

STRENGTHENING CRYPTOGRAPHY USING QUANTUM PROPERTIES

A PROJECT REPORT

Submitted by

NAVEEN S R 211418104172

PRADEISH C 211418104196

MOHAN RAJ RISHI S 211418104158

in partial fulfilment for the award of the degree

Of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING



PANIMALAR ENGINEERING COLLEGE

(An Autonomous Institution, Affiliated to Anna University, Chennai)

MAY 2022

PANIMALAR ENGINEERING COLLEGE
(An Autonomous Institution, Affiliated to Anna University, Chennai)

BONAFIDE CERTIFICATE

Certified that this project report “**STRENGTHENING CRYPTOGRAPHY USING QUANTUM PROPERTIES**” is the bonafide work of “**NAVEEN S R, PRADEISH C, MOHAN RAJ RISHI S**” who carried out this project under my supervision.

SIGNATURE

Dr. S. MRUGAVALLI, M.E., Ph.D.,
HEAD OF THE DEPARTMENT
DEPARTMENT OF CSE,
PANIMALAR ENGINEERING
COLLEGE,
NAZARATHPETTAI,
POONAMALLEE,
CHENNAI – 600 123

SIGNATURE

Mr. M. KRISHNAMOORTHY, M.E.,
MBA.,
SUPERVISOR
DEPARTMENT OF CSE,
PANIMALAR ENGINEERING
COLLEGE,
NAZARATHPETTAI,
POONAMALLEE,
CHENNAI – 600 123

Certified that the above mentioned students were examined in End Semester project viva-voice held on _____.

INTERNAL EXAMINER

EXTERNAL EXAMINER

DECLARATION

We **NAVEEN S R (211418104172)**, **PRADEISH C (211418104196)**, **MOHAN RAJ RISHI (211418104158)** hereby declare that this project report titled “**STRENGTHENING CRYPTOGRAPHY USING QUANTUM PROPERTIES** ” under the guidance of **Mr. M. KRISHNAMOORTHY** is the original work done by us and we have not plagiarized or submitted to any other degree in any university by us.

ACKNOWLEDGEMENT

We express our deep gratitude to our respected Secretary and Correspondent **Dr. P. CHINNADURAI, M.A., Ph.D.** for his kind words and enthusiastic motivation, which inspired us a lot in completing this project.

We would like to extend our heartfelt and sincere thanks to our directors **Tmt. C.VIJAYARAJESWARI, Thiru. C. SAKTHIKUMAR, M.E.,** and **Tmt. SARANYASREE SAKTHIKUMAR B.E., M.B.A.,** for providing us with the necessary facilities for the completion of this project.

We also express our gratitude to our Principal **Dr. K. MANI, M.E., Ph.D.** for his prompt concern and encouragement provided to us throughout the course.

We thank the HOD of the CSE (COMPUTER SCIENCE ENGINEERING) Department, **Dr. S. MURUGAVALLI, M.E., Ph.D.** for the support extended throughout the project.

We would like to thank our **Project Guide Mr. Krishnamoorthy, M.E.,** and all the faculty members of the Department of CSE for their advice and suggestions for the successful completion of the project.

NAVEEN S R (211418104178)

PRADEISH C (211418104196)

MOHAN RA RISHI S (211418104158)

ABSTRACT

For most of the cryptography algorithms, randomness plays a vital role. But generation of pure random number is impossible in classical computer. Even for generating decent random numbers it needed a random seed. But since the seed is a number in source code or algorithm to generate seed which is also in source, it can be revealed to hacker if the system is compromised. Which made possible to predict all the random numbers the computer will generate by the hacker. So, to solve this issue we introduce quantum computer to create a true randomness using its superposition capability.

But quantum machines are not accessible to all the computers, especially to clients. So, we found that it possible to generate a pure random number on client side with the help of quantum computer on server side using Diffie Hellman Key Exchange. Even through un-encrypted channel with any compromise.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	iii
	LIST OF FIGURES	1
1.	INTRODUCTION	2
	1.1 Overview	3
	1.2 Problem Definition	3
2.	LITERATURE SURVEY	5
	2.1 A Study of the basics of Quantum Computing	6
	2.2 Quantum Cloning Machines and the Application	15
3.	SYSTEM ANALYSIS	26
	3.1 Existing System	27
	3.2 Proposed system	27
	3.3 Requirement Analysis	28
	3.3.1 Hardware Requirements	28
	3.3.2 Functional Requirements	28
	3.4 Software Environment	28

	3.5 Software Description	29
4.	SYSTEM DESIGN	37
	4.1 UML Diagrams	38
	4.1.1 Use Case diagram	39
	4.1.2 Sequence diagram	41
	4.1.3 Component diagram	43
5.	SYSTEM ARCHITECTURE	45
	5.1 Architecture Overview	46
6.	SYSTEM IMPLEMENTATION	48
	6.1 Tkinter	49
	6.1.1 Installation	49
	6.1.2 Implementation in Project	50
	6.2 Qiskit	52
	6.2.1 Installation	52
	6.2.2 Implementation in Project	54
7.	SYSTEM TESTING	57
	7.1 Unit Testing	58
	7.1.1 White box Testing	58
	7.1.2 Black box Testing	60
	7.2 Integration Testing	61
8.	CONCLUSION	62
	8.1 Performance analysis report	63
9	APPENDICES	65

9.1 Sample Screens	66
9.2 Coding	68
REFERENCES	72

LIST OF FIGURES

FIGURE NUMBER	FIGURE NAME
3.1	Icon of Python
3.2	Icon of Visual Studio
3.3	IBMQ cloud quantum machine
3.4	Qiskit quantum circuit optimizer
3.5	Structure of Python Tkinter GUI
3.6	Structure of Python Socket
3.7	Working of the TCP protocol
5.1	System Architecture
6.1	Jupyter Notebook
6.2	Qiskit import
6.3	IBMQ Load Account
6.4	Output of IBM load connection
9.1	Quantum circuit and result
9.2	Encrypted conversation between server and client

CHAPTER 1

INTRODUCTION

1. INTRODUCTION

1.1 OVERVIEW

The Generation of pure random numbers from the classical computers. For any encryption we need randomness, we have to feed the seed to generate pseudorandom numbers. But anyone can perfectly predict all the random numbers generated by the computer with the seed of what we feed. So, in our project we introduce quantum computing, to create a true random bit so numbers by its superposition concept and also due to its no-cloning property, if one attempts to read the random numbers, the quantum state will be changed.

But quantum machines are not accessible to all the computers, especially to clients. So, we found that it possible to generate a pure random number on client side with the help of quantum computer on server side using Diffie Hellman Key Exchange. Even through un-encrypted channel with any compromise.

- Type of Technology used: Quantum computing
- Fundamental unit of information: Qubit
- Fundamental Theorem: No-cloning of Quantum mechanics

1.2 PROBLEM DEFINITION

The Generation of pure random numbers from classical computing is impossible, it requires an external source for getting the pure randomness. In most of the solutions, systems use the information like POSIX time, CPU status, internet traffic, etc,. Since most of the servers used to log this information, it is possible to get the right feed if the server app source code is revealed or the server app is compromised.

According to the principle of pseudo random number generators, a stream of random numbers which the server generated and used as a requirement for cryptography

can be revealed if the server is compromised and which can lead to retrieval of sensitive data and vital information which should not be found outside the organization.

Combination of retrieving server app source code and log and monitoring all the server's communication can lead to a massive cyber-attack. Although these attacks are rare and hard to happen, there is no proof it can't happen in future.

So, to prevent these kinds of attacks this project uses the quantum properties.

CHAPTER 2

LITERATURE SURVEY

2. LITERATURE SURVEY

2.1 A Study of the basics of Quantum Computing

Title: A Study of the basics of Quantum Computing

Year: 2005

Author: Prashant Singh, Department d'Informatique et de recherche operationnelle,
Universite de Montreal, Montreal. Canada.

Quantum theory is one of the most successful theories that have influenced the course of scientific progress during the twentieth century. It has presented a new line of scientific thought, predicted entirely inconceivable situations and influenced several domains of modern technologies. There are many different ways for expressing laws of science in general and laws of physics in particular. Similar to physical laws of nature, information can also be expressed in different ways. The fact that information can be expressed in different ways without losing its essential nature, leads for the possibility of the automatic manipulation of information.

All ways of expressing information use physical system, spoken words are conveyed by air pressure fluctuations: “No information without physical representation”. The fact that information is insensitive to exactly how it is expressed and can be freely translated from one form to another, makes it an obvious candidate for fundamentally important role in physics, like interaction, energy, momentum and other such abstractors. This is a project report on the general attributes of Quantum Computing and Information Processing from a layman's point of view.

A New Outlook to the Principle of Linear Superposition

Modification of Wave function as a requirement of Quantum Teleportation:

Here we introduce a possibility of modification of Expansion Theorem as applicable to an EPR situation for correlated terms. In this section we attempt to introduce a correlation term in the definition of wave function (ψ) as given by Linear

Superposition Principle. A general expression of wave function (ψ) can be expressed in word equation as:

Wave Function (ψ) = Term of Correlated states (δ) + Sum of Un-correlated terms.

This word equation can be expressed numerically as:

$\psi = \delta + \sum a_n \psi_n$ where $n \in$ allowed un-correlated states & $\psi(n)$ are eigen states.

Hence in general with the introduction for our formalism of wave function, we can now divide wave function into 3 categories which are:

1. Quantum Systems with Corr. = 0 $\psi = \sum a_n \psi_n$ where $n \in (1,2,3,4..)$ & $\psi(n)$ are eigen states.
2. Quantum Systems with $0 < \text{Corr.} < 1$ $\psi = \delta + \sum a_n \psi_n$ where $n \in$ allowed uncorrelated states & $\psi(n)$ are eigen states.
3. Quantum Systems with Corr. = 1 $\psi = \delta$ and $\sum a_n \psi_n = 0$ $\psi(n)$ are eigen states.

Thus the above treatment of Expansion Theorem in Quantum Systems suggests that the definition of wave function should be modified to take into account the representation of EPR states and further investigation should be done to determine the value of δ in the wave function definition. To further support and validate the above formalism I have applied the Schrodinger Wave Eq. both time dependent and time independent to check if the form of Expansion Theorem changes or not. In the standard formulation of Quantum Mechanics we have Schrodinger Wave Eq. given by :

$$\nabla^2 \psi + 2m/\hbar^2 (E - V) \psi = 0 \text{ -----(1) Schrodinger time independent Eq. in this eq.}$$

ψ : wave function, m is effective mass of the system, E is total energy, V is potential energy of the system and $\hbar^2 = h^2 / 4\pi^2$ (h : Plank Constant) and Schrodinger Time dependent wave eq. is given by:

$$i\hbar/2\pi (\partial/\partial t) \psi = H \psi \text{ -----(2)}$$

where ψ : wave function of the system, H is Hamiltonian of the system.^{8 24} Now, by putting all the three definitions of ψ into these fundamental wave equations of Quantum Mechanics that the modified definition of ψ is correct as we get the expected results.

Suitability of Quantum Bits for Quantum Computation:

Since there are three types of wave function systems as we have discussed in previous sections, thus how do they relate to qubits is given in following lines:

1. For purely unentangled states in a Quantum system i.e. $\text{Corr.} = 0$, in such type of systems wave functions is given by Linear Superposition Principle: $\psi = \sum a_n \psi_n$ & $\sum |a_n|^2 = 1$ where $n \in (1,2,3,4,\dots)$ and $\psi(n)$ are eigen states.

In these type of system of qubits quantum algorithms can be made “efficiently” as in this case the underlying parallelism of computation and vast storage of information is possible according to the conception of Bloch Sphere or otherwise since every state ψ_n is independent of each other and hence can be used for computation and storing of information.

2. For mixed entangled states in a Quantum System i.e. $0 < \text{Corr.} < 1$, in such type of systems wave function is given by:

$\psi = \delta + \sum a_n \psi_n$ & $\sum |a_n|^2 \neq 1$ where $n \in$ allowed un-correlated states & $\psi(n)$ are Eigen states. In such a case since entangled and unentangled states cannot be separated as that would amount to an interaction with the system leading to information loss and wave function collapse. Hence such type of a state is not fit for computational purposes as it may lead to spurious results.

3. For purely entangled states in a Quantum system i.e. $\text{Corr.} = 1$, in such type of systems wave function is given by:

$\psi = \delta$ and $\sum a_n \psi_n = 0$ $\psi(n)$ are Eigen states.

These states are suitable for teleportation of information using EPR states and not for information storage or computational purposes. Thus case 3 is well suited only for information communication keeping the validity of Quantum No Cloning Theorem.

An alternative interpretation of Quantum No Cloning Theorem:

The definition of Quantum No Cloning Theorem states that:

**“It is not possible to copy quantum bits of an unknown quantum state.” Or
“Replication of information of arbitrary qubits is not possible.”**

This fundamental limit is brought by Heisenberg’s Uncertainty Principle given by: Δx .

$$\Delta p \geq h/2\pi$$

Δx is the uncertainty in the measurement of position.

Δp is the uncertainty in the measurement of momentum.

Thus Uncertainty Principle brings about the problem of Measurement in Quantum Mechanics which says that act of observation leads to wave function collapse and information carried by a Quantum System is lost i.e.:

$$\psi = \alpha|0\rangle + \beta|1\rangle \xrightarrow{\text{observation}} \text{either } |0\rangle \text{ or } |1\rangle$$

Now, another explanation of information loss associated with measurement of a qubit can be given by energy exchange leading to irreversibility brought into the system according to the 2nd Law of Thermodynamics. The very act of measurement or observation has brought an element of irreversibility into the system which can also be appreciated by the fact that there is a concept of canonically conjugate variables in Quantum Mechanics. That is if we have two observables say A and B in a quantum system which are canonically conjugate, they follow the inequality $AB \neq BA$ and it is represented in Quantum Mechanics as commutator brackets $[A, B] = h/2\pi \Rightarrow AB - BA = h/2\pi$

Now the very fact that $AB \neq BA$ suggests that there is some irreversibility brought in the process of measurement of a quantum system and hence the uncertainty creeps in which is given by Heisenberg Uncertainty Principle. This leads to some important questions like:

1. Does the notion of quantum of action h ($h, 2h, 3h \dots$) lead to irreversibility in the measurement process since $\frac{1}{2}h$ or any fraction value of h cannot take place?
2. Irreversibility is brought in due to entropy change in the system, since entropy is a statistical phenomenon, does this anyway relate to statistical behaviour of quantum probabilities?
3. Is irreversibility in a Quantum measurement more fundamental than uncertainty in the measurement process and leads to uncertainty in the system, thus explaining the need for Uncertainty Principle?

Thus in Classical Mechanics entropy (S) of a system is associated with the loss of information in the system due to the energy interaction and 2nd Law of Thermodynamics. But in Quantum Mechanics we have the concept of choice of information due to wave function collapse, so we can define a new parameter analogous to entropy in case of Quantum systems related to the choice of information. Thus we introduce here a possibility of the need of a new parameter for depicting the concept of choice of information analogous to entropy. The new parameter can be an area of further research. The concept of irreversibility in the system due to measurement process leading to non-replication of information on a qubit is an alternative explanation of the Quantum No Cloning Theorem.

Explanation of Measurement Problem by Symmetry breaking process:

Symmetry is the most fundamental property of physical systems. It is a very general term with regard to physical systems. There are various symmetries in Nature like local space time symmetries and global space time symmetries. Global space time symmetries apply to all the laws of Physics. Here in this section we will attempt to give

a direction on which a further thought can be developed for the resolution of the famous Measurement Problem in Quantum Mechanics. It is the Measurement Problem which lies at the heart of various problems and observation anomalies in Quantum Mechanics. Here I am trying to give an explanation of Measurement Problem using Symmetry breaking process. The basic equation of Quantum Mechanics i.e. $AB \neq BA$ valid for conjugate variables shows some kind of irreversibility in the system. Now by the Leibniz's Principle of Sufficient Reason (PSR): If there is no sufficient reason for one thing to happen instead of another, the principle says that nothing happens or in a situation with certain symmetry evolves in such a way that, in the absence of an asymmetric cause, the initial symmetry is preserved. In other words, a breaking of initial symmetry cannot happen without a reason or an asymmetry cannot originate spontaneously. Asymmetry is what creates a phenomenon. Thus symmetry breaking follows Principle of Causality that is "For every effect there is a cause." By the Measurement process we purposefully throw the Quantum System as a "choice of information" by the observer unlike the collapsing of the wave function into an uncertain Eigen state. Thus in a Measurement process the system does not lose information like classical systems but is thrown in a particular eigen state based on the information choice made by the observer. Hence we need to introduce a parameter in Quantum Mechanics which takes care of this "choice of information" rather the "loss of information" whose parameter is entropy in classical mechanics. Curie's Principle also is an interesting way to represent symmetry requirements in a system. The symmetry elements of the causes must be found in their effects, but the converse is not true, that is the effects can be more symmetric than their causes. Conditions valid for Curie's Principle are:

1. The causal connection must be valid.
2. Cause and Effect and their respective symmetries must be well defined.

There must be a sufficient reason for the Measurement Problem to occur and by the Leibniz's Principle of Sufficient Reason (PSR) we can conclude that there must be some symmetry breaking phenomenon occurring as a reason for the

asymmetry in the system and thus leading to Measurement Problem. Thus it is a matter of further research to find the precise way how symmetry breaks in the system? What type of symmetry exists? Is it a local or a global symmetry which breaks in the Reduction(R) Process of the wave function as it is called in the Quantum Mechanics parlance? Thus introducing symmetry in the Reduction Process in Quantum Mechanics is a novel idea and can put some more light into the reason and the precise process by which wave function collapse occurs. Thus this may fundamentally give the reason of Heisenberg Uncertainty Principle. It is yet to be seen that it is the Uncertainty Principle which is more fundamental or the symmetry breaking which is occurring in the system.

EPR Correlation: Interacting Hamiltonian Vs Non-Linear Wave function:

Basically there are two ways to introduce the effect of correlation in an EPR situation in the definition of wave function of the Quantum System those are:

1. In the form of Schrodinger Time Dependent Wave Eq.

$$i\hbar/2\pi (\partial/\partial t) \psi = H \psi$$

We can introduce the correlation effect in the modification of the Hamiltonian (H) of the system by introducing a potential term in the expression of H of the system and keeping wave function (ψ) as it is defined by the Expansion Theorem.

2. In the second case we can keep Hamiltonian (H) as it is and change the definition of wave function (ψ) by introducing a δ term to account for correlation effect of EPR states.

Hence we have attempted the second case in our formalism of accounting for correlation in the definition of wave function (ψ) of the system. Though both the approaches are new to the scene and have not been attempted for this type of a situation. Hence the application of first case is an area of further research in this formalism. Eigen states are essentially entangled or correlated in an EPR situation and since EPR implies non-local communication and apparent breakdown of Heisenberg Uncertainty Principle hence it may lead to the modification of Linear Superposition Principle's linearity gets

violated and different eigen states start interfering with each other and lead to the introduction of δ in the definition of wave function (ψ).

EPR states \longrightarrow modification of Linear Superposition Principle \longrightarrow new definition of wave function (ψ) or introduction of a new parameter altogether.

We find from the above two possible scenarios that EPR correlation can be incorporated in the Schrodinger time dependent wave Eq. by modifying the Hamiltonian and introducing the δ term due to correlation i.e. The wave eq. can be written as:

Hamiltonian = Kinetic energy term + Potential energy + energy due to EPR correlation

Thus this can be one of the attempted explanations for introducing correlation in the scene. In the second case we can put the Hamiltonian as it is without “energy due to EPR correlation” term and can rather modify the wave function (ψ) in a non-linear way by introducing the δ term and accounting for EPR correlation as it has already been done in previous sections.

Possibility of third paradigm in Quantum Mechanics:

It is a well known fact that though Quantum Mechanics gives correct mathematical results of quantum phenomena but the interpretation of Quantum Mechanics is very peculiar and is still not complete till date because of paradoxes like Schrodinger Cat paradox and Einstein Podolsky Rosen (EPR) Paradox, Wave Function Collapse and Measurement Problem. In this we attempt to give a new direction of thinking for a new formulation of the fundamental principle of Uncertainty in Quantum Mechanics since all peculiarities of Quantum Mechanics are derived from the Uncertainty Principle. We describe the two mechanics we know of and predict the possibility of the third one:

1. In Classical Mechanics the fundamental concept is the concept of Determinism which says that given a state of the system at one point of time the state of the system at some later time can be predicted with 100% accuracy. This can be

reformulated in a statement including dynamical variables position (x) and momentum (p) as

“Both position (x) & momentum (p) can be measured to any desired accuracy.”

2. In Quantum Mechanics we have the fundamental Uncertainty Principle given by:

$$\Delta x \cdot \Delta p \geq h/2\pi$$

It is interpreted as “Either position (x) or momentum (p) can be measured with desired accuracy but not both at the same time.”

3. Here we give a new line of thinking to the above formalisms by saying that:

“Neither position (x) nor momentum (p) to any degree of accuracy can be measured.”

This is a very interesting statement since in case 1 above we have determinism in the definition of the system. In case 2 we have probability but still it is limited probability since one of the parameters can still be measured to desired degree of accuracy. But in case 3 we have brought in a complete indeterminism in which both the variables are completely indeterministic. This formalism requires a new definition of Schrodinger Wave Eq., Uncertainty Principle etc. It may be possible that Quantum Mechanics is a special case of this third mechanics as we move from complete indeterminism to limited indeterminism (Quantum Mechanics) to complete determinism (Classical Mechanics). This concept of complete indeterminism may be related in some way to Consciousness of the observer which is hot area of research in Quantum Mechanics these days. This last proposal was given by Stephen Hawking as a new direction of thinking on the modification of Quantum Mechanics right at the fundamental level.

2.2 Quantum Cloning Machines and the Application

Title: Quantum Cloning Machines and the Application

Year: 2014

Authors: Heng Fan, Yi-Nan Wang, Li Jing, Jie-Dong Yue, Han-Duo Shi, Yong-Liang Zhang, and Liang-Zhu Mu

Beijing National Laboratory for Condensed Matter Physics, Institute of Physics, Chinese Academy of Sciences, Beijing 100190, China

Collaborative Innovation Center of Quantum Matter, Beijing 100190,

China 3 School of Physics, Peking University, Beijing 100871, China

Quantum information, qubit and quantum entanglement:

We have a quantum system constituted by two states $|0\rangle$ and $|1\rangle$. They are orthogonal,

$$\langle 0|1\rangle = 0. \quad (1)$$

Those two states can be energy levels \uparrow of an atom, photon polarizations, electron spins, Bose-Einstein condensate with two intrinsic freedoms or any physical material with quantum properties. In this review, we also use some other standard notations $|0\rangle = |\uparrow\rangle$, $|1\rangle = |\downarrow\rangle$ and exchange them without mentioning. Simply, those two states can be represented as two vectors in linear algebra,

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}. \quad (2)$$

Corresponding to bit in classical information science, a qubit in quantum information science is a superposition of two orthogonal states,

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle, \quad (3)$$

where a normalization equation should be satisfied,

$$|\alpha|^2 + |\beta|^2 = 1. \quad (4)$$

Here both α and β are complex parameters which include amplitude and phase information, $\alpha = |\alpha|e^{i\varphi\alpha}$ and $\beta = |\beta|e^{i\varphi\beta}$. So, a qubit $|\psi\rangle$ is defined on a two-dimensional Hilbert space C^2 . In quantum mechanics, a whole phase cannot be detected and thus can

be omitted, only the relative phase of α and β is important which is $\varphi = \varphi_\alpha - \varphi_\beta$. Now we can find that a qubit can be represented in another form,

$$|\psi\rangle = \cos \frac{\theta}{2} |0\rangle + \sin \frac{\theta}{2} e^{i\varphi} |1\rangle, \quad (5)$$

where $\theta \in [0, \pi]$, $\varphi \in [0, 2\pi]$. It corresponds to a point in the Bloch sphere, see FIG. 1.

The two qubits in separable form can be written as,

$$\begin{aligned} |\psi\rangle|\varphi\rangle &= (\alpha|0\rangle + \beta|1\rangle)(\gamma|0\rangle + \delta|1\rangle) \\ &= \alpha\gamma|00\rangle + \alpha\delta|01\rangle + \beta\gamma|10\rangle + \alpha\delta|11\rangle. \end{aligned} \quad (6)$$

If those two qubits are identical, one can find,

$$\begin{aligned} |\psi\rangle^{\otimes 2} &\equiv (\alpha|0\rangle + \beta|1\rangle)(\alpha|0\rangle + \beta|1\rangle) \\ &= \alpha^2 |00\rangle + \sqrt{2}\alpha\beta \frac{1}{\sqrt{2}} (|01\rangle + |10\rangle) + \beta^2 |11\rangle. \end{aligned} \quad (7)$$

For convenience, we write the second term as a normalized symmetric state $\frac{1}{\sqrt{2}} (|01\rangle + |10\rangle)$ which will be used later.

For two-qubit state, besides those separable state, we also have the entangled state, for example,

$$|\Phi^+\rangle \equiv \frac{1}{\sqrt{2}} (|00\rangle + |11\rangle). \quad (8)$$

This is state cannot be written as a product form like $|\psi\rangle|\varphi\rangle$, so it is “entangled”. It is actually a maximally entangled state. In quantum information science, quantum entanglement is the valuable resource which can be widely used in various tasks and protocols. Complementary to entangled state $|\Phi^+\rangle$, we have other three orthogonal and maximally entangled state which constitute a complete basis for $C^2 \otimes C^2$. Those four states are Bell states, here we list them all as follows,

$$|\Phi^+\rangle \equiv \frac{1}{\sqrt{2}} (|00\rangle + |11\rangle), \quad (9)$$

$$|\Phi^-\rangle \equiv \frac{1}{\sqrt{2}} (|00\rangle - |11\rangle), \quad (10)$$

$$|\Psi^+\rangle \equiv \frac{1}{\sqrt{2}} (|01\rangle + |10\rangle), \quad (11)$$

$$|\Psi^-\rangle \equiv \frac{1}{\sqrt{2}} (|01\rangle - |10\rangle). \quad (12)$$

Those four Bell states can be transformed to each other by local unitary transformations. Consider three Pauli matrices defined as,

$$\sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \sigma_y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, \quad (13)$$

Since $\sigma_y = i\sigma_x\sigma_z$, if the imaginary unit, “i”, is the whole phase, we sometimes use $\sigma_x\sigma_z$ instead of σ_y . Bear in mind that we have $|0i\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$, and $|h0\rangle = \begin{pmatrix} 1 & 0 \end{pmatrix}$, so in linear algebra, we have the representation,

$$|0ih0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \begin{pmatrix} 1 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}, \quad (14)$$

Now three Pauli matrices have an operator representation,

$$\sigma_x = |0\rangle\langle 1| + |1\rangle\langle 0|, \quad (15)$$

$$\sigma_y = -i|0\rangle\langle 1| + i|1\rangle\langle 0|, \quad (16)$$

$$\sigma_z = |0\rangle\langle 0| - |1\rangle\langle 1|. \quad (17)$$

In this review, we will not distinguish the matrix representation and the operator representation. Acting Pauli matrices σ_x and σ_z on a qubit, we find,

$$\sigma_x|0\rangle = |1\rangle, \sigma_x|1\rangle = |0\rangle, \quad (18)$$

$$\sigma_z|0\rangle = |0\rangle, \sigma_z|1\rangle = -|1\rangle, \quad (19)$$

which are the bit flip action and phase flip action, respectively, while σ_y will cause both bit flip and phase flip for a qubit. In this review, for convenience, we sometimes use notations $X \equiv \sigma_x$, $Z \equiv \sigma_z$ to represent the corresponding Pauli matrices. Also, those Pauli matrices can also be defined in higher dimensional system, while the same notations might be used if no confusion is caused.

For four Bell states, their relationship by local transformations can be as follows,

$$|\Phi^-\rangle = (I \otimes \sigma_z)|\Phi^+\rangle, \quad (20)$$

$$|\Psi^+\rangle = (I \otimes \sigma_x)|\Phi^+\rangle, \quad (21)$$

$$|\Psi^-\rangle = (I \otimes \sigma_x\sigma_z)|\Phi^+\rangle, \quad (22)$$

where I is the identity in C^2 , the Pauli matrices are acting on the second qubit.

Here we have already used the tensor product. Consider two operators, $O1 = \begin{pmatrix} A1 & B1 \\ C1 & D1 \end{pmatrix}$,

$O2 = \begin{pmatrix} A2 & B2 \\ C2 & D2 \end{pmatrix}$, the tensor product $O1 \otimes O2$ is defined and calculated as follows,

$$O1 \otimes O2 = \begin{pmatrix} A102 & B102 \\ C102 & D102 \end{pmatrix} = \begin{pmatrix} A1A2 & A1B2 & B1A2 & B1B2 \\ A1C2 & A1D2 & B1C2 & B1D2 \\ C1A2 & C1B2 & D1A2 & D1B2 \\ C1C2 & C1D2 & D1C2 & D1D2 \end{pmatrix}. \quad (23)$$

When we apply a tensor product of, $O1 \otimes O2$, on a two-qubit quantum state, operator $O1$ is acting on the first qubit, operator $O2$ is acting on the second qubit. We have already extended one qubit to two-qubit state. Similarly, multipartite qubit state can be obtained. Also we may try to extend qubit from two-dimension to higher-dimensional system, generally named as “qutrit” for dimension three and “qudit” for dimension d in more general case, the Hilbert space is extended from C^2 to C^d . For example, we sometimes have “qutrit” when we consider a quantum state in three dimensional system. For more general case, a qudit is also a superposed state,

$$|\psi\rangle = \sum_{j=0}^{d-1} x_j |j\rangle, \quad (24)$$

where x_j , $j = 0, 1, \dots, d-1$, are normalized complex parameters. Quantum entanglement can also be in higher dimensional, multipartite systems.

A qubit $|\psi\rangle$ can be represented by its density matrix,

$$\begin{aligned} |\psi\rangle\langle\psi| &= (\alpha|0\rangle + \beta|1\rangle)(\alpha^* \langle 0| + \beta^* \langle 1|) \\ &= \begin{pmatrix} |\alpha|^2 & \alpha\beta^* \\ \alpha^* \beta & |\beta|^2 \end{pmatrix}. \end{aligned} \quad (25)$$

However, a general qubit may be not only the superposed state, which is actually the pure state, but also a mixed state which is in a probabilistic form. It can only be represented by a density operator ρ ,

$$\rho = \sum_j p_j |\psi_j\rangle\langle\psi_j|, \quad (26)$$

where p_j is the probabilistic distribution with $\sum p_j = 1$.

A density matrix is positive semi-definite, and its trace equals to 1,

$$\rho \geq 0, \text{Tr} \rho = 1. \quad (27)$$

The density matrices of a pure state and a mixed state can be easily distinguished by the following conditions,

$$\text{Tr} \rho^2 = 1, \text{ pure state}; \quad (28)$$

$$\text{Tr} \rho^2 < 1, \text{ mixed state.} \quad (29)$$

For multipartite state, one part of the state is the reduced density matrix obtained by tracing out other parts. For example, for two-qubit maximally entangled state $|\Phi^+_{AB}\rangle$ constituted by A and B parts, each qubit is a mixed state,

$$\rho_A = \text{Tr}_B |\Phi^+_{AB}\rangle \langle \Phi^+_{AB}| = \frac{1}{2} I. \quad (30)$$

This case is actually a completely mixed state, here I is the identity operator. The identity operator can be written as any pure state and its orthogonal state with equal probability,

$$\rho_A = \frac{1}{2} I = \frac{1}{2} |\psi\rangle \langle \psi| + \frac{1}{2} |\psi^\perp\rangle \langle \psi^\perp|, \quad (31)$$

where if $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, its orthogonal state can take the form,

$$|\psi^\perp\rangle = \beta^*|0\rangle - \alpha^*|1\rangle, \quad (32)$$

where $*$ means the complex conjugation.

Quantum gates:

In QIP, all operations should satisfy the laws of quantum mechanics such as the generally used unitary transformation and quantum measurement. Similar as in classical computation, all quantum computation can be effectively implemented by several fundamental gates. The single qubit rotation gate and controlled-NOT (CNOT) gate constitute a complete set of fundamental gates for universal quantum computation (Barenco et al., 1995). The single qubit rotation gate is just a unitary transformation on a qubit, $R(\vartheta, \varphi)$, defined as

$$\begin{aligned} R(\vartheta, \varphi)|0\rangle &= \cos \vartheta |0\rangle + e^{i\varphi} \sin \vartheta |1\rangle, \\ R(\vartheta, \varphi)|1\rangle &= -e^{-i\varphi} \sin \vartheta |0\rangle + \cos \vartheta |1\rangle, \end{aligned} \quad (33)$$

where the phase parameter φ should also be controllable. The CNOT gate is defined as a unitary transformation on two-qubit system, one qubit is the controlled qubit and another qubit is the target qubit. For a CNOT gate, when the controlled qubit is $|0\rangle$, the target qubit does not change; when the controlled qubit is $|1\rangle$, the target qubit should be flipped. Explicitly it is defined as,

$$\text{CNOT} : |0\rangle|0\rangle \rightarrow |0\rangle|0\rangle;$$

$$\begin{aligned}
\text{CNOT} : |0\rangle|1\rangle &\rightarrow |0\rangle|1\rangle; \\
\text{CNOT} : |1\rangle|0\rangle &\rightarrow |1\rangle|1\rangle; \\
\text{CNOT} : |1\rangle|1\rangle &\rightarrow |1\rangle|0\rangle,
\end{aligned} \tag{34}$$

where the first qubit is the controlled qubit and the second qubit is the target qubit. By matrix representation, CNOT gate takes the form,

$$\text{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}. \tag{35}$$

Depending on physical systems, we can use different universal sets of quantum gates to realize the universal quantum computation.

NO-CLONING THEOREM

A simple proof of no-cloning theorem

For classical information, the possibility of cloning it is an essential feature. In classical systems, cloning, in other words, copying seems no problem. Information stored in computers can be easily made several copies as backup; the accurate semiconservative replication of DNA steadily passes gene information between generations. But for quantum systems, this is not the case. As proved by Wootters and Zurek (Wootters and Zurek, 1982), deterministic cloning of pure states is not possible. After this seminal work, much interest has been shown in extending and generalizing the original no-cloning theorem (Barnum et al., 1996; Luo, 2010b; Luo et al., 2009; Luo and Sun, 2010; Piani et al., 2008), which gives us new insight to boundaries of the classical and quantum. On the other hand, no-signaling, guaranteed by Einstein's theory of relativity, is also delicately preserved by no-cloning. This chapter will focus on these topics, hoping to give a thorough description of the no-cloning theorem. As it is known, a single measurement on a quantum system will only reveal minor information about it, but as a result of which, the quantum system will collapse to an eigenstate of the measurement operator and all the other information about the original state becomes

lost. Suppose there exists a cloning machine with a quantum operation U , which duplicates an arbitrary pure state

$$U(|\phi\rangle \otimes |R\rangle \otimes |M\rangle) = |\phi\rangle \otimes |\phi\rangle \otimes |M(\phi)\rangle \quad (36)$$

here $|\phi\rangle$ denotes an arbitrary pure state, $|R\rangle$ an initial blank state of the cloning machine, $|M\rangle$ the initial state of the auxiliary state(ancilla), and $M(\phi)$ is the ancillary state after operation which depends on $|\phi\rangle$. With such machine, one can get any number of copies of the original quantum state, and then complete information of it can be determined. However, is it possible to really build such a machine? No-cloning theorem says no.

THEOREM : No quantum operation exist which can perfectly and deterministically duplicate a pure state. The proof can be in two methods.

(1). Using the linearity of quantum mechanics. This proof is first proposed by Wootters and Zurek (Wootters and Zurek, 1982), and also by Dieks (Dieks, 1982). Suppose there exists a perfect cloning machine that can copy an arbitrary quantum state, that is, for any state $|\phi\rangle$

$$|\phi\rangle|\Sigma\rangle|M\rangle \rightarrow |\phi\rangle|\phi\rangle|M(\phi)\rangle$$

where $|\Sigma\rangle$ is a blank state, and $|M\rangle$ is the state of auxiliary system(ancilla). Thus for state $|0\rangle$ and $|1\rangle$, we have

$$|0\rangle|\Sigma\rangle|M\rangle \rightarrow |0\rangle|0\rangle|M(0)\rangle,$$

$$|1\rangle|\Sigma\rangle|M\rangle \rightarrow |1\rangle|1\rangle|M(1)\rangle$$

In this way, for the state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$

$$(\alpha|0\rangle + \beta|1\rangle)|\Sigma\rangle|M\rangle \rightarrow \alpha|00\rangle|M(0)\rangle + \beta|11\rangle|M(1)\rangle$$

On the other hand, $|\psi\rangle$ itself is a pure state, so

$$|\psi\rangle|\Sigma\rangle|M\rangle \rightarrow (\alpha^2|00\rangle + \alpha\beta|01\rangle + \alpha\beta|10\rangle + \beta^2|11\rangle)|M(\psi)\rangle.$$

Obviously, the right hand sides of the two equations cannot be equal, as a result, the premise is false that such a perfect cloning machine exists, which concludes the proof. The linearity of quantum mechanics is also used to show that the superluminal is not possible (Dieks, 1982).

(2). Using the properties of unitary operation. This proof is first proposed by Yuen in (Yuen, 1986), see also Sec. 9-4 of Peres's textbook(Peres, 1995). Consider the

process of cloning machine as a unitary operator U , then for any two state $|\phi\rangle$ and $|\psi\rangle$, since under unitary operation the inner product is preserved, we have

$$\langle\psi|\phi\rangle = \langle\psi|U^\dagger U|\phi\rangle = \langle\psi|\langle\psi|\phi\rangle|\phi\rangle = \langle\psi|\phi\rangle^2$$

So $\langle\psi|\phi\rangle$ is either 0 or 1. If the value is 0, it means the two states being copied should be orthogonal, while if 1, the two states are the same.

No-cloning and no-signaling:

According to Einstein's relativity theory, superluminal signaling cannot be physically realized. Yet due to the non-local property of quantum entanglement, superluminal signaling is possible provided perfect cloning machine can be made. The scheme has been well-known since Herbert (Herbert, 1982) first proposed his "FLASH" in 1982. The idea is as follows: suppose Alice and Bob, at an arbitrary distance, share a pair of entangled qubits in the state $|\psi\rangle = (1/\sqrt{2})(|01\rangle - |10\rangle)$. Alice can measure her qubit by either σ_x or σ_z . If the measurement is σ_z , Alice's qubit will collapse to the state $|0\rangle$ or $|1\rangle$, with probability 50%. Respectively, this prepares Bob's qubit in the state $|1\rangle$ or $|0\rangle$. Without knowing the result of Alice's measurement, the density matrix of Bob's qubit is $\frac{1}{2}|0\rangle\langle 0| + \frac{1}{2}|1\rangle\langle 1| = \frac{1}{2}I$. On the other hand, if Alice's measurement is σ_x , Alice's qubit will collapse to the state $|\phi_{x+}\rangle$ or $|\phi_{x-}\rangle$, where $|\phi_{x+}\rangle = 1/\sqrt{2}(|0\rangle + |1\rangle)$, $|\phi_{x-}\rangle = 1/\sqrt{2}(|0\rangle - |1\rangle)$, being eigenvectors of σ_x . Thus Bob's qubit is prepared in the state $|\phi_{x-}\rangle$ or $|\phi_{x+}\rangle$ respectively, in this case the density matrix of Bob's qubit is still $\frac{1}{2}|\phi_{x+}\rangle\langle\phi_{x+}| + \frac{1}{2}|\phi_{x-}\rangle\langle\phi_{x-}| = \frac{1}{2}I$. Obviously, Bob gets no information about which measurement is made by Alice. While, if perfect cloning is allowed, the scenario will change. Bob can use the cloning machine to make arbitrarily many copies of his qubit, in which way he is able to determine the exact state of his qubit, that is, whether an eigenstate of σ_z or σ_x . With this information, Bob knows the measurement Alice has taken. Fortunately, since no-cloning theorem has been proved, the above superluminal signaling scheme cannot be realized, which leaves theory of relativity and quantum mechanics in coexistence.

Up to now, there are many cloning schemes found, naturally one may ask, whether it is possible by using imperfect cloning, to extract information about which measuring

basis Alice has used. According to the property of quantum transformation, the answer is no. To see this, we may first consider a simple scheme, that is, Bob can use the universal quantum cloning machine (UQCM) proposed by Buřek and Hillery (Buřek and Hillery, 1996) to process his qubit. The UQCM transformation reads,

$$\begin{aligned} |0\rangle|Q\rangle &\rightarrow \sqrt{2/3} |00\rangle|\uparrow\rangle + \sqrt{1/3}|+\rangle|\downarrow\rangle, \\ |1\rangle|Q\rangle &\rightarrow \sqrt{2/3} |11\rangle|\downarrow\rangle + \sqrt{1/3}|+\rangle|\uparrow\rangle, \end{aligned}$$

where $|Q\rangle$ is the original state of the copying-machine, $|+\rangle$ and $|-\rangle$ are two orthogonal states of the output, $|+\rangle = 1/\sqrt{2} (|01\rangle + |10\rangle)$, $|-\rangle = 1/\sqrt{2} (|01\rangle - |10\rangle)$, and $|\uparrow\rangle, |\downarrow\rangle$ are the ancillary states which are orthogonal to each other. If Alice chooses σ_z , the density matrix of Bob's qubit after the process is

$$\begin{aligned} \rho_b &= \frac{1}{2} \left(\frac{2}{3} |00\rangle\langle 00| + \frac{1}{3} |+\rangle\langle +| \right) + \frac{1}{2} \left(\frac{2}{3} |11\rangle\langle 11| + \frac{1}{3} |+\rangle\langle +| \right) \\ &= \frac{1}{3} (|00\rangle\langle 00| + |11\rangle\langle 11| + |+\rangle\langle +|). \end{aligned}$$

If Alice chooses σ_x , it can be easily verified that the density matrix will not change, thus no information can be gained by Bob. In fact, Bruss et al. have pointed out in (Bruř et al., 2000b) that the density matrix of Bob's qubit will not change no matter what operation is taken on it, as long as the operation is linear and trace-preserving. Suppose the original density matrix shared between Alice and Bob is ρ^{ab} , and Alice has done a measurement A_m on her qubit, Bob makes a transformation B on his, then the shared density matrix becomes $A_m \otimes B(\rho^{ab})$, here m specifies which measurement Alice has taken. In Bob's view, with the linear and trace-preserving property of A_m , the density matrix of his qubit is

$$\begin{aligned} \text{tr}_a(A_m \otimes B(\rho^{ab})) &= B\text{tr}_a(A_m \otimes I(\rho^{ab})) \\ &= B\text{tr}_a(\rho^{ab}). \end{aligned}$$

Note that tr and Tr both denote trace similarly in this review. Here we see that the density matrix of Bob's qubit has nothing to do with Alice's measurement A_m , therefore no information is transferred to Bob. Note that to get the above conclusion, we have only used the linear and trace-preserving property of A_m . Since any quantum operator is

linear and completely positive, no-signaling should always hold, thus providing a method to determine the fidelity limit of a cloning machine.

The situation might be more complicated when the no-signaling correlation is considered. It is found that, however, no-signaling might be more non-local than that of quantum mechanics. Then it seems that besides of no-signaling, some extra principle, like local orthogonality (Fritz et al., 2013), should be satisfied such that the no-signaling nonlocality might be realizable by quantum mechanics (Popescu, 2014).

Gisin studied the case of $1 \rightarrow 2$ qubit UQCM in (Gisin, 1998). We continue the scheme that Alice and Bob share a pair of entangled states. Now Alice has done some measurement by σ_x or σ_z , and thus Bob's state has been prepared in a respect mixed state. Let there be a UQCM, suppose the input density matrix is $|\phi\rangle\langle\phi| = \frac{1}{2}(\mathbf{I} + \mathbf{m} \cdot \boldsymbol{\sigma})$, with \mathbf{m} being the Bloch vector of $|\phi\rangle$, then after cloning the reduced state on party a and b should read, $\rho_a = \rho_b = (1 + \eta \mathbf{m} \cdot \boldsymbol{\sigma})$, yielding the fidelity to be $F = (1 + \eta)/2$. According to the form of ρ^a and ρ^b , the composite output state of the cloning machine should be

$$\rho^{\text{out}} = \frac{1}{4}(\mathbf{I}_4 + \eta(\mathbf{m} \cdot \boldsymbol{\sigma} \otimes \mathbf{I} + \mathbf{I} \otimes \mathbf{m} \cdot \boldsymbol{\sigma}) + \sum_{i,j=x,y,z} t_{ij} \sigma_i \otimes \sigma_j).$$

The universality of UQCM requires

$$\rho_{\text{out}}(\mathbf{U}_m) = \mathbf{U} \otimes \mathbf{U} \rho_{\text{out}}(\mathbf{m}) \mathbf{U}^\dagger \otimes \mathbf{U}^\dagger. \quad (43)$$

The no-signaling condition requires

$$\frac{1}{2} \rho^{\text{out}}(+x) + \frac{1}{2} \rho^{\text{out}}(-x) = \frac{1}{2} \rho^{\text{out}}(+z) + \frac{1}{2} \rho^{\text{out}}(-z), \quad (44)$$

where $\rho^{\text{out}}(+z)$ represents the output state of the UQCM under the condition that Alice has take the measurement σ_x and got result $+$.

Also we should notice that ρ^{out} must be positive. Putting the positive condition together with (43) and (44), we shall get $\eta \leq \frac{2}{3}$ ($F \leq \frac{5}{6}$). Although we have found an upper bound of F , the question remains whether it can be reached. But we know it can be, since a practical UQCM scheme with $F = \frac{5}{6}$ has been proposed (Buřek and Hillery, 1996).

Navez et al. have derived the upper bound of fidelity for d-dimensional $1 \rightarrow 2$ UQCM using no-signaling condition (Navez and Cerf, 2003), and the bound also has been proved to be tight. Simon et al. have shown how no-signaling condition together with the static property of quantum mechanics can lead to properties of quantum dynamics (Simon et al., 2001). By static properties we mean: 1) The states of quantum systems are described as vectors in Hilbert space. 2) The usual observables are represented by projections in Hilbert space and the probabilities for measurement are described by the usual trace rule. The two properties with no-signaling condition shall imply that any quantum map must be completely positive and linear, which is what we already have in mind. This may help to understand why bound derived by no-signaling condition is always tight. The experimental test of the no-signaling theorem is also performed in optical system (De Angelis et al., 2007). From no-signaling condition, the monogamy relation of violation of Bell inequalities can be derived, and this can be used to obtain the optimal fidelity for asymmetric cloning (Pawłowski and Brukner, 2009). And some general properties of no-signaling theorem are presented in (Masanes et al., 2006). The relationship between optimal cloning and no signaling is presented in (Ghosh et al., 1999). The no-signaling is shown to be related with optimal state estimation (Han et al., 2010). Also the no-signaling is equivalent to the optimal condition in minimum-error quantum state discrimination (Bae et al., 2011), more results of those topics can be found in (Bae and Hwang, 2012) for qubit case and (Bae, 2012b) for the general case. The optimal cloning of arbitrary fidelity by using no-signaling is studied in (Gedik and C, akmak, 2012).

CHAPTER – 3

SYSTEM ANALYSIS

3. SYSTEM ANALYSIS

3.1 Existing System

The Generation of pure random numbers from classical computing is impossible, it requires an external source for getting the pure randomness. In most of the solutions, systems use the information like POSIX time, cpu status, internet traffic, etc,. Since most of the servers used to log this information, it is possible to get the right feed if the server app source code is revealed or the server app is compromised.

According to the principle of pseudo random number generators, a stream of random numbers which the server generated and used as a requirement for cryptography can be revealed if the server is compromised and which can lead to retrieval of sensitive data and vital information which should not be found outside the organization.

Combination of retrieving server app source code and log and monitoring all the server's communication can lead to a massive cyber-attack. Although these attacks are rare and hard to happen, there is no proof it can't happen in future.

3.2 Proposed System

In this project we directly generate seed for cryptography using Quantum computing. In Quantum computer we set the value in superposition state, hence the bits set the value on the 0's and 1's 50% equally.

The generated seed in superposition state comes as definite random number. Quantum computer is the only source to generate true random numbers. Hence, no one can predict the seed in quantum computer because it is true random.

3.3 REQUIREMENTS ANALYSIS AND SPECIFICATIONS

The Requirement Analysis and Specification phase starts after the feasibility study stage is complete and the project is financially viable and technically feasible. This phase ends when the requirements specification document has been developed and

reviewed. It is usually called the Software Requirements Specification (SRS) Document.

3.3.1 Hardware Requirements

Computer: Quantum Computer

Qubits: more than 5

Processor type: atleast Falcon r5 and must support Hadamard gate.

3.3.2 Functional Requirements

Quantum computer capable of processing the given quantum circuit without noise if used as a seeder or else a readily accessible quantum computer with high accuracy and capable of processing quantum circuits instantly.

3.4 Software Environment

- Operating System: Linux | Mac | Windows | Solaris
- Language used: Python
- Tools: VS Code, IBMQ cloud quantum machines, Qiskit quantum circute optimizer
- Framework: Qiskit, Tkinter (GUI)
- Server client connection: Python socket
- Server client protocol: TCP

3.5 Software Description

Python:



Fig 3.1: Icon of Python

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built-in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components. Python's simple, easy-to-learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms and can be freely distributed.

Often, programmers fall in love with Python because of the increased productivity it provides. Since there is no compilation step, the edit-test-debug cycle is incredibly fast. Debugging Python programs is easy: a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception. When the program doesn't catch the exception, the interpreter prints a stack

trace. A source-level debugger allows inspection of local and global variables, evaluation of arbitrary expressions, setting breakpoints, stepping through the code a line at a time, and so on. The debugger is written in Python itself, testifying to Python's introspective power. On the other hand, often the quickest way to debug a program is to add a few print statements to the source: the fast edit-test-debug cycle makes this simple approach very effective.

Visual Studio Code:

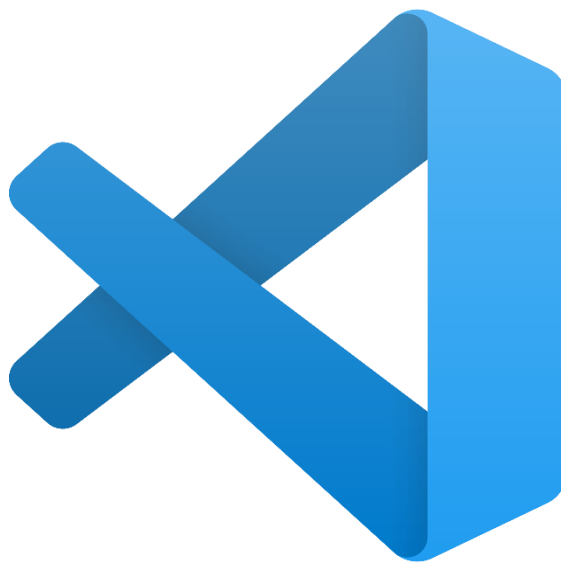


Fig 3.2: Icon of Visual Studio

Visual Studio Code is a source-code editor that can be used with a variety of programming languages, including Java, JavaScript, Go, Node.js, Python, C++ and Fortran. It is based on the Electron framework, which is used to develop Node.js Web applications that run on the Blink layout engine. Visual Studio Code employs the same editor component (codenamed "Monaco") used in Azure DevOps (formerly called Visual Studio Online and Visual Studio Team Services).

Out of the box, Visual Studio Code includes basic support for most common programming languages. This basic support includes syntax highlighting, bracket matching, code folding, and configurable snippets. Visual Studio Code also ships with IntelliSense for JavaScript, TypeScript, JSON, CSS, and HTML, as well as

debugging support for Node.js. Support for additional languages can be provided by freely available extensions on the VS Code Marketplace.

Instead of a project system, it allows users to open one or more directories, which can then be saved in workspaces for future reuse. This allows it to operate as a language-agnostic code editor for any language. It supports many programming languages and a set of features that differs per language. Unwanted files and folders can be excluded from the project tree via the settings. Many Visual Studio Code features are not exposed through menus or the user interface but can be accessed via the command palette.

IBMQ cloud quantum machines:



Fig 3.3: IBMQ cloud quantum machine

IBM Quantum systems, based on IBM Quantum System One, are built using world-leading quantum processors, cryogenic components, control electronics, and classical computing technology. These systems are denoted by names that start with `ibmq_*`. All systems return a configuration file containing all the information needed for executing quantum circuits on the systems. Additionally, quantum systems return properties information that details the characteristics of the system qubits, as well as the

gates acting on these qubits. This includes noise information obtained from system calibration scripts.

Today, IBM Quantum makes real quantum hardware -- a tool scientists only began to imagine three decades ago -- available to thousands of developers. Our engineers deliver ever-more-powerful superconducting quantum processors at regular intervals, building toward the quantum computing speed and capacity necessary to change the world.

These machines are very different from the classical computers that have been around for more than half a century. Here's a primer on this transformative technology.

Qiskit quantum circuite optimizer:

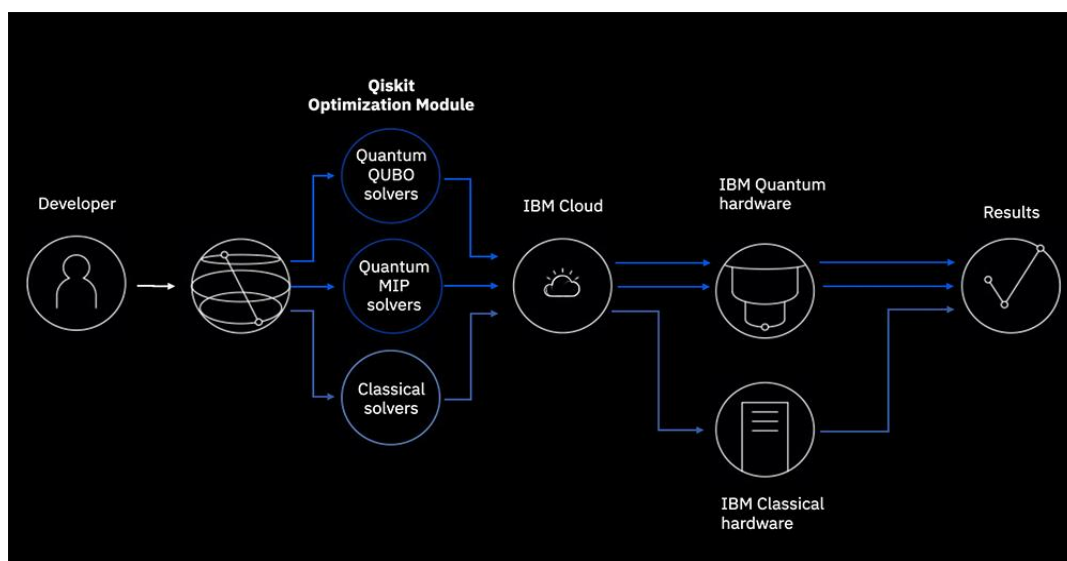


Fig 3.4: Qiskit quantum circuit optimizer

Qiskit is an open-source software development kit (SDK) for working with quantum computers at the level of circuits, pulses, and algorithms. It provides tools for creating and manipulating quantum programs and running them on prototype quantum devices on IBM Quantum Experience or on simulators on a local computer. It follows the circuit model for universal quantum computation, and can be used for

any quantum hardware (currently supports superconducting qubits and trapped ions) that follows this model.

Qiskit was founded by IBM Research to allow software development for their cloud quantum computing service, IBM Quantum Experience. Contributions are also made by external supporters, typically from academic institutions.

The primary version of Qiskit uses the Python programming language. Versions for Swift and JavaScript were initially explored, though the development for these versions have halted. Instead, a minimal re-implementation of basic features is available as *MicroQiskit*, which is made to be easy to port to alternative platforms.

A range of Jupyter notebooks are provided with examples of quantum computing being used. Examples include the source code behind scientific studies that use Qiskit, as well as a set of exercises to help people to learn the basics of quantum programming. An open-source textbook based on Qiskit is available as a university-level quantum algorithms or quantum computation course supplement.

Tkinter (GUI):

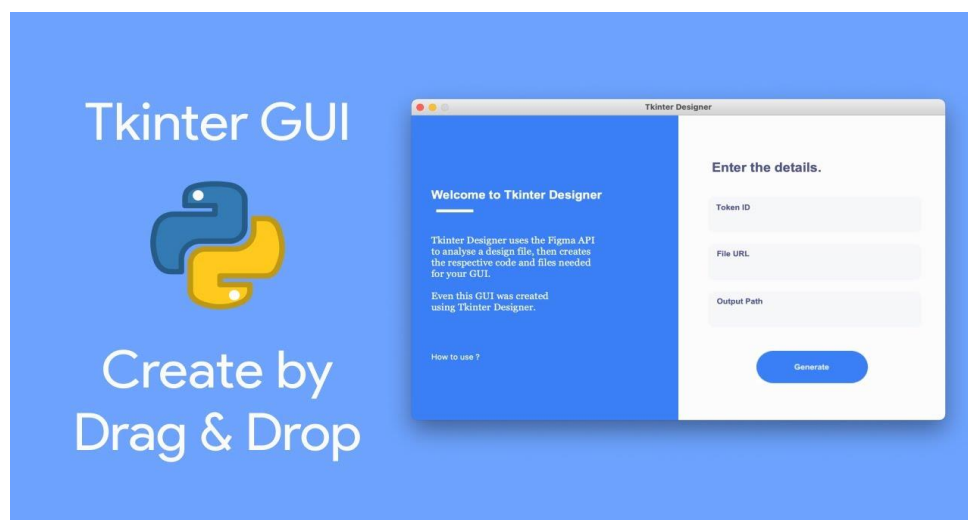


Fig 3.5: Structure of Python Tkinter GUI

The tkinter package (“Tk interface”) is the standard Python interface to the Tcl/Tk GUI toolkit. Both Tk and tkinter are available on most Unix platforms, including macOS, as well as on Windows systems.

Running `python -m tkinter` from the command line should open a window demonstrating a simple Tk interface, letting you know that tkinter is properly installed on your system, and also showing what version of Tcl/Tk is installed, so you can read the Tcl/Tk documentation specific to that version.

Tkinter supports a range of Tcl/Tk versions, built either with or without thread support. The official Python binary release bundles Tcl/Tk 8.6 threaded. See the source code for the `_tkinter` module for more information about supported versions.

Tkinter is not a thin wrapper, but adds a fair amount of its own logic to make the experience more pythonic. This documentation will concentrate on these additions and changes, and refer to the official Tcl/Tk documentation for details that are unchanged.

Python socket:

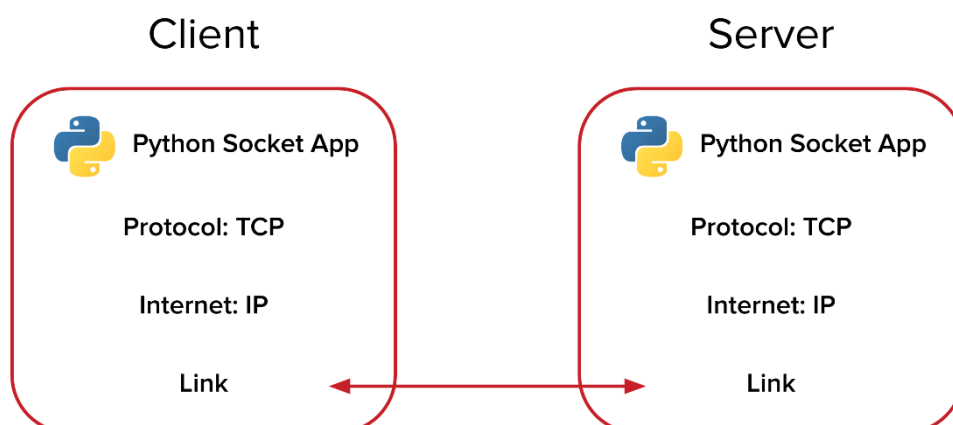


Fig 3.6: Structure of Python socket

Sockets are the endpoints of a bidirectional communications channel. Sockets may communicate within a process, between processes on the same machine, or between processes on different continents.

Sockets may be implemented over a number of different channel types: Unix domain sockets, TCP, UDP, and so on. The *socket* library provides specific classes for handling the common transports as well as a generic interface for handling the rest.

Python provides two levels of access to network services. At a low level, you can access the basic socket support in the underlying operating system, which allows you to implement clients and servers for both connection-oriented and connectionless protocols.

Python also has libraries that provide higher-level access to specific application-level network protocols, such as FTP, HTTP, and so on.

TCP:

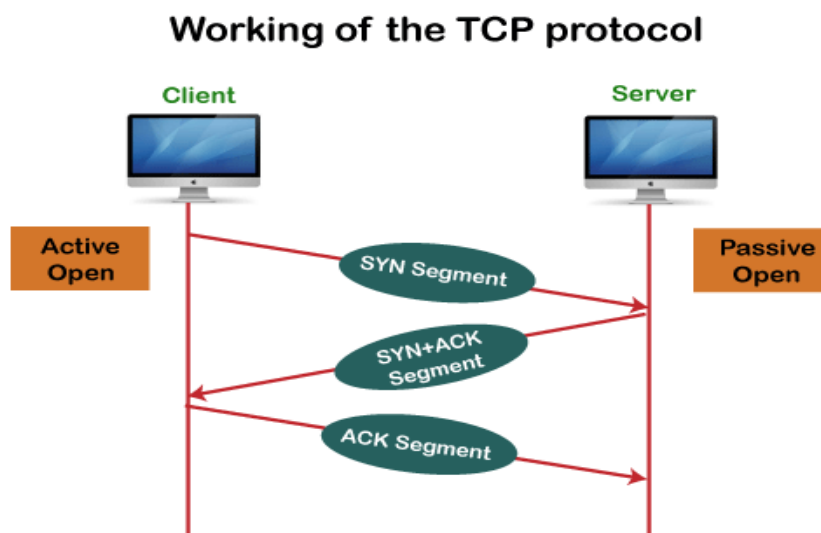


Fig 3.7: Working of the TCP protocol

TCP stands for Transmission Control Protocol a communications standard that enables application programs and computing devices to exchange messages over a network. It is designed to send packets across the internet and ensure the successful delivery of data and messages over networks.

TCP is one of the basic standards that define the rules of the internet and is included within the standards defined by the Internet Engineering Task Force (IETF). It is one of the most commonly used protocols within digital network communications and ensures end-to-end data delivery.

TCP organizes data so that it can be transmitted between a server and a client. It guarantees the integrity of the data being communicated over a network. Before it transmits data, TCP establishes a connection between a source and its destination, which it ensures remains live until communication begins. It then breaks large amounts of data into smaller packets, while ensuring data integrity is in place throughout the process.

As a result, high-level protocols that need to transmit data all use TCP Protocol. Examples include peer-to-peer sharing methods like File Transfer Protocol (FTP), Secure Shell (SSH), and Telnet. It is also used to send and receive email through Internet Message Access Protocol (IMAP), Post Office Protocol (POP), and Simple Mail Transfer Protocol (SMTP), and for web access through the Hypertext Transfer Protocol (HTTP).

CHAPTER – 4

SYSTEM DESIGN

4. SYSTEM DESIGN

System design is the phase that bridges the gap between problem domain and the existing system in a manageable way. This phase focuses on the solution domain, i.e. *“how to implement?”*

It is the phase where the SRS document is converted into a format that can be implemented and decides how the system will operate.

In this phase, the complex activity of system development is divided into several smaller sub-activities, which coordinate with each other to achieve the main objective of system development.

4.1 UML Diagrams:

UML is an acronym that stands for **Unified Modeling Language**. Simply put, UML is a modern approach to modeling and documenting software. In fact, it's one of the most popular business process modeling techniques.

It is based on **diagrammatic representations** of software components. As the old proverb says: “a picture is worth a thousand words”. By using visual representations, we are able to better understand possible flaws or errors in software or business processes.

4.1.1 Use case Diagram:

A use case diagram can summarize the details of your system's users (also known as actors) and their interactions with the system. To build one, we'll use a set of specialized symbols and connectors. An effective use case diagram can help your team discuss and represent:

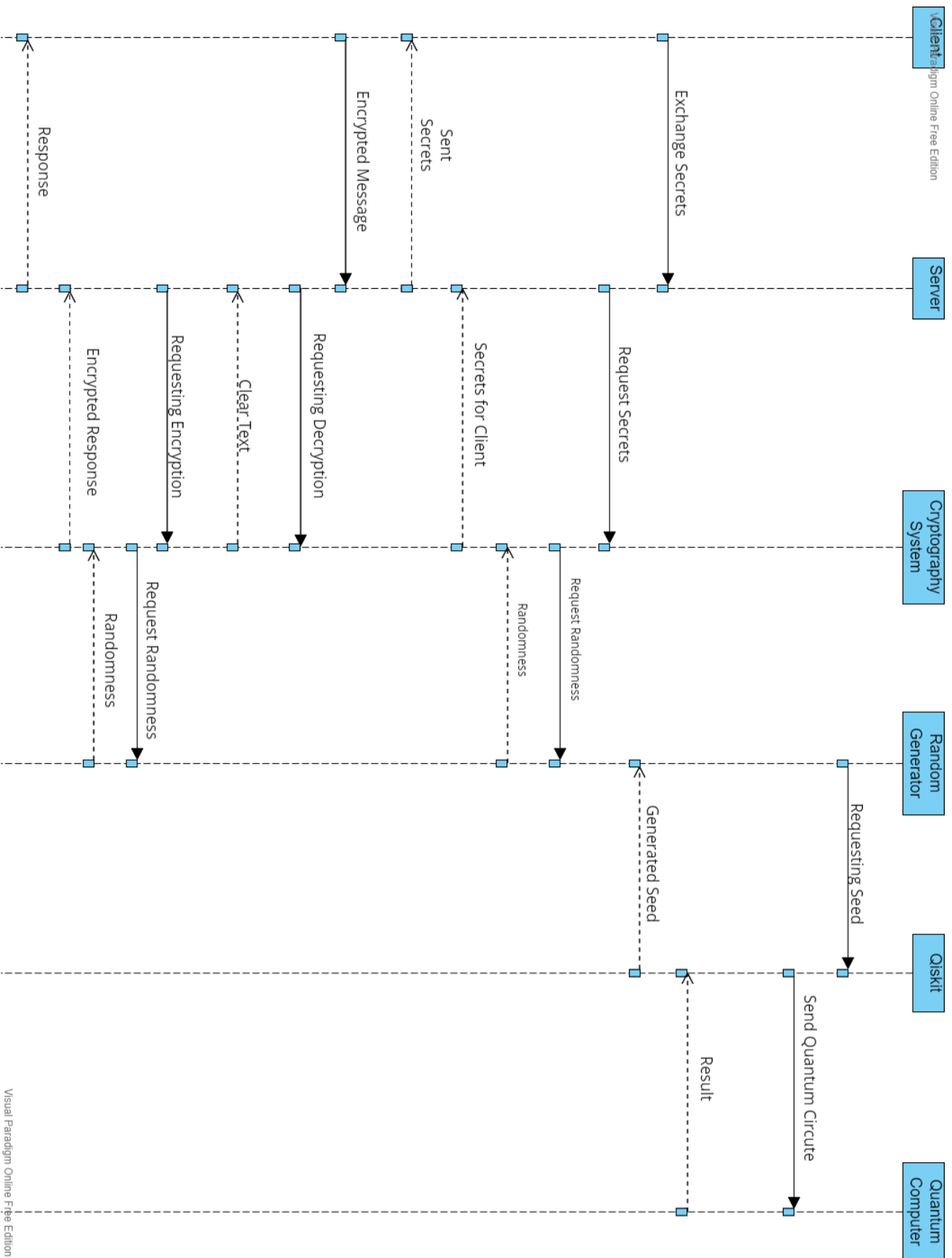
- Scenarios in which our system or application interacts with people, organizations, or external systems
- Goals that our system or application helps those entities (known as actors) achieve
- The scope of our system



4.1.2 Sequence Diagram:

A **sequence diagram** or **system sequence diagram (SSD)** shows process interactions arranged in time sequence in the field of software engineering. It depicts the processes involved and the sequence of messages exchanged between the processes needed to carry out the functionality. Sequence diagrams are typically associated with use case realizations in the 4+1 architectural view model of the system under development. Sequence diagrams are sometimes called **event diagrams** or **event scenarios**.

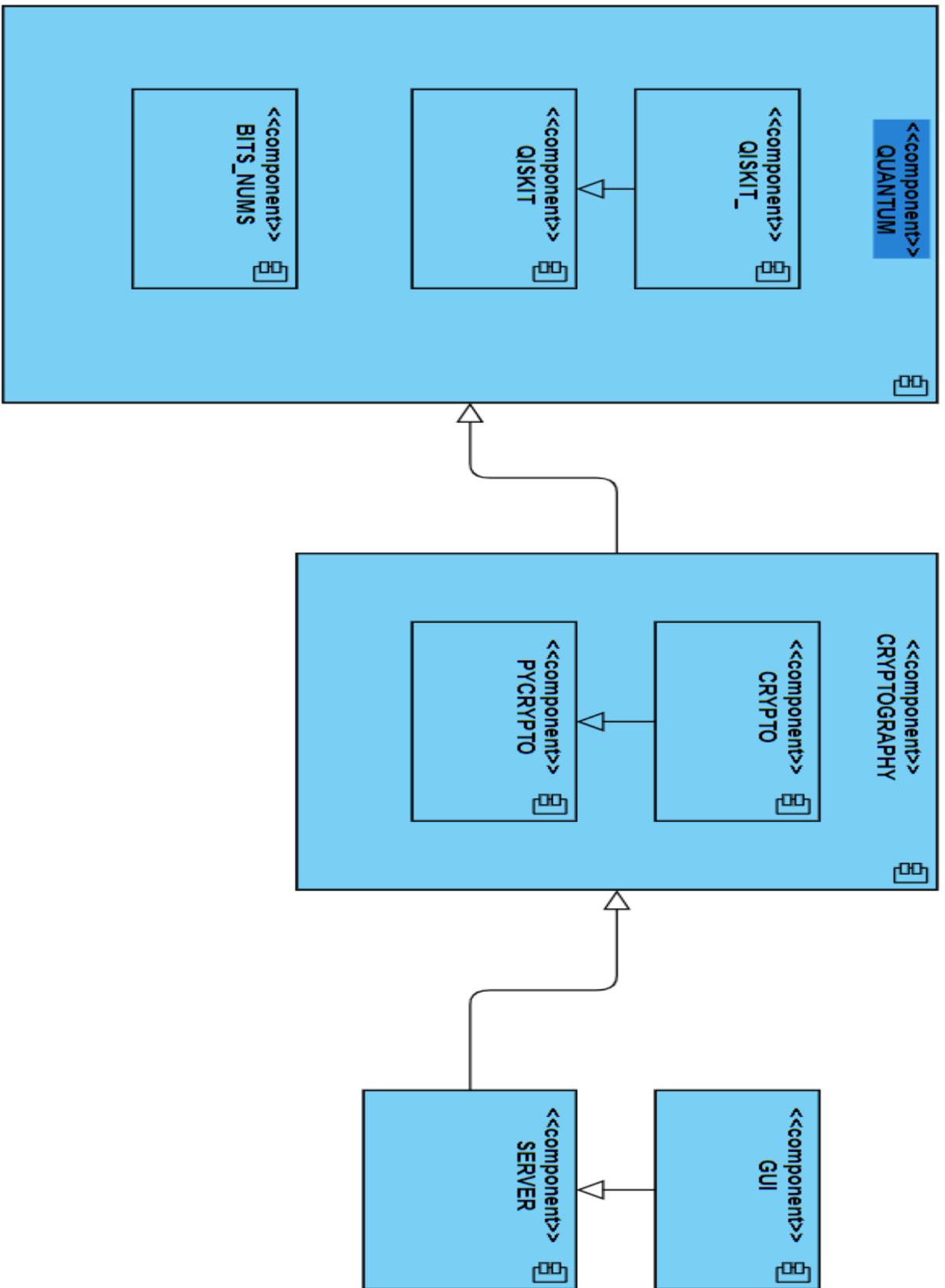
For a particular scenario of a use case, the diagrams show the events that external actors generate, their order, and possible inter-system events. All systems are treated as a black box; the diagram places emphasis on events that cross the system boundary from actors to systems. A system sequence diagram should be done for the main success scenario of the use case, and frequent or complex alternative scenarios.



4.1.3 Component Diagram:

A component diagram allows verification that a system's required functionality is acceptable. These diagrams are also used as a communication tool between the developer and stakeholders of the system. Programmers and developers use the diagrams to formalize a roadmap for the implementation, allowing for better decision-making about task assignment or needed skill improvements. System administrators can use component diagrams to plan ahead, using the view of the logical software components and their relationships on the system.

The component diagram extends the information given in a component notation element. One way of illustrating the provided and required interfaces by the specified component is in the form of a rectangular compartment attached to the component element. Another accepted way of presenting the interfaces is to use the ball-and-socket graphic convention. A *provided* dependency from a component to an interface is illustrated with a solid line to the component using the interface from a "lollipop", or ball, labelled with the name of the interface. A *required* usage dependency from a component to an interface is illustrated by a half-circle, or socket, labelled with the name of the interface, attached by a solid line to the component that requires this interface. Inherited interfaces may be shown with a lollipop, preceding the name label with a caret symbol. To illustrate dependencies between the two, use a solid line with a plain arrowhead joining the socket to the lollipop.



CHAPTER – 5

SYSTEM ARCHITECTURE

5. SYSTEM ARCHITECTURE

5.1 Architecture Overview

A **system architecture** is the conceptual model that defines the structure, behavior, and more views of a system. An architecture description is a formal description and representation of a system, organized in a way that supports reasoning about the structures and behaviors of the system.

A system architecture can consist of system components and the sub-systems developed, that will work together to implement the overall system. There have been efforts to formalize languages to describe system architecture, collectively these are called architecture description languages (ADLs).

One can think of system architecture as a set of representations of an existing (or future) system. These representations initially describe a general, high-level functional organization, and are progressively refined to more detailed and concrete descriptions.

System architecture conveys the informational content of the elements consisting of a system, the relationships among those elements, and the rules governing those relationships. The architectural components and set of relationships between these components that an architecture description may consist of hardware, software, documentation, facilities, manual procedures, or roles played by organizations or people.

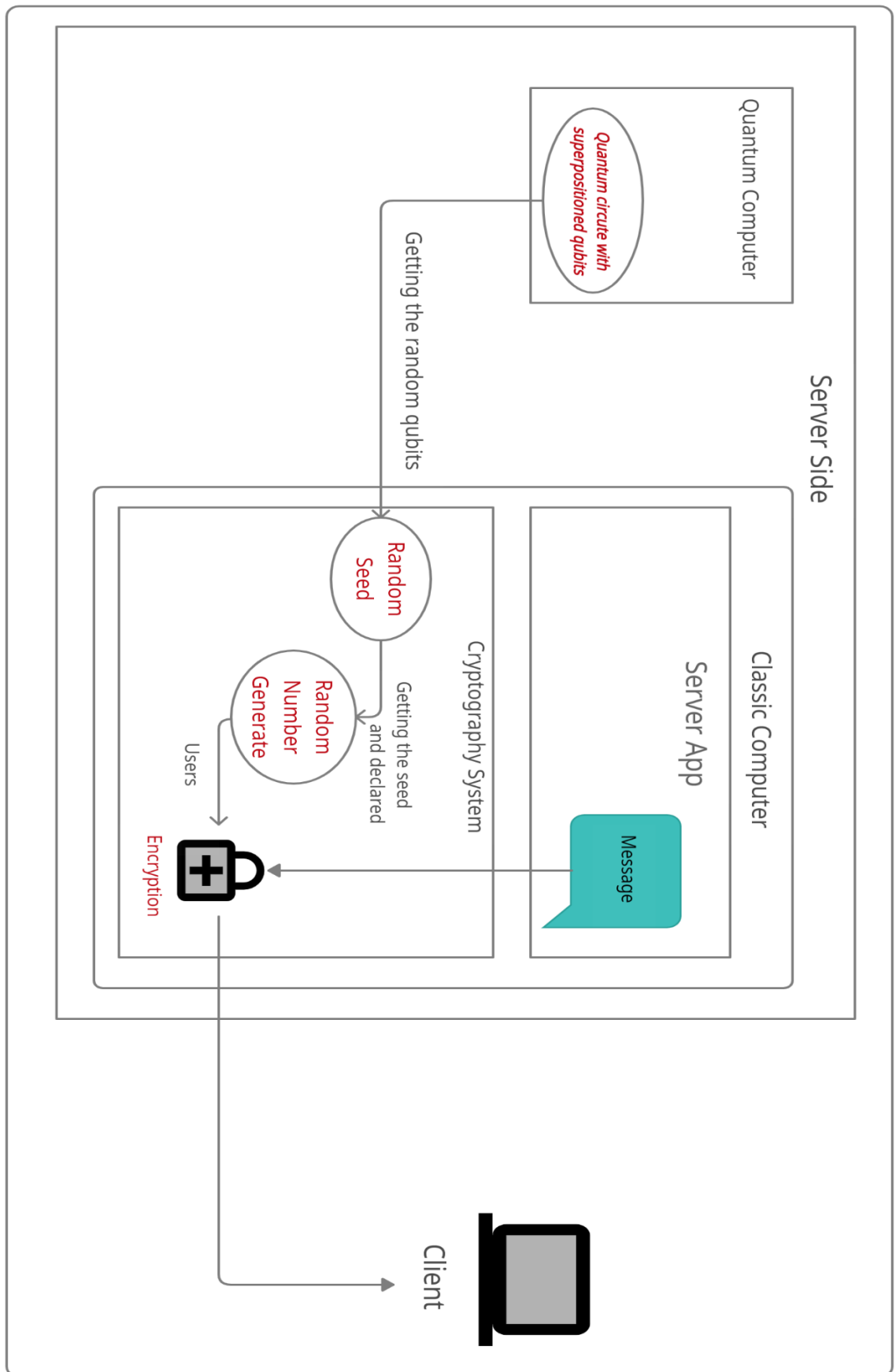


Fig 5.1: System Architecture

CHAPTER – 6

SYSTEM IMPLEMENTATION

6. SYSTEM IMPLEMENTATION

The implementation phase plays the most important role in the software development process. It is at this stage that the physical source code of the system being built is created. Programmers code the IT system on the basis of the collected requirements and the developed project documentation. They are based on experience and proven software development techniques.

6.1 Tkinter:

The tkinter package (“Tk interface”) is the standard Python interface to the Tcl/Tk GUI toolkit. Both Tk and tkinter are available on most Unix platforms, including macOS, as well as on Windows systems.

Running `python -m tkinter` from the command line should open a window demonstrating a simple Tk interface, letting you know that tkinter is properly installed on your system, and also showing what version of Tcl/Tk is installed, so you can read the Tcl/Tk documentation specific to that version.

Tkinter supports a range of Tcl/Tk versions, built either with or without thread support. The official Python binary release bundles Tcl/Tk 8.6 threaded. See the source code for the `_tkinter` module for more information about supported versions.

Tkinter is not a thin wrapper, but adds a fair amount of its own logic to make the experience more pythonic. This documentation will concentrate on these additions and changes, and refer to the official Tcl/Tk documentation for details that are unchanged.

6.1.1 Installation:

Step 1 – Make sure Python and pip is preinstalled on your system

Type the following commands in command prompt to check if python and pip is installed on your system.

To check Python

```
python --version
```

If python is successfully installed, the version of python installed on your system will be displayed.

To check pip

```
pip -V
```

The version of pip will be displayed, if it is successfully installed on your system.

Step 2 – Install Tkinter

Tkinter can be installed using pip. The following command is run in the command prompt to install Tkinter.

```
pip install tk
```

This command will start downloading and installing packages related to the Tkinter library. Once done, the message of successful installation will be displayed.

6.1.2 Implementation in Project:

Developing desktop based applications with python Tkinter is not a complex task. An empty Tkinter top-level window can be created by using the following steps.

1. import the Tkinter module.
2. Create the main application window.
3. Add the widgets like labels, buttons, frames, etc. to the window.
4. Call the main event loop so that the actions can take place on the user's computer screen.

Python Tkinter Geometry

The Tkinter geometry specifies the method by using which, the widgets are represented on display. The python Tkinter provides the following geometry methods.

1. The pack() method
2. The grid() method
3. The place() method

Python Tkinter pack() method

The pack() widget is used to organize widget in the block. The positions widgets added to the python application using the pack() method can be controlled by using the various options specified in the method call.

Syntax: widget.pack(options)

Python Tkinter grid() method

The grid() geometry manager organizes the widgets in the tabular form. We can specify the rows and columns as the options in the method call. We can also specify the column span (width) or rowspan(height) of a widget.

Syntax: widget.grid(options)

Python Tkinter place() method

The place() geometry manager organizes the widgets to the specific x and y coordinates.

Syntax: widget.place(options)

6.2 Qiskit

Qiskit is an open-source software development kit (SDK) for working with quantum computers at the level of circuits, pulses, and algorithms. It provides tools for creating and manipulating quantum programs and running them on prototype quantum devices on IBM Quantum Experience or on simulators on a local computer. It follows the circuit model for universal quantum computation, and can be used for any quantum hardware (currently supports superconducting qubits and trapped ions) that follows this model.

Qiskit was founded by IBM Research to allow software development for their cloud quantum computing service, IBM Quantum Experience. Contributions are also made by external supporters, typically from academic institutions.

The primary version of Qiskit uses the Python programming language. Versions for Swift and JavaScript were initially explored, though the development for these versions have halted. Instead, a minimal re-implementation of basic features is available as *MicroQiskit*, which is made to be easy to port to alternative platforms.

A range of Jupyter notebooks are provided with examples of quantum computing being used. Examples include the source code behind scientific studies that use Qiskit, as well as a set of exercises to help people to learn the basics of quantum programming. An open-source textbook based on Qiskit is available as a university-level quantum algorithms or quantum computation course supplement.

6.2.1 Installation

Step 1: Download the Anaconda Python installer for Windows.

Step 2: Once the installation is complete, launch the “Anaconda Prompt”

Step 3: In the anaconda prompt, type “pip install qiskit”

Step 4: You are all set to launch the Jupyter Notebook by typing the command “jupyter notebook”

Once the Jupyter Notebook is started, you can access the notebook using the URL printed in the command output.

Step 5: Let’s create a new Python 3 Notebook by clicking New > Python 3. This will open a new tab with an editor to write and execute Python programs.



Fig 6.1: Jupyter NoteBook

Step 6: Import Qiskit by typing “import qiskit” and hit “Shift+Enter”.

The above command will not display any output, as we have just imported the Qiskit library.

Step 7: Let’s now check the version of Qiskit imported by typing the below line.

qiskit.__qiskit_version__

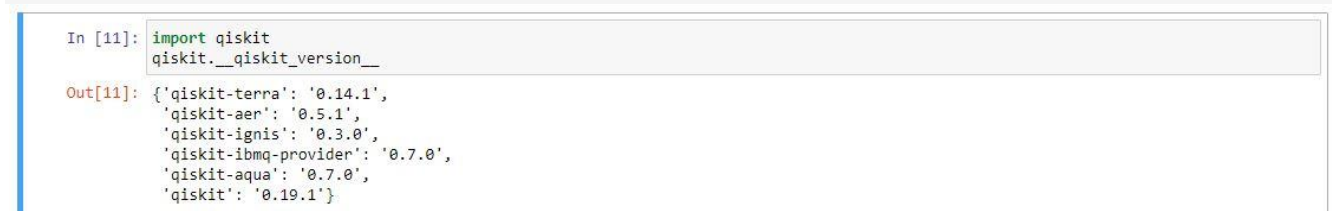


Fig 6.2: qiskit import

6.2.2 Implementation in Project:

Step 1: Go to <https://quantum-computing.ibm.com/> and sign-up to create your account

Step 2: Upon sign-in, click on the user icon on the top right corner and select "My Account"

Step 3: Click the "Copy token" (blue button) to copy your API token to the clipboard.

Step 4: Let us now go ahead and import IBMQ from Qiskit by typing the below line in the Jupyter Notebook and hit "Shift+Enter"

```
from qiskit import IBMQ
```

Step 5: You now have an IBM Q Experience's API token, so let's save that to your computer by typing the below line and hit "Shift+Enter"

```
IBMQ.save_account('<YOUR-IBM-API-TOKEN>')
```

Remember to replace YOUR-IBM-API-TOKEN with your own.

Once executed, IBM's API token is now saved onto your computer and you are now ready to access IBM's Quantum devices.

Step 6: To see that we have connected to the IBM's Quantum devices, type the below code in the Notebook and hit "Shift+Enter"

```
IBMQ.load_account()
```

Upon execution, you will see the output as shown below:

```
In [8]: IBMQ.load_account()
Out[8]: <AccountProvider for IBMQ(hub='ibm-q', group='open', project='main')>
```

Fig 6.3: IBMQ Load Account

Step 7: You are now ready to write your first Qiskit program and execute it on a simulator and also on real Quantum hardware.

```
import numpy as np

from qiskit import(

    QuantumCircuit,

    execute,

    Aer)

from qiskit.visualization import plot_histogram

# Use Aer's qasm_simulator

simulator = Aer.get_backend('qasm_simulator')

# Create a Quantum Circuit acting on the q register

circuit = QuantumCircuit(2, 2)

# Add a H gate on qubit 0

circuit.h(0)

# Add a CX (CNOT) gate on control qubit 0 and target qubit 1

circuit.cx(0, 1)

# Map the quantum measurement to the classical bits

circuit.measure([0,1], [0,1])
```

```

# Execute the circuit on the qasm simulator

job = execute(circuit, simulator, shots=1000)

# Grab results from the job

result = job.result()

# Returns counts

counts = result.get_counts(circuit)

print("\nTotal count for 00 and 11 are:",counts)


# Draw the circuit

print(circuit.draw(output='text'))

```

You should see the circuit printed as below:

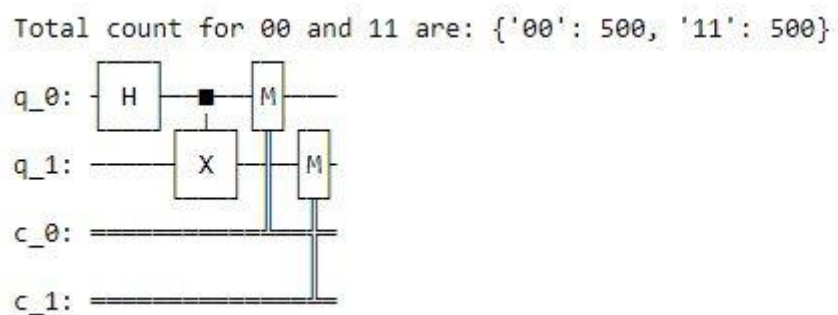


Fig 6.4: Output of IBM connection

CHAPTER – 7

SYSTEM TESTING

7. SYSTEM TESTING

7.1 Unit Testing

Unit testing is conducted to verify the functional performance of each modular component of the software. Unit testing focuses on the smallest unit of the software design (i.e.), the module.

These types of tests are usually written by developers as they work on code (white-box style), to ensure that the specific function is working as expected. One function might have multiple tests, to catch corner cases or other branches in the code. Unit testing alone cannot verify the functionality of a piece of software, but rather is used to ensure that the building blocks of the software work independently from each other.

Unit testing is a software development process that involves a synchronized application of a broad spectrum of defect prevention and detection strategies to reduce software development risks, time, and costs. It is performed by the software developer or engineer during the construction phase of the software development life cycle. Unit testing aims to eliminate construction errors before code is promoted to additional testing; this strategy is intended to increase the quality of the resulting software as well as the efficiency of the overall development process.

Depending on the organization's expectations for software development, unit testing might include static code analysis, data-flow analysis, metrics analysis, peer code reviews, code coverage analysis, and other software testing practices.

7.1.1 WHITE BOX TESTING

White Box Testing is a test case design method that uses the control structure of the procedural design to drive cases. Using the white box testing methods, we derived test cases that guarantee that all independent paths within a module have been exercised at least once. White-box testing (also known as clear box testing, glass box testing, transparent box testing, and structural testing) verifies the internal structures or workings of a program, as opposed to the functionality exposed to the end-user. In

white-box testing, an internal perspective of the system (the source code), as well as programming skills, are used to design test cases. The tester chooses inputs to exercise paths through the code and determines the appropriate outputs. This is analogous to testing nodes in a circuit, e.g., in-circuit testing (ICT).

While white-box testing can be applied at the unit, integration, and system levels of the software testing process, it is usually done at the unit level. It can test paths within a unit, paths between units during integration, and between subsystems during a system-level test. Though this method of test design can uncover many errors or problems, it might not detect unimplemented parts of the specification or missing requirements.

API testing – testing of the application using public and private APIs (application programming interfaces)

Code coverage – creating tests to satisfy some criteria of code coverage (e.g., the test designer can create tests to cause all statements in the program to be executed at least once),

Fault injection methods – intentionally introducing faults to gauge the efficacy of testing strategies.

Code coverage tools can evaluate the completeness of a test suite that was created with any method, including black-box testing. This allows the software team to examine parts of a system that are rarely tested and ensures that the most important function points have been tested. Code coverage as a software metric can be reported as a percentage.

Statement coverage, which reports on the number of lines executed to complete the test.

Decision coverage, which reports on whether both the True and the False branch of a given test has been executed

100% statement coverage ensures that all code paths or branches (in terms of control flow) are executed at least once. This helps ensure correct functionality, but not sufficient since the same code may process different inputs correctly or incorrectly.

Pseudo-tested functions and methods are those that are covered but not specified (it is possible to remove their body without breaking any test case).

7.1.2 BLACK BOX TESTING

Black-box testing (also known as functional testing) treats the software as a "black box," examining functionality without any knowledge of internal implementation, without seeing the source code. The testers are only aware of what the software is supposed to do, not how it does it. Black-box testing methods include equivalence partitioning, boundary value analysis, all-pairs testing, state transition tables, decision table testing, fuzz testing, model-based testing, use case testing, exploratory testing, and specification-based testing. Specification-based testing aims to test the functionality of software according to the applicable requirements. This level of testing usually requires thorough test cases to be provided to the tester, who then can simply verify that for a given input, the output value (or behavior), either "is" or "is not" the same as the expected value specified in the test case. Test cases are built around specifications and requirements, i.e., what the application is supposed to do. It uses external descriptions of the software, including specifications, requirements, and designs to derive test cases. These tests can be functional or non-functional, though usually functional. Specification-based testing may be necessary to assure correct functionality, but it is insufficient to guard against complex or high-risk situations. One advantage of the black box technique is that no programming knowledge is required. Whatever biases the programmers may have had, the tester likely has a different set and may emphasize different areas of functionality. On the other hand, black-box testing has been said to be "like a walk in a dark labyrinth without a flashlight. Because they do not examine the source code, there are situations when a tester writes many test cases to check something that could have been tested by only one test case or leaves some parts of the program untested. This method of test can be applied to all levels of software testing: unit, integration, system, and acceptance. It typically comprises most if not all testing at higher levels, but can also dominate unit testing as well. Black box testing is done to find incorrect or missing function Interface error Error in external database access Performance error Initialization and termination errors.

7.2 INTEGRATION TESTING

Integration Testing is a systematic technique for the construction of the program structure while at the same time conducting tests to uncover errors associated with interfacing. i.e., integration testing is the complete testing of the set of modules that makes up the product. The objective is to take untested modules and build a program structure tester should identify critical modules. Critical modules should be tested as early as possible. One approach is to wait until all the units have passed testing, and then combine them and then tested. This approach is evolved from unstructured testing of small programs. Another strategy is to construct the product in increments of tested units. A small set of modules are integrated and tested, to which another module is added and tested in combination. And so on. The advantages of this approach are that interface dispenses can be easily found and corrected. The major error that was faced during the project is linking. The connection between the cloud database and module app is very crucial and some improper casting of different types like Object and Geopoint caused the app to 63 crash and this was rectified by properly casting the types for Object data types. Another major error that was faced was the real-time location update to the cloud when the truck is on the move, instead of black black-box on individually fetching the current location the utility called location change listener was used to automatically listen for location changes and it was moved to the cloud database.

CHAPTER – 8

CONCLUSION

8. Conclusion

With this proposed system the seed used for generate random number is secured and can't be revealed to any one because of we communication with quantum computer directly and not cached anywhere.

Also, with the help this method we can securely send true random number generated using quantum computer to client even in un-encrypted channel without any compromise.

Combination of retrieving server app source code and monitoring all the server communication can lead to a massive cyber-attack. Although these attacks are rare and hard to happen, there is no proof it can't happen in future. So to prevent these kinds of attacks this project uses the quantum properties.

8.1 Performance analysis report

Existing System:

The Generation of pure random numbers from classical computing is impossible, it requires an external source for getting the pure randomness. In most of the solutions, systems use the information like POSIX time, cpu status, internet traffic, etc,. Since most of the servers used to log this information, it is possible to get the right feed if the server app source code is revealed or the server app is compromised.

According to the principle of pseudo random number generators, a stream of random numbers which the server generated and used as a requirement for cryptography can be revealed if the server is compromised and which can lead to retrieval of sensitive data and vital information which should not be found outside the organization.

Combination of retrieving server app source code and log and monitoring all the server's communication can lead to a massive cyber-attack. Although these attacks are rare and hard to happen, there is no proof it can't happen in future.

Proposed System:

In this project we directly generate seed for cryptography using Quantum computing. In Quantum computer we set the value in superposition state, hence the bits set the value on the 0's and 1's 50% equally.

The generated seed in superposition state comes as definite random number. Quantum computer is the only source to generate true random numbers. Hence, no one can predict the seed in quantum computer because it is true random.

Future Enhancement:

To create a generic algorithm to support all type of server-client protocols and cryptography techniques using the pure randomness (which is shared using proposed method)

To find out the efficient way to implement both the quantum computer and the previously mentioned algorithm to all the domestic and commercial servers.

CHAPTER – 9

APPENDICES

9. APPENDICES

9.1 SAMPLE SCREENS

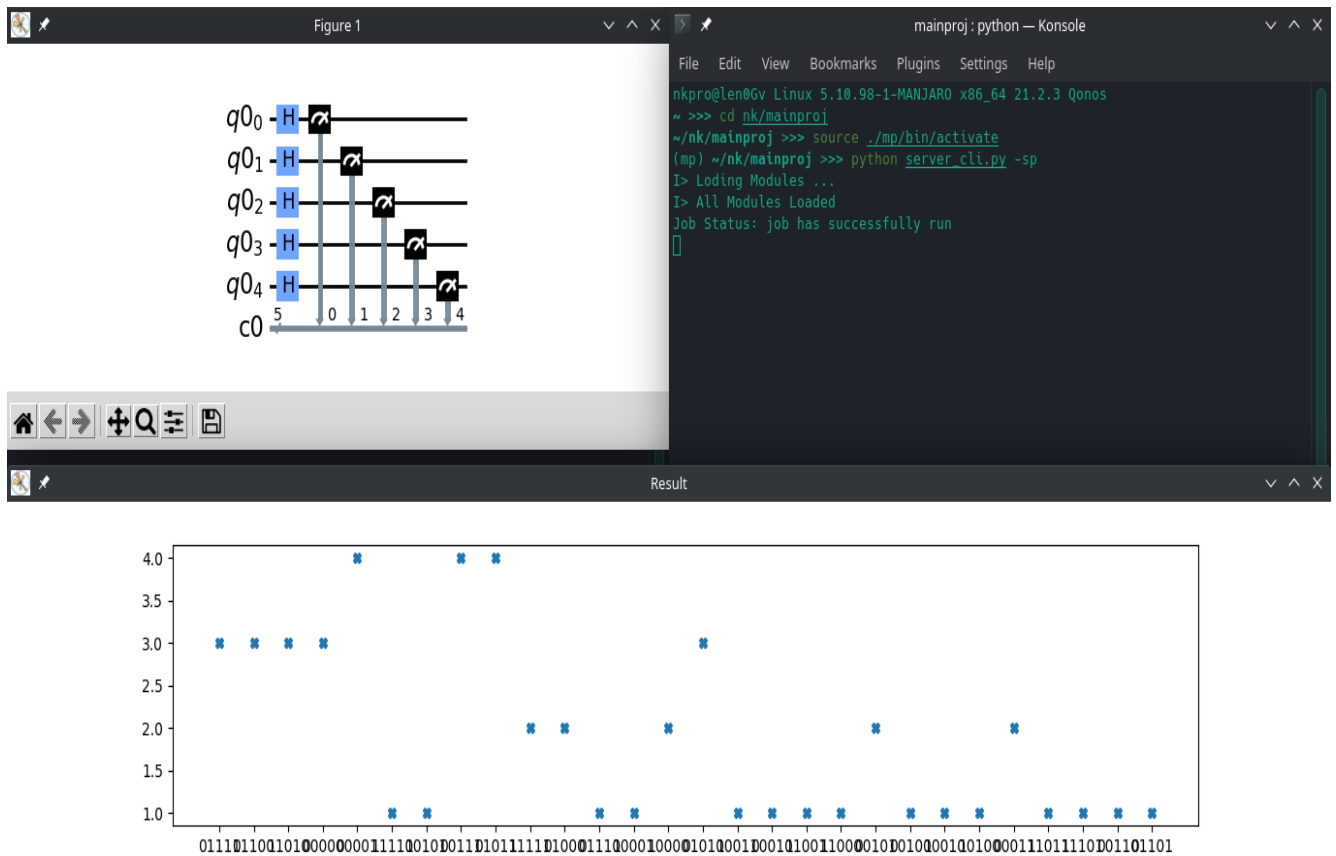
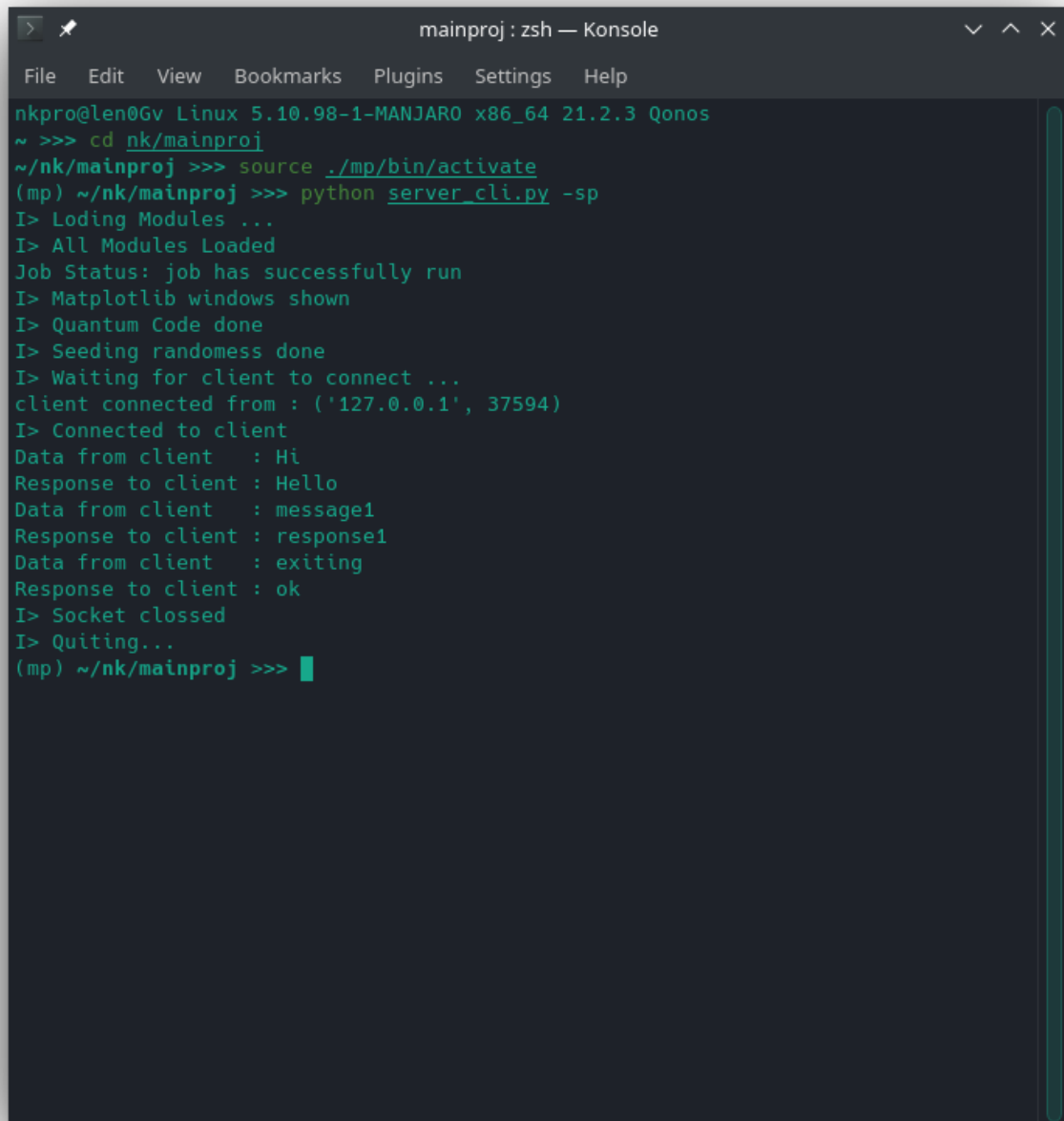


Fig 9.1: Quantum circute and result



```
mainproj : zsh — Konsole
File Edit View Bookmarks Plugins Settings Help
nkpro@len0Gv Linux 5.10.98-1-MANJARO x86_64 21.2.3 Qonos
~ >>> cd nk/mainproj
~/nk/mainproj >>> source ./mp/bin/activate
(mp) ~/nk/mainproj >>> python server_cli.py -sp
I> Loding Modules ...
I> All Modules Loaded
Job Status: job has successfully run
I> Matplotlib windows shown
I> Quantum Code done
I> Seeding randomness done
I> Waiting for client to connect ...
client connected from : ('127.0.0.1', 37594)
I> Connected to client
Data from client : Hi
Response to client : Hello
Data from client : message1
Response to client : response1
Data from client : exiting
Response to client : ok
I> Socket closed
I> Quitting...
(mp) ~/nk/mainproj >>>
```

Fig 9.2: Encrypted conversation between server and client

9.2 CODING

Client.py

```
import socket

HOST_NAME = 'localhost'
PORT = 5225

def connect():
    c = socket.socket()
    c.connect((HOST_NAME, PORT))
    return c
```

Server.py

```
import socket

HOST_NAME = 'localhost'
PORT = 5225

s = socket.socket()
s.bind((HOST_NAME, PORT))

s.listen()

def client():
    conn, addr = s.accept()
    print('client connected from :', addr)
    return conn
```

quantum.py

```
import qiskit_
from bits_nums import *

import matplotlib.pyplot as plt

N = 5 # max no. of qbits we have in IBMQ free

def get_qcir(n=None):
    if n is None:
        n = N
```

```

q, c = QuantumRegister(n), ClassicalRegister(n)
qcir = QuantumCircuit(q, c)
qcir.h(q)

# insert qbit manipulate circuit

qcir.measure(q, c)

return qcir

def show(qcir):
    qcir.draw(output='mpl').show()

def output(qcir, backend=None):
    if backend == 1:
        backend = qbackend1
    elif backend == 2:
        backend = qbackend2
    else:
        backend = sbackend

    job = execute(qcir, backend)

    return result(job, 0)

def plot(counts):
    if type(counts) is not dict:
        counts = counts.get_counts()
    plt.plot(counts.keys(), counts.values(), 'X')
    plt.show(block=False)

```

bit_nums.py

```

from qiskit_ import *

def bis(counts, len=None):
    if type(counts) is not dict:
        counts = counts.get_counts()
    results = [(j,i) for i,j in counts.items()]
    results.sort(reverse=True, key=lambda x:x[0])

```

```

        binary_string = ''.join(map(str, [i[1] for i in
results]))
        return binary_string[:len]

def num(counts, bits=None):
    return int(bis(counts, bits), 2)

```

qiskit_.py

```

from functools import partial

import qiskit
from qiskit import QuantumRegister, ClassicalRegister,
QuantumCircuit
from qiskit import BasicAer, IBMQ
from qiskit.tools.monitor import job_monitor

SHOTS = 52

sbackend = BasicAer.get_backend('qasm_simulator')
qaccount = IBMQ.load_account()
qbackend1 = qaccount.get_backend('ibmq_bogota')
qbackend2 = qaccount.get_backend('ibmq_santiago')

def execute(circute, backend=sbackend, shots=None,
memory=True):
    if shots is None:
        shots = SHOTS
    return qiskit.execute(circute, backend, shots=shots,
memory=memory)

def result(job, moc=2):
    job_monitor(job)
    r = job.result()

    if moc == 0:
        return r
    elif moc == 1:
        return r.get_counts()
    elif moc == 2:
        return r.get_memory()
    elif moc == 3:

```

```

        return r.get_memory(), r.get_counts()
    else:
        return r, r.get_memory(), r.get_counts()

'''
>>> for i in range(32):
...     print(' ' if i%3 else '>', str(i).zfill(2), i%3,
bin(i)[2:].zfill(5), end=' ')
...     b=bin(i)[2:].zfill(5)
...     a,aa = (sum(map(int,b[::2])), sum(map(int,b[1::2])))
...     print(a, aa, abs(a-aa), ' ' if i%3 else '<')
...
> 00 0 00000 0 0 0 <
    01 1 00001 1 0 1
    02 2 00010 0 1 1
> 03 0 00011 1 1 0 <
    04 1 00100 1 0 1
    05 2 00101 2 0 2
> 06 0 00110 1 1 0 <
    07 1 00111 2 1 1
    08 2 01000 0 1 1
> 09 0 01001 1 1 0 <
    10 1 01010 0 2 2
    11 2 01011 1 2 1
> 12 0 01100 1 1 0 <
    13 1 01101 2 1 1
    14 2 01110 1 2 1
> 15 0 01111 2 2 0 <
    16 1 10000 1 0 1
    17 2 10001 2 0 2
> 18 0 10010 1 1 0 <
    19 1 10011 2 1 1
    20 2 10100 2 0 2
> 21 0 10101 3 0 3 <
    22 1 10110 2 1 1
    23 2 10111 3 1 2
> 24 0 11000 1 1 0 <
    25 1 11001 2 1 1
    26 2 11010 1 2 1
> 27 0 11011 2 2 0 <
    28 1 11100 2 1 1
    29 2 11101 3 1 2
> 30 0 11110 2 2 0 <
    31 1 11111 3 2 1
>>> '''

```

REFERENCES

REFERENCES

1) **Title:** A Study of the basics of Quantum Computing

Year: 2005

Author: Prashant Singh, Department d'Informatique et de recherche operationnelle,
Universite de Montreal, Montreal. Canada.

Source: <https://arxiv.org/vc/quant-ph/papers/0511/0511061v1.pdf>

2) **Title:** Quantum Cloning Machines and the Application

Year: 2014

Authors: Heng Fan, Yi-Nan Wang, Li Jing, Jie-Dong Yue, Han-Duo Shi, Yong-Liang Zhang, and Liang-Zhu Mu

Beijing National Laboratory for Condensed Matter Physics, Institute of
Physics, Chinese Academy of Sciences, Beijing 100190, China

Collaborative Innovation Center of Quantum Matter, Beijing 100190,
China 3 School of Physics, Peking University, Beijing 100871, China

Source: <https://arxiv.org/pdf/1301.2956.pdf>