1.3 Encryption and Decryption

After the public key is obtained over the secure communication channel, the data entered is encrypted using the pubic key using modular exponentiation scheme. The message is first convert into integers and then the ciphertext is computed using the public key as shown in the Equation 3.5. The encoded message is then sent to the recipient.

$$C = M^e mod N \tag{3.5}$$

The message can be recovered from the ciphertext by using the private key component as per the Equation 3.6. Given the ciphertext, the original text can be recovered by reversing the padding scheme.

$$M = C^d mod N \tag{3.6}$$

Decryption of the message is carried out using Shor's Algorithm in this project. Shor's algorithm is quantum algorithm used to find the period of cyclic or periodic functions. By representing a product of two prime numbers, called the coprime, as a periodic function using the modulo operator, and converting this equation into a form that a quantum computer can process, Shor's algorithm can determine the period of that function. Interestingly, using the period of this function, a quantum computer could factor the coprime number. The value of N which is shared through the communication channel is given as an input to the Shor instance which gives factors as the output. These factors are then used to determine the private key for the message. The encrypted message then received can be easily decrypted using the private key.

The following chapter describes designing the flow of the algorithm on a quantum computer. The chapter highlights each steps and details on performing arithmetic computation using Quantum Fourier Transform. The chapter also describes in detail the protocol for Quantum Key distribution which will be implemented in next chapter to transmit the public key. Shor's Algorithm has been implemented to decrypt the message.

Chapter 4

Implementation of RSA Algorithm

CHAPTER 4

IMPLEMENTATION OF RSA ALGORITHM

RSA is the most commonly used asymmetric cryptographic algorithm used in transmitting data over the web. The algorithm with the method of QKD has been implemented on IBM Quantum Experience [22] where the message is encrypted from the sender side and is decrypted at the receiver side. The first section gives an introduction to the platform. The next section discusses the flowchart of the program along with its implementation using Python.

4.1 IBM Quantum Experience

IBM Quantum Experience is the collective form of IBM Quantum Composer and IBM Quantum Lab. It creates an online stage permitting general public and premium admittance given by IBM for cloud-based quantum processing administrations and incorporates admittance to bunch of model quantum processors from IBM. Presently, IBM has added around 20 processors for the service available in different geographical locations with difference in number of qubits. Clients cooperate with a quantum processor through quantum circuits created programmatically using Jupyter Notebooks. These quantum circuits are compiled down to OpenQASM for execution on real processors [23].

4.2 RSA implementation on IBMQ

As discussed in the previous chapter, arithmetic operations has been implemented on quantum computer based on QFT. The key is generated using these arithmetic operations and the decryption is implemented using Shor's Algorithm. The actual implementation is done by utilizing the IBM Quantum Information Science Kit (QISKIT) using Python programming language and IBM Quantum Assembly (QASM). Figure 4.1 describes the flowchart of the algorithm. The steps of the algorithm are described as follows:

1. **Import libraries** - The first step is to import all the Quantum libraries in the workspace. Grover algorithm instance is imported from the qiskit.aqua.algorithms library. The simulator background is initialised with 'simulator_mps' which is a 100 qubit. Code 4.1 indicates all the instances imported from various quantum libraries.

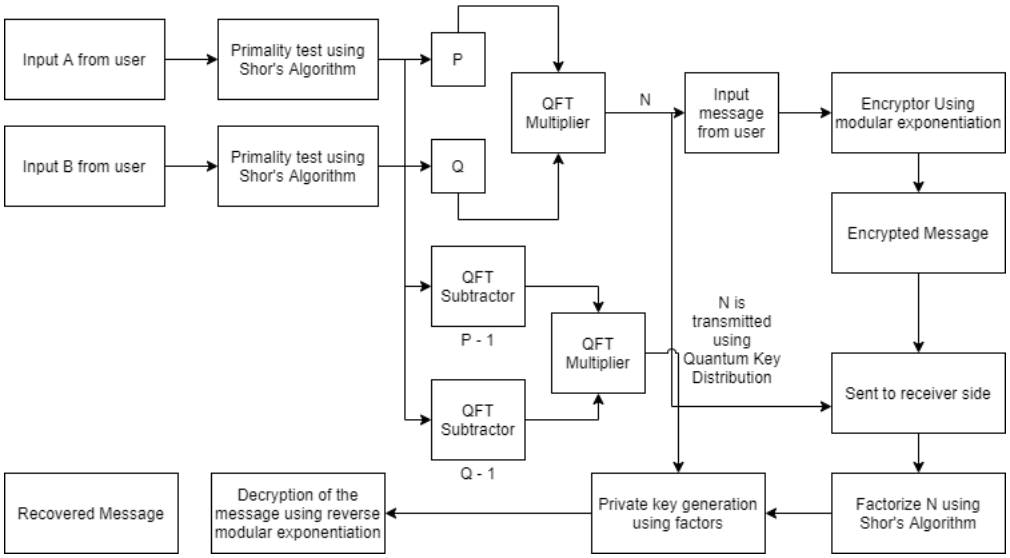


Figure 4.1: Flow Chart of the algorithm

```
import time
from qiskit import QuantumRegister, ClassicalRegister
from qiskit import QuantumCircuit
from qiskit import execute
from qiskit import IBMQ
from qiskit import Aer
import math
import operator
from random import seed
from random import randint
from numpy import gcd
import numpy as np
IBMQ.save_account('API_token')
IBMQ.load_account()
provider = IBMQ.get_provider(hub='ibm-q',group='open')
sim = provider.get_backend('simulator_mps')
```

Figure 4.1: Importing libraries in workspace

2. **Primality Test** - Primality test is a test to check whether the number is prime or not. The test is implemented using Shor's algorithm. The function is defined which takes a number as its input arguments and Shor's algorithm instance from qiskit.aqua.algorithms

library is initialised and the number is then factorised. This particular function is called for two applications one for checking whether the number is prime or not and second for factorising the number. The factor list is returned as an output of this function. Code 4.2 describes the code to factorise the number and check for the primality.

```
def isPrime(number):
    from qiskit import IBMQ
    from qiskit.aqua import QuantumInstance
    from qiskit.aqua.algorithms import Shor
    IBMQ.save_account('API_token')
    IBMQ.load_account()
    provider = IBMQ.get_provider(hub='ibm-q')
    sim = provider.get_backend('simulator_mps')
    factors = Shor(number)
    result_dict = factors.run(QuantumInstance(sim, shots=1024,
       skip_qobj_validation=False))
    result = result_dict['factors'] # Get factors from results
    return result
```

Figure 4.2: Factorising using Shor's Algorithm

3. **Implementing Quantum Fourier Transform** - All the algebraic expression is implemented using QFT. This function initialises the QFT operation where initially it creates the input state and then evolves the QFT state to it and finally inverse QFT operation is done to get the value back in computational basis. Code 4.3 describes the code to implement the QFT operation.

```
def CreateInputState(qc,reg,n,pie):
    qc.h(reg[n])
    for i in range(0,n):
        qc.cu1(pie/float(2**(i+1)),reg[n-(i+1)],reg[n])
def evolveQFTstate(qc,reg_a,reg_b,n,pie,factor):
    l = len(reg_b)
    for i in range (0,n+1):
        if((n-i)>l-1):
            pass
```

```
        else:
            qc.cu1(factor * pie / float(2**(i)), reg_b[n - i],
                reg_a[n])
def inverseQFT(qc,reg,n,pie):
    for i in range(0,n):
        qc.cu1(-1*pie/float(2**(n-i)),reg[i],reg[n])
    qc.h(reg[n])
```

Figure 4.3: Implementing the QFT operation

4. **Arithmetic computation** - This step involves using the above step to compute arithmetic operations based on QFT. The multiplication operation is implemented using N controlled additions. Code 4.4 describes the function to implement the arithmetic computation using QFT.

```
def QFTADD(reg_a,reg_b,circ,factor=1):
    pie = math.pi
    n = len(reg_a)
    for i in range(0,n):
        CreateInputState(circ,reg_a,n-(i+1),pie)
    for i in range(0,n):
        evolveQFTstate(circ,reg_a,reg_b,n-(i+1),pie,factor)
    for i in range(0,n):
        inverseQFT(circ,reg_a,i,pie)
def QFTsub(reg_a,reg_b,circ):
    QFTADD(reg_a,reg_b,circ,-1)
def mul(multiplicand_in,multiplier_in):
    l1=len(multiplicand_in)
    l2 = len(multiplier_in)
    if l2>l1:
        multiplier_in,multiplicand_in = multiplicand_in,
            multiplier_in
        l2,l1 = l1,l2
    accumulator = QuantumRegister(l1+l2)
    multiplicand = QuantumRegister(l1)
    multiplier = QuantumRegister(l2)
```

```
    d = QuantumRegister(1)
    cl_accumulator = ClassicalRegister(l1+l2)
    cl_multiplier = ClassicalRegister(l2)
    circ_accumulator = QuantumCircuit(accumulator,multiplicand,
       cl_accumulator)
    circ_multiplier = QuantumCircuit(multiplier,d,cl_multiplier)
    circ_multiplier.x(d[0])
    for i in range(l1):
        if multiplicand_in[i] == '1':
            circ_accumulator.x(multiplicand[l1-i-1])
    for i in range(l2):
        if multiplier_in[i] == '1':
            circ_multiplier.x(multiplier[l2-i-1])
    multiplier_str = '1'
    while(int(multiplier_str) != 0):
        QFTADD(accumulator, multiplicand, circ_accumulator)
        QFTsub(multiplier, d, circ_multiplier)
        circ_multiplier.measure(multiplier, cl_multiplier)
        result = execute(circ_multiplier, backend=Aer.
           get_backend('qasm_simulator'), shots=2).result().
           get_counts(circ_multiplier)
        multiplier_str = list(result.keys())[0]

    circ_accumulator.measure(accumulator, cl_accumulator)
    result = execute(circ_accumulator, backend=Aer.get_backend('
       qasm_simulator'), shots=2).result().get_counts(
       circ_accumulator)
    total = list(result.keys())[0]
    return total
```

Figure 4.4: Arithmetic computation using QFT

5. **Encryption using RSA algorithm** - This code snippet describes the encryption using RSA Algorithm. Two prime number inputs are taken from user and its primality is tested using the second step. Both numbers are passed to the step 2 if the returned factor list is

empty then the numbers are prime else these numbers are again taken from user. Public key is calculated using the value of N which is product of two input and Phi of N which is product of number one less than inputs. N and Phi of N is used to calculate the public key for encoding the message. The encrypt function is run which takes the input message and public key as arguments and returns the encrypted using modular exponentiation. Code 4.5 details the encrypt function and the encrypted message is returned as the output.

```
print('\n Encryption using RSA Algorithm')
print('--------------------')
print('\n Please enter the values of P and Q')
P = int(input('Enter a prime number for P: '))
Q = int(input('Enter a prime number for Q: '))
print('--------------------')
#Check if P and Q are prime or not
check_P = isPrime(P)
check_Q = isPrime(Q)
while((check_P != []) or (check_Q != [])):
    P = int(input('Enter a prime number for P: '))
    Q = int(input('Enter a prime number for Q: '))
    check_P = isPrime(P)
    check_Q = isPrime(Q)
#Product of P and Q
N = mul(format(P,"b"),format(Q,"b"))
print('The Product of P and Q:',int(N,2))
PhiOfN = mul(format((P-1),"b"),format((Q-1),"b"))
print('The Product of P and Q:',int(PhiOfN,2))
L = int(PhiOfN,2)
for E in range(2,L):
    if gcd(int(N,2),E) * gcd(int(PhiOfN,2),E)==1:
        break
public_key = (E,int(N,2))
print("The public key is: ",public_key)
def encrypt(pub_key,n_text):
    e,n = pub_key
    x=[]
```

```
    m=0
    for i in n_text:
        if(i.isupper()):
            m = ord(i)-65
            c = (m**e)%n
            x.append(c)
        elif(i.islower()):
            m = ord(i)-97
            c = (m**e)%n
            x.append(c)
        elif(i.isspace()):
            spc=400
            x.append(spc)
    return x
message = input("Please enter a message to encrypt: ")
enc_msg = encrypt(public_key,message)
print('The Encrypted Message is: ',enc_msg)
```

Figure 4.5: RSA encryption and public key generation

6. **Quantum Key Distribution** - This step involves transmission of the value of N over an open communication channel using Quantum Key Distribution. The value of N is stored in binary and bases for each bit is randomly generated and each bit is encoded using the generated bases using the function encode_message. The encoded message is sent to the receiver side which also generates its set of bases randomly and measures the output using the measure_message function. The results are stored and are used further for decryption. Any interference from untrusted part will change the state and hence the output won't match. Code 4.6 demonstrates the QKD protocol to transmit the message.

```
def encode_message(bits, bases):
    message = []
    for i in range(n):
        qc = QuantumCircuit(1,1)
        if bases[i] == 0: # Prepare qubit in Z-basis
            if bits[i] == 0:
                pass
```

```
            else:
                qc.x(0)
        else: # Prepare qubit in X-basis
            if bits[i] == 0:
                qc.h(0)
            else:
                qc.x(0)
                qc.h(0)
        qc.barrier()
        message.append(qc)
    return message
def measure_message(message, bases):
    backend = Aer.get_backend('qasm_simulator')
    measurements = []
    for q in range(n):
        if bases[q] == 0: # measuring in Z-basis
            message[q].measure(0,0)
        if bases[q] == 1: # measuring in X-basis
            message[q].h(0)
            message[q].measure(0,0)
        qasm_sim = Aer.get_backend('qasm_simulator')
        qobj = assemble(message[q], shots=1, memory=True)
        result = qasm_sim.run(qobj).result()
        measured_bit = int(result.get_memory()[0])
        measurements.append(measured_bit)
    return measurement
key_to_distribute = '000000'+ format(N,"b")
key_to_distribute
message =  np.array(list(key_to_distribute), dtype=int)
print(message)
n = len(message)
alice_bases = randint(2, size=n)
print(alice_bases)
message_to_send = encode_message(message, alice_bases)
```

```
bob_bases = randint(2, size=n)
print(bob_bases)
bob_results = measure_message(message_to_send, bob_bases)
print(bob_results)
```

Figure 4.6: Quantum Key Distribution protocol

7. **Decryption using Shor's Algorithm** - The RSA encrypted message is sent to the receiver side with the value of N which is sent over an open communication channel without getting detected. With the value of N, private key is generated using factorisation operation. The value is passed to step 2 which uses Shor's Algorithm to factorise the number into its prime components and returns the factor list. The private key is generated using these factors. The encrypted message is sent to the decrypt function along with the private key which reverses the encryption operation using inverse modular exponentiation operation. Hence the original message is recovered. Code 4.7 describes the code for implementing the decrypt function.

```
print('\n Decryption using Shor Algorithm')
print('--------------------')
#Finding factors using Shor Algorithm
factors = isPrime(int(N,2))
factors
#Finding the Private Key
P_dec = factors[0][0]
Q_dec = factors[0][1]
L = (P_dec - 1)*(Q_dec - 1)
D=1
while True:
    if D * E % L == 1 and D!=E and D!=int(N,2):
        break
    D += 1
private_key = (D,int(N,2))
print('The Private Key is: ',private_key)
def decrypt(priv_key,c_text):
    d,n=priv_key
```

```
    txt=c_text.split(',')
    x=''
    m=0
    for i in txt:
        if(i=='400'):
            x+=' '
        else:
            m=(int(i)**d)%n
            m+=65
            c=chr(m)
            x+=c
    return x
message = input("Enter the message to decrypt(Separate numbers
   with ','):")
dec_message = decrypt(private_key,message)
dec_message
```

Figure 4.7: Decryption using Shor's Algorithm

This chapter describes each steps of the algorithm which was used to encrypt and decrypt the message using RSA algorithm on quantum computer. This chapter also indicates the outputs of this code which will be described in the next chapter.

Chapter 5

Results & Discussions

CHAPTER 5

RESULTS & DISCUSSIONS

The implementation of RSA algorithm from the previous chapter is run on IBM Quantum experience and the results are verified. The first section shows the output of the encryptor after a message is entered by a user. The second section verifies the working of Quantum Key distribution and the last section describes the output of the decryptor to recover the original message entered by the user.

5.1 Output of the Encryptor

The encryptor part of the algorithm is implemented by applying arithmetic operations required for the encryption using QFT. The user inputs two prime number and its primality is tested. Figure 5.1 shows the entry of two number and the verification whether they are prime or not using Shor's Algorithm.

```
 Please enter the values of P and Q
Enter a prime number for P: 7
Enter a prime number for Q: 5
--------------------


configrc.store_credentials:WARNING:2021-06-03 15:01:46,935: Credentials already present. Set overwrite=True to overwrite.
ibmqfactory.load_account:WARNING:2021-06-03 15:01:47,142: Credentials are already in use. The existing account in the session w
ill be replaced.
/opt/conda/lib/python3.8/site-packages/qiskit/aqua/algorithms/factorizers/shor.py:69: DeprecationWarning: The package qiskit.aq
ua.algorithms.factorizers is deprecated. It was moved/refactored to qiskit.algorithms.factorizers (pip install qiskit-terra). F
or more information see <https://github.com/Qiskit/qiskit-aqua/blob/master/README.md#migration-guide>
  warn_package('aqua.algorithms.factorizers',
/opt/conda/lib/python3.8/site-packages/qiskit/aqua/quantum_instance.py:135: DeprecationWarning: The class qiskit.aqua.QuantumIn
stance is deprecated. It was moved/refactored to qiskit.utils.QuantumInstance (pip install qiskit-terra). For more information
see <https://github.com/Qiskit/qiskit-aqua/blob/master/README.md#migration-guide>
  warn_class('aqua.QuantumInstance',
/opt/conda/lib/python3.8/site-packages/qiskit/providers/ibmq/ibmqbackend.py:814: DeprecationWarning: Passing a Qobj to Backend.
run is deprecated and will be removed in a future release. Please pass in circuits or pulse schedules instead.
  return super().run(circuits, job_name=job_name, job_share_level=job_share_level,
configrc.store_credentials:WARNING:2021-06-03 15:02:03,602: Credentials already present. Set overwrite=True to overwrite.
ibmqfactory.load_account:WARNING:2021-06-03 15:02:03,789: Credentials are already in use. The existing account in the session w
ill be replaced.
/opt/conda/lib/python3.8/site-packages/qiskit/providers/ibmq/ibmqbackend.py:814: DeprecationWarning: Passing a Qobj to Backend.
run is deprecated and will be removed in a future release. Please pass in circuits or pulse schedules instead.
  return super().run(circuits, job_name=job_name, job_share_level=job_share_level,


The inputs entered are prime number
```

Figure 5.1: Input from user and its primality test

After the primality test, value of N and PhiOfN is calculated using QFT multiplier and using these values public key is generated which will be helped in encoding the data. Figure 5.2 shows the output of the public key generation for encoding the data.

Once the public key is generated, the user enters a message to be encoded. The encryptor function runs taking the input message and the public key as arguments and return the encrypted message. Figure 5.3 shows the output of the last step of the encryption. Running primality test using any classical algorithm takes superpolynomial time to find results but checking for prime

```
<ipython-input-3-e2ae8752acde>:5: DeprecationWarning: The QuantumCircuit.cu1 method is deprecated as of 0.16.0. It will be remo
ved no earlier than 3 months after the release date. You should use the QuantumCircuit.cp method instead, which acts identicall
y.
  qc.cu1(pie/float(2**(i+1)),reg[n-(i+1)],reg[n])

The Product of P and Q: 35
The Product of P and Q: 24
The public key is:  (11, 35)
```

Figure 5.2: Public Key Generation

```
Please enter a message to encrypt: Implementation of RSA on Quantum Computer
The Encrypted Message is:  [22, 3, 15, 16, 9, 3, 9, 27, 24, 0, 24, 22, 14, 27, 400, 14, 10, 400, 33, 2, 0, 400, 14, 27, 400, 1
1, 20, 0, 27, 24, 20, 3, 400, 18, 14, 3, 15, 20, 24, 9, 33]
```

Figure 5.3: Encrypting the message from the user

number in this case using Shor's Algorithm on IBM server was implemented in 89ms. Hence, it requires only polynomial time to execute the primality test.

5.2 Quantum Key Distribution

The quantum key distribution protocol is implemented to share the value of N which can be used to generate the private key at the receiver side. The sender generates a random set of bases and encodes the value of N using that and is then sent to receiver which also generates a random set of bases and measures the results. Figure 5.4 shows the measurement at the receiver side. If an untrusted person starts to measure the result the value changes and hence interference can be detected at the receiver side.

```
[0 0 0 0 0 0 1 0 0 0 1 1]
[0 1 1 0 1 1 1 1 1 1 1 0]
[0 1 0 0 0 0 0 1 0 1 1 0]
The measured result is:  [0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1]
```

Figure 5.4: Quantum key distribution protocol

The probability of not detecting an interference is given by the equation 5.1.

$$P(undetected) = 0.75^{x} \tag{5.1}$$

where x is number of bits. In this case, number of bits is 12. Hence the probability is

$$P(undetected) = 0.75^{12} = 0.031. \tag{5.2}$$

5.3 Decryption using Shor's Algorithm

The encrypted message is then sent to the receiver which first generates the private key. The value of N is first factorised using Shor's Algorithm which take this value as input and returns

a list with the factors. The factors are then used to generate the private key. Figure 5.5 shows the output of the private key generated.

```
configrc.store_credentials:WARNING:2021-06-03 19:51:57,867: Credentials already present. Set overwrite=True to overwrite.


 Decryption using Shor Algorithm
--------------------


ibmqfactory.load_account:WARNING:2021-06-03 19:51:58,063: Credentials are already in use. The existing account in the session w
ill be replaced.
/opt/conda/lib/python3.8/site-packages/qiskit/providers/ibmq/ibmqbackend.py:814: DeprecationWarning: Passing a Qobj to Backend.
run is deprecated and will be removed in a future release. Please pass in circuits or pulse schedules instead.
  return super().run(circuits, job_name=job_name, job_share_level=job_share_level,


The Private Key is:  (59, 35)
```

Figure 5.5: Private key generation

The decrypt function is called which takes the encrypted message and private key and implements inverse modular exponentiation to decipher the message and recover the original message. Figure 5.6 shows the output as the original text recovered after deciphering the encrypted message.

```
Enter the message to decrypt(Separate numbers with ','):11,20,0,27,24,20,3,400,22,3,15,6,9,3,9,27,24,0,24,22,14,27,400,14,10,40
0,33,2,0,400,0,16,6,14,33,22,24,28,3
The decrypted message is: QUANTUM IMPLEMENTATION OF RSA ALGORITHM
```

Figure 5.6: Decrypting the message from the user

The results of the proposed implementation of RSA algorithm on quantum computer has been shown in this chapter with describing the results of each step. Decrypting the algorithm using Shor's Algorithm decrypts the message and using QKD to transmit the value of N guarantees to detect the interference if occurred.

Chapter 6

Conclusion and Future Scope

CHAPTER 6

CONCLUSION AND FUTURE SCOPE

6.1 Conclusion

In today's world, where information play a particularly important role, the transmission and the storage of data must be maximally secure. This project is an effort towards studying the impact of quantum computing on present day cryptographic schemes and proposing a quantum implementation of one of the most widely used RSA algorithm along with a secure method to transmit the key using a secure open communication channel. The very first objective was to elucidate the implications of quantum computing in present cryptographic systems. The second objective was to implement a novel quantum cryptography scheme when the users are fully classical. The third and final objective was to implement a classical encryption standard as Quantum circuits.

Initially, a comparative study has been done to know more about present day cryptographic systems and impact of quantum algorithms to it. Quantum algorithms like Shor's algorithm can make present cryptographic system collapse since it is based on large prime number factorisation which is the important element of widely used algorithms like RSA. whereas Grover's algorithm which is used to track an element from unsorted database can find the key used to implement the cryptography is some iterations and hence can easily crack a password. The widely used RSA algorithm is implemented using quantum computation in order to make it less vulnerable to quantum attacks. The arithmetic operations are implemented using QFT which designs a reversible gate circuit to implement arithmetic operations and the decryption is done using Shor's algorithm by factorising the value. The transmission of the key is done using Quantum key distribution. This method is used to transmit message between two classical user over an open communication channel securely.

The algorithm was successfully able to carry out arithmetic operations using QFT. The check for primality was also implemented using Shor's algorithm on quantum computer and the run time for the primality test was around 89ms on quantum server which is fast as compared to its best classical counterpart which takes super polynomial time to execute this test. The public key is generated and the message is encrypted using the quantum algorithm. The key is then shared using quantum key distribution method which uses BB84 secured protocol. The QKD implemented in this guarantees a secured transmission where if anyone else interferes

the state of photons will change and hence the data gets corrupted. This interference have a chance of 97% to be detected with the size of key in this project and can be increased if used a longer key. The decryption is carried out to recover the original message. The key received is factorised using Shor's algorithm which can factorise large number much faster than its classical counterparts and hence generates the private key to decrypt the message. Hence QKD can help in transmitting any data securely and quantum algorithm can be applied to encrypt and decrypt the data.

6.2 Future Scope

Quantum computing can be a game-changer in such fields as cryptography and chemistry, material science, agriculture, and pharmaceuticals. The quantum key distribution technique discussed in this project can be further developed with user authentication and authorisation to transmit highly classified data and make it less vulnerable to quantum attacks. Symmetric cryptographic algorithms can be implemented on quantum computer using reversible circuits which can make the communication more reliable as the key will be hard to crack even for other quantum algorithms.

BIBLIOGRAPHY

[1] J. S. Chen, D. A. Desai, S. P. Heyns, and F. Pietra, "Literature review of numerical simulation and optimisation of the shot peening process," *Advances in Mechanical Engineering*, vol. 11, no. 3, p. 1 687 814 018 818 277, 2019.

[2] L. Gyongyosi, "Objective function estimation for solving optimization problems in gate-model quantum computers," *Scientific reports*, vol. 10, no. 1, pp. 1–21, 2020.

[3] C. H. Bennett and G. Brassard, "An update on quantum cryptography," in *Advances in Cryptology*, G. R. Blakley and D. Chaum, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 1985, pp. 475–480, ISBN: 978-3-540-39568-3.

[4] L. Gyongyosi, "Unsupervised quantum gate control for gate-model quantum computers," *Scientific reports*, vol. 10, no. 1, pp. 1–16, 2020.

[5] C. H. Bennett and G. Brassard, "Quantum cryptography: Public key distribution and coin tossing," *Theoretical Computer Science*, vol. 560, pp. 7–11, Dec. 2014, ISSN: 0304-3975. DOI: 10.1016/j.tcs.2014.05.025. [Online]. Available: http://dx.doi.org/10.1016/j.tcs.2014.05.025.

[6] T. Haener, M. Soeken, M. Roetteler, and K. M. Svore, "Quantum circuits for floating-point arithmetic," in *International Conference on Reversible Computation*, Springer, 2018, pp. 162–174.

[7] T. Hong, Y. Li, S.-B. Park, D. Mui, D. Lin, Z. A. Kaleq, N. Hakim, H. Naeimi, D. S. Gardner, and S. Mitra, "Qed: Quick error detection tests for effective post-silicon validation," in *2010 IEEE International Test Conference*, 2010, pp. 1–10. DOI: 10.1109/TEST.2010.5699215.

[8] M. Ostaszewski, E. Grant, and M. Benedetti, "Structure optimization for parameterized quantum circuits," *Quantum*, vol. 5, p. 391, 2021.

[9] Z. Zilic and K. Radecka, "The role of super-fast transforms in speeding up quantum computations," in *Proceedings 32nd IEEE International Symposium on Multiple-Valued Logic*, IEEE, 2002, pp. 129–135.

[10] X. Zhou, D. W. Leung, and I. L. Chuang, "Methodology for quantum logic gate construction," *Physical Review A*, vol. 62, no. 5, p. 052 316, 2000.

[11] P. Shor, "Algorithms for quantum computation: Discrete logarithms and factoring," in *Proceedings 35th Annual Symposium on Foundations of Computer Science*, 1994, pp. 124–134. DOI: 10.1109/SFCS.1994.365700.

[12] G. Florio and D. Picca, "Quantum implementation of elementary arithmetic operations," *arXiv preprint quant-ph/0403048*, 2004.

[13] M. Mariantoni, H. Wang, T. Yamamoto, M. Neeley, R. C. Bialczak, Y. Chen, M. Lenander, E. Lucero, A. D. O'Connell, D. Sank, *et al.*, "Implementing the quantum von neumann architecture with superconducting circuits," *Science*, vol. 334, no. 6052, pp. 61–65, 2011.

[14] T. Humble, "Consumer applications of quantum computing: A promising approach for secure computation, trusted data storage, and efficient applications," *IEEE Consumer Electronics Magazine*, vol. 7, no. 6, pp. 8–14, 2018.

[15] A. U. Khalid, Z. Zilic, and K. Radecka, "Fpga emulation of quantum circuits," in *IEEE International Conference on Computer Design: VLSI in Computers and Processors, 2004. ICCD 2004. Proceedings.*, IEEE, 2004, pp. 310–315.

[16] L. Gyongyosi and S. Imre, "Quantum circuit design for objective function maximization in gate-model quantum computers," *Quantum Information Processing*, vol. 18, no. 7, pp. 1–33, 2019.

[17] S. Sun, Z. Cao, H. Zhu, and J. Zhao, "A survey of optimization methods from a machine learning perspective," *IEEE transactions on cybernetics*, vol. 50, no. 8, pp. 3668–3681, 2019.

[18] P. W. Purdom and C. A. Brown, "Evaluating search methods analytically," in *In Proceedings of the National Conference on Artificial Intelligence*, 1982, pp. 124–127.

[19] M. Alam, A. Ash-Saki, and S. Ghosh, "Analysis of quantum approximate optimization algorithm under realistic noise in superconducting qubits," *arXiv preprint arXiv:1907.09631*, 2019.

[20] E. Farhi, J. Goldstone, S. Gutmann, and H. Neven, "Quantum algorithms for fixed qubit architectures," *arXiv preprint arXiv:1703.06199*, 2017.

[21] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, 1978.

[22] *IBM Quantum*, https://quantum-computing.ibm.com/, 2021.

[23] *Qiskit: An open-source framework for quantum computing*, 2019. DOI: 10.5281/zenodo.2562110.