
Multigraphs: Emergence from Hilbert Space Subdivision in Superposed Quantum Systems and their Image Encoding Application

Luis José Mantilla Santa Cruz



UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE COMPUTAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Uberlândia
2024

Luis José Mantilla Santa Cruz

**Multigraphs: Emergence from Hilbert Space
Subdivision in Superposed Quantum Systems
and their Image Encoding Application**

Dissertação de mestrado apresentada ao Programa de Pós-graduação da Faculdade de Computação da Universidade Federal de Uberlândia como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

Área de concentração: Ciência da Computação

Supervisor: PhD. João Henrique de Souza Pereira

Co-supervisor: PhD. Luis F. Faina

Uberlândia

2024

Ficha Catalográfica Online do Sistema de Bibliotecas da UFU
com dados informados pelo(a) próprio(a) autor(a).

C957
2024

Cruz, Luís José Mantilla Santa, 1991-
Multigraphs: Emergence from Hilbert Space Subdivision
in Superposed Quantum Systems and their Image Encoding
Application [recurso eletrônico] / Luís José Mantilla
Santa Cruz. - 2024.

Orientador: João Henrique de Souza Pereira.

Coorientador: Luís Fernando Faina.

Dissertação (Mestrado) - Universidade Federal de
Uberlândia, Pós-graduação em Ciência da Computação.

Modo de acesso: Internet.

Disponível em: <http://doi.org/10.14393/ufu.di.2024.489>

Inclui bibliografia.

Inclui ilustrações.

1. Computação. I. Pereira, João Henrique de Souza,
1975-, (Orient.). II. Faina, Luís Fernando, 1965-,
(Coorient.). III. Universidade Federal de Uberlândia.
Pós-graduação em Ciência da Computação. IV. Título.

CDU: 681.3

Bibliotecários responsáveis pela estrutura de acordo com o AACR2:

Gizele Cristine Nunes do Couto - CRB6/2091
Nelson Marcos Ferreira - CRB6/3074



UNIVERSIDADE FEDERAL DE UBERLÂNDIA
Coordenação do Programa de Pós-Graduação em Ciência da
Computação

Av. João Naves de Ávila, 2121, Bloco 1A, Sala 243 - Bairro Santa Mônica, Uberlândia-MG,
CEP 38400-902

Telefone: (34) 3239-4470 - www.ppgco.facom.ufu.br - cpgfacom@ufu.br



ATA DE DEFESA - PÓS-GRADUAÇÃO

Programa de Pós-Graduação em:	Ciência da Computação				
Defesa de:	Dissertação de Mestrado, 17/2024, PPGCO				
Data:	15 de julho de 2024	Hora de início:	09:00	Hora de encerramento:	11:15
Matrícula do Discente:	12222CCP027				
Nome do Discente:	Luis José Mantilla Santa Cruz				
Título do Trabalho:	Multi-Graphs: Emergence from Hilbert Space Subdivision in Superposed Quantum Systems and their Image Encoding Application				
Área de concentração:	Ciência da Computação				
Linha de pesquisa:	Sistemas de Computação				
Projeto de Pesquisa de vinculação:	-----				

Reuniu-se por videoconferência, a Banca Examinadora, designada pelo Colegiado do Programa de Pós-graduação em Ciência da Computação, assim composta: Professores Doutores: Luis Fernando Faina - FACOM/UFU (Coorientador), Pedro Frosi Rosa - FACOM/UFU, Romerson Deiny Oliveira - University of Bristol e João Henrique de Souza Pereira - FACOM/UFU, orientador do candidato.

Os examinadores participaram desde as seguintes localidades: Romerson Deiny Oliveira - Bristol/Inglaterra . Os outros membros da banca e o aluno participaram da cidade de Uberlândia.

Iniciando os trabalhos o presidente da mesa, Prof. Dr. João Henrique de Souza Pereira, apresentou a Comissão Examinadora e o candidato, agradeceu a presença do público, e concedeu ao Discente a palavra para a exposição do seu trabalho. A duração da apresentação do Discente e o tempo de arguição e resposta foram conforme as normas do Programa.

A seguir o senhor presidente concedeu a palavra, pela ordem sucessivamente, aos examinadores, que passaram a arguir ao candidato. Ultimada a arguição, que se desenvolveu dentro dos termos regimentais, a Banca, em sessão secreta, atribuiu o resultado final, considerando o candidato:

Aprovado

Esta defesa faz parte dos requisitos necessários à obtenção do título de Mestre.

O competente diploma será expedido após cumprimento dos demais requisitos,

conforme as normas do Programa, a legislação pertinente e a regulamentação interna da UFU.

Nada mais havendo a tratar foram encerrados os trabalhos. Foi lavrada a presente ata que após lida e achada conforme foi assinada pela Banca Examinadora.



Documento assinado eletronicamente por **João Henrique de Souza Pereira, Professor(a) do Magistério Superior**, em 23/07/2024, às 15:27, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do [Decreto nº 8.539, de 8 de outubro de 2015](#).



Documento assinado eletronicamente por **Pedro Frosi Rosa, Usuário Externo**, em 23/07/2024, às 15:27, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do [Decreto nº 8.539, de 8 de outubro de 2015](#).



Documento assinado eletronicamente por **Luís Fernando Faina, Professor(a) do Magistério Superior**, em 23/07/2024, às 22:41, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do [Decreto nº 8.539, de 8 de outubro de 2015](#).



Documento assinado eletronicamente por **Romerson Deiny Oliveira, Usuário Externo**, em 25/07/2024, às 13:48, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do [Decreto nº 8.539, de 8 de outubro de 2015](#).



A autenticidade deste documento pode ser conferida no site https://www.sei.ufu.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **5550785** e o código CRC **70E66527**.

For the passionate learners and dedicated researchers who strive to make the world a better place through science.

Acknowledgements

I am immensely grateful to the Organization of American States and the Grupo Coimbra de Universidades Brasileiras for their generous support through the Brazil Scholarship PAEC OAS-GCUB program. Their financial assistance has played a crucial role in facilitating my research endeavors and academic pursuits.

Furthermore, I extend my deepest appreciation to my parents, friends, and girlfriend for their unwavering emotional support throughout this journey. Their encouragement, understanding, and belief in my abilities have been a constant source of motivation, driving me forward even in the face of challenges.

I am immensely grateful to Dr. Pedro Frosi Rosa for his invaluable guidance and mentorship. Additionally, I extend my heartfelt appreciation to Dr. João Henrique de Souza Pereira and Dr. Luis F. Faina for their unwavering support, expertise, and encouragement. Their insights have not only shaped my research trajectory but also contributed significantly to my personal growth and development.

Lastly, I extend my heartfelt thanks to the academic and research community at Universidade Federal de Uberlândia, as well as my colleagues and friends in the research group, for fostering a stimulating and supportive environment conducive to intellectual growth and collaboration.

“Hilbert space is a big place!”
(Prof. Carlton M. Caves)

Resumo

Computação quântica emergiu como um campo transformador, oferecendo capacidades sem precedentes que prometem revolucionar diversos domínios, como criptografia, codificação de dados e processamento de informações. Este estudo explora o potencial dos sistemas quânticos e seus fenômenos associados na geração de números pseudo-aleatórios, criptografia de imagem e representação de multigrafos.

Inicialmente, esta pesquisa propõe um algoritmo que aproveita as quasi-probabilidades inerentes à mecânica quântica para construir um Gerador de Números Pseudo-aleatórios (PRNG - *Pseudorandom Number Generator*) a partir de um sistema quântico. Através de testes rigorosos, é demonstrada a capacidade deste método em gerar sequências determinísticas de números pseudo-aleatórios, destacando a eficiência e imprevisibilidade dos números aleatórios gerados quanticamente.

Posteriormente, é desenvolvido e avaliado um algoritmo de criptografia e descryptografia baseado no PRNG, analisando sua robustez contra ataques comuns, como recorte, ruído e correlação. Os resultados enfatizam a segurança aumentada oferecida pelos esquemas de criptografia baseados em quântica, pavimentando o caminho para a transmissão e armazenamento de imagens fundamentalmente seguros.

Além disso, é introduzido um novo framework para representação de multigrafos dentro de sistemas quânticos. Esta abordagem inovadora permite a geração de multigrafos com densidade variável de vértices ou arestas, abrindo novos caminhos para a representação e processamento de dados.

Assim, este estudo contribui para o avanço da computação quântica ao demonstrar a eficácia dos sistemas quânticos na geração de números pseudo-aleatórios, melhorando a segurança na criptografia de imagem e oferecendo *insights* na representação de multigrafos dentro de contextos quânticos.

Palavras-chave: Quantum Computing, Pseudo-Random Number Generation, Image Encryption, Multigraph Representation, Quantum Information Processing..

Abstract

Quantum computing has emerged as a transformative field, offering unprecedented capabilities that promise to revolutionize various domains such as cryptography, data encoding, and information processing. This study explores the potential of quantum systems and their associated phenomena in generating pseudo-random numbers, image encryption, and multigraph representation.

Initially, this research proposes an algorithm leveraging quasi-probabilities inherent in quantum mechanics to construct a Pseudo Random Number Generator (PRNG) from a small quantum system. Through rigorous testing, the ability of this approach to generate deterministic sequences of pseudo-random numbers is demonstrated, highlighting the efficiency and unpredictability of quantum-generated random numbers.

Subsequently, an encryption and decryption algorithm based on the PRNG is developed and evaluated, analyzing its robustness against common attacks such as cropping, noise, and correlation. The findings underscore the heightened security offered by quantum-based encryption schemes, paving the way for fundamentally secure image transmission and storage.

Moreover, this research introduces a novel framework for representing multigraphs within quantum systems. This innovative approach allows the generation of graphs with varying vertex or edge label density, opening new avenues for data representation and processing.

Therefore, this study contributes to the advancement of quantum computing by demonstrating the efficacy of quantum systems in generating pseudo-random numbers, enhancing image encryption security, and offering insights into the representation of multigraphs within quantum contexts.

Keywords: Quantum Computing, Pseudo-Random Number Generation, Image Encryption, Multi graph Representation, Quantum Information Processing..

List of Figures

Figure 1 – Representation of qubit using Bloch sphere.	34
Figure 2 – Quantum circuit composed of 4 qubits and 16 quantum gates.	34
Figure 3 – Encryption process for an image.	37
Figure 4 – Decryption process for an image.	37
Figure 5 – Graph G with $V = \{1, 2, \dots, 8\}$ and $E = \{\{1, 2\}, \{2, 3\}, \dots, \{7, 8\}\}$. . .	39
Figure 6 – The line graph, $\vec{G}$, illustrates a particular set of adjacent edges in G . . .	40
Figure 7 – Quantum circuit with n qubits and a unitary operator that alters the system.	41
Figure 8 – Circuit based on three qubits.	41
Figure 9 – Directed graph for the circuit shown in Figure 8.	41
Figure 10 – This figure illustrates the entropy of True Random Number Generators (TRNGs) vs Pseudo Random Number Generators (PRNGs).	47
Figure 11 – Sequence of gates comprising the configuration of the quantum system.	49
Figure 12 – Representation of a qubit and its spin path through the Bloch sphere.	50
Figure 13 – Quantum circuit with n qubits.	53
Figure 14 – Multigraph illustrating transitions and vertices corresponding to the circuit shown in Figure 8.	53
Figure 15 – Basic quantum circuit based on four qubits and gates (CNOT and X).	56
Figure 16 – Multigraph constructed using the inputs and output from Table 1.	57
Figure 17 – Multigraph constructed using the inputs and output from Table 2.	58
Figure 18 – Multigraph constructed using the inputs and output from Table 3.	59
Figure 19 – General configuration of a quantum circuit with n qubits and the with image structure.	61
Figure 20 – IBM Quantum Composer platform.	64
Figure 21 – Selected images for experiments.	64
Figure 22 – Original images and their corresponding histograms.	68
Figure 23 – Encrypted images.	69
Figure 24 – Histograms for the red channel of encrypted images.	69

Figure 25 – Decrypted images.	69
Figure 26 – Encrypted images with various clipping operations and their decrypted counterparts.	70
Figure 27 – Cipher images and their decrypted counterparts under varying noise levels.	71
Figure 28 – Correlation distribution for Airplane image.	74
Figure 29 – Quantum circuit based on four qubits and gates (CNOT and X). . . .	75
Figure 30 – Multigraph generated using the quantum circuit shown in Figure 29. .	75
Figure 31 – Graph that contain the corresponding vertices and Multigraph Transition Label (MTL) for the encoded data.	77
Figure 32 – Illustration depicting the process of obtaining small pieces of raw data for encoding.	78
Figure 33 – Quantum circuits and their corresponding quasi-probabilities.	79
Figure 34 – Multigraph generated using the quantum more probable states shown in Figure 33d.	79

List of Tables

Table 1	– This table presents the expected results for determinate input after the execution of the circuit shown in Figure 15, where the input space is a subdivision $\mathcal{H} = \mathcal{H}^{\otimes 3} \otimes \mathcal{H}$	57
Table 2	– This table presents the expected results to determinate input after the execution of the circuit shown in Figure 15 over a subdivision $\mathcal{H} = \mathcal{H}^{\otimes 3} \otimes \mathcal{H}$	58
Table 3	– This table presents the expected results to determinate input after the execution of the circuit shown in Figure 15 over a subdivision $\mathcal{H} = \mathcal{H}^{\otimes 2} \otimes \mathcal{H}^{\otimes 2}$	59
Table 4	– Parameters for QQ-PRG algorithm configuration.	65
Table 5	– Outcome of the NIST SP800-22 randomness test suite for QQ-PRG. . .	66
Table 6	– Comparative analysis of quantum generator JUR01 and QQ-PRG. . . .	66
Table 7	– Parameters for QQ-PRG algorithm configuration.	67
Table 8	– Diagonal correlation distribution for Red, Green, Blue (RGB) images. .	72
Table 9	– Correlation distribution for grayscale Lena and Baboon images.	72
Table 10	– Preprocessing of raw data.	76
Table 11	– Summary of Data Encoding Process	76
Table 12	– The rotations needed to encode Lena image segments (Figure 21d), along with the top initial states, their occurrences, and grayscale binary matrix representation.	80

List of Algorithms

1	Algorithm for obtaining a set of probabilities.	51
2	Algorithm for encryption based in Exclusive OR (XOR) operation.	52
3	Algorithm for decryption based in XOR operation.	52
4	Algorithm for obtaining the graph line from a set of vertices.	60
5	Algorithm for obtaining the sequential set of vertices using a graph line. . .	61
6	Algorithm for selecting an adequate rotation for desired states.	62

List of Symbols

α	Complex coefficient associated with the state $ 0\rangle$ for a simple qubit.
$\bar{E}$	Mean of expected variables.
$\bar{O}$	Mean of observed variables.
β	Complex coefficient associated with the state $ 1\rangle$ for a simple qubit.
$\otimes$	Tensor operator
$\hbar$	Planck constant
$ 0\rangle$	Qubit on 0 state.
$ 1\rangle$	Qubit on 1 state.
$ \psi\rangle$	A quantum system.
$ i\rangle$	The possible state in a quantum system $ \psi\rangle$.
$\mathbb{V}$	vector space.
$\mathcal{H}^{\otimes n}$	Hilbert space for n qubits
$\mathcal{H}_T$	Hilbert space associated with MTL
$\mathcal{H}_V$	Hilbert space associated with elements in V
$ c_{ij} ^2$	Normalized quasi probability associated to the state $ ij\rangle$.
$\oplus$	XOR operation
θ	Angle, usually expressed as $\frac{\pi}{a}$, where $a \in \mathbb{R}$
Δ	Standard Deviation
φ	Phase Angle (in quantum mechanics)

$\vec{G}$	Graph Line
$\{u, v\}$	Edge in E
A_G	Adjacency matrix of graph G
c	Normalization Scale Parameter ($c = 10^{10}$)
c_i	A complex component associated to the state $ i\rangle$.
C_{OE}	Correlation between the observed and expected values.
$D(E)$	Variance of expected variables.
$D(O)$	Variance of observed variables.
$\dim(\mathcal{H})$	Number of dimensions of the Hilbert space.
E	Set of edges, $\{u, v\}, E \subseteq V \times V$
E_i	Expected variable.
f_s	Scale Factor
$G = (V, E)$	Graph
H	Hadamard Operator (Quantum gate for creating superposition)
$H(A)$	Entropy of the Sample Set
I	Quantum Identity Operator
M^S	Support of a matrix
$N(u)$	Neighborhood of vertex u
O_i	Observed variable.
$R(\theta)$	Quantum Rotation Operator with angle θ
S	Quantum Phase Operator (also known as $\sqrt{Z}$)
T	Quantum T Operator (also known as $\sqrt{S}$)
T	Set of Transitions
U	Unitary operator
u, v	Vertex in V
V	Set of vertices

$W(x, p)$ Wigner Function

X Quantum Pauli-X Operator

Y Quantum Pauli-Y Operator

Z Quantum Pauli-Z Operator

List of Abbreviations

CNOT Controlled-NOT

FFT Fast Fourier Transform

IBM International Business Machines

MTL Multigraph Transition Label

NIST National Institute of Standards and Technology

PRNG Pseudo-Random Number Generator

QQ-PRG Quantum Quasi-Probability Pseudo Random Generator

QD Quasi-probability Distributions

RGB Red, Green, Blue

TRNG True-Random Number Generator

XOR Exclusive OR

Contents

List of Figures	10
List of Tables	12
List of Algorithms	13
List of Symbols	16
List of Abbreviations	18
1 INTRODUCTION	22
1.1 Motivation	24
1.2 Research Objectives and Challenges	25
1.2.1 General Objective	25
1.2.2 Specific Objectives	25
1.3 Dissertation Organization	26
2 FUNDAMENTALS	27
2.1 Complex Vector Spaces	27
2.1.1 Basis and Dimension	28
2.1.2 Hilbert Space	28
2.1.2.1 Definition and Properties	28
2.1.2.2 Vectors and Basis	29
2.1.2.3 Inner Product	29
2.1.2.4 Linear Operators	29
2.1.2.5 Completeness and Orthonormal Basis	30
2.2 Quantum Computing Fundamentals	30
2.2.1 Quantum System	31
2.2.2 Qubits	31

2.2.3	Quantum Superposition	31
2.2.4	Quantum Gates	32
2.2.4.1	Hadamard Gate	32
2.2.4.2	X Gate	32
2.2.4.3	Y Gate	33
2.2.4.4	Z Gate	33
2.2.4.5	T Gate	33
2.2.4.6	Rotation Gate	33
2.2.5	Quantum Circuits	34
2.2.6	Quasi-probability Distributions (QDs)	35
2.2.6.1	Wigner Function	35
2.2.6.2	Calculation of Quasi-Probabilities	35
2.3	Data Encryption	35
2.3.1	Random Numbers	35
2.3.1.1	True Random Number Generation	36
2.3.1.2	Pseudo Random Number Generation	36
2.3.1.3	Assessment of Random Number Generation	36
2.3.2	Image Encryption and Decryption Schemes	36
2.3.3	Image Encryption and Decryption Evaluation	38
2.3.3.1	Histogram Analysis	38
2.3.3.2	Correlation between adjacent pixels	38
2.3.3.3	Cropping Attack Test	39
2.3.3.4	Noise Attack Test	39
2.4	Graph Theory	39
2.4.1	Line Graph	40
2.4.2	Support of a Matrix	40
2.4.3	Graph of a Matrix	40
2.4.4	Graph in Quantum Computers	40
3	PROPOSALS	42
3.1	Quantum Quasi-Probability Pseudo Random Generator	42
3.1.1	Theoretical Analysis of Probability Distribution	43
3.1.1.1	Uniform Probability Distribution	43
3.1.1.2	Nonuniform Probability Distribution	44
3.1.1.3	Analyzing Deviation Between Uniform and Nonuniform Probability Distri- butions	45
3.1.2	Proposed Steps for Pseudo-Random Number Generation	47
3.1.3	Proposed Algorithm	50
3.2	Image Encryption and Decryption Schemes	51
3.2.1	Normalization of quasi-probabilities values	51

3.2.2	Encryption Algorithm	52
3.2.3	Decryption Algorithm	52
3.3	Quantum Paradigm for Multigraph Analysis	52
3.3.1	Quantum Multigraph Preparation without Superposition	53
3.3.1.1	Step 1: Initialization of Simple States	53
3.3.1.2	Step 2: Multigraph as Quantum Unitary Operator	54
3.3.2	Quantum Multigraph Preparation with Superposition	54
3.3.2.1	Step 1: Initialization of Simple States	54
3.3.2.2	Step 2: Creating a System in Superposition	55
3.3.2.3	Step 3: Representing Multigraphs with a Quantum Circuit	55
3.3.2.4	Step 4: Wrapping Up the Multigraph Model	55
3.3.3	Insights into Multigraph Behavior	55
3.3.3.1	$\dim(\mathcal{H}_V) > \dim(\mathcal{H}_T)$	56
3.3.3.2	$\dim(\mathcal{H}_V) < \dim(\mathcal{H}_T)$	57
3.3.3.3	$\dim(\mathcal{H}_V) = \dim(\mathcal{H}_T)$	58
3.3.4	Data Encoding and Compression with Multigraph Structures.	59
3.3.4.1	Direct Encoding of Binary Data on Multigraphs	60
3.3.4.2	Enhanced Encoding with Superposition on Multigraphs	61
4	EXPERIMENTAL RESULTS	63
4.1	Setup for Experiments	63
4.2	Quantum Quasi-Probability Pseudo Random Generator (QQ-PRG)	65
4.2.1	Setup for Testing Algorithm QQ-PRG	65
4.2.2	Statistical test	65
4.3	Image Encryption and Decryption Tests	67
4.3.1	Setup for Testing Encryption and Decryption Algorithms	67
4.3.2	Encryption and Decryption Tests	67
4.3.2.1	Histogram analysis	68
4.3.2.2	Cropping attack test	69
4.3.2.3	Noise attack test	70
4.3.2.4	Correlation Analysis	71
4.4	Image Encoding Using Multigraph	73
4.4.1	Setup for Testing Image Encoding using Multigraph	73
4.4.1.1	Quantum Circuit	73
4.4.1.2	Hilbert Space Partitioning	73
4.4.2	Image Encoding Using Multigraph Encoding without Superposition	75
4.4.2.1	Preprocessing of Raw Data	76
4.4.2.2	Encoding Process	76
4.4.2.3	Decoding Process	77

4.4.3	Image Encoding Using Multigraph Encoding with Superposition	77
4.4.3.1	Preprocessing the Raw Data	77
4.4.3.2	Encoding the Data	78
4.4.3.3	Results for Image Encoding using Multigraphs	80
5	CONCLUSIONS	81
5.1	Contributions	81
5.2	Scientific Production	82
5.2.1	Journal Submissions	82
5.2.2	Workshop Presentations	83
5.3	Future Works	83
	REFERENCES	85
A	DATA AND IMAGE PROCESSING ALGORITHMS	90
A.1	Algorithms for General Data Processing	90
A.2	Algorithms for Image Processing	96
B	QUANTUM CIRCUITS IMPLEMENTATIONS	102
B.1	Quantum Circuits as PRNG	102
B.2	Multigraphs on Quantum Circuits	104
C	PROTOCOLS FOR PROPOSED ALGORITHMS	112
C.1	Protocol for Quantum Quasi-Probability Pseudo Random Gen- erator (QQ-PRG)	112
C.2	Protocol for Image Encryption and Decryption	113
C.3	Protocol for Data Encoding with Multi-Graph Structures . . .	115

Introduction

There are several technologies that have the potential to change the world - Augmented and virtual reality, 3D printing, internet of things, autonomous robotics, artificial intelligence, quantum computing, and others. Quantum technology offers several new features and has opened the door to a new era - the quantum age. This age began in the early 1980s, when Benioff and Feynman introduced the concept of quantum computing, which uses quantum mechanics to create a more effective computer. This theoretical machine utilizes qubits as the fundamental unit of information, making it a more advanced and versatile computing system compared to classical computers. The realization of quantum computers brings about the concept of quantum supremacy, which refers to the point where quantum computers can solve problems that are practically unsolvable for classical digital computers. Achieving quantum supremacy has profound implications, as it represents a significant leap in computational power and efficiency, potentially transforming industries and scientific research that rely on complex computations (CALEFFI et al., 2019; BAE et al., 2020; THAPLIYAL; MUÑOZ-COREAS; KHALUS, 2021).

Over the past few decades, several realizations of qubits, the cornerstone of quantum computers, have been presented. Among the most important ones are Josephson junctions, linear optics, quantum dots, and nuclear resonance. In addition, some theoretical realizations have also been proposed. For example, Le, Cykiert e Ginossar (2023) presents a scheme for scalable and robust quantum computations on a two-dimensional array of qubits whose main feature is fixed longitudinal coupling.

Based on these theoretical, physical, and technological advances, a few companies offer services with a small number of qubits to test quantum technology, such as International Business Machines (IBM) with IBM Quantum Condor, Intel with Horse Ridge II, Huawei with HiQ platform, Google with Sycamore, Microsoft with Azure Quantum service, Fujitsu with Digital Annealing Unit (DAU), Alibaba, Airbus, and others.

These new theoretical advances and the new hardware make it possible to achieve quantum supremacy. This concept was created to describe the disruptive impact of quantum computers in our world dominated by digital computers. There are some proposals

to achieve this goal, some of which were presented by (ARUTE et al., 2019; NEILL et al., 2018; ACAMPORA et al., 2023). These papers present an innovative set of computing techniques to ensure the fidelity of the data, entanglement duration, coherence, integration of some computing nodes and other important factors. In addition, some countries have made their efforts to achieve quantum supremacy public (Russia, US, China, EU, UK, Canada, Japan).

As a consequence of quantum computers, some counterparts of computer science areas were created. These new areas are quantum communications, quantum computing, quantum sensing, and quantum simulation.

Quantum sensing focuses on the development of new quantum devices (KUMAR et al., 2022a). Quantum simulation focuses on developing quantum representations of nature and creating new materials.

Quantum communication focuses on the development of a new generation of functional communication networks. To achieve this goal, researchers have to deal with quantum entanglement, quantum teleportation, necessary quantum hardware, methods to improve network efficiency, and other related issues. Quantum entanglement, for instance, requires the use of new methods, protocols, and hardware (YU et al., 2020; YIN et al., 2017; COUTINHO et al., 2022; CACCIAPUOTI et al., 2020). Furthermore, optimization is needed to ensure efficient implementation of quantum protocols and prevent noise (CHIN et al., 2021; ANSHU; JAIN, 2022; SHI et al., 2021).

Quantum computing aims to create a counterpart of traditional computing and incorporate new properties such as superposition, entanglement, squeezing and coherence into algorithms. Some important subareas include quantum circuits and software, quantum hardware, quantum information science and quantum machine learning. Some of these new methods, techniques and algorithms are being used to solve regular problems such as quantum pattern matching and offer innovative solutions to traditional computing problems (NIROULA; NAM, 2021; ZIDAN et al., 2023). Also are applied to solve complex real-world optimization problems (AJAGEKAR; YOU, 2019; SIGOV; RATKIN; IVANOV, 2022; KRENN; ERHARD; ZEILINGER, 2020).

Quantum information is intricately intertwined with the manipulation, encoding, process and safeguarding of data. In this dynamic field, the focus lies not only on the transmission and storage of information but also on ensuring its security through quantum encryption techniques. Two of the most interesting subjects are algorithms to generate pseudo random numbers applied to cryptography and underlying graph structures into quantum systems.

Generation of random numbers is one of the most important topic for data encryption. Random numbers hold significance in everyday applications, impacting various aspects of our lives (JACAK et al., 2021). There are two categories for generating random numbers: True-Random Number Generator (TRNG), utilizing physical sources of true randomness,

and Pseudo-Random Number Generator (PRNG), employing deterministic algorithms to generate a set of numbers.

TRNG are inherently unpredictable. These devices capture and convert these naturally occurring processes into random numbers, providing statistically independent and uniformly distributed numbers. This makes TRNGs ideal for applications requiring the highest level of randomness and security. On the other hand, PRNG are a class of algorithms used to generate sequences of numbers that appear random but are, in fact, generated deterministically. These generators rely on initial values, called seeds, and a series of mathematical operations to produce a sequence of numbers with statistical properties resembling truly random sequences.

There exist some traditional algorithms to generate random numbers (Yarrow 160, Fortuna, Blum Blum Shub, Counter-Based) (KUMAR; MISHRA, 2022). New algorithms for cryptography based on quantum principles, such as quantum walks, have been introduced (EL-LATIF; ABD-EL-ATTY; VENEGAS-ANDRACA, 2020; SU; WANG, 2022; HAO et al., 2023; EL-LATIF; ABD-EL-ATTY; VENEGAS-ANDRACA, 2020; ABD-EL-ATTY et al., 2021; GAO et al., 2022a). Algorithms for image encryption based on quantum walks were presented: on dimensional quantum walks perspective (SU; WANG, 2022) and two-dimensional quantum walks (HAO et al., 2023). Furthermore, new devices based on quantum technology, like QRNG JUR01 (JACAK et al., 2021), have been presented and commercialized.

Moreover, graph theory plays a significant role in information theory and computer science. It utilizes a mathematical framework to capture the relationships among its elements, facilitating the development of precise solutions for intricate challenges. The correspondence between a quantum circuit and a graph has paved the way for the formulation of more sophisticated algorithms. Quantum algorithms, different from classical counterparts, employ novel mechanisms for problem-solving (WEIGOLD et al., 2021). These new mechanisms are based on quantum mechanics and the phenomenon that arises from it.

Graphs in quantum computing are utilized for information encoding, agglomerative clustering and data representation. Exploration of graph properties and qdits further advances quantum computing (KASHEFI et al., 2009; PALER; FOWLER; WILLE, 2019; ILI; BAI, 2010; D'HONDT, 2009; ZHANG et al., 2023; GIUSTINA et al., 2024; ZHANG; ZHAO; WANG, 2013; YAN et al., 2021; EMMS et al., 2009; BHADURI et al., 2023; WANG et al., 2020).

1.1 Motivation

The importance of this research lies in its potential to advance our understanding and utilization of quantum systems, particularly through the exploration of quasi-probability

states. This study aims to uncover new applications and methodologies within quantum computing, which could significantly impact fields such as cryptography, data security, and computational theory.

Quasi-probability is a critical concept in quantum mechanics, representing the outcomes of complex quantum algorithms. It provides insights into the probabilistic nature of quantum states, which can be harnessed for various advanced applications. For example, the Quantum Walks algorithm demonstrates how the quasi-probability of states within a quantum system can be leveraged to enhance security measures. Several studies have highlighted the practical applications of this concept. SU; WANG (2022) introduced a controlled quantum walk-based scheme, GAO et al. (2022b) proposed a local substitution and global permutation scheme, and ABD-EL-ATTY et al. (2021) developed a privacy-preserving encryption mechanism to protect patient privacy. Additionally, EL-LATIF; ABD-EL-ATTY; VENEGAS-ANDRACA (2020) presented a new approach to PRNG.

This research aims to extend the current understanding of quasi-probability by integrating it into more complex computational frameworks, such as multigraphs. By conceptualizing quasi-probability as a link between quantum states, this study seeks to develop more sophisticated algorithms that utilize these structures. This approach could lead to the creation of highly secure encryption methods and other advanced computational techniques, thereby significantly enhancing the capabilities and applications of quantum computing.

1.2 Research Objectives and Challenges

This section outlines the general and specific objectives of the research.

1.2.1 General Objective

- ❑ To apply the concept of quasi-probability states in quantum systems for developing innovative approaches to pseudo-random number generation and image encryption, as well as advancing multigraph analysis.

1.2.2 Specific Objectives

1. Quantify the statistical properties of quasi-probability states and their effects on the randomness and unpredictability of pseudo-random number generation in quantum systems.
2. Develop algorithms for image encryption and decryption that leverage the distinct characteristics of pseudo-random number generators associated with quasi-probability states in quantum systems, with the explicit goal of enhancing security measures.

3. Investigate the potential advantages and limitations of utilizing quasi-probability states of quantum systems for image encryption, with a focus on robustness against attacks.
4. Explore the utility of quasi-probability states in quantum systems for the development and analysis of multigraphs, aiming to uncover patterns and correlations inaccessible through classical methods.

1.3 Dissertation Organization

The remainder of this document is organized as follows:

- ❑ Chapter 2 presents the fundamental concepts, including qubits, quantum gates, quantum circuits, the application of quantum circuits for deterministic processes, and graph theory associated with quantum computers.
- ❑ Chapter 3 introduces the proposals:
 1. A quantum circuit to generate pseudo-random numbers.
 2. A scheme for image encryption and decryption which employs quasi-probability values derived from a quantum system.
 3. Labeled multigraphs resulting from the division of Hilbert space over a quantum circuit.
- ❑ Chapter 4 presents empirical evidence for each proposal:
 1. Statistical testing using the National Institute of Standards and Technology (NIST) SP 800-22 to validate the proposed PRNG.
 2. Histogram analysis, cropping attack, noise attack, and correction between adjacent pixels to test the image encryption and decryption scheme.
 3. Image encoding results for a multigraph over quantum circuit without and with superposition.
- ❑ Finally, Chapter 5 presents the achieved conclusions, the contributions of the work, as well as some suggestions for future work.

Fundamentals

This chapter explores the fundamental concepts necessary for understanding quantum computing and the basic operations involved in quantum computation. It aims to provide concise yet comprehensive coverage, enabling readers to grasp not only the advancements done in this research, but also gain insight into the broader field of quantum computing.

2.1 Complex Vector Spaces

A complex vector space $\mathbb{V}$ is characterized by a nonempty set, along with three operations and a distinguished element called the zero vector (YANOFSKY; MANNUCCI, 2008). These operations are defined as follows:

- Addition: $+: \mathbb{V} \times \mathbb{V} \rightarrow \mathbb{V}$
- Negation: $-: \mathbb{V} \rightarrow \mathbb{V}$
- Scalar multiplication: $\cdot: \mathbb{C} \times \mathbb{V} \rightarrow \mathbb{V}$

These operations, along with zero, satisfy the following properties for all $V, W, X \in \mathbb{V}$ and for all $c, c_1, c_2 \in \mathbb{C}$ (YANOFSKY; MANNUCCI, 2008):

- Commutativity of addition: $V + W = W + V$
- Associativity of addition: $(V + W) + X = V + (W + X)$
- Zero is an additive identity: $V + 0 = V = 0 + V$
- Every vector has an inverse: $V + (-V) = 0 = (-V) + V$
- Scalar multiplication has a unit: $1 \cdot V = V$
- Scalar multiplication respects complex multiplication: $c_1 \cdot (c_2 \cdot V) = (c_1 \cdot c_2) \cdot V$
- Scalar multiplication distributes over addition: $c \cdot (V + W) = c \cdot V + c \cdot W$
- Scalar multiplication distributes over complex addition: $(c_1 + c_2) \cdot V = c_1 \cdot V + c_2 \cdot V$

2.1.1 Basis and Dimension

In the realm of quantum computing, three fundamental concepts play a vital role: linear combination, linear independence, and basis. These concepts provide a foundational understanding of qubits as mathematical structures and the fundamental mechanisms used in computation.

Let $\mathbb{V}$ be a complex (or real) vector space. A vector $V \in \mathbb{V}$ is considered a linear combination of particular vectors $V_0, V_1, \dots, V_{n-1}$ in $\mathbb{V}$ if V can be expressed as:

$$V = c_0 \cdot V_0 + c_1 \cdot V_1 + \dots + c_{n-1} \cdot V_{n-1} \quad (2.1)$$

for some coefficients $c_0, c_1, \dots, c_{n-1}$ in $\mathbb{C}(\mathbb{R})$, which denotes either the complex numbers $\mathbb{C}$ or the real numbers $\mathbb{R}$ (YANOFSKY; MANNUCCI, 2008).

A set $\{V_0, V_1, \dots, V_{n-1}\}$ of vectors in $\mathbb{V}$ is called linearly independent if

$$0 = c_0 \cdot V_0 + c_1 \cdot V_1 + \dots + c_{n-1} \cdot V_{n-1} \quad (2.2)$$

implies that $c_0 = c_1 = \dots = c_{n-1} = 0$. This means that the only way that a linear combination of the vectors can be the zero vector is if all the c_j are zero (YANOFSKY; MANNUCCI, 2008).

A set of vectors $\mathbb{B} = \{V_0, V_1, \dots, V_{n-1}\} \subseteq \mathbb{V}$ is called a basis of a (complex) vector space $\mathbb{V}$ if it satisfies the following conditions (YANOFSKY; MANNUCCI, 2008):

- Every vector $V \in \mathbb{V}$ can be expressed as a linear combination of vectors from $\mathbb{B}$.
- The set $\mathbb{B}$ is linearly independent.

2.1.2 Hilbert Space

Quantum theory is founded on two fundamental concepts: wave functions and operators. The state of a quantum system is represented by a wave function, while observables are described by operators (ESSIN; GRIFFITHS, 2006). These wave functions reside in a mathematical framework known as Hilbert space, which is a complete vector space endowed with an inner product. This structure allows for a rigorous definition of quantum states and observables, forming the backbone of quantum mechanics.

2.1.2.1 Definition and Properties

A Hilbert space $\mathcal{H}$ is a vector space over the complex numbers $\mathbb{C}$ equipped with an inner product $\langle \cdot | \cdot \rangle$ that satisfies the following properties:

- **Linearity:** For all $|\psi\rangle, |\phi\rangle, |\chi\rangle \in \mathcal{H}$ and all $\alpha, \beta \in \mathbb{C}$,

$$\langle \alpha\psi + \beta\phi | \chi \rangle = \alpha \langle \psi | \chi \rangle + \beta \langle \phi | \chi \rangle$$

□ **Conjugate Symmetry:** For all $|\psi\rangle, |\phi\rangle \in \mathcal{H}$,

$$\langle\psi|\phi\rangle = \overline{\langle\phi|\psi\rangle}$$

□ **Positive-Definiteness:** For all $|\psi\rangle \in \mathcal{H}$,

$$\langle\psi|\psi\rangle \geq 0 \quad \text{and} \quad \langle\psi|\psi\rangle = 0 \iff |\psi\rangle = 0$$

2.1.2.2 Vectors and Basis

In an n -dimensional Hilbert space, a quantum state is represented as a vector $|\psi\rangle$, which can be expressed as a linear combination of basis vectors. If $\{|e_i\rangle\}$ denotes an orthonormal basis for this space, any state $|\psi\rangle$ can be written as:

$$|\psi\rangle = \sum_{i=0}^n x_i |e_i\rangle \quad (2.3)$$

where x_i are complex coefficients. For simplicity, the state can also be represented as an n -tuple column vector:

$$|\psi\rangle = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix} \quad (2.4)$$

These coefficients x_i indicate the projection of $|\psi\rangle$ onto the basis vectors $|e_i\rangle$.

2.1.2.3 Inner Product

The inner product of two vectors $|\psi\rangle$ and $|\phi\rangle$ in Hilbert space, denoted $\langle\psi|\phi\rangle$, is a fundamental operation that yields a complex number. This inner product is defined as:

$$\langle\psi|\phi\rangle = \sum_{i=0}^n x_i^* y_i \quad (2.5)$$

where $|\psi\rangle = [x_0 \ x_1 \ \cdots \ x_n]^T$ and $|\phi\rangle = [y_0 \ y_1 \ \cdots \ y_n]^T$, with x_i^* being the complex conjugate of x_i . The inner product provides a measure of the "overlap" between two states and is instrumental in calculating probabilities and expectation values.

2.1.2.4 Linear Operators

Linear operators on Hilbert space, which represent quantum observables and transformations, can be described by matrices. An operator $\mathbf{T}$ acting on a state $|\psi\rangle$ transforms it to another state $|\psi'\rangle$:

$$|\psi'\rangle = \mathbf{T} |\psi\rangle \quad (2.6)$$

Expressed in matrix form, this becomes:

$$|\psi'\rangle = \begin{bmatrix} t_{00} & t_{01} & \cdots & t_{0n} \\ t_{10} & t_{11} & \cdots & t_{1n} \\ \vdots & \vdots & \ddots & \vdots \\ t_{n0} & t_{n1} & \cdots & t_{nn} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix} \quad (2.7)$$

Here, $\mathbf{T}$ is an $n \times n$ matrix with elements t_{ij} that are generally complex numbers. This matrix representation allows for the concise description of how operators transform quantum states. For example, the evolution of a quantum state over time can be described by a unitary operator, ensuring that the inner product (and hence the probability interpretation) is preserved.

2.1.2.5 Completeness and Orthonormal Basis

A Hilbert space is complete, meaning that every Cauchy sequence of vectors in the space converges to a vector within the space. This property is crucial for the mathematical consistency of quantum mechanics. Additionally, the orthonormal basis provides a convenient means to decompose and analyze quantum states. An orthonormal basis satisfies:

$$\langle e_i | e_j \rangle = \delta_{ij} \quad (2.8)$$

where δ_{ij} is the Kronecker delta, which is 1 if $i = j$ and 0 otherwise.

The completeness and orthonormality ensure that any vector in the Hilbert space can be uniquely expressed as a linear combination of the basis vectors, making Hilbert space an indispensable tool in both theoretical and applied quantum physics (ESSIN; GRIFFITHS, 2006).

2.2 Quantum Computing Fundamentals

Quantum computing is a rapidly evolving field that harnesses the principles of quantum mechanics to process information in fundamentally different ways from classical computing. At the core of quantum computing are quantum bits, or qubits, which can exist in superpositions of states, allowing them to represent both 0 and 1 simultaneously (YANOFSKY; MANNUCCI, 2008).

Quantum computers perform computations using quantum gates, which manipulate the quantum states of qubits. These gates include operations such as the Hadamard gate, Controlled-NOT (CNOT) gate and phase gate. Quantum algorithms exploit quantum phenomena such as superposition and entanglement to solve certain problems exponentially faster than classical algorithms. However, quantum computing is still in its early

stages, facing significant challenges such as qubit decoherence, errors in quantum operations and the development of scalable quantum hardware. Despite these challenges, quantum computing holds promise for revolutionizing fields such as cryptography, optimization and more (NIELSEN; CHUANG, 2001).

2.2.1 Quantum System

A quantum system can be described as a linear combination of singular states called basis states. Common bases include $|0\rangle$ and $|1\rangle$. A superposition state of the system is given by a linear combination of its basis states:

$$|\psi\rangle = \sum_{i=0}^{2^n-1} c_i |i\rangle \quad (2.9)$$

where, n is the number of qubits, c_i is a complex number, and $|i\rangle$ is a basis element.

2.2.2 Qubits

A qubit is the fundamental unit of information in a two-dimensional quantum system. Unlike classical bits, qubits can exist in multiple states simultaneously, including states represented as 0, 1, or a combination of both. This combination is typically represented as a linear combination of two orthogonal basis states, denoted as $|0\rangle$ and $|1\rangle$, with corresponding complex coefficients α and β which are complex numbers.

$$|\psi\rangle = \alpha |0\rangle + \beta |1\rangle \quad (2.10)$$

A qubit is represented as a 2-by-1 matrix with complex numbers.

$$\begin{bmatrix} \alpha \\ \beta \end{bmatrix}, \quad (2.11)$$

where the components α and β obey the restriction $|\alpha|^2 + |\beta|^2 = 1$.

2.2.3 Quantum Superposition

Quantum superposition stands as one of the most captivating phenomena in quantum computing. It allows for the simultaneous existence of multiple states within a system, granting access to a vast array of states within Hilbert space. A quantum computer possesses the remarkable property wherein the number of potential states increases exponentially with 2^n , where n denotes the number of qubits in superposition.

For example, a system composed of two qubits in superposition can be described as a linear combination:

$$|\psi\rangle = c_{00} |00\rangle + c_{01} |01\rangle + c_{10} |10\rangle + c_{11} |11\rangle \quad (2.12)$$

Here, the set of possible states is denoted by $S = \{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$. The coefficients c_{ij} , which are complex numbers, satisfy the condition $|c_{00}|^2 + |c_{01}|^2 + |c_{10}|^2 + |c_{11}|^2 = 1$, with $|c_{ij}|^2$ representing the probability of each respective state.

2.2.4 Quantum Gates

Typically represented by unitary matrices that allow us the manipulation of quantum state. There are several quantum gates used in quantum computing, and each one has an specific purpose and effect on the qubit's state. Some of the most commonly used are included in Equations 2.13.

$$I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}, Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, S = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}, T = \begin{bmatrix} 1 & 0 \\ 0 & e^{\frac{i\pi}{4}} \end{bmatrix} \quad (2.13)$$

2.2.4.1 Hadamard Gate

When applied to a single qubit, the Hadamard gate transforms the basis states $|0\rangle$ and $|1\rangle$ into a superposition state, allowing exploration of the Hilbert space.

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \quad (2.14)$$

The transformation of the Hadamard gate for a qubit in state $|0\rangle$ or $|1\rangle$ can be represented as follows:

$$H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \quad (2.15)$$

$$H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \quad (2.16)$$

This new state has an equal probability (50% each) of being measured in either the $|0\rangle$ state or the $|1\rangle$ state (YANOFSKY; MANNUCCI, 2008).

2.2.4.2 X Gate

The X gate is also known as the Pauli-X gate. It is a single-qubit quantum gate that performs a bit-flip operation. When applied to a qubit, it switches the amplitudes of the basis states $|0\rangle$ and $|1\rangle$. The X gate is represented as follows:

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

2.2.4.3 Y Gate

The Y gate is another member of the Pauli gates family. It is a single-qubit quantum gate that performs a bit-flip operation along with a phase-flip operation. When applied to a qubit, it switches the amplitudes of the basis states $|0\rangle$ and $|1\rangle$ and introduces a phase difference. The Y gate is represented as follows:

$$Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$$

The Y gate is essential for creating and manipulating superposition states and is an integral part of quantum circuits.

2.2.4.4 Z Gate

The Z gate, also known as the Pauli-Z gate, is a single-qubit quantum gate that performs a phase-flip operation. When applied to a qubit, it changes the sign of the amplitude of the $|1\rangle$ state while leaving the $|0\rangle$ state unchanged. The Z gate is represented as follows:

$$Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

2.2.4.5 T Gate

The T gate is a single-qubit quantum gate that introduces a $\pi/4$ phase shift to the state $|1\rangle$ without affecting the state $|0\rangle$. It is often used in quantum algorithms for quantum phase estimation and other quantum circuits. The T gate is represented as follows:

$$T = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix}$$

2.2.4.6 Rotation Gate

A rotation gate is a fundamental quantum gate that induces a rotation in the Bloch sphere representation of a qubit. It is often denoted as $R(\theta)$, where θ represents the angle of rotation in radians. The general form of a rotation gate can be expressed as:

$$R(\theta) = \begin{bmatrix} \cos\left(\frac{\theta}{2}\right) & -e^{i\varphi} \sin\left(\frac{\theta}{2}\right) \\ e^{i\varphi} \sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{bmatrix}$$

Here, φ is the phase angle associated with the rotation gate. The rotation gate allows for arbitrary rotations around the Bloch sphere, providing a versatile tool for quantum algorithms and quantum circuit design. See Figure 1.

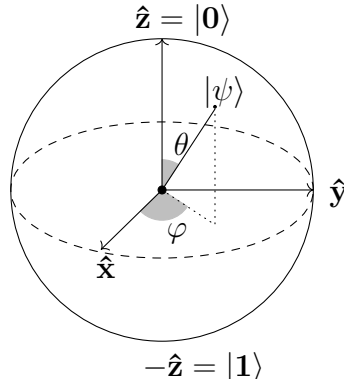


Figure 1 – Representation of qubit using Bloch sphere.

2.2.5 Quantum Circuits

A quantum circuit is a fundamental concept in quantum computing and quantum information processing. It serves as a framework for the manipulation of qubits and the execution of quantum operations. A quantum circuit comprises a collection of qubits and a specific sequence of quantum gates. These gates act on individual qubits, inducing transformations that impact the quantum system's quasi-probability distribution. Quantum circuits are pivotal tools for exploring and analyzing the complex Hilbert space of quantum states, making them essential for solving a wide range of problems in quantum computing.

The execution of a quantum circuit involves the systematic application of quantum gates in a prescribed order. As gates are applied sequentially, they modify the quantum state, allowing quantum computers to process information in ways that classical computers cannot. Quantum circuits are the foundation of quantum algorithms and are at the core of quantum supremacy demonstrations. Figure 2 shows a simple quantum circuit.

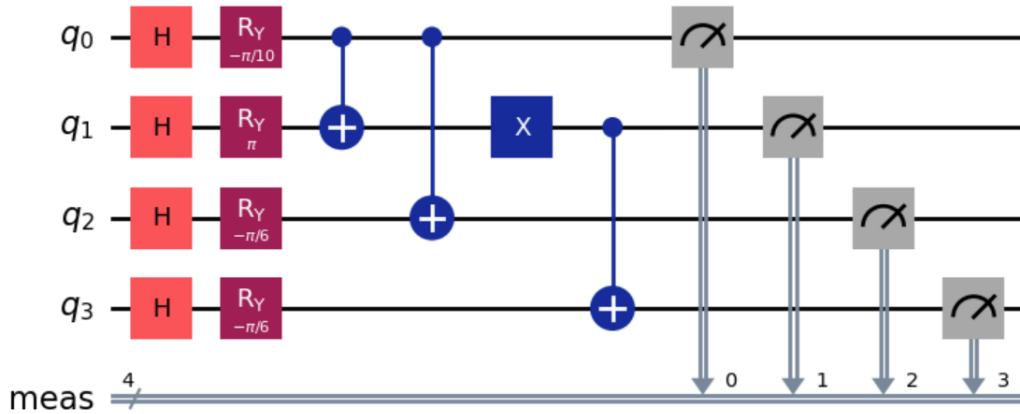


Figure 2 – Quantum circuit composed of 4 qubits and 16 quantum gates.

2.2.6 Quasi-probability Distributions (QDs)

Quasi-probability Distributions (QD) provide a powerful framework for describing the joint probability distribution of physical properties or measurements that cannot be simultaneously precisely known or predicted with arbitrary precision in quantum systems. One of the most renowned and widely used examples of QDs is the Wigner function, which associates the quantum state with a quasi-probability distribution (WIGNER, 1932; LEVY; LOSTAGLIO, 2020).

2.2.6.1 Wigner Function

The Wigner function, introduced by Eugene Wigner in 1932, is a fundamental QD in quantum mechanics. It provides a unique way to represent quantum states in phase space, using the position and momentum coordinates. Unlike classical probability distributions, the Wigner function can take on negative values, signifying the inherently non-classical nature of quantum mechanics. It essentially allows us to visualize the quantum state in a manner that retains some analogies to classical probability distributions while preserving the essential characteristics of quantum superposition and uncertainty.

The Wigner function is defined for a quantum state ρ as follows:

$$W(x, p) = \frac{1}{\pi\hbar} \int_{-\infty}^{\infty} \langle x - y | \rho | x + y \rangle e^{2ipy/\hbar} dy, \quad (2.17)$$

where x and p are position and momentum coordinates, respectively, and $\hbar$ is the reduced Planck constant.

2.2.6.2 Calculation of Quasi-Probabilities

The computation of quasi-probability distributions, including the Wigner function, often involves sampling techniques. These methods allow researchers to estimate the quasi-probability distribution associated with a quantum state by performing measurements or simulations.

2.3 Data Encryption

Data encryption is a fundamental aspect of cybersecurity, crucial for safeguarding sensitive information during transmission or storage. Random numbers play a pivotal role in encryption algorithms, providing the necessary unpredictability to generate cryptographic keys and create secure ciphertext.

2.3.1 Random Numbers

Random numbers are essential in various computing domains, particularly in cryptography and simulations where unpredictability is critical. There are two primary methods

for generating random numbers: True Random Number Generation (TRNG) and Pseudo Random Number Generation (PRNG).

2.3.1.1 True Random Number Generation

TRNG aims to produce numbers that are genuinely unpredictable, even given complete knowledge of the system. In classical computing, this often involves utilizing physical phenomena like thermal noise or radioactive decay. In the quantum realm, the intrinsic randomness of quantum mechanics is leveraged.

Quantum TRNG can be achieved through various means, such as measuring the spin of a single photon or detecting the outcome of a quantum process. For instance, the unpredictable nature of quantum superpositions and entanglement allows for the creation of random bits with high entropy.

2.3.1.2 Pseudo Random Number Generation

PRNG is a fundamental component of many cryptographic and security systems. It plays a crucial role in generating keys, initialization vectors, and other random parameters used in encryption and decryption processes. In the next subsection, we delve into the principles and methods of pseudo-random number generation, with a focus on their relevance to image encryption and quantum computing.

2.3.1.3 Assessment of Random Number Generation

Assessing the reliability and randomness of generated random numbers is paramount for their cryptographic utility. Leveraging the comprehensive NIST SP 800-22 suite, which encompasses 15 rigorous tests tailored for such evaluations, it is meticulously scrutinize the statistical properties of the generated sequence. Through a battery of tests including frequency, runs, and correlation analyses, it is ensured adherence to expected statistical norms, verifying uniformity and absence of discernible patterns.

2.3.2 Image Encryption and Decryption Schemes

Image encryption is essential for safeguarding visual data against unauthorized access or tampering. Utilizing PRNG in conjunction with cryptographic techniques, such as the XOR operation, offers a robust method for encrypting images while maintaining efficiency and security.

The XOR operation is a binary operation that compares corresponding bits of two operands. When applied in image encryption, each pixel of the image is XORed with a pseudo-random bit stream generated by a PRNG. This process scrambles the pixel values, introducing randomness and obscuring the original image content.

The encryption process based on XOR operation and PRNG typically involves the following steps:

- ❑ **Key Generation:** A secret cryptographic key is generated using a cryptographically secure PRNG. This key serves as the basis for generating the pseudo-random bit stream required for the XOR operation.
- ❑ **Pseudo-Random Bit Stream Generation:** The PRNG is initialized with the secret key, and a pseudo-random bit stream is generated. This bit stream should be as long as the image data to ensure every pixel is encrypted.
- ❑ **XOR Encryption:** Each pixel of the image is XORed with the corresponding bit from the pseudo-random bit stream. This bitwise operation alters the pixel values in a manner that appears random but can be reversed with the same key and bit stream. The encryption process is illustrated in Figure 3.
- ❑ **Decryption:** To decrypt the encrypted image, the same key is used to regenerate the pseudo-random bit stream. Each encrypted pixel is then XORed with the corresponding bit from the regenerated bit stream, effectively reversing the encryption process and restoring the original image. The decryption process is depicted in Figure 4.

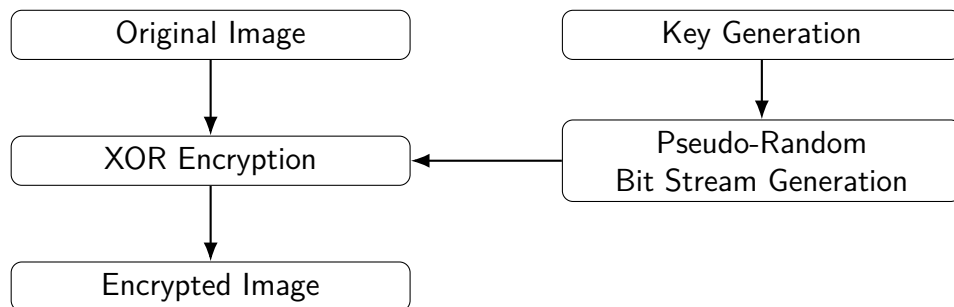


Figure 3 – Encryption process for an image.

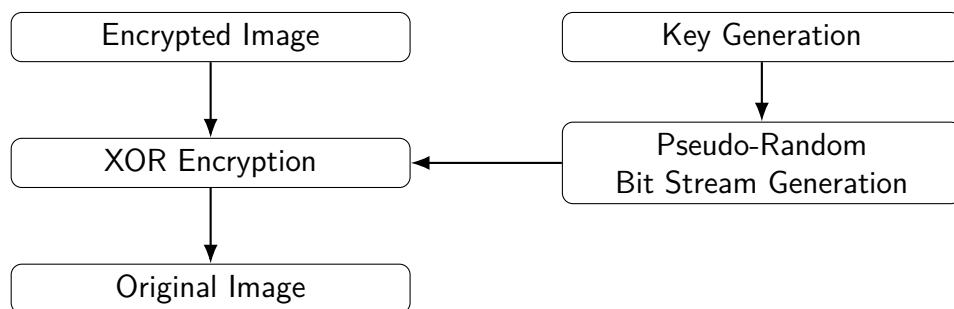


Figure 4 – Decryption process for an image.

2.3.3 Image Encryption and Decryption Evaluation

This subsection delves into the core methods for assessing the effectiveness and robustness of image encryption and decryption. It is utilized histogram analysis to ensure that encrypted images have a uniform distribution of pixel intensities, which obscures original patterns. Correlation analysis of adjacent pixels quantifies how well the encryption disrupts natural pixel relationships. Additionally, cropping and noise attack tests simulate real-world scenarios of data loss and interference, respectively, to evaluate the encryption's resilience. These evaluations are crucial for ensuring that the encryption scheme can maintain image integrity and security in various conditions.

2.3.3.1 Histogram Analysis

Histogram analysis is used to evaluate the effectiveness of image encryption by comparing the histograms of the original, encrypted, and decrypted images. A histogram represents the distribution of pixel intensities in an image. In a well-encrypted image, the histogram should appear uniform, indicating that pixel values are evenly distributed, thereby obfuscating the original image structure. This randomness ensures that the encrypted image does not reveal any patterns from the original image.

2.3.3.2 Correlation between adjacent pixels

In a regular image, each pixel exhibits a high correlation with its neighboring pixels. Correlation analysis is used to estimate the relationship between a particular pixel and its surrounding pixels. This analysis is critical for understanding how encryption affects the natural structure of an image. Equation 2.18 provides a method to quantify this correlation, yielding values close to 1 for data with high correlation and values near zero for data with low correlation.

$$C_{OE} = \frac{\sum_{i=1}^N (O_i - \bar{O})(E_i - \bar{E})}{\sqrt{D(O)D(E)}}. \quad (2.18)$$

In this equation, C_{OE} represents the correlation coefficient between two variables, O and E , which can be thought of as the observed and expected values, respectively. The summation term considers all N pairs of corresponding pixels in the observed (O_i) and expected (E_i) datasets. The terms $\bar{O}$ and $\bar{E}$ denote the mean values of the observed and expected datasets, respectively, while $D(O)$ and $D(E)$ represent the variances of the observed and expected data. The resulting correlation coefficient provides insights into the degree of similarity or dissimilarity between the two datasets.

High correlation values suggest that the pixel values in the observed data closely match the expected values, indicating that the natural structure of the image remains intact. On the other hand, low correlation values indicate a substantial change in the pixel

relationships, typical in encrypted images where disrupting these natural correlations is intended to increase security.

2.3.3.3 Cropping Attact Test

To evaluate the robustness of the encrypted images against partial data loss, it was conducted a cropping attack test. This test involves removing specific portions of the encrypted image to simulate data loss scenarios and then attempting to decrypt the partially corrupted images. By analyzing the decrypted images, it assess how well the encryption method handles such attacks and the extent to which the original image can be reconstructed.

2.3.3.4 Noise Attact Test

To evaluate the resilience of the encrypted images against potential information degradation during transmission and preservation, it was conducted a noise attack test. This test involved deliberately introducing salt and pepper noise to simulate random pixel corruption, akin to interference commonly encountered in data transmission or storage processes.

2.4 Graph Theory

A graph G is formally defined as a pair $G = (V, E)$, where V is the set of vertices and $E \subseteq V \times V$ represents the set of edges. Each edge e in E is an unordered pair $\{u, v\}$, indicating that it connects vertices u and v . This is denoted as $u \sim v$. The neighborhood of a vertex u , denoted $N(u)$, is the set of vertices adjacent to u (BOLLOBÁS, 2013; DIESTEL; DIESTEL, 2017). Figure 5 illustrates an example of a graph.

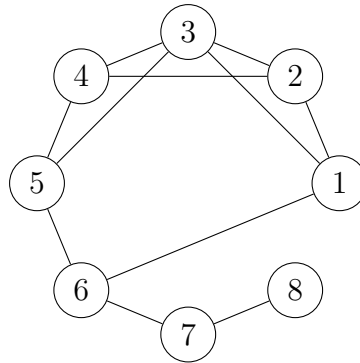


Figure 5 – Graph G with $V = \{1, 2, \dots, 8\}$ and $E = \{\{1, 2\}, \{2, 3\}, \dots, \{7, 8\}\}$.

The adjacency matrix A_G of a graph $G = (V, E)$, with $|V| = n$, is defined as:

$$A_G(u, v) = \begin{cases} 1 & \text{if } (u, v) \in E \\ 0 & \text{otherwise.} \end{cases}$$

A multigraph $G = (V, E)$ allows multiple edges between the same pair of nodes.

2.4.1 Line Graph

The line graph $\vec{G}$ of a multigraph $G = (V, E)$ is a graph defined as $\vec{G} = (\vec{V}, \vec{E})$, where $\vec{V} = E$ and $\vec{E} = \{(u, v) : u, v \in E \text{ and } u \text{ and } v \text{ are adjacent in } G\}$. In the line graph $\vec{G}$, each vertex represents an edge of G , and two vertices are connected by an edge if and only if their corresponding edges in G share a common endpoint. Figure 6 presents a visual representation representation.

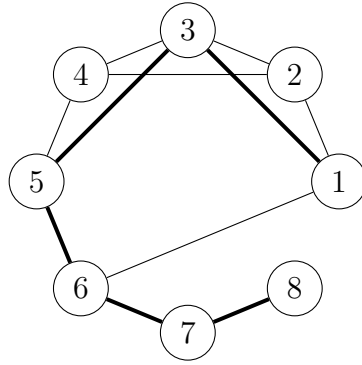


Figure 6 – The line graph, $\vec{G}$, illustrates a particular set of adjacent edges in G .

2.4.2 Support of a Matrix

Given a matrix M , its support M^S is a $(0 - 1)$ -matrix defined such that $M^{S_{i,j}} = 1$ if and only if $M_{i,j} \neq 0$. The support of a matrix establishes a connection between the matrix and a graph.

2.4.3 Graph of a Matrix

The graph associated with a matrix M is defined as the graph whose adjacency matrix corresponds to the support matrix M^S .

2.4.4 Graph in Quantum Computers

Graphs are typically defined as $G = (V, E)$, where $V = \{0, 1, 2, \dots, n - 1\}$ represents the set of vertex labels, and $E = \{\{i, j\} : i, j \in V\}$ represents the set of unweighted edges. In the context of quantum computing, graph vertices reside in the Hilbert space $\mathcal{H}^{\otimes n}$.

A common method for constructing a multigraph involves a unitary operator U and the set of possible states of the system. Figure 7 presents a general perspective, which takes u as input and returns v after a transformation. Here, the transition set E emerges as a consequence of the regular operation of the unitary operator.

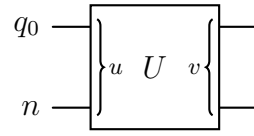


Figure 7 – Quantum circuit with n qubits and a unitary operator that alters the system.

For clarity, consider Figure 8, which presents a simple circuit composed of three qubits. It takes as inputs the vertices $V = \{000, 001, 010, \dots, 111\}$. As a result of executing the circuit, the set of edges E can be constructed.

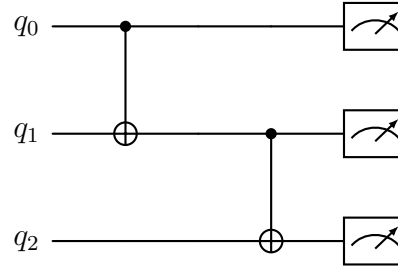


Figure 8 – Circuit based on three qubits.

Figure 9 corresponds to the underlying multigraph depicted in Figure 8, illustrating all possible states and their transitions. Utilizing a superposition state allows the multigraph to represent any of the possible transitions from u to v depending on the configuration of the qubits.

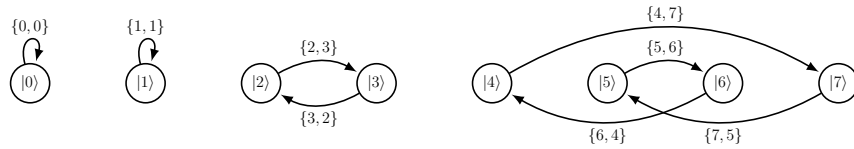


Figure 9 – Directed graph for the circuit shown in Figure 8.

Proposals

This chapter explores three proposals of this research to leverage quasi-probability over a quantum system in superposition. Each proposal delves into mechanisms for obtaining quasi-probability tailored for specific applications.

Section 3.1 provides an in-depth exploration of quasi-probability in quantum systems in superposition and their utilization as PRNG. The suggested algorithm Quantum Quasi-Probability Pseudo Random Generator (QQ-PRG) employs a configuration that is both efficient and effective to accomplish this goal.

Section 3.2 introduces schemes for image encryption and decryption that leverage a PRNG as a fundamental component, thereby offering enhanced security measures.

Finally, Section 3.3 presents a novel perspective on quantum circuits through the interpretation of multi-transition labels for multigraphs generated. This approach utilizes quasi-probability to handle vertices and edges within a multigraph, opening new avenues for analysis and understanding.

3.1 Quantum Quasi-Probability Pseudo Random Generator

This method harnesses the principles of quantum mechanics, utilizing quantum circuits, sequential execution, and quasi-probabilities to generate sequences of numbers with random properties.

Based on a quantum system, a set of Hadamard gates, and a set of rotation gates, the proposed method generates a set of binary numbers with pseudo random properties. The system consists of a small set of qubits initialized in the $|0\rangle$ state. The inclusion of Hadamard gates allows the algorithm to leverage superposition mechanisms to explore the Hilbert Space effectively. Additionally, a sequence of rotation gates induce changes that modify the probabilities of the resulting states.

The uniqueness of this algorithm lies in its utilization of a small number of qubits and a sequence of rotational gates. It stands out because it only requires at least 3 qubits to generate pseudo-random sequences of numbers.

Due to its characteristics, this method holds potential for communication and data encryption. However, the main limitation of the system lies in the time required to calculate the set of pseudo-random numbers. Each set, consisting of 2^n elements, relies on a quantum system and rotation gates for computation, leading to significant computational overhead.

3.1.1 Theoretical Analysis of Probability Distribution

This subsection analyzes the entropy of two types of theoretical quantum systems: those with uniformly and non-uniformly distributed probabilities. In both cases, the probability distribution is partitioned among the quantum states, ensuring that $\sum_{i=0}^{2^n-1} P(a_i) = 1$, where $a_i \in [0, 2^n - 1]$ represents a particular state, $P(a_i)$ is the associated probability, and n denotes the number of qubits.

In an ideal situation, when the probability is evenly distributed across all states of a quantum system, it reflects a state of complete uncertainty. This means that each possible outcome has an equal chance of occurring, resembling the randomness observed in a purely random system. This concept of uniform probability distribution across quantum states has been thoroughly examined and tested by Kumar et al. in their research (2022) (KUMAR et al., 2022b).

On the other hand, the opposite scenario occurs when the probability distribution across the quantum states is not uniform. This typically happens when specific operations are applied to the quantum system, altering its state and consequently leading to a non-uniform distribution of probabilities among its possible outcomes. In essence, this means that certain outcomes become more likely than others due to the intentional manipulation of the quantum system.

By examining how probability is distributed across the quantum states in these two scenarios, researchers aim to understand the underlying principles that govern randomness within quantum systems. This understanding is crucial for designing efficient and reliable pseudo-random number generators that leverage the unique properties of quantum mechanics.

3.1.1.1 Uniform Probability Distribution

A uniform probability distribution, often referred to simply as a uniform distribution, is a type of probability distribution where every possible outcome has an equal probability of occurring. In other words, if you were to represent the probabilities graphically, it would result in a flat, constant line. This means that all values within a certain range are equally likely to occur.

Quantum computers have the potential to achieve a uniform probability distribution through the utilization of a fundamental quantum gate known as the Hadamard gate.

This gate is pivotal in generating a superposition state, wherein there is an equal probability (50%) of measuring the quantum state as either ‘0’ or ‘1’. This intrinsic property forms the basis of the proposal outlined by Kumar et al. (2022b). Their work establishes that by leveraging multiple Hadamard gates, it becomes feasible to generate random sequences of numbers. This capability stems from the unique ability of quantum systems to explore and exploit quantum superposition, enabling the creation of probabilistic outcomes that exhibit uniformity.

Additionally, Shannon’s entropy serves as a metric to quantify the uncertainty or randomness inherent in a probability distribution Vajapeyam (2014). In the context of a uniform probability distribution, entropy is defined as

$$H(A) = H(P(a_1), P(a_2), \dots, P(a_{2^n})) = - \sum_{i=0}^{2^n-1} P(a_i) \log_2 P(a_i). \quad (3.1)$$

In the case of a truly random process, the probability values are identical for every element, indicating maximal uncertainty. A uniform distribution within a quantum system, consisting of 2^n elements observed over N shots, can be represented as

$$H(A) = \sum_{i=0}^{2^n-1} H\left(\frac{1}{2^n}\right), \quad (3.2)$$

where $P(a_i) = \frac{1}{2^n}$, $a_i = |i\rangle, i \in 1, \dots, 2^n$. This representation is utilized throughout the document, with some modifications and constraints to effectively depict nonuniform probability distributions.

3.1.1.2 Nonuniform Probability Distribution

A nonuniform probability distribution depicts a situation in which the likelihood of different outcomes is not uniform or equal. Unlike a uniform distribution, where each outcome has an identical probability of occurring, a nonuniform distribution assigns differing probabilities to distinct outcomes. This discrepancy in probabilities can stem from various factors, including inherent biases within the system, disparate conditions influencing the outcomes, or specific characteristics unique to the scenario under examination.

In the realm of quantum systems, nonuniform probability distributions manifest as alterations to the initially uniform probability distribution. This signifies that the probability associated with each quantum state can be adjusted through the application of quantum gates. These gates have the capability to manipulate the probabilities, potentially increasing or decreasing them relative to the mean probability. Consequently, the resulting distribution reflects a deviation from uniformity, exhibiting varied probabilities across different quantum states.

A representation for a nonuniform distribution derived of a transformation of a quantum system can be expressed as:

$$1 = \sum_{i=0}^{2^n-1} \frac{w_i}{2^n}, \quad (3.3)$$

where, w_i represents the weight associated with the i -th outcome, and 2^n denotes the total number of possible outcomes. The sum over i ensures that the sum of all probabilities equals 1, adhering to the normalization condition required for a valid probability distribution. Additionally, the inequality $w_i \geq 0$ ensures that each weight is non-negative.

The entropy of the nonuniform probability distribution, denoted as $H'(A)$, is defined as:

$$H'(A) = H\left(\frac{w_1}{2^n}, \frac{w_2}{2^n}, \dots, \frac{w_{2^n}}{2^n}\right) = - \sum_{i=0}^{2^n-1} \frac{w_i}{2^n} \log_2 \frac{w_i}{2^n}, \quad (3.4)$$

This equation calculates the uncertainty or randomness inherent in the nonuniform distribution. It sums over all outcomes, with each outcome's contribution weighted by its probability and the logarithm of its probability. The primary constraint $\sum_{i=1}^{2^n} \frac{w_i}{2^n} = 1$ ensures that the probabilities of all outcomes sum up to 1, fulfilling the normalization requirement for a valid probability distribution.

3.1.1.3 Analyzing Deviation Between Uniform and Nonuniform Probability Distributions

This subsection delves into the quantification of the deviation between a nonuniform distribution, denoted as $\Delta H'(A)$, and a uniform probability distribution, represented by $H(A)$. Both quantum systems comprise an identical number of qubits and states, yet the nonuniform system underwent alterations through a sequence of quantum gates.

The deviation, expressed as $\Delta H'(A)$, is calculated as the difference between the entropies of the uniform and nonuniform distributions:

$$\Delta H'(A) = H(A) - H'(A). \quad (3.5)$$

This equation measures the extent to which the nonuniform distribution diverges from uniformity. By subtracting the entropy of the nonuniform distribution, $H'(A)$, from that of the uniform distribution, $H(A)$, this approach provides a clearer understanding of the deviation's extent.

The entropy for a uniform distribution (3.6) is given by:

$$H(A) = -2^n \log_2 \left(\frac{1}{2^n}\right) = n. \quad (3.6)$$

The entropy for a non-uniform distribution under the restriction $\sum_{i=0}^{2^n-1} \frac{w_i}{2^n} = 1$ is expressed as:

$$H'(A) = - \sum_{i=0}^{2^n-1} \left(\frac{w_i}{2^n}\right) \log_2 \left(\frac{w_i}{2^n}\right), \quad (3.7)$$

Further simplification:

$$H'(A) = (n) \sum_{i=0}^{2^n-1} \frac{w_i}{2^n} - \sum_{i=0}^{2^n-1} \frac{w_i}{2^n} (\log_2 w_i), \quad (3.8)$$

Simplifying this expression:

$$H'(A) = n - \sum_{i=0}^{2^n-1} \frac{w_i}{2^n} \log_2 w_i. \quad (3.9)$$

Finally, the deviation of $H'(A)$ in relation to $H(A)$ is defined as:

$$\Delta H'(A) = \sum_{i=0}^{2^n-1} \frac{w_i \log_2 w_i}{2^n}. \quad (3.10)$$

The deviation between a nonuniformly distributed probability and a uniformly distributed probability yields valuable insights into the characteristics of the probability distribution within a quantum system:

- The maximum entropy achievable by a uniform distribution, denoted as $H(A)$, equals the number of elements in the quantum system, represented by n . This serves as a reference point for understanding the degree of randomness or uncertainty inherent in the system.
- In the absence of deviation ($\Delta H'(A) = 0$), which occurs only in the trivial scenario where all w_i are equal to 1 ($i = 1, \dots, 2^n$), the entropy of the nonuniform distribution, $H'(A)$, equals the maximum entropy $H(A)$. However, in non-trivial cases where probabilities w_i are not all equal to 1, it follows that $0 \leq H'(A) < H(A)$. This inequality signifies that the entropy of the nonuniform distribution is less than the maximum entropy achievable by a uniform distribution.
- The magnitude of $\Delta H'(A)$ is directly proportional to the deviation of each w_i from unity. Smaller deviations result in a smaller difference between $H(A)$ and $H'(A)$, indicating that the distribution approaches uniformity. Conversely, larger deviations lead to a greater discrepancy between the entropies, signifying a more pronounced departure from randomness.

Moreover, Figure 10 provides a visual representation of the two fundamental methodologies for generating random numbers. The blue line denotes the entropy profile of a genuine random generator, which relies on a defined number of qubits within a quantum system. In contrast, the red area signifies the potential entropy range attainable by a pseudo-random number generator (PRNG) implemented within a quantum framework. This intuitive visualization effectively contrasts the inherent randomness of true random generators with the controlled randomness of PRNGs in quantum systems. It underscores the vast spectrum of opportunities for harnessing quantum technology to develop sophisticated PRNGs capable of catering to diverse applications.

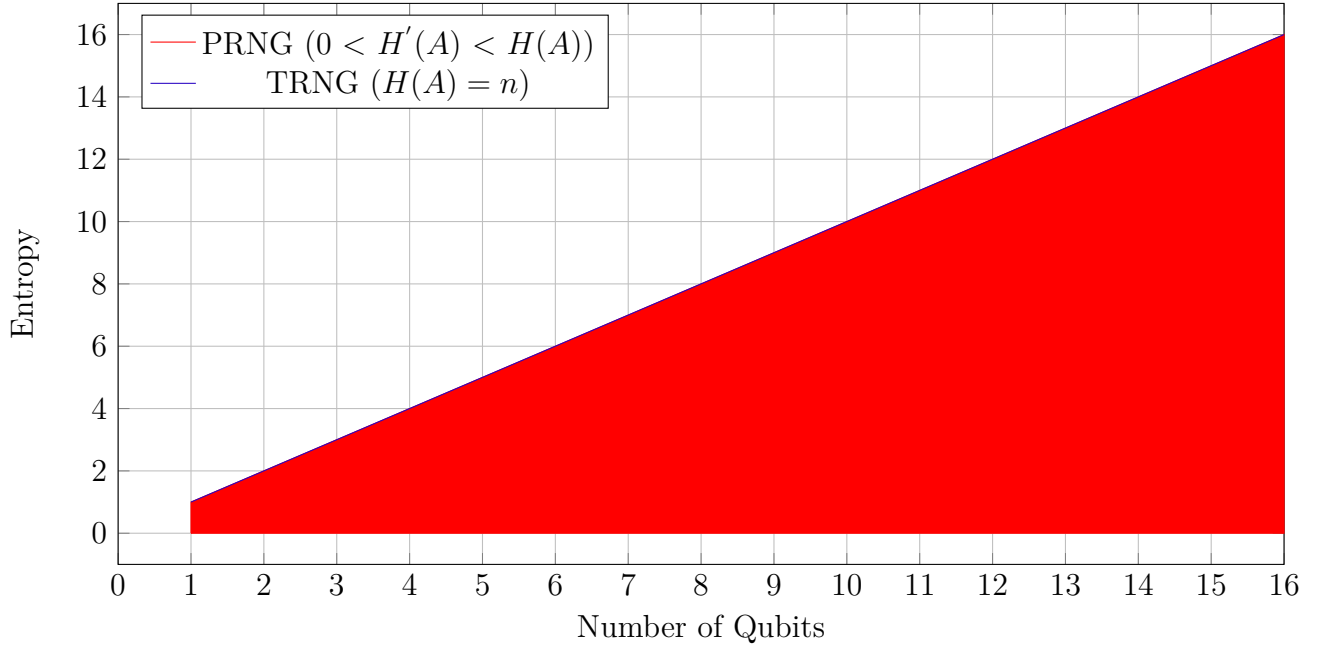


Figure 10 – This figure illustrates the entropy of True Random Number Generators (TRNGs) vs Pseudo Random Number Generators (PRNGs).

3.1.2 Proposed Steps for Pseudo-Random Number Generation

This proposal delineates a sequential process comprised of several distinct steps aimed at generating pseudo random numbers. Each step contributes to the generation of a large sequence of numbers by creating, modifying, and measuring quantum systems.

Firstly, the algorithm calculates the number of quantum systems based on known configuration parameters, number of qubits, and the desired length of the list of numbers.

Secondly, the algorithm engages in an iterative process where quantum systems are created, modified, and measured. This iterative procedure is pivotal in generating pseudo-randomness and ensures that the resulting numbers exhibit desirable statistical properties.

Finally, the algorithm transforms the quantum system measurements into the corresponding pseudo random numbers.

Step 1. Determining the Number of Required Quantum Systems

The number of required quantum systems, denoted as k , to generate a sequence of $L \geq 0$ elements is calculated using the formula:

$$k = \frac{L}{2^n}, \quad (3.11)$$

where n represents the number of qubits utilized in the quantum system.

Step 2. Quantum System

This proposal requires a quantum system composed of $n = 3$ qubits. This configuration is based on the study carried out by Jozwiak et al. which extensively delves into the intricacies of quantum entanglement for random number generation (JÓŻWIAK; JACAK; JACAK, 2024). This approach illustrates the potential for TRNG to be realized using a quantum computer. This study suggest using 3 qubits as minimum configuration for random number generators based on quantum systems.

$$\mathcal{H} = \mathcal{H}^{\otimes n} = \mathcal{H}_1 \otimes \mathcal{H}_2 \otimes \cdots \otimes \mathcal{H}_n, \quad (3.12)$$

which is fundamental to our exploration. Here, $\mathcal{H}_i$ represents the Hilbert space associated with the i th qubit, and the tensor product symbolizes the combination of these individual spaces into a unified quantum entity.

The system is initialized in the $|0\rangle$ state. However, it is not a constraint, as the system admits any state for its initialization.

$$|\psi(0)\rangle = |0\rangle^{\otimes n}. \quad (3.13)$$

Sequentially, to obtain a superposed system, a set of Hadamard gates are applied:

$$|\psi(0)'\rangle = H^{\otimes n} |\psi(0)\rangle. \quad (3.14)$$

A critical component is the definition of PRNG is the unitary operator. This operator functions as a deterministic process, harnessing the principles of quantum mechanics to induce changes within the quantum system. Mathematically, it is expressed as follows:

$$U = S(R_j^{\otimes n}), \quad (3.15)$$

Here, $\mathcal{R}_j(\theta)$ represents a rotation around a specific axis of the qubit j , with $\theta \in [0, \frac{\pi}{2}]$, while S denotes the shift operator. The shift operator, denoted as S , introduces additional complexity to the deterministic process. It allows us to change dynamically the quantum state (WANG; MANOUCHEHRI, 2013). However, in the context of this proposal, the shift operator simplifies the identity matrix. The shift operator is formally defined as

$$S = \sum_{i=0}^n |i\rangle \langle i|. \quad (3.16)$$

Simplifying the unitary operator by configuring the shift operator as an identity matrix enables the definition of a deterministic process primarily based on rotations. This simplified unitary operator is expressed as:

$$U = I(\mathcal{R}_j(\theta)^{\otimes n}). \quad (3.17)$$

The evolution of a quantum system occurs discretely, advancing through distinct time steps under the influence of a specified unitary operator denoted as U . At each time step t , the unitary operator U acts on the quantum state $|\psi(t)\rangle$. Mathematically, the quantum state at time t , denoted as $|\psi(t)\rangle$, is derived by applying the unitary operator U t times to the initial state $|\psi(0)\rangle$, represented by the equation $|\psi(t)\rangle = U^t |\psi(0)\rangle$. Figure 11 presents the circuit for this process.

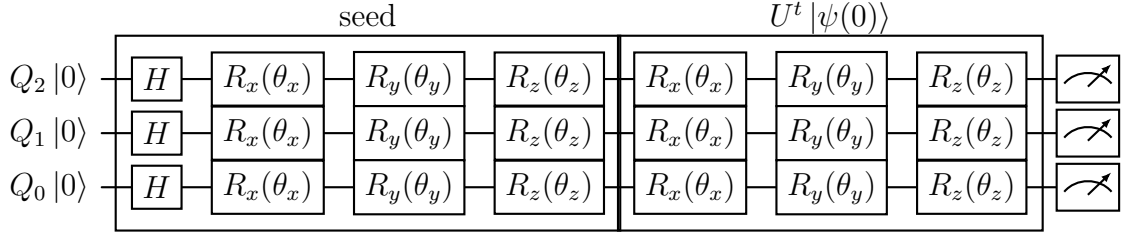


Figure 11 – Sequence of gates comprising the configuration of the quantum system.

Impact of Additional Qubits on Randomness

Additionally, it is necessary to remark that even though the proposed quantum system can accommodate more than 3 qubits, the addition of qubits increases the number of possible quantum states. Specifically, the state space grows exponentially with the number of qubits. For a system with n qubits, there are 2^n possible states. Consequently, the associated quasi-probability is divided among these states.

The main consequence of this is that the unitary operator, U , makes the probability values for some states close to zero as the number of states increases. Mathematically, this can be expressed as:

$$P(|\psi_i\rangle) \approx 0 \quad \text{for some states } |\psi_i\rangle \in \mathcal{H}^{\otimes n}. \quad (3.18)$$

In practical terms, this means that when the probabilities for certain states become very small, those states are unlikely to be observed. As a result, the generated numbers lose their random characteristic because the system does not explore all possible states uniformly.

Step 3. Pseudo-Random Number Generation

This section outlines the preprocessing steps required to generate pseudo-random numbers from a quantum system's quasi-probability distribution. Initially, values are collected from the quantum system's output. These values undergo a series of transformations to yield pseudo-random numbers. This includes scaling each value by 10^9 to increase its magnitude and then taking the modulus with 10^3 to confine the output within the range $[0, 1000]$. These transformed numbers are subsequently utilized as pseudo-random numbers in various applications requiring randomness.

3.1.3 Proposed Algorithm

This deterministic process is comprehensively elucidated in Figure 12, which serves as a visual representation of one potential trajectory that a quantum spin can traverse across the Bloch sphere. This figure effectively illustrates the underlying mechanics of the PRNG, shedding light on the intricate interplay of quantum states and transformations that result in the generation of pseudo-random numbers.

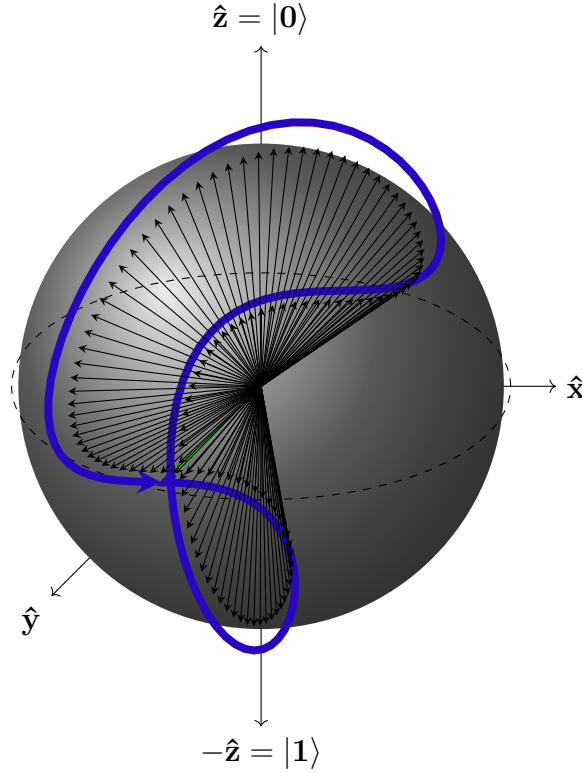


Figure 12 – Representation of a qubit and its spin path through the Bloch sphere.

The strategy involves a system of n qubits, a unitary operator U , and rotation parameters. To accomplish this, the following steps are followed:

- 1 Commence with a quantum system denoted as $|\psi\rangle$.
- 2 Apply a unitary operator U^t .
- 3 Retrieve the quasi-probability values for the states and store them in a list.
- 4 Iterate through steps 1 and 3 until the desired list length is reached.

The Algorithm 1 summarizes all the required steps to generate an adequate array of data. The implementation details of this algorithm can be found in Appendix B.1.

Algorithm 1 Algorithm for obtaining a set of probabilities.

Input: n - Number of qubits, $\theta = fs = \frac{1}{100}$ - Scale factor, $\theta_x, \theta_y, \theta_z$ - Rotation angles, L - Required length

Output: *values*

Input: $backend \leftarrow \text{Aer.get_backend('statevector_simulator')}$

$k = \frac{L}{2^n}$

$values \leftarrow []$

$cont = 0$

while k **do**

$circ \leftarrow \text{QUANTUMCIRCUIT}(n)$

for i **in** $\text{range}(n)$ **do**

$circ.h(i)$

$circ.rx(\theta_x, i)$

$circ.ry(\theta_y, i)$

$circ.rz(\theta_z, i)$

end for

$r_x, r_y, r_z \leftarrow cont \times \theta$

for i **in** $\text{range}(n)$ **do**

$circ.rx(r_x, i)$

$circ.ry(r_y, i)$

$circ.rz(r_z, i)$

end for

$counts \leftarrow backend.run(circ, shots = 2048).result().get_counts()$

$newValues \leftarrow \text{list}(counts.values())$

for i **in** $newValues$ **do**

$num = np.floor(i * 10^9) \% 10^3$

$values.append(num)$

end for

$cont \leftarrow cont + 1$

$k \leftarrow k - 1$

end while

return *values*

3.2 Image Encryption and Decryption Schemes

Image encryption and decryption processes rely on reversible operations applied to sequences of binary numbers. Typically, the XOR operation is utilized for this task. XOR fundamentally modifies bit information by combining two sources: the data used for encryption and a generated set of random numbers. It is essential that both sources have the same size for proper encryption and decryption.

3.2.1 Normalization of quasi-probabilities values

To perform the normalization of the list of quasi-probability values, a standard normalization is used (PATRO; SAHU, 2015):

$$v_i = \frac{v_i - \min(V)}{\max(V) - \min(V)} * c \quad (3.19)$$

Where, c is a constant which allow us to obtain big numbers.

3.2.2 Encryption Algorithm

The encryption algorithm is based on the $\oplus$ (bitwise XOR) operation between the image values and the normalized quasi-probability values. Algorithm 2 outlines the process for transforming the original image into its encrypted form. The implementation details of the algorithm can be found in Appendix A.2.

Algorithm 2 Algorithm for encryption based in XOR operation.

Input: I, V
 $I' = \{\}$
 $N = |I|$
for $i \leftarrow 0$ to N **do**
 $I'_i = I_i \oplus V_i$
end for

3.2.3 Decryption Algorithm

Since the binary operation $\oplus$ is reversible, it is possible to apply the same set of operations to recover the original data. Algorithm 3 outlines the decryption process.

Algorithm 3 Algorithm for decryption based in XOR operation.

Input: I', V
 $I = \{\}$
 $N = |I'|$
for $i \leftarrow 0$ to N **do**
 $I_i = I'_i \oplus V_i$
end for

3.3 Quantum Paradigm for Multigraph Analysis

This section introduces a more sophisticated methodology for handling multigraphs, utilizing a portion of the system to label underlying each overlapped multigraphs. This approach underscores that the overall state resides within the general Hilbert space $\mathcal{H} = \mathcal{H}_{\mathcal{T}} \otimes \mathcal{H}_{\mathcal{V}}$, where $\mathcal{H}_{\mathcal{T}}$ represents the space for MTL and $\mathcal{H}_{\mathcal{V}}$ represents the state space for vertices.

Figure 13 shows an scheme for the system division between labels for transitions and vertices. The first t qubits serve as inputs for circuit T , which corresponds to the labeling

of the underlying multigraph. The remaining qubits from t to n are inputs for circuit G , where the vertex u is transformed into the vertex v .

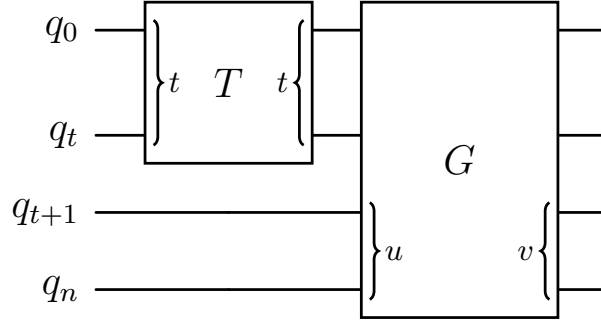


Figure 13 – Quantum circuit with n qubits.

Notice that the system does not undergo physical changes. Consequently, the underlying multigraph maintains its relations in E , although the number of vertices decreases proportionally to the qubits used for MTL. The redistribution of multigraph elements plays a crucial role in creating a new perspective. This modification enables the utilization of overlapping multigraphs, each labeled by $t \in T$. Moreover, it facilitates access to various multigraphs through the superposition mechanism. In Figure 14, the corresponding MTL set is composed of $T = \{0, 1\}$. Each label identifies an underlying multigraph, and depending on the inputs $|t, u\rangle$, the circuit will produce the output $|t, v\rangle$, i.e., depending on the circuit configuration, it is possible to traverse specific graph paths.

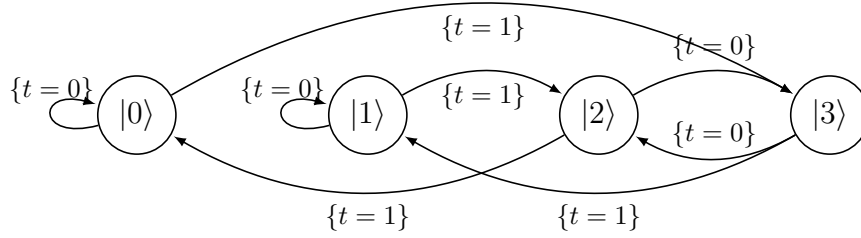


Figure 14 – Multigraph illustrating transitions and vertices corresponding to the circuit shown in Figure 8.

3.3.1 Quantum Multigraph Preparation without Superposition

This subsection presents the basic initialization steps for a multigraph over a quantum system without superposition.

3.3.1.1 Step 1: Initialization of Simple States

An empty quantum system of size $n + m$ qubits, denoted as the middle state $|I\rangle_1$.

$$|I\rangle_1 = |0\rangle^{n+m} \quad (3.20)$$

3.3.1.2 Step 2: Multigraph as Quantum Unitary Operator

It is defined the quantum operator, Ω_{XY} , which contain the multigraph structure as:

$$\Omega_{XY} = \bigotimes_{i=0}^{m-1} \Omega_{XY}^i. \quad (3.21)$$

Each transformation of the quantum system is determined by the action of these unitary operators on the qubits that represent transitions and vertices. Mathematically, the action of Ω_{XY}^i can be described as:

$$\Omega_{XY}^i : |y\rangle \rightarrow |y \oplus (x \cdot C_{XY}^i)\rangle \quad (3.22)$$

If $C_{XY}^i = 1$, then Ω_{XY}^i is a controlled-NOT (CNOT) gate; otherwise, the operator acts as the identity operator. The final simple transformation of a vertex $|Y\rangle$ into $|Y'\rangle$ is given by:

$$\Omega_{XY}(|0\rangle^{n+m-1}) = \bigotimes_{i=0}^{m-1} \Omega_{XY}^i(|0\rangle^{n+m-1}) \quad (3.23)$$

3.3.2 Quantum Multigraph Preparation with Superposition

This subsection outlines the fundamental steps for initializing a multigraph on a quantum system with superposition.

3.3.2.1 Step 1: Initialization of Simple States

For a multigraph composed of 2^n MTL and a set of nodes ranging from 0 to 2^m , it is initialized a quantum register with $n + m$ qubits as follows:

$$|I\rangle_0 = |0\rangle^{\otimes(n+m)} \quad (3.24)$$

To create a superposed system that allows every possible element for both subdivisions of the system, it is required an operator, U_1 , composed of a set of Hadamard gates which transform the elementary states:

$$U_1 = H^{\otimes(n+m)} \quad (3.25)$$

The transformation from $|I\rangle_0$ to the middle state $|I\rangle_1$ is achieved by applying U_1 :

$$|I\rangle_1 = U_1 |I\rangle_0 = \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle \otimes \frac{1}{\sqrt{2^m}} \sum_{y=0}^{2^m-1} |y\rangle \quad (3.26)$$

3.3.2.2 Step 2: Creating a System in Superposition

To bias the probabilities and favor certain edges and vertices, it is applied a series of single-qubit rotations using the unitary operator U_2 :

$$U_2 = \bigotimes_{i=0}^{2^n-1} R_y(\theta_i) \otimes \bigotimes_{j=0}^{2^m-1} R_y(\theta_j) \quad (3.27)$$

A modified superposed state generated by U_2 over $|I\rangle_1$ is equal to

$$U_2 |I\rangle_1 = \frac{1}{\sqrt{2^n}} \sum_{X=0}^{2^n-1} \left(\bigotimes_{i=0}^{2^n-1} R_y(\theta_i) \right) |X\rangle \otimes \frac{1}{\sqrt{2^m}} \sum_{Y=0}^{2^m-1} \left(\bigotimes_{j=0}^{2^m-1} R_y(\theta_j) \right) |Y\rangle \quad (3.28)$$

This equation applies the rotation gate $R_y(\theta_i)$ to each qubit in the n -qubit subsystem and $R_y(\theta_j)$ to each qubit in the m -qubit subsystem, ensuring that the correct indices are used for each subsystem.

3.3.2.3 Step 3: Representing Multigraphs with a Quantum Circuit

To define the quantum circuit which represent a multigraph can be expressed as:

$$M(\psi) = \frac{1}{\sqrt{2^n}} \sum_{X'=0}^{2^n-1} \frac{1}{\sqrt{2^m}} \sum_{Y'=0}^{2^m-1} |X'\rangle \bigotimes_{i=n}^{n+m-1} \Omega_{X'Y'}^i |Y'\rangle \quad (3.29)$$

Let Ω_{XY} represent the quantum operator describing the transformation of vertex Y when subjected to the transformation labeled by X . Equation 3.22 define Ω_{XY}^i as the quantum gate corresponding to the i -th bit of the binary representation of X .

3.3.2.4 Step 4: Wrapping Up the Multigraph Model

To compute the complete equation for $M(U_2 |I\rangle_1)$, it is necessary apply the operator M to $U_2 |I\rangle_1$. For this, it is substitute the expression $U_2 |I\rangle_1$ into the expression for $M |\psi\rangle$.

$$\begin{aligned} M(U_2 |I\rangle_1) &= \frac{1}{\sqrt{2^n}} \sum_{X'=0}^{2^n-1} \frac{1}{\sqrt{2^m}} \sum_{Y'=0}^{2^m-1} |X'\rangle \bigotimes_{i=n}^{n+m-1} \Omega_{X'Y'}^i |Y'\rangle \\ &= \frac{1}{\sqrt{2^n}} \sum_{X'=0}^{2^n-1} \frac{1}{\sqrt{2^m}} \sum_{Y'=0}^{2^m-1} \left(\bigotimes_{i=0}^{2^n-1} R_y(\theta_i) \right) |X'\rangle \bigotimes_{i=n}^{n+m-1} \Omega_{X'Y'}^i \left(\bigotimes_{j=0}^{2^m-1} R_y(\theta_j) \right) |Y'\rangle \end{aligned}$$

This expression involves a superposition over all possible combinations of states X' and Y' , with quantum operations acting on these states.

3.3.3 Insights into Multigraph Behavior

Examining the dimensions of Hilbert spaces associated with transition labels ($\mathcal{H}_T$) and vertices ($\mathcal{H}_V$) offers valuable insights into the behavior of multigraphs.

When $\dim(\mathcal{H}_V) > \dim(\mathcal{H}_T)$, the surplus of dimensions for vertices suggests the potential existence of disconnected multigraphs. In such cases, there may be excess vertices that are not linked to any transitions. Conversely, if $\dim(\mathcal{H}_V) < \dim(\mathcal{H}_T)$, more dimensions for transition labels facilitate the formation of multiple edges between vertices, leading to the emergence of intricate graph structures.

In a balanced scenario where $\dim(\mathcal{H}_V) = \dim(\mathcal{H}_T)$, every vertex maintains an equal level of connectivity to transitions. This equilibrium fosters the creation of well-connected multigraphs where all vertices are interconnected.

Given that the Hilbert space division between $\mathcal{H}_T$ and $\mathcal{H}_V$ is a virtual construct, it does not require alterations in the quantum circuit to utilize the generated underlying structures. To enhance clarity, let's elucidate these cases using a simple circuit depicted in Figure 15.

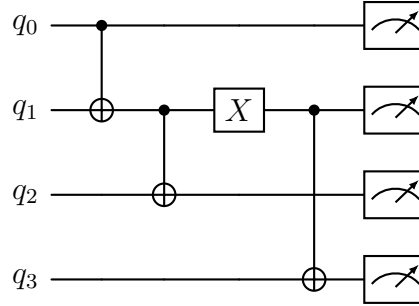


Figure 15 – Basic quantum circuit based on four qubits and gates (CNOT and X).

3.3.3.1 $\dim(\mathcal{H}_V) > \dim(\mathcal{H}_T)$

In this scenario, where the dimensionality of the Hilbert space associated with vertices, denoted as $\mathcal{H}_V$, surpasses that of the transition labels, represented by $\mathcal{H}_T$, intriguing dynamics emerge within the generated environment. Such a configuration suggests the presence of a surplus of vertices compared to the available transitions or arcs that connect them.

To illustrate this concept vividly, envision a network of cities where the number of locations (vertices) outnumbers the direct routes (transitions) between them. Essentially, $\mathcal{H}_V$ encompasses a greater number of potential instances than $\mathcal{H}_T$.

Practically speaking, this disparity manifests as instances where certain transitions lack direct connections to all vertices, resulting in disconnected regions within the network. In the context of our city analogy, this implies the existence of areas where certain cities are not directly reachable from others due to insufficient or absent connecting routes.

An illuminating approach to visualize and comprehend the implications of this scenario is through the examination of circuit outputs. By mapping the elements of transitions $\mathcal{H}_T$ and vertices (V), it unveils new insights into the diverse outcomes and configurations

achievable within the system. This process facilitates the identification of disconnected regions and highlights the intricate interplay between vertices and transitions within the multigraph framework.

To illustrate this case it was defined $\mathcal{H} = \mathcal{H} \otimes \mathcal{H}^{\otimes 3}$, where the elements of $T \in \mathcal{H}$ and $V \in \mathcal{H}^{\otimes 3}$. Table 1 provides a tangible demonstration of the outcomes generated through this modeling approach, offering invaluable insights into the behavior of multigraphs characterized by an abundance of vertices relative to available transitions.

Input vertex $ \psi_0\rangle$	$U 0, \psi_0\rangle$	$U 1, \psi_0\rangle$
$ 000\rangle$	$ 0101\rangle$	$ 1010\rangle$
$ 001\rangle$	$ 0100\rangle$	$ 1011\rangle$
$ 010\rangle$	$ 0111\rangle$	$ 1000\rangle$
$ 011\rangle$	$ 0110\rangle$	$ 1001\rangle$
$ 100\rangle$	$ 0000\rangle$	$ 1111\rangle$
$ 101\rangle$	$ 0001\rangle$	$ 1110\rangle$
$ 110\rangle$	$ 0010\rangle$	$ 1101\rangle$
$ 111\rangle$	$ 0011\rangle$	$ 1100\rangle$

Table 1 – This table presents the expected results for determinate input after the execution of the circuit shown in Figure 15, where the input space is a subdivision $\mathcal{H} = \mathcal{H}^{\otimes 3} \otimes \mathcal{H}$.

Additionally, the corresponding underlying multigraph is compound by the set of elements $\{u, v\} \in E$. The complete set of relationships is shown in Figure 16.

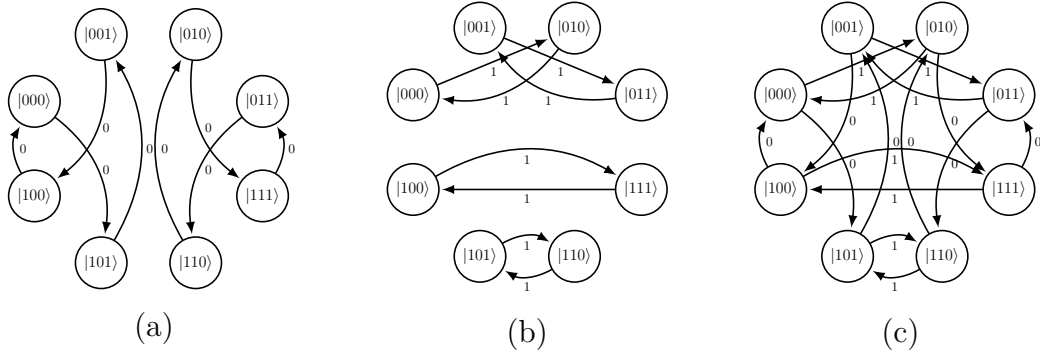


Figure 16 – Multigraph constructed using the inputs and output from Table 1.

3.3.3.2 $\dim(\mathcal{H}_V) < \dim(\mathcal{H}_T)$

In this scenario, the dimensionality of the Hilbert space associated with transition labels, denoted as $\mathcal{H}_T$, surpasses that of vertices, represented by $\mathcal{H}_V$. This surplus of dimensions for transition labels opens up the possibility of having multiple edges between two vertices.

To illustrate this concept further, let's imagine a subway map where each station represents a vertex and each subway line corresponds to a transition. With more dimensions

available for labels assigned to underlying multigraphs, it becomes feasible to accommodate multiple lines connecting the same pair of stations. Consequently, the network transforms into a complex web of interconnections.

Mathematically, this scenario can be captured by defining a composite Hilbert space $\mathcal{H} = \mathcal{H}_T^{\otimes m} \otimes \mathcal{H}_V^{\otimes n}$, where m denotes the degree of freedom associated with each transition label and n signifies the degree of freedom associated with each vertex, and $m > n$. By exploring such tensor products, it is effectively represented the multitude of possible configurations within the multigraph system, where transitions exhibit greater complexity than vertices.

To schematize this case it was defined $\mathcal{H} = \mathcal{H}^{\otimes 3} \otimes \mathcal{H}^{\otimes 1}$, where the elements of $T \in \mathcal{H}^{\otimes 3}$ and $V \in \mathcal{H}^{\otimes 1}$. Examining the resulting outputs, as showcased in Table 2, provides valuable insights into the diverse array of pathways and connections facilitated by this configuration.

$ \psi_0\rangle U$	$ 000\psi_0\rangle U$	$ 001\psi_0\rangle U$	$ 010\psi_0\rangle U$	$ 011\psi_0\rangle U$	$ 100\psi_0\rangle U$	$ 101\psi_0\rangle U$	$ 110\psi_0\rangle U$	$ 111\psi_0\rangle U$
$ 0\rangle$	$ 0101\rangle$	$ 0111\rangle$	$ 0000\rangle$	$ 0010\rangle$	$ 1010\rangle$	$ 1000\rangle$	$ 1111\rangle$	$ 1101\rangle$
$ 1\rangle$	$ 0100\rangle$	$ 0110\rangle$	$ 0001\rangle$	$ 0011\rangle$	$ 1011\rangle$	$ 1001\rangle$	$ 1110\rangle$	$ 1100\rangle$

Table 2 – This table presents the expected results to determinate input after the execution of the circuit shown in Figure 15 over a subdivision $\mathcal{H} = \mathcal{H}^{\otimes 3} \otimes \mathcal{H}$.

The corresponding vertices and edges are summarized in Figure 17. It shows a large set of small multigraphs that obey the underlying circuit mechanic.

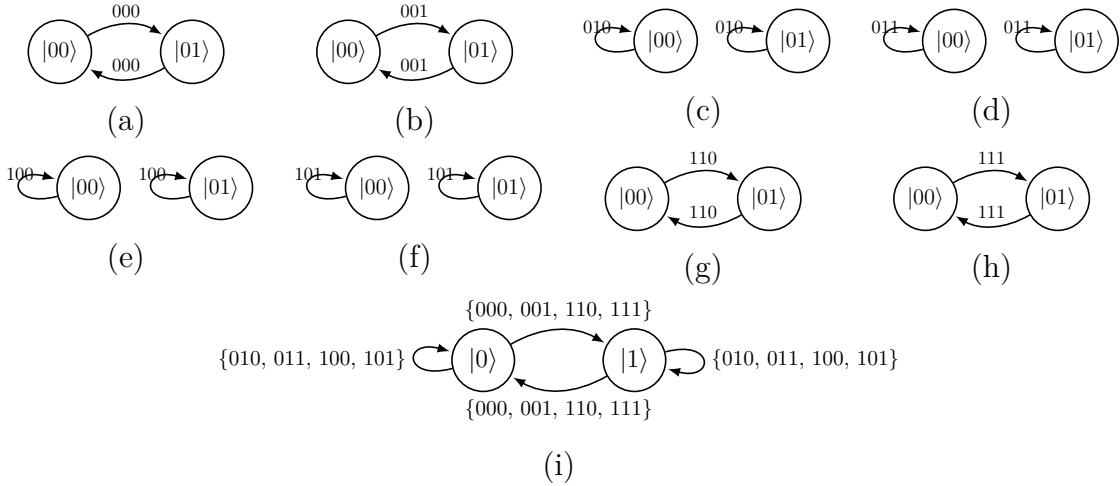


Figure 17 – Multigraph constructed using the inputs and output from Table 2.

3.3.3.3 $\dim(\mathcal{H}_V) = \dim(\mathcal{H}_T)$

In this case the dimensions of the Hilbert spaces associated with both vertices ($\mathcal{H}_V$) and transition labels ($\mathcal{H}_T$) are equal, signifying a balance between connectivity and complexity. Each vertex maintains an equal level of connectivity to transitions, fostering a well-connected multigraph where every vertex is interconnected with every other vertex.

To illustrate this equilibrium, envision a city grid where each intersection represents a vertex, and each road segment corresponds to a transition. In this scenario, every intersection is linked to precisely the right combination of roads, facilitating seamless travel throughout the city without unnecessary detours or redundancies.

The division of the Hilbert space for this case is defined as $\mathcal{H} = \mathcal{H}^{\otimes 2} \otimes \mathcal{H}^{\otimes 2}$, where the elements of $T \in \mathcal{H}^{\otimes 2}$ and $V \in H^{\otimes 2}$. Analyzing the resulting outputs, as presented in Table 3, provides a comprehensive depiction of the seamless connectivity and optimal routing facilitated by this balanced configuration.

$ \psi_0\rangle$	$U 00, \psi_0\rangle$	$U 01, \psi_0\rangle$	$U 10, \psi_0\rangle$	$U 11, \psi_0\rangle$
$ 00\rangle$	$ 0101\rangle$	$ 0000\rangle$	$ 1010\rangle$	$ 1111\rangle$
$ 01\rangle$	$ 0100\rangle$	$ 0001\rangle$	$ 1011\rangle$	$ 1110\rangle$
$ 10\rangle$	$ 0111\rangle$	$ 0010\rangle$	$ 1000\rangle$	$ 1101\rangle$
$ 11\rangle$	$ 0110\rangle$	$ 0011\rangle$	$ 1001\rangle$	$ 1100\rangle$

Table 3 – This table presents the expected results to determinate input after the execution of the circuit shown in Figure 15 over a subdivision $\mathcal{H} = \mathcal{H}^{\otimes 2} \otimes \mathcal{H}^{\otimes 2}$.

The underlying digraph is shown in Figure 18. It is possible to notice that each vertex is full connected to the remaining vertices, resulting in a line graph connecting any two vertices.

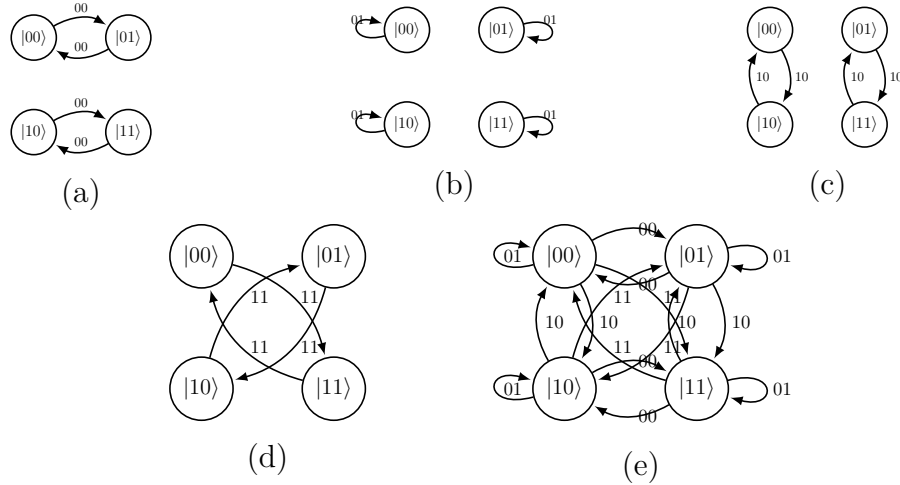


Figure 18 – Multigraph constructed using the inputs and output from Table 3.

3.3.4 Data Encoding and Compression with Multigraph Structures.

Efficient encoding and compression techniques are paramount for managing and transmitting vast amounts of information. Multigraph structures offer a unique approach to data representation, providing a framework that allows for both encoding and compression of binary data. By leveraging the inherent properties of multigraphs, such as their ability to represent complex relationships and their flexibility in storing sequential data, it is possible to develop robust algorithms for data manipulation.

Data encoding with multigraph structures involves mapping binary data onto vertices within the graph, with transitions between vertices representing portions of the binary string. This approach allows for the efficient representation of binary data in a structured manner, facilitating easy traversal and manipulation.

3.3.4.1 Direct Encoding of Binary Data on Multigraphs

Consider a multigraph $G = (V, E)$, where V denotes the set of vertices and E represents the set of edges. Let $T = \{t_0, t_1, \dots\}$ be a set of transition functions defined over an environment where the dimension of the Hilbert space H_V is greater than or equal to the dimension of H_T . Given a long binary string, it is possible to encode this data utilizing the concept of a line graph within the multigraph framework.

In this encoding scheme, each vertex in the multigraph represents a segment of the binary string and the appropriate transition set specify the sequential order of these segments. To begin encoding, it is designated the vertex u as initial and then each transition t_i in the line graph, corresponds to an adjacent portion of the binary string. Figure 6 presents a graphical representation of this concept.

This approach offers a structured and efficient means of representing binary data within a multigraph framework. Algorithm 4 transforms a binary string input and an initial state u into a set of transitions, based on a specified graph structure, $G = (V, E)$.

Algorithm 4 Algorithm for obtaining the graph line from a set of vertices.

Input: $U \equiv G = (V, E)$, *BinaryString*, initial state u_0 , $T = \{t_0, t_1, \dots\}$

Output: *SetOfTransitions*

```

SetOfTransitions  $\leftarrow \{\}$ 
splitData  $\leftarrow \text{split}(\text{Data}, \text{dim}(V))$ 
vertex  $\leftarrow u_0$ 
for  $i$  in splitData do
  for  $t$  in  $T$  do
     $|\psi\rangle \leftarrow |t\rangle | \text{vertex} \rangle$ 
     $|\psi'\rangle \leftarrow (\hat{I} \oplus \hat{U}) |\psi\rangle$ 
    if  $|\psi'\rangle = |l\rangle | \text{splitData}[i] \rangle$  then
      SetOfTransitions  $\leftarrow \text{SetOfTransitions} \cup \{t\}$ 
      vertex  $\leftarrow \text{splitData}[i]$ 
    end if
  end for
end for

```

To recover the original binary string is necessary go over the multigraph and store the visited vertices. Algorithm 5 is designed to operate on a quantum system described by an input graph $G = (V, E)$, an initial vertex u and a set of transitions T . Refer to Appendix A.2 for additional implementation details.

Algorithm 5 Algorithm for obtaining the sequential set of vertices using a graph line.

Input: $U \equiv G = (V, E)$, u_0 , and $T = \{t_0, t_1, \dots, t_m\}$

Output: $setOfOrderedVertices$

$labelSize \leftarrow \dim(T)$

$setOfOrderedVertices \leftarrow \{\}$

$vertex \leftarrow u_0$

for i in T **do**

$|\psi\rangle \leftarrow |i\rangle |vertex\rangle$

$|\psi'\rangle \leftarrow (U_1 \otimes U_2)(|i\rangle \otimes |vertex\rangle)$

$\rho_{labelSize, \dots, n} \leftarrow \text{Tr}_{0, \dots, labelSize-1}(|\psi'\rangle \langle \psi'|)$

$setOfOrderedVertices \leftarrow setOfOrderedVertices \cup \{\rho_{labelSize, \dots, n}\}$

$vertex \leftarrow \rho_{labelSize, \dots, n}$

end for

3.3.4.2 Enhanced Encoding with Superposition on Multigraphs

For image encoding, a common approach involves splitting the quantum register into two distinct parts (ZHANG et al., 2013). The first part is dedicated to encoding the arrangement of pixels within the image. This is achieved by employing Hadamard gates, which enable the generation of complete indexes for both the rows and columns of the image. Essentially, this step establishes the spatial layout of the image.

The second part of the encoding process focuses on representing color information using quantum gates. These gates manipulate the quantum states to encode the various color components of the image.

Both components of the quantum register pixel arrangement and color encoding are intricately connected to accurately depict the corresponding image.

To maximize the benefits of using multigraph structures for encoding images, it's important to dedicate part of the quantum circuit to encoding the spatial layout of the image, H_T , and the other part to encode the color information, H_V . Figure 19 presents the general configuration of the circuit for image encoding.

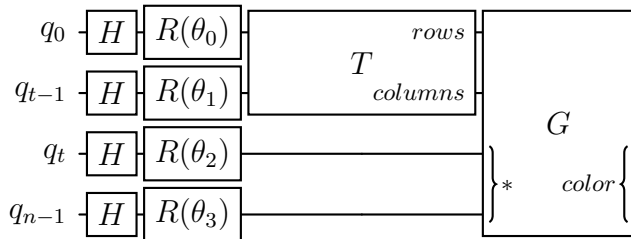


Figure 19 – General configuration of a quantum circuit with n qubits and the with image structure.

Additionally, a crucial component is the unitary operator, denoted as $U = \otimes_{i=0}^3 R(\theta_i)$. This operator plays a significant role in manipulating the probability distribution of quantum states. Essentially, it allows us to adjust probabilities to favor particular outcomes.

By applying this operator, it increases the likelihood of specific configurations, such as vertices and their connections.

Algorithm 6 offers an alternative approach to determining the appropriate rotations needed to create the necessary states for encoding a portion of an image. This algorithm provides a systematic method for achieving the desired quantum states, optimizing the encoding process for image data. The implementation detail are in Appendix B.2.

Algorithm 6 Algorithm for selecting an adequate rotation for desired states.

Input: $U \equiv G = (V, E)$, $setOfRotations$, $listOfDesiredStates$

Output: $selectedRotation$

$selectedRotation \leftarrow \text{None}$

$listObtainedStates \leftarrow \{\}$

for r in $setOfRotations$ **do**

$|\psi'\rangle \leftarrow H^{\otimes 4} |\psi\rangle$

$|\psi''\rangle \leftarrow U(Ry(\frac{\pi}{r_0}) \otimes Ry(\frac{\pi}{r_1}) \otimes Ry(\frac{\pi}{r_2}) \otimes Ry(\frac{\pi}{r_3})) |\psi'\rangle$

$states \leftarrow M^{\otimes 4} |\psi''\rangle$

$listObtainedStates \leftarrow listObtainedStates \cup states$

end for

$selectedRotation \leftarrow \text{find}(listObtainedStates, listOfDesiredStates)$

Experimental Results

This chapter serves as a comprehensive exposition of the empirical outcomes derived from the implemented structures and algorithms, aimed at elucidating the diverse applications, advantages and capabilities of quantum computing.

The experimental analyses conducted in this chapter demonstrates the performance and efficacy of the proposed methods. These results not only demonstrate the practical viability of quantum computing, in the area of this research, but also provide valuable benchmarks for evaluating its potential in various domains.

Section 4.2 offers a meticulous examination of the statistical tests employed to evaluate the proposed pseudo-random number generation (PRNG) scheme outlined in Section 3.1. Through the adequate tests, this section unveils the reliability and randomness of the generated sequences, crucial for a plethora of cryptographic and computational applications.

In Section 4.3, an exhaustive analysis of statistical tests conducted on encryption and decryption schemes, as detailed in Section 3.2, is presented. By dissecting security and efficiency metrics, this section solidifies the robustness and cryptographic strength of the proposed methods.

Lastly, Section 4.4 unveils pivotal findings derived from the application of multigraph structures for data encoding, as developed in Section 3.3. Through comprehensive experimentation, this section elucidates the innovative approaches and efficiencies facilitated by multigraph representations, paving the way for heightened data manipulation and processing capabilities within quantum computing paradigms.

4.1 Setup for Experiments

The experiments outlined in this chapter were conducted with utmost care and precision within a meticulously designed environment, ensuring the reliability and accuracy of the results obtained. Specifically, the IBM Qiskit Aer simulator was selected as the platform for these experiments due to its renowned robustness and precision in simu-

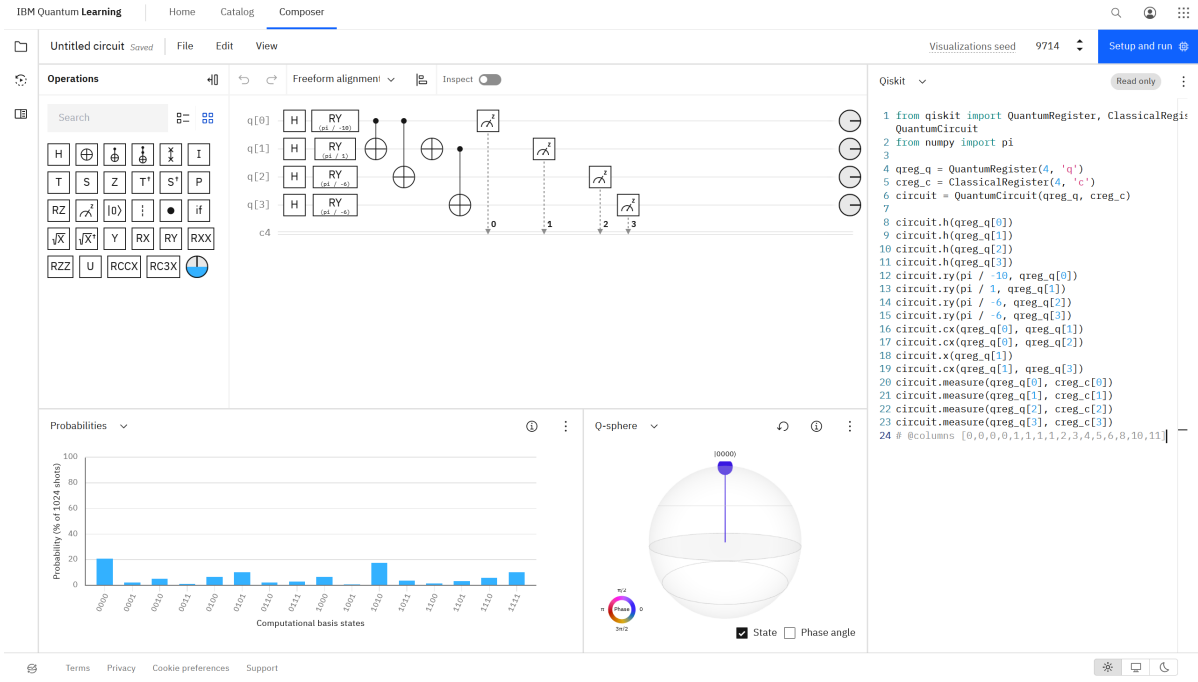


Figure 20 – IBM Quantum Composer platform.

lating quantum systems. The Qiskit Aer simulator provided an optimal framework for both the development and testing phases of our research. Additionally, to provide further insights into quantum computing, Figure 20 showcases the IBM Quantum Composer platform, which offers a user-friendly interface for visualizing and understanding quantum algorithms and circuits.

Moreover, the flexibility and scalability of the Qiskit Aer simulator facilitated seamless experimentation across various quantum computing paradigms. Its comprehensive suite of tools and functionalities enabled the implementation and evaluation of complex algorithms and structures, thereby enriching the experimental process and enhancing the reliability of the obtained results.

Furthermore, the data chosen for the experiment is presented in Figure 21. This set is composed by four RGB images which are usually use for image processing.

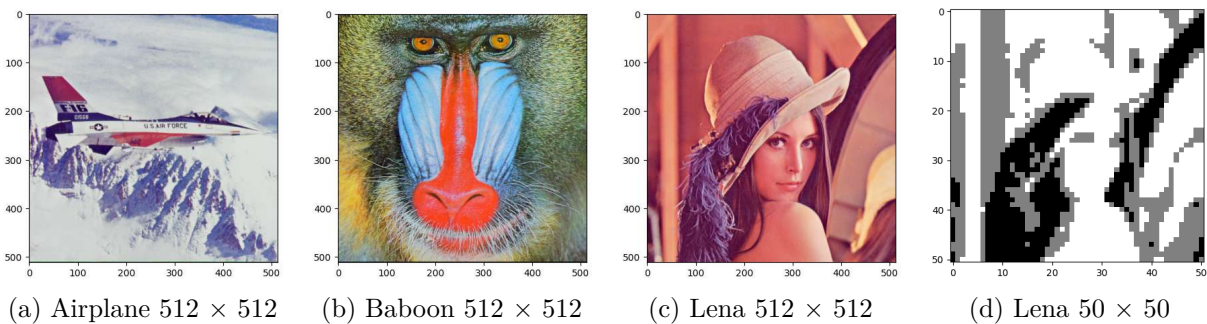


Figure 21 – Selected images for experiments.

4.2 Quantum Quasi-Probability Pseudo Random Generator (QQ-PRG)

This section presents the experimental results for Algorithm 1, designed to elucidate the randomness of its outputs. The discussion is divided into two main parts. This algorithm is detailed in Appendix B. The first part details the setup for testing, summarizing the key parameters and their corresponding values. The second part presents the statistical tests performed, including comparisons with other algorithms to assess the performance and randomness of the generated sequences.

4.2.1 Setup for Testing Algorithm QQ-PRG

The testing of QQ-PRG necessitates an initial configuration, encompassing several crucial parameters essential for its operation. These parameters include the number of required qubits, a scale factor, a set of rotations, and the desired length of the generated list. Table 4 provides a comprehensive overview of these parameters and their corresponding values.

Parameter	Value
Qubits	3
Scale factor	$sf = \frac{1}{100}$
Length	$L = 2048 \times 2048 = 4,194,304$
Number of quantum systems	$k = \frac{2048 \times 2048}{2^3} = 524288$
Rotations	$R_x = \frac{\pi}{3}, R_y = \frac{\pi}{3}, R_z = \frac{\pi}{3}$

Table 4 – Parameters for QQ-PRG algorithm configuration.

4.2.2 Statistical test

The generated sequence of numbers with pseudo-random properties, obtained through the Algorithm 3.1 execution, is subjected to rigorous statistical testing using the NIST SP 800-22 suite. See Appendix C. Comprising 15 distinct tests, each designed to calculate various indices for the provided binary data, these evaluations are crucial to determining the randomness and reliability of the generated sequence.

For each of the 15 tests, a predefined threshold value $\alpha = 0.01$ is established, signifying that the generated sequence must exhibit a 99% probability of being genuinely random to pass the test. The sequence under examination consists of a vast array of 4,194,304 decimal numbers. Each decimal number, comprised of three digits falling within the range of $[0, 999]$, is represented as a 10-bit string. In consequence, the entire raw dataset comprises a sequence of 41,943,040 binary digits, obtained from the execution of 524,288 quantum circuits. Each circuit was composed of 3 qubits, utilized a scaling factor of

$fs = 1/100$, employed a uniform set of rotation gates for each axis and set rotation angles at $\frac{\pi}{2} \times 0.19$ for each rotation gate.

The results obtained from the application of the NIST SP800-22 test battery are presented in Table 5. These results demonstrate success in 3 of the 15 tests. To provide context, the results are compared with those of a real-world quantum random number generator named JUR01, and this comparison is presented in Table 6. Both generators employ different methodologies for generating sets of numbers, each with its own advantages. The primary advantage of the proposed approach is its capability to pass a universal test. However, a notable limitation is the absence of an adequate methodology to rotate the qubits to generate a non-uniform probability distribution that closely approaches the maximum entropy.

C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	P-VALUE	PROPORTION	STATISTICAL TEST
10	0	0	0	0	0	0	0	0	0	0.000000*	0/10*	Frequency
10	0	0	0	0	0	0	0	0	0	0.000000*	0/10*	BlockFrequency
10	0	0	0	0	0	0	0	0	0	0.000000*	0/10*	CumulativeSums
10	0	0	0	0	0	0	0	0	0	0.000000*	0/10*	CumulativeSums
10	0	0	0	0	0	0	0	0	0	0.000000*	0/10*	Runs
10	0	0	0	0	0	0	0	0	0	0.000000*	0/10*	LongestRun
0	2	2	2	1	1	0	1	1	0	0.739918	10/10	Rank
10	0	0	0	0	0	0	0	0	0	0.000000*	0/10*	Fast Fourier Transform (FFT)
0	0	0	0	0	1	0	0	0	0	—	71/148*	NonOverlappingTemplate
10	0	0	0	0	0	0	0	0	0	0.000000 *	0/10*	OverlappingTemplate
2	1	2	0	1	1	0	0	2	1	0.739918	10/10	Universal
10	0	0	0	0	0	0	0	0	0	0.000000 *	0/10*	ApproximateEntropy
0	0	0	0	0	0	0	0	0	0	—	—	RandomExcursions
10	0	0	0	0	0	0	0	0	0	0.000000*	0/10*	Serial
2	1	2	0	0	0	1	2	2	0	0.534146	9/10	LinearComplexity

Table 5 – Outcome of the NIST SP800-22 randomness test suite for QQ-PRG.

Test	Quantum generator JUR01 ¹	QQ-PRG	Success
Frequency (block = 128)	0.000000* (0/1000)	0.000000 * (0/10)	*
BlockFrequency	0.000000* (845/1000)	0.000000* (0/10)	*
CumulativeSums (forward)	0.000000* (0/1000)	0.000000* (0/10)	*
Runs	0.000000* (0/1000)	0.000000* (0/10)	*
LongestRun	0.459717 (985/1000)	0.000000* (0/10)	JUR01
Rank	0.536163 (994/1000)	0.739918 (10/10)	JUR01 and QQ-PRG
FFT	0.749884 (991/1000)	0.000000* (0/10)	JUR01
NonOverlappingTemplate	0.000000* (631/1000)	Varies	*
Overlapping patterns	0.000023* (984/1000)	0.000000* (0/10)	*
Universal	0.000000* (723/1000)	0.739918 (10/10)	QQ-PRG
Entropy	0.000000* (0/1000)	0.000000* (0/10)	*
Excursions	-	-	*
Variational excursions	-	-	*
LinearComplexity (block = 500)	0.641284 (990/1000)	0.534146 (9/10)	JUR01 and QQ-PRG

Table 6 – Comparative analysis of quantum generator JUR01 and QQ-PRG.

¹ Quantum generator JUR01 presented by (JACAK et al., 2021).

4.3 Image Encryption and Decryption Tests

This section is dedicated to a comprehensive evaluation of the efficiency and effectiveness of the encryption and decryption schemes outlined in Section 2.3.2. These schemes offer a versatile approach to data encryption, providing a foundational framework for securing information.

The encryption and decryption processes are facilitated by Algorithms 4 and 5, respectively, as detailed in Section 3.2 and implemented in Appendix A. Both algorithms are designed to operate with a symmetric key, crucial for ensuring the security of the encrypted data. This symmetric key is generated using the Algorithm 1, QQ-PRG, a deterministic algorithm known for its robustness and reliability in generating pseudo-random sequences.

4.3.1 Setup for Testing Encryption and Decryption Algorithms

The testing of encryption and decryption algorithms necessitates the utilization of a symmetric key, which comprises a list of pseudo-random numbers. Generating this list of numbers involves configuring the QQ-PRG Algorithm. Table 7 presents an overview of these parameters and their corresponding values.

Parameter	Value
Qubits	3
Scale factor	$sf = \frac{1}{100}$
Length	$L = 262144 = 512 \times 512$
Number of quantum systems	$k = \frac{512 \times 512}{2^3} = 32768$
Rotations	$R_x = \frac{\pi}{3}, R_y = \frac{\pi}{3}, R_z = \frac{\pi}{3}$

Table 7 – Parameters for QQ-PRG algorithm configuration.

Additionally, Appendix A provides several functions that facilitate image processing. These functions enable us to load an image, convert it into a vector of elements, perform operations on this vector, and then convert it back into an image.

4.3.2 Encryption and Decryption Tests

To verify the efficacy and efficiency of cryptography algorithms, various tests can be conducted to assess their performance under different conditions. Among the selected tests are histogram analysis, cropping attack, noise attack, and correlation between pixels. Each test is designed to unveil specific characteristics of the encrypted data and the relationships between its elements.

Histogram analysis provides insights into the distribution of pixel intensities, aiding in understanding the overall structure and information content of the encrypted image. A cropping attack involves the elimination of certain parts of the image to gain insight

into the dependence of the pixels. Noise attacks introduce random disturbances to the encrypted data, evaluating the algorithm's resilience to external interference. Correlation between pixels examines the degree of interdependence between adjacent pixels in the encrypted and decrypted image, shedding light on the algorithm's ability to obscure patterns and maintain data integrity.

4.3.2.1 Histogram analysis

Original images are characterized by unique histograms that unveil specific details of each image. In a histogram, the x-axis represents the gray values, indicating different intensity levels within a specific color band (such as red). The y-axis shows the number of occurrences of each gray value, essentially counting how often each intensity level appears in the image. Figure 21 presents the original images with names Airplane, Baboon, and Lena. The corresponding histograms for each image are shown in panels (a), (b), and (c) of Figure 22. It reveals the pixel intensity distributions for the red band.

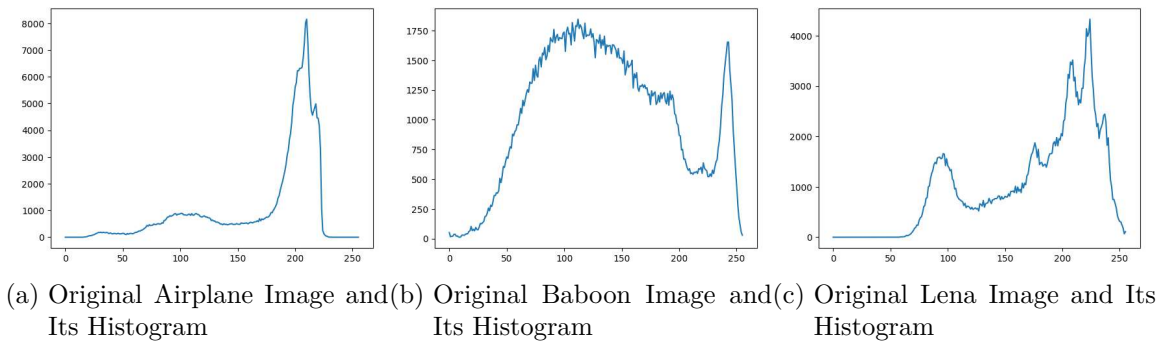


Figure 22 – Original images and their corresponding histograms.

Figure 23 shows the encrypted versions of the Airplane, Baboon, and Lena images. These encrypted images were generated using Algorithm 4, which employs XOR operation to modify the images, thereby enhancing their security. Notice that no recognizable pattern emerges from the encrypted images. The corresponding procedure for encrypting images is detailed in Appendix C.

Moreover, these encrypted images serve as crucial elements for further analysis in subsequent sections. They are utilized to evaluate their resilience against noise and cropping attack, as well as to examine the correlation analysis between the generated pixels. These assessments provide comprehensive insights into the effectiveness of the encryption process, offering valuable information about its robustness and reliability.

Figure 24 showcases histograms corresponding to each encrypted image, illustrating pixel intensity for the red band. This representation vividly demonstrates how the algorithm converts the image into an unintelligible sequence of numbers. In essence, the data becomes encrypted and inaccessible, ensuring enhanced security and confidentiality.

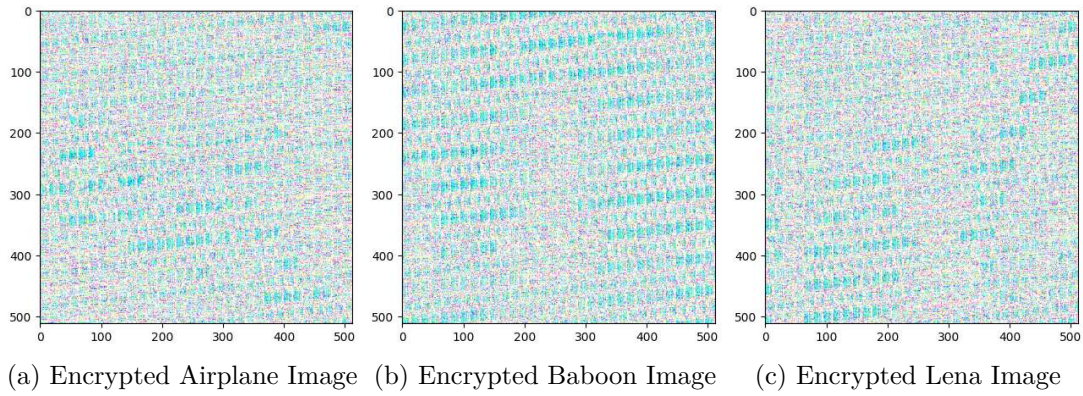


Figure 23 – Encrypted images.

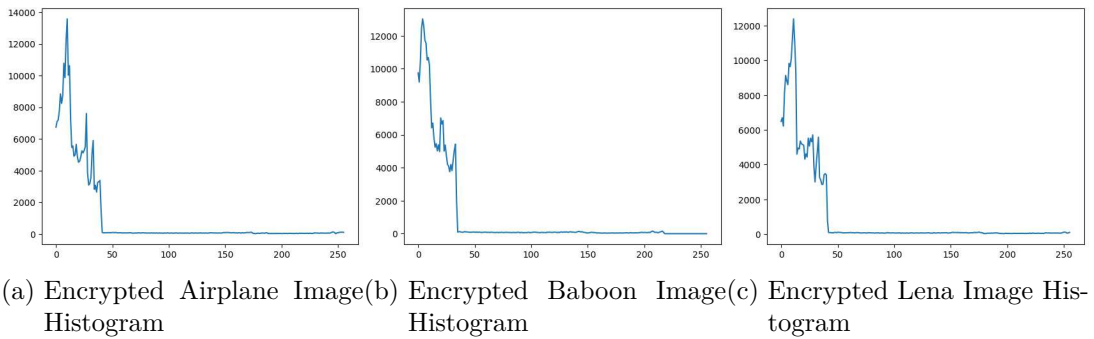


Figure 24 – Histograms for the red channel of encrypted images.

Finally, the decrypted images are reconstructed from their encrypted counterparts and Algorithm 5. Figure 25 presents the decrypted versions of the encrypted Airplane, Baboon, and Lena images. The related protocol is described in detail in Appendix C.

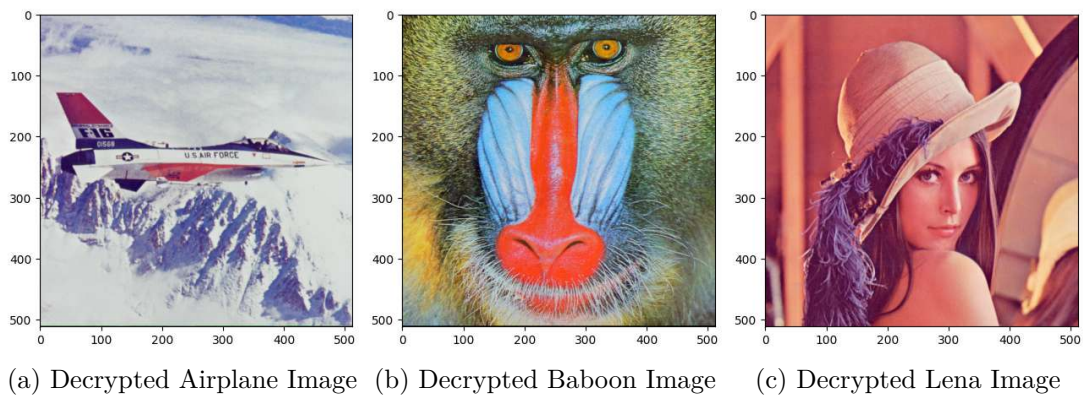


Figure 25 – Decrypted images.

4.3.2.2 Cropping attack test

To evaluate the resilience of the encrypted images against potential information degradation during transmission and preservation, it was conducted a cropping attack test.

This involved applying clipping operations to the cipher images, simulating the removal of specific regions similar to cropping, which can occur during data transmission or storage processes. The robustness of the images against such attacks is evaluated by examining their decrypted versions.

Figure 26 presents the results of the conducted tests, showcasing the cipher images influenced by various clipping operations alongside their corresponding decrypted versions. The applied clipping operations are depicted as removal of square regions of different sizes. Panels (a), (b), and (c) of Figure 26 illustrate images with 60x60, 180x180, and 360x360 pixel squares removed, respectively. The corresponding decrypted versions of these images are depicted in panels (d), (e), and (f), providing insights into the efficacy of the decryption process under different clipping conditions.

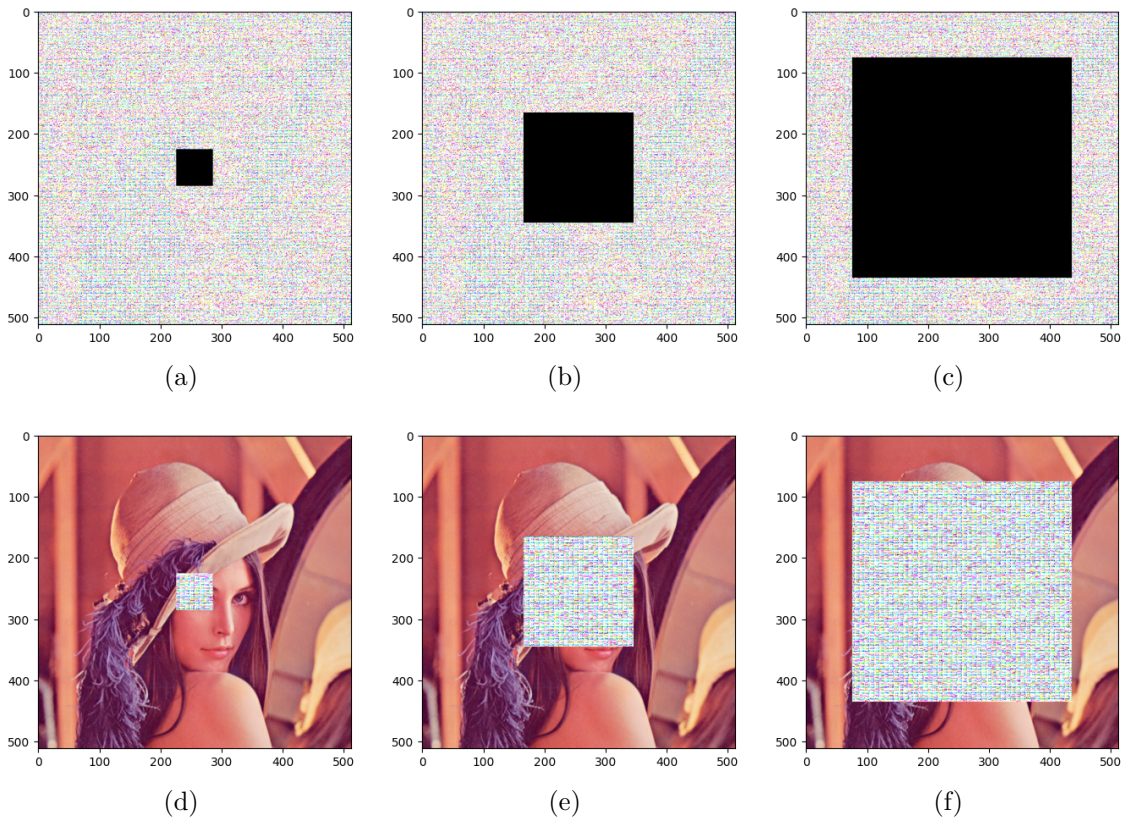


Figure 26 – Encrypted images with various clipping operations and their decrypted counterparts.

4.3.2.3 Noise attack test

To evaluate the resilience of the encrypted images against potential information degradation during transmission and storage, it was conducted a noise attack test. This test involved deliberately introducing salt and pepper noise to simulate random pixel corruption, akin to interference commonly encountered in data transmission or storage processes.

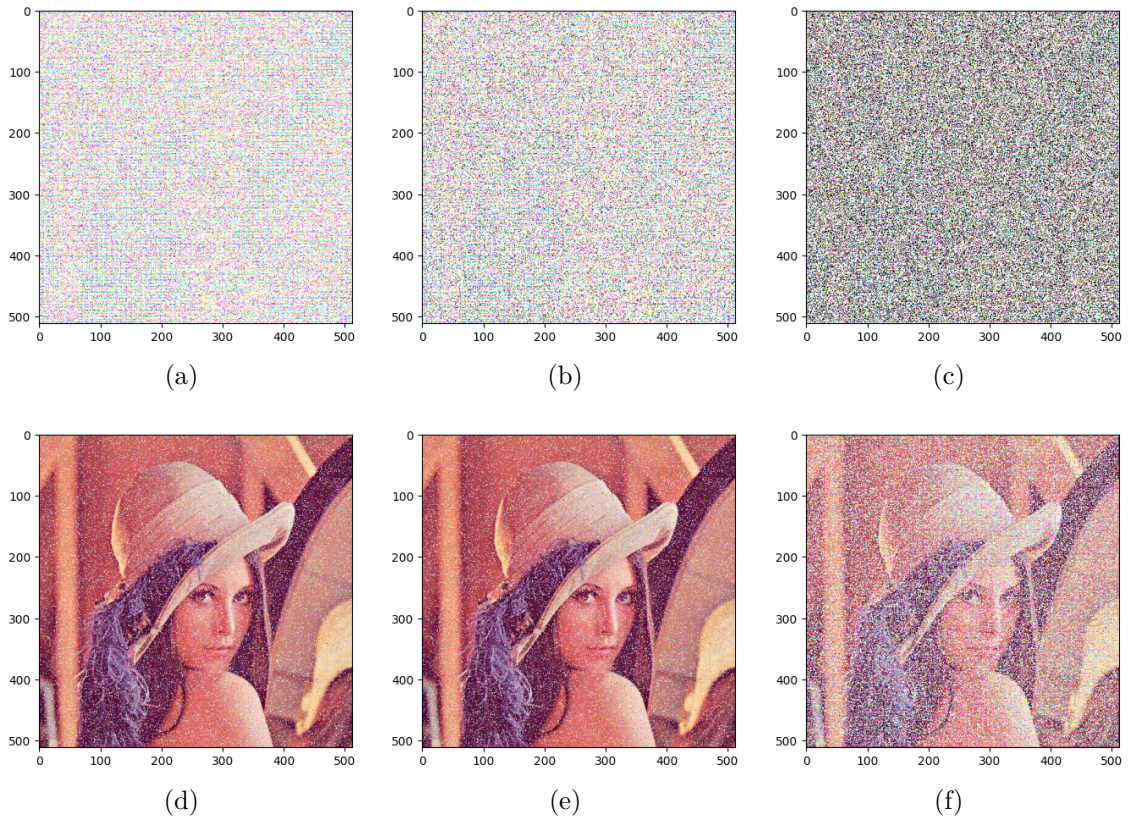


Figure 27 – Cipher images and their decrypted counterparts under varying noise levels.

Figure 27 illustrates the conducted tests, presenting cipher images affected by varying levels of salt and pepper noise alongside their corresponding decrypted versions. The noise was introduced using OpenCV functions, which randomly alter the values of selected pixels. The levels of introduced noise are expressed as percentages, indicating the proportion of pixels affected in relation to the entire image.

Panels (a), (b), and (c) of Figure 27 depict cipher images with 0.004%, 0.04%, and 0.4% salt and pepper noise, respectively. These levels were chosen to represent mild, moderate, and severe noise interference scenarios. The corresponding decrypted versions of these images, shown in panels (d), (e), and (f), respectively, provide valuable insights into the efficacy of the decryption process under varying noise conditions.

4.3.2.4 Correlation Analysis

The correlation analysis is a crucial tool for understanding how the relationships between pixels change between the original and encrypted images. In this analysis, it is examined horizontal, vertical, and diagonal pixel correlations to gain insights into the encryption process's impact on pixel arrangements.

To ensure a thorough evaluation, correlation indices are computed for selected images Airplane, Baboon, and Lena in both RGB and grayscale scenarios using Equation 2.18. Furthermore, the implementation of this analysis in Python is elaborated in Appendix A.

The results of these analyses are detailed in Tables 8 and 9, respectively. Additionally, each table incorporates reported results from prior research to facilitate a robust comparison. This comparative approach enhances our understanding of the efficacy and uniqueness of the proposed methodology, allowing us to benchmark our results against established standards in the field.

Image	Proposal			Ref. ²			Ref. ³		
	R	G	B	R	G	B	R	G	B
Airplane	0.923	0.92	0.90	-	-	-	0.88	0.89	0.89
EncAirplane	0.025	-0.002	-0.002	-	-	-	-0.005	-0.005	-0.002
Baboon	0.848	0.73	0.84	0.91	0.80	0.88	-	-	-
EncBaboon	0.028	0.002	-0.003	-0.002	-0.003	10^{-3}	-	-	-
Lena	0.969	0.96	0.92	0.93	0.92	0.90	-	-	-
EncLena	0.025	0.003	-0.002	0.001	0.004	-0.005	-	-	-

Table 8 – Diagonal correlation distribution for RGB images.

Image	Direction	Proposal		Ref. ⁴		Ref. ⁵	
		Original	Cipher	Original	Cipher	Original	Cipher
Lena	H	0.951	-0.00014	0.944	-0.004	0.985	-0.0002
	V	0.981	-0.00012	0.972	6.23e-5	0.971	0.0001
	D	0.965	0.0504	0.940	-0.006	0.958	-0.0013
Baboon	H	0.753	0.025	0.872	0.0046	0.754	0.0009
	V	0.869	0.0010	0.824	-0.001	0.861	-0.0008
	D	0.791	0.0019	0.762	0.005	0.723	-0.0005

Table 9 – Correlation distribution for grayscale Lena and Baboon images.

The correlation analysis provides several key insights into the effectiveness of the encryption process:

Firstly, the original images exhibit a high level of pixel correlation. This is clearly shown in Table 8 for the R, G, and B channels, as well as in Table 9 for the grayscale channel. These high correlation values reflect the natural coherence and structure inherent in the original images, indicating that neighboring pixels have similar intensity values.

Secondly, the encrypted images show significantly lower correlation values across all color channels. This dramatic reduction in correlation, as highlighted in Tables 8 and 9, demonstrates the encryption method’s ability to effectively disrupt the original pixel relationships. This reduction is crucial for ensuring that the encrypted image does not retain any of the predictable patterns found in the original image, thereby enhancing the security of the encryption process.

² Presented by (EL-LATIF; ABD-EL-ATTY; VENEGAS-ANDRACA, 2020)

³ Presented by (EL-LATIF et al., 2020)

⁴ Presented by (GAO et al., 2022b)

⁵ Presented by (ABD-EL-ATTY et al., 2021)

Finally, the analysis underscores the impact of specific parameters on correlation values. Figure 28 visually represents the correlation distribution for the Airplane image, illustrating how the encryption process affects pixel relationships across different channels. The figure details the R, G, and B channels of both the original (a, b, c) and encrypted images (d, e, f). Notably, Figure 28d emphasizes the normalization effect of the parameter c , which plays a crucial role in determining the final correlation values. This highlights the importance of c in the overall encryption scheme, as it significantly influences the integrity and security of the encrypted image.

4.4 Image Encoding Using Multigraph

This section presents the obtained results from employing multigraphs for image encoding. As detailed in Section 3.3, the utilization of superposition offers two distinct perspectives which are tested in next sections.

Given the vast array of potential configurations for quantum circuits and the multitude of underlying multigraph structures, this study selects a straightforward quantum circuit and a widely applicable configuration, specifically $\mathcal{H}_T = \mathcal{H}_V$. This configuration facilitates the creation of a complete multigraph, enabling the construction of a line graph and ensuring the existence of any particular subgraph.

4.4.1 Setup for Testing Image Encoding using Multigraph

The setting up of the environment for testing image encoding using a multigraph involves two primary configurations. First, it requires designing a quantum circuit that generates all possible allowable relationships. Second, it involves logically partitioning the Hilbert space to create a consistent multigraph.

4.4.1.1 Quantum Circuit

This step involves constructing a quantum circuit designed to generate all possible relationships between vertices and edges. The specific quantum circuit code used for this purpose is detailed in Appendix B. The design of this circuit ensures that every potential connection is accounted for, providing the foundation for creating a multigraph structure.

Figure 29 depicts the chosen quantum circuit configuration for the tests. It is important to note that there are numerous quantum circuits variations, each capable of generating a wide array of possible multigraph configurations.

4.4.1.2 Hilbert Space Partitioning

In this step, the Hilbert space is logically partitioned to facilitate the generation of a coherent multigraph. This partitioning is essential as it determines how vertices and

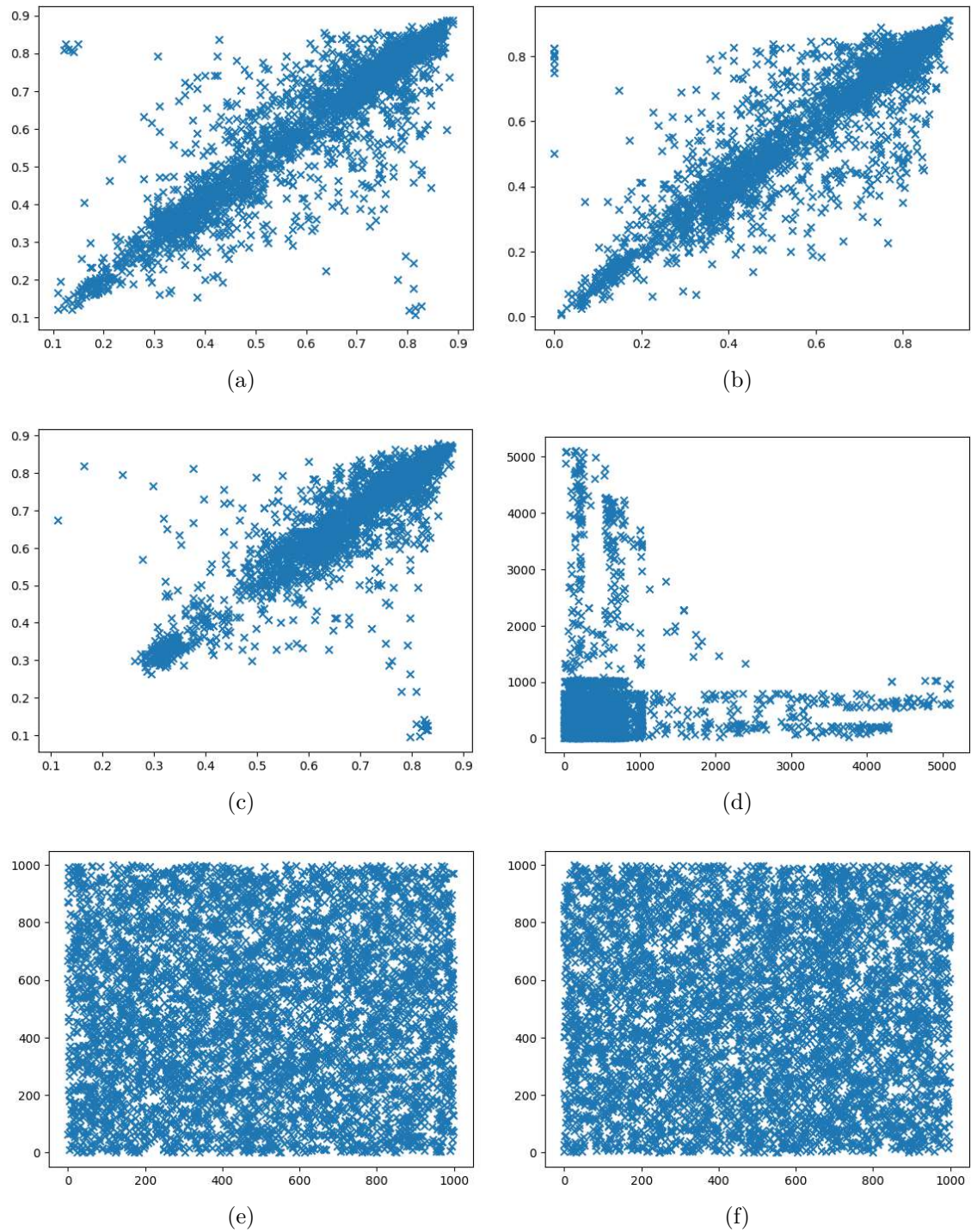


Figure 28 – Correlation distribution for Airplane image.

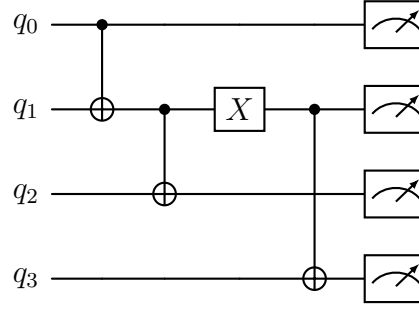


Figure 29 – Quantum circuit based on four qubits and gates (CNOT and X).

edges are represented within the quantum framework, ensuring the resultant multigraph accurately reflects the intended relationships.

For the testing, the chosen partitioning corresponds to the third case developed in Section 3.3, where $\mathcal{H}_T = \mathcal{H}_V$. In this configuration, the general Hilbert space is composed as follows: $\mathcal{H}^{\otimes 4} = \mathcal{H}^2 \otimes \mathcal{H}^2 = \mathcal{H}_T \otimes \mathcal{H}_V$.

This setup generates a complete multigraph where each node is directly connected to every other node. This approach not only simplifies the testing process but also provides a comprehensive structure to evaluate the encoding algorithms.

Figure 30 illustrate the underlying multigraph structure based on this configuration.

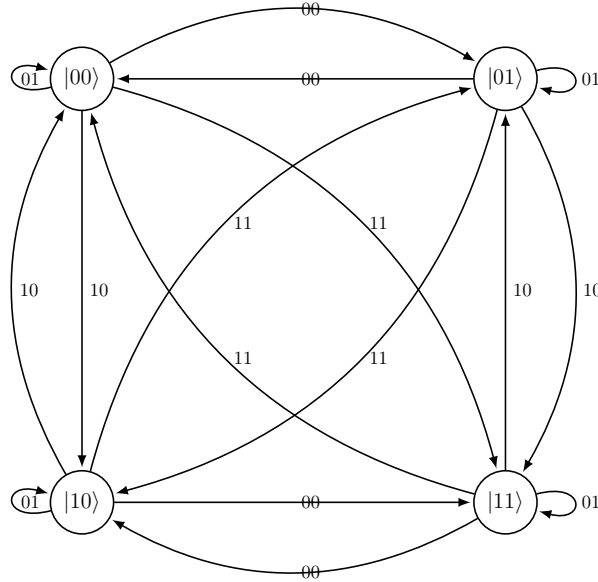


Figure 30 – Multigraph generated using the quantum circuit shown in Figure 29.

4.4.2 Image Encoding Using Multigraph Encoding without Superposition

The process of encoding an image using a multigraph structure without superposition leverages the concept of a line graph as the central element. This approach ensures that the image data is effectively mapped onto the multigraph.

Algorithm 4 outlines the steps for calculating the line graph over a multigraph. This algorithm systematically stores the entire dataset within the multigraph structure by utilizing transitions to encode the image information. The transitions ensure that every piece of data is accurately represented within the multigraph framework.

To decode the image data, Algorithm 5 is employed. This algorithm reconstructs the original image by reversing the encoding process, utilizing the line graph to accurately retrieve and reconstruct the stored data. This ensures that the image can be decoded precisely, preserving the integrity of the original data.

4.4.2.1 Preprocessing of Raw Data

This step involves transforming the raw data into a format suitable for representation within a graph structure. In the context of multigraph encoding, the raw data is converted into a ordered list of vertices that the line graph will traverse. Table 10 illustrates the preprocessing of raw data:

Description	Data
Raw data	01011011101010101110111010111110001101110010
Vertices	[01], [01], [10], [11], [10], [10], [10], [10], [11], [10], [11], [10], [10], [11], [11], [10], [00], [11], [01], [11], [00], [10]

Table 10 – Preprocessing of raw data.

4.4.2.2 Encoding Process

The encoding process involves mapping the preprocessed image data onto the multigraph structure. This is achieved through Algorithm 4, which constructs the line graph from the multigraph. In this approach, each data element is represented as a node in the multigraph. The relationships and order of these data elements are captured through the connections in the line graph, ensuring that the structure and sequence of the original image are preserved. This method allows for an efficient and organized encoding of the image data within the multigraph framework.

Description	Data
Initial vertex u	00
Edges	$\{\{00,01\},\{01,01\},\{01,11\},\{11,10\},\{10,10\},\{10,10\},\{10,10\},$ $\{10,11\},$ $\{11,10\},\{10,10\},\{10,11\},\{11,11\},\{11,10\},\{10,00\},\{00,11\},\{11,01\},$ $\{01,11\},\{11,00\},\{00,10\}\}$
Output	$MTL = \{00, 01, 11, 00, 00, 01, 01, 01, 00, 00, 00, 00,$ $01,00, 01, 00,10, 11, 10, 10, 11, 10\}$

Table 11 – Summary of Data Encoding Process

Finally, the result of the encoding process is the MTL. This list contains the corresponding transition labels which encode the raw data. Figure 31 presents a visual representation of the MTL or line graph generated.

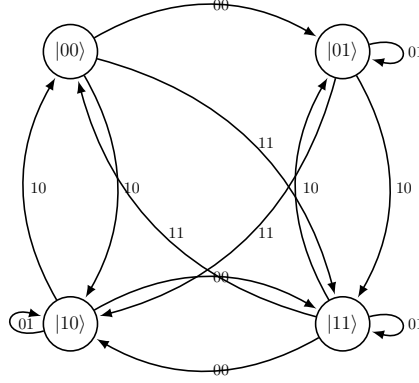


Figure 31 – Graph that contains the corresponding vertices and MTL for the encoded data.

4.4.2.3 Decoding Process

Once the image data has been encoded using the multigraph approach, decoding becomes essential to recover the original data. This process involves reversing the encoding steps to reconstruct the original image from the multigraph representation. Algorithm 5 delineates the systematic procedure for decoding the encoded data.

The decoding algorithm relies on two crucial elements: the initial state, denoted by u , which serves as the starting point for the decoding process, and the MTL, which encapsulates the transitions between vertices in the multigraph structure. These components enable the algorithm to efficiently navigate through the multigraph, retracing the steps of the encoding process to reconstruct the original image data. Through this systematic approach, the decoding algorithm ensures the faithful retrieval of the encoded image data, maintaining the integrity and structure of the original image.

4.4.3 Image Encoding Using Multigraph Encoding with Superposition

Image encoding using a multigraph with superposition entails a methodical approach. Leveraging superposition enables the acquisition of a set of elements along with their associated probabilities, as discussed in Sections 2 and 3.3 respectively. This approach encompasses three pivotal phases: preprocessing the raw data, encoding the data using the multigraph with superposition technique, and evaluating the performance of the encoded system.

4.4.3.1 Preprocessing the Raw Data

Preparing the raw data for encoding is crucial to ensure optimal data representation within the chosen framework. This initial phase involves transforming the raw data to

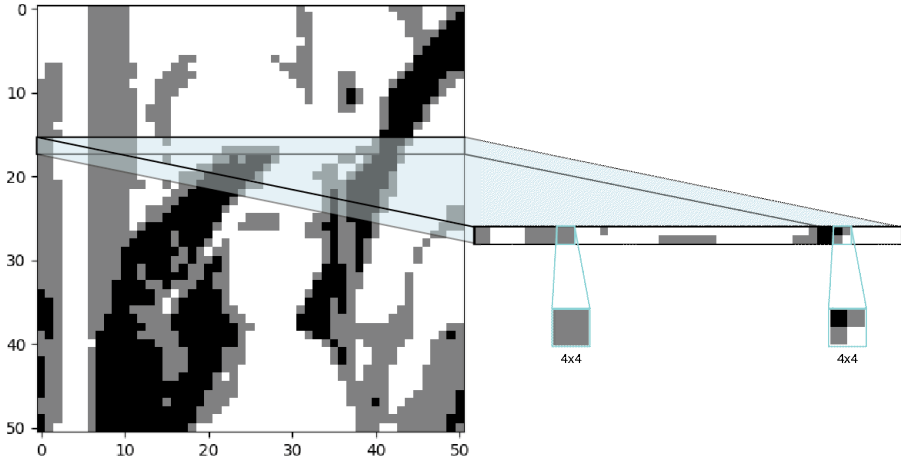


Figure 32 – Illustration depicting the process of obtaining small pieces of raw data for encoding.

make it compatible with the encoding process.

In our case, the raw data consist of a grayscale image with pixel values ranging from 0 to 3. Figure 21d provides a visual representation of the selected image, showcasing its grayscale intensity levels.

In the preprocessing phase, the image undergoes division into smaller matrices, each consisting of 4×4 pixels. This division enables the image to be broken down into more manageable data units, which is essential for efficient encoding using a multigraph technique. Figure 32 depicts a schematic representation of this matrix generation process.

4.4.3.2 Encoding the Data

The crucial part of our evaluation is the encoding of preprocessed data using the multigraph with superposition technique. Algorithm 6 is central to this step, as it computes the necessary rotations to effectively encode the data within a multigraph structure enhanced by superposition.

The algorithm works by calculating the required qubit rotations based on a provided list of rotations. It matches the data for encoding with the outputs of the quantum system to determine the appropriate set of rotations for each specific piece of data. This ensures that the data is accurately represented within the quantum framework.

Figure 33 provides a detailed illustration of the probability distributions two particular quantum system in superposition and its quantum states associated. Subfigure (a) showcases a quantum circuit consisting of 4 qubits, each initially put into a superposition state using Hadamard gates, followed by a series of CNOT gates which over a Hilbert space division composes a multigraph. Subfigure (b) presents a bar chart displaying the probability distribution of each quantum state resulting from the circuit in (a). In Subfigure(c), the quantum circuit is modified to include additional rotations around the y-axis for each qubit using $R_y(\theta_i), \theta_i \in [0, a\frac{\pi}{2}], a \in \mathcal{R}$ gates with specified angles. Subfigure (d)

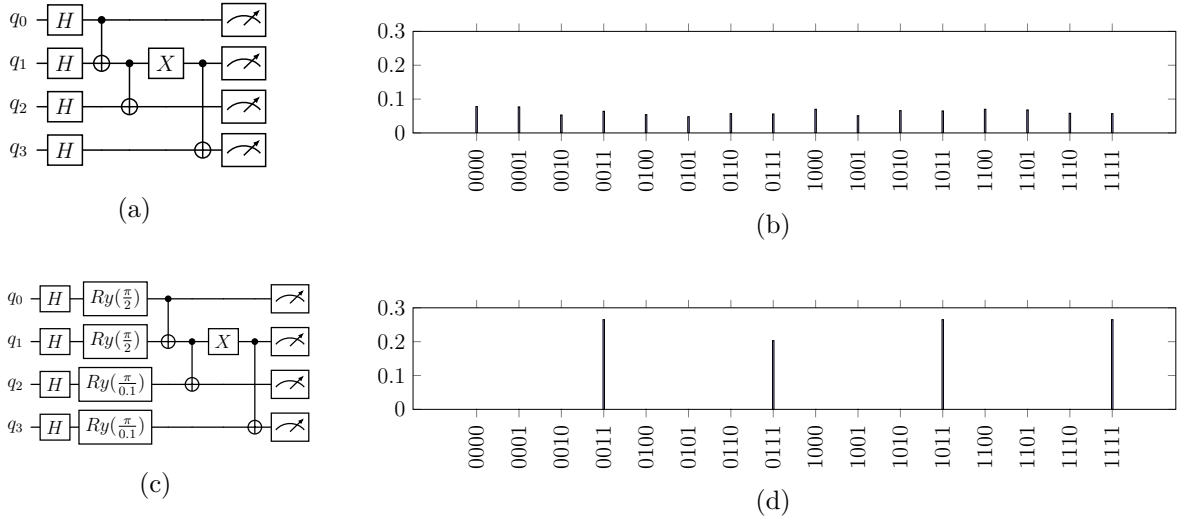


Figure 33 – Quantum circuits and their corresponding quasi-probabilities.

then illustrates the probabilities of measuring each possible computational basis state after executing the modified circuit in (c), also shown as a bar chart. The visual comparison across these subfigures highlights the impact of different quantum operations on the state probabilities.

For enhanced comprehension, Figure 34 visually depicts the encoded data, specifically focusing on the scenario outlined in Figure 33(d). In this representation, transitions within the multigraph signify the positions of individual pixels, while nodes encapsulate the corresponding grayscale levels. It's essential to note that this particular graph constitutes a subset of the overarching structure presented in Figure 30. This illustration serves to elucidate how pixel positions and grayscale information are encapsulated within the encoded data, providing a tangible visualization of the encoding process.

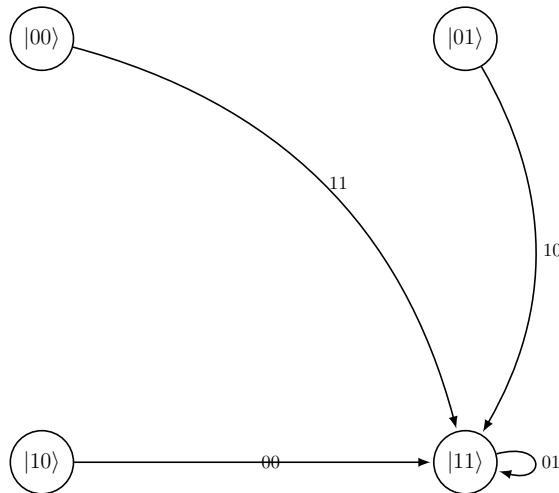


Figure 34 – Multigraph generated using the quantum more probable states shown in Figure 33d.

4.4.3.3 Results for Image Encoding using Multigraphs

The outcome of the encoding process is a series of rotation angles that effectively represent the image. Table 12 offers insights into this representation, presenting some of the frequently encountered matrices in the test image along with the required rotations for encoding. By employing the multigraph with superposition technique and utilizing the calculated rotations, it is established a robust encoding scheme that faithfully captures the essence of the original data.

Index	$Ry(\theta_i), i \in \{0, 1, 2, 3\}$	Representative states	Occurrences	M
0	$(\frac{\pi}{-150.0}, \frac{\pi}{-83.3}, \frac{\pi}{-150.0}, \frac{\pi}{-16.6})$	0011, 0111, 1011, 1111	245	
1	$(\frac{\pi}{-150.0}, \frac{\pi}{-83.3}, \frac{\pi}{-50.0}, \frac{\pi}{-16.6})$	0010, 0110, 1010, 1110	88	
2	$(\frac{\pi}{-150.0}, \frac{\pi}{-83.3}, \frac{\pi}{-16.6}, \frac{\pi}{-50.0})$	0010, 0111, 1010, 1111	42	
3	$(\frac{\pi}{50.0}, \frac{\pi}{-150.0}, \frac{\pi}{16.6}, \frac{\pi}{150.0})$	0011, 0110, 1011, 1110	11	
4	$(\frac{\pi}{-83.3}, \frac{\pi}{16.6}, \frac{\pi}{-150.0}, \frac{\pi}{150.0})$	0011, 0111, 1010, 1111	8	
5	$(\frac{\pi}{-150.0}, \frac{\pi}{-50.0}, \frac{\pi}{-150.0}, \frac{\pi}{-150.0})$	0010, 0110, 1011, 1110	3	
6	$(\frac{\pi}{-116.6}, \frac{\pi}{-116.6}, \frac{\pi}{-116.6}, \frac{\pi}{-116.6})$	0011, 0111, 1010, 1110	7	
7	$(\frac{\pi}{-150.0}, \frac{\pi}{-116.6}, \frac{\pi}{50.0}, \frac{\pi}{150.0})$	0010, 0101, 1001, 1101	12	
8	$(\frac{\pi}{-83.3}, \frac{\pi}{16.6}, \frac{\pi}{16.6}, \frac{\pi}{150.0})$	0011, 0111, 1011, 1110	11	
9	$(\frac{\pi}{-150.0}, \frac{\pi}{-16.6}, \frac{\pi}{-116.6}, \frac{\pi}{16.6})$	0001, 0101, 1001, 1101	68	
10	$(\frac{\pi}{50.0}, \frac{\pi}{-50.0}, \frac{\pi}{-50.0}, \frac{\pi}{-83.3})$	0010, 0111, 1010, 1110	3	
11	$(\frac{\pi}{-83.3}, \frac{\pi}{-83.3}, \frac{\pi}{-116.6}, \frac{\pi}{83.3})$	0011, 01100, 1010, 1110	10	
12	$(\frac{\pi}{-116.6}, \frac{\pi}{16.6}, \frac{\pi}{50.0}, \frac{\pi}{-83.3})$	0010, 0111, 1011, 1111	13	
13	$(\frac{\pi}{-116.6}, \frac{\pi}{-150.0}, \frac{\pi}{-150.0}, \frac{\pi}{-150.0})$	0010, 0111, 1011, 1110	2	
14	$(\frac{\pi}{-150.0}, \frac{\pi}{116.6}, \frac{\pi}{-116.6}, \frac{\pi}{-50.0})$	0011, 0101, 1010, 1101	2	
15	$(\frac{\pi}{-150.0}, \frac{\pi}{-16.6}, \frac{\pi}{-50.0}, \frac{\pi}{-116.6})$	0010, 0110, 1010, 1111	7	
16	$(\frac{\pi}{83.3}, \frac{\pi}{-116.6}, \frac{\pi}{-116.6}, \frac{\pi}{83.3})$	0011, 0110, 1010, 1101	6	
17	$(\frac{\pi}{-150.0}, \frac{\pi}{-116.6}, \frac{\pi}{16.6}, \frac{\pi}{150.0})$	0001, 0110, 1010, 1110	6	
18	$(\frac{\pi}{-116.6}, \frac{\pi}{-16.6}, \frac{\pi}{-50.0}, \frac{\pi}{-83.3})$	0010, 0101, 1010, 1101	12	
19	$(\frac{\pi}{-50.0}, \frac{\pi}{-50.0}, \frac{\pi}{150.0}, \frac{\pi}{-16.6})$	0001, 0101, 1001, 1110	8	
20	$(\frac{\pi}{-150.0}, \frac{\pi}{50.0}, \frac{\pi}{-116.6}, \frac{\pi}{-150.0})$	0010, 0110, 1011, 1111	4	
21	$(\frac{\pi}{-150.0}, \frac{\pi}{-150.0}, \frac{\pi}{116.6}, \frac{\pi}{-150.0})$	0011, 0110, 1011, 1101	3	
22	$(\frac{\pi}{-150.0}, \frac{\pi}{-150.0}, \frac{\pi}{-50.0}, \frac{\pi}{50.0})$	0001, 0110, 1001, 1111	4	
23	$(\frac{\pi}{-83.3}, \frac{\pi}{-50.0}, \frac{\pi}{16.6}, \frac{\pi}{83.3})$	0011, 0110, 1011, 1111	6	
24	$(\frac{\pi}{-150.0}, \frac{\pi}{83.3}, \frac{\pi}{-16.6}, \frac{\pi}{150.0})$	0010, 0110, 1010, 1101	8	
25	$(\frac{\pi}{-150.0}, \frac{\pi}{-16.6}, \frac{\pi}{-50.0}, \frac{\pi}{-16.6})$	0001, 0110, 1001, 1110	12	
26	$(\frac{\pi}{-50.0}, \frac{\pi}{-83.3}, \frac{\pi}{116.6}, \frac{\pi}{83.3})$	0001, 0110, 1010, 1111	1	
27	$(\frac{\pi}{-116.6}, \frac{\pi}{-16.6}, \frac{\pi}{-150.0}, \frac{\pi}{83.3})$	0001, 0101, 1010, 1101	7	
28	$(\frac{\pi}{-116.6}, \frac{\pi}{-116.6}, \frac{\pi}{-50.0}, \frac{\pi}{50.0})$	0010, 0101, 1001, 1110	2	
29	$(\frac{\pi}{-150.0}, \frac{\pi}{-50.0}, \frac{\pi}{16.6}, \frac{\pi}{116.6})$	0010, 0110, 1001, 1110	6	
30	$(\frac{\pi}{-150.0}, \frac{\pi}{-16.6}, \frac{\pi}{-116.6}, \frac{\pi}{-83.3})$	0001, 0101, 1010, 1110	2	
31	$(\frac{\pi}{-83.3}, \frac{\pi}{-16.6}, \frac{\pi}{83.3}, \frac{\pi}{83.3})$	0001, 0110, 1001, 1101	3	
32	$(\frac{\pi}{16.6}, \frac{\pi}{-83.3}, \frac{\pi}{-50.0}, \frac{\pi}{116.6})$	0011, 0101, 1011, 1111	1	
33	$(\frac{\pi}{-150.0}, \frac{\pi}{-16.6}, \frac{\pi}{-16.6}, \frac{\pi}{-150.0})$	0010, 0101, 10100, 1110	1	
34	$(\frac{\pi}{-116.6}, \frac{\pi}{-150.0}, \frac{\pi}{116.6}, \frac{\pi}{-50.0})$	0010, 0101, 1011, 1110	1	

Table 12 – The rotations needed to encode Lena image segments (Figure 21d), along with the top initial states, their occurrences, and grayscale binary matrix representation.

Conclusions

The emergence of quantum computers and their associated phenomena has opened up a new frontier for exploration, offering promising avenues for developing innovative methods in searching, encoding, encryption, decryption, and various other applications. Quantum computers possess unique capabilities that surpass those of traditional digital computers, providing new opportunities for insights and advancements that were previously unattainable.

This research shows the utilization of quasi-probabilities, rooted in quantum mechanics, to establish a Pseudo Random Number Generator PRNG based on a quantum system. Through testing, was demonstrated that this approach generate a deterministic sequence of numbers with pseudo-random characteristics. The experiments underscore the efficiency and unpredictability of quantum-generated random numbers, laying the foundation for future image encryption schemes.

Furthermore, was developed an image encryption scheme leveraging the unique properties of quantum states and their associated quasi-probabilities. These schemes offer enhanced security against classical attacks and pave the way for fundamentally secure image transmission and storage.

Lastly, was presented a new perspective on the utilization of multigraphs, with labels assigned to transitions within the underlying multigraphs. This innovative structure bridges multigraphs and the environment defined by the dimensions assigned to each part of the underlying multigraph, enabling the generation of graphs with varying vertex or edge label density. This perspective highlights the pivotal role of transitions in data representation and processing, exploring environments where different configurations can lead to the generation of multiple multigraphs with potential new applications, that become possible from the discoveries, in this research.

5.1 Contributions

This section outlines the main contributions achieved in this work:

- ❑ Proposal of an algorithm utilizing a quantum system and the quasi-probability of its states to generate a long sequence of numbers exhibiting pseudo-random properties.
- ❑ Development and testing of an encryption and decryption algorithm based on a PRNG. Evaluation of this algorithm includes analysis through histogram examination, cropping attack, noise attack, and correlation analysis.
- ❑ Introduction of a novel framework for multigraphs within quantum systems. Through rigorous theoretical analysis and extensive practical experimentation, the feasibility and effectiveness of this framework are demonstrated. The results exhibit significant advancements in data encoding, compression, and manipulation within multigraph contexts, with potential applications in quantum computing, data science, and cryptography.

5.2 Scientific Production

These contributions include paper submissions in peer-reviewed journals and workshop presentations that have advanced the understanding of quantum computing, pseudo-random number generation, image encryption, and multigraph representation.

5.2.1 Journal Submissions

1. Quantum-Based PRNG for Image Encryption: A Comprehensive Study

- **Title:** Quantum-Based PRNG for Image Encryption: A Comprehensive Study
- **Authors:** Luis José Mantilla Santa Cruz, Luis F. Faina, J. H. de Souza Pereira
- **Journal:** Communications in Statistics - Simulation and Computation
- **Status:** Major review (January 2024)

2. Exploring Quantum Systems for Pseudo-Random Number Generation

- **Title:** Exploring Quantum Systems for Pseudo-Random Number Generation
- **Authors:** Luis José Mantilla Santa Cruz, Luis F. Faina, J. H. de Souza Pereira
- **Journal:** Journal of Quantum Computing and Cryptography
- **Status:** Major review (February 2024)

3. Quantum Multigraphs: Expanding the Horizons of Quantum Information Theory

- **Title:** Quantum Multigraphs: Expanding the Horizons of Quantum Information Theory
- **Authors:** Luis José Mantilla Santa Cruz, Luis F. Faina, Fabíola Souza Fernandes Pereira, J. H. de Souza Pereira
- **Journal:** Digital Signal Processing
- **Status:** Submitted in March 2024

5.2.2 Workshop Presentations

1. Exploring Quantum Computing and Quasi-Probability-Based Image Encryption

- **Event:** XVII Workshop of Thesis and Dissertations in Computer Science
- **Location:** Campus Santa Mônica - Universidade Federal de Uberlândia
- **Date:** 06/10/2023

2. Quantum Computing's Symphony: Superposition, Gates, and Cryptographic Possibilities

- **Event:** XVII Workshop of Thesis and Dissertations in Computer Science
- **Location:** Campus Santa Mônica - Universidade Federal de Uberlândia
- **Date:** 06/10/2023

5.3 Future Works

This section discusses potential future directions for research and development in the field.

- Currently, was explored the Pseudo Random Number Generator PRNG. However, the proposed unitary operator, $U = S(R_j^{\otimes n})$, opens the door to a more complex methodology for generating pseudo-random numbers. By incorporating more complex operations into the shift operator, S , can enhance the security and randomness of the generator.

- Labeled Multigraphs represent a novel approach to encoding and processing data, utilizing probability as a link between edges and vertices in a multigraph. The critical next step in this research is to develop an algorithm that can precisely calculate the necessary qubit rotations for encoding data within a superposed quantum circuit.

References

- ABD-EL-ATTY, B. et al. Optical image encryption based on quantum walks. **Optics and Lasers in Engineering**, Elsevier, v. 138, p. 106403, 2021. Disponível em: <<https://doi.org/10.1016/j.optlaseng.2020.106403>>.
- ACAMPORA, G. et al. D-nisq: A reference model for distributed noisy intermediate-scale quantum computers. **Information Fusion**, Elsevier B.V., v. 89, p. 16–28, 1 2023. ISSN 15662535. Disponível em: <<https://doi.org/10.1016/j.inffus.2022.08.003>>.
- AJAGEKAR, A.; YOU, F. Quantum computing for energy systems optimization: Challenges and opportunities. **Energy**, Elsevier, v. 179, p. 76–89, 2019. Disponível em: <<https://doi.org/10.1016/j.energy.2019.04.186>>.
- ANSHU, A.; JAIN, R. Efficient methods for one-shot quantum communication. **npj Quantum Information**, Nature Publishing Group, v. 8, n. 1, p. 1–7, 2022. Disponível em: <<https://doi.org/10.1038/s41534-022-00608-1>>.
- ARUTE, F. et al. Quantum supremacy using a programmable superconducting processor. **Nature**, Nature Publishing Group, v. 574, n. 7779, p. 505–510, 2019.
- BAE, J.-H. et al. Quantum circuit optimization using quantum karnaugh map. **Scientific reports**, Nature Publishing Group, v. 10, n. 1, p. 1–8, 2020. Disponível em: <<https://doi.org/10.1038/s41598-020-72469-7>>.
- BHADURI, A. S. et al. Robust quantum circuit for clique problem with intermediate qudits. **Nano Communication Networks**, Elsevier B.V., v. 37, 9 2023. ISSN 18787789. Disponível em: <<https://doi.org/10.1016/j.nancom.2023.100466>>.
- BOLLOBÁS, B. **Modern graph theory**. [S.l.]: Springer Science & Business Media, 2013. v. 184.
- CACCIAPUOTI, A. S. et al. When entanglement meets classical communications: Quantum teleportation for the quantum internet. **IEEE Transactions on Communications**, IEEE, v. 68, n. 6, p. 3808–3833, 2020. Disponível em: <<https://doi.org/10.1109/TCOMM.2020.2978071>>.
- CALEFFI, M. et al. The quantum internet: Networking challenges in distributed quantum computing. **arXiv preprint quantph/1810.08421**, 2019. Disponível em: <<https://doi.org/10.1109/MNET.001.1900092>>.

- CHIN, H.-M. et al. Machine learning aided carrier recovery in continuous-variable quantum key distribution. **npj Quantum Information**, Nature Publishing Group, v. 7, n. 1, p. 1–6, 2021. Disponível em: <<https://doi.org/10.1038/s41534-021-00361-x>>.
- COUTINHO, B. C. et al. Robustness of noisy quantum networks. **Communications Physics**, Nature Publishing Group, v. 5, n. 1, p. 1–9, 2022. Disponível em: <<https://doi.org/10.1038/s42005-022-00866-7>>.
- D'HONDT, E. Quantum approaches to graph colouring. **Theoretical Computer Science**, v. 410, p. 302–309, 2 2009. ISSN 03043975. Disponível em: <<https://doi.org/10.1016/j.tcs.2008.09.055>>.
- DIESTEL, R.; DIESTEL, R. The basics. **Graph Theory**, Springer, p. 1–34, 2017. Disponível em: <https://doi.org/10.1007/978-3-662-53622-3_1>.
- EL-LATIF, A. A. A. et al. Controlled alternate quantum walks based privacy preserving healthcare images in internet of things. **Optics and Laser Technology**, Elsevier Ltd, v. 124, 4 2020. ISSN 00303992. Disponível em: <<https://doi.org/10.1016/j.optlastec.2019.105942>>.
- EL-LATIF, A. A. A.; ABD-EL-ATTY, B.; VENEGAS-ANDRACA, S. E. Controlled alternate quantum walk-based pseudo-random number generator and its application to quantum color image encryption. **Physica A: Statistical Mechanics and its Applications**, Elsevier, v. 547, p. 123869, 2020. Disponível em: <<https://doi.org/10.1016/j.physa.2019.123869>>.
- EMMS, D. et al. Coined quantum walks lift the cospectrality of graphs and trees. **Pattern Recognition**, v. 42, p. 1988–2002, 9 2009. ISSN 00313203. Disponível em: <<https://doi.org/10.1016/j.patcog.2008.10.025>>.
- ESSIN, A. M.; GRIFFITHS, D. J. Quantum mechanics of the $1/x^2$ potential. **American journal of physics**, AIP Publishing, v. 74, n. 2, p. 109–117, 2006. Disponível em: <<https://doi.org/10.1119/1.2165248>>.
- GAO, Y.-j. et al. A novel quantum image encryption technique based on improved controlled alternated quantum walks and hyperchaotic system. **Physica A: Statistical Mechanics and its Applications**, Elsevier, v. 598, p. 127334, 2022. Disponível em: <<https://doi.org/10.1016/j.physa.2022.127334>>.
- GAO, Y. jun et al. A novel quantum image encryption technique based on improved controlled alternated quantum walks and hyperchaotic system. **Physica A: Statistical Mechanics and its Applications**, Elsevier B.V., v. 598, 7 2022. ISSN 03784371. Disponível em: <<https://doi.org/10.1016/j.physa.2022.127334>>.
- GIUSTINA, D. D. et al. Quantum encoding of dynamic directed graphs. **Journal of Logical and Algebraic Methods in Programming**, Elsevier Inc., v. 136, 1 2024. ISSN 23522216. Disponível em: <<https://doi.org/10.1016/j.jlamp.2023.100925>>.
- HAO, W. et al. A hybrid neqr image encryption cryptosystem using two-dimensional quantum walks and quantum coding. **Signal Processing**, Elsevier B.V., v. 205, 4 2023. ISSN 01651684. Disponível em: <<https://doi.org/10.1016/j.sigpro.2022.108890>>.

- ILI, A.; BAI, M. On the chromatic number of integral circulant graphs. **Computers and Mathematics with Applications**, v. 60, p. 144–150, 7 2010. ISSN 08981221. Disponível em: <<https://doi.org/10.1016/j.camwa.2010.04.041>>.
- JACAK, M. M. et al. Quantum generators of random numbers. **Scientific Reports**, Nature Publishing Group UK London, v. 11, n. 1, p. 16108, 2021. Disponível em: <<https://doi.org/10.1038/s41598-021-95388-7>>.
- JÓŹWIAK, P.; JACAK, J. E.; JACAK, W. A. New concepts and construction of quantum random number generators. **Quantum Information Processing**, Springer, v. 23, n. 4, p. 132, 2024. Disponível em: <<https://doi.org/10.1007/s11128-024-04335-8>>.
- KASHEFI, E. et al. Twisted graph states for ancilla-driven universal quantum computation. **Electronic Notes in Theoretical Computer Science**, v. 249, p. 307–331, 8 2009. ISSN 15710661. Disponível em: <<https://doi.org/10.1016/j.entcs.2009.07.096>>.
- KRENN, M.; ERHARD, M.; ZEILINGER, A. Computer-inspired quantum experiments. **Nature Reviews Physics**, Nature Publishing Group, v. 2, n. 11, p. 649–661, 2020. Disponível em: <<https://doi.org/10.1038/s42254-020-0230-4>>.
- KUMAR, A.; MISHRA, A. Evaluation of cryptographically secure pseudo random number generators for post quantum era. In: IEEE. **2022 IEEE 7th International conference for Convergence in Technology (I2CT)**. 2022. p. 1–5. Disponível em: <<https://doi.org/10.1109/I2CT54291.2022.9824543>>.
- KUMAR, A. et al. Futuristic view of the internet of quantum drones: review, challenges and research agenda. **Vehicular Communications**, Elsevier, p. 100487, 2022. Disponível em: <<https://doi.org/10.1016/j.vehcom.2022.100487>>.
- KUMAR, V. et al. Quantum true random number generation on ibms cloud platform. **Journal of King Saud University-Computer and Information Sciences**, Elsevier, v. 34, n. 8, p. 6453–6465, 2022. Disponível em: <<https://doi.org/10.1016/j.jksuci.2022.01.015>>.
- LE, N.; CYKIERT, M.; GINOSSAR, E. Scalable and robust quantum computing on qubit arrays with fixed coupling. **npj Quantum Information**, Nature Publishing Group, v. 9, n. 1, p. 1–8, 2023. Disponível em: <<https://doi.org/10.1038/s41534-022-00668-3>>.
- LEVY, A.; LOSTAGLIO, M. Quasiprobability distribution for heat fluctuations in the quantum regime. **PRX Quantum**, APS, v. 1, n. 1, p. 010309, 2020. Disponível em: <<https://doi.org/10.1103/PRXQuantum.1.010309>>.
- NEILL, C. et al. A blueprint for demonstrating quantum supremacy with superconducting qubits. **Science**, American Association for the Advancement of Science, v. 360, n. 6385, p. 195–199, 2018. Disponível em: <<https://doi.org/10.1126/science.aao4309>>.
- NIELSEN, M. A.; CHUANG, I. L. **Quantum computation and quantum information**. [S.l.]: Cambridge university press Cambridge, 2001. v. 2.
- NIROULA, P.; NAM, Y. A quantum algorithm for string matching. **npj Quantum Information**, Nature Publishing Group, v. 7, n. 1, p. 1–5, 2021. Disponível em: <<https://doi.org/10.1038/s41534-021-00369-3>>.

- PALER, A.; FOWLER, A. G.; WILLE, R. Faster manipulation of large quantum circuits using wire label reference diagrams. **Microprocessors and Microsystems**, Elsevier B.V., v. 66, p. 55–66, 4 2019. ISSN 01419331. Disponível em: <<https://doi.org/10.1016/j.micpro.2019.02.008>>.
- PATRO, S.; SAHU, K. K. Normalization: A preprocessing stage. **arXiv preprint arXiv:1503.06462**, 2015. Disponível em: <<https://doi.org/10.17148/IARJSET.2015.2305>>.
- SHI, H. et al. Entanglement-assisted capacity regions and protocol designs for quantum multiple-access channels. **npj Quantum Information**, Nature Publishing Group, v. 7, n. 1, p. 1–9, 2021. Disponível em: <<https://doi.org/10.1038/s41534-021-00412-3>>.
- SIGOV, A.; RATKIN, L.; IVANOV, L. A. Quantum information technology. **Journal of Industrial Information Integration**, Elsevier B.V., v. 28, 7 2022. ISSN 2452414X. Disponível em: <<https://doi.org/10.1016/j.jii.2022.100365>>.
- SU, Y.; WANG, X. A robust visual image encryption scheme based on controlled quantum walks. **Physica A: Statistical Mechanics and its Applications**, Elsevier B.V., v. 587, 2 2022. ISSN 03784371. Disponível em: <<https://doi.org/10.1016/j.physa.2021.126529>>.
- THAPLIYAL, H.; MUÑOZ-COREAS, E.; KHALUS, V. Quantum circuit designs of carry lookahead adder optimized for t-count t-depth and qubits. **Sustainable Computing: Informatics and Systems**, Elsevier, v. 29, p. 100457, 2021. Disponível em: <<https://doi.org/10.1016/j.suscom.2020.100457>>.
- VAJAPHEYAM, S. Understanding shannon’s entropy metric for information. **arXiv preprint arXiv:1405.2061**, 2014.
- WANG, J.; MANOUCHEHRI, K. Physical implementation of quantum walks. **Heidelberg, Springer Berlin**, Springer, v. 10, p. 978–3, 2013. Disponível em: <<https://doi.org/10.1007/978-3-642-36014-5>>.
- WANG, Y. et al. Qudits and high-dimensional quantum computing. **Frontiers in Physics**, Frontiers Media SA, v. 8, p. 589504, 2020. Disponível em: <<https://doi.org/10.3389/fphy.2020.589504>>.
- WEIGOLD, M. et al. Encoding patterns for quantum algorithms. **IET Quantum Communication**, Wiley Online Library, v. 2, n. 4, p. 141–152, 2021. Disponível em: <<https://doi.org/10.1049/qtc2.12032>>.
- WIGNER, E. On the quantum correction for thermodynamic equilibrium. **Physical review**, APS, v. 40, n. 5, p. 749, 1932. Disponível em: <<https://doi.org/10.1103/PhysRev.40.749>>.
- YAN, P. et al. Quantum probability-inspired graph neural network for document representation and classification. **Neurocomputing**, Elsevier B.V., v. 445, p. 276–286, 7 2021. ISSN 18728286. Disponível em: <<https://doi.org/10.1016/j.neucom.2021.02.060>>.
- YANOFSKY, N. S.; MANNUCCI, M. A. **Quantum computing for computer scientists**. Cambridge University Press, 2008. Disponível em: <<https://doi.org/10.1017/CBO9780511813887>>.

- YIN, J. et al. Satellite-based entanglement distribution over 1200 kilometers. **Science**, American Association for the Advancement of Science, v. 356, n. 6343, p. 1140–1144, 2017. Disponível em: <<https://doi.org/10.1126/science.aan3211>>.
- YU, Y. et al. Entanglement of two quantum memories via fibres over dozens of kilometres. **Nature**, Nature Publishing Group, v. 578, n. 7794, p. 240–245, 2020. Disponível em: <<https://doi.org/10.1038/s41586-020-1976-7>>.
- ZHANG, W.; ZHAO, D.; WANG, X. Agglomerative clustering via maximum incremental path integral. **Pattern Recognition**, v. 46, p. 3056–3065, 11 2013. ISSN 00313203. Disponível em: <<https://doi.org/10.1016/j.patcog.2013.04.013>>.
- ZHANG, X. et al. Graphic lattices made by graph felicitous-type labelings and colorings of topological coding. **Discrete Applied Mathematics**, Elsevier B.V., v. 336, p. 37–46, 9 2023. ISSN 0166218X. Disponível em: <<https://doi.org/10.1016/j.dam.2023.03.023>>.
- ZHANG, Y. et al. Neqr: a novel enhanced quantum representation of digital images. **Quantum information processing**, Springer, v. 12, p. 2833–2860, 2013. Disponível em: <<https://doi.org/10.1007/s11128-013-0567-z>>.
- ZIDAN, M. et al. Rapid solution of logical equivalence problems by quantum computation algorithm. **Applied Soft Computing**, Elsevier Ltd, v. 132, 1 2023. ISSN 15684946. Disponível em: <<https://doi.org/10.1016/j.asoc.2022.109844>>.

Data and Image Processing Algorithms

This section presents algorithms for general data processing tasks such as sorting, and transformation. These algorithms are fundamental for efficiently handling and manipulating images.

A.1 Algorithms for General Data Processing

```
def normalizationB(values):  
    """  
    Normalize a vector using min-max scaling.  
  
    Parameters:  
    values (ndarray): Input vector to be normalized.  
  
    Returns:  
    ndarray: Normalized vector.  
    """  
    arr = np.array(values)  
    normalized_vector = arr / np.linalg.norm(arr)  
    normalized_vector = (normalized_vector - np.min(normalized_vector))  
    / (np.max(normalized_vector) - np.min(normalized_vector))  
  
    return normalized_vector  
  
def normalization(values):  
    """  
    Normalize a vector to integer values.  
  
    Parameters:
```

values (ndarray): Input vector to be normalized.

Returns:

ndarray: Normalized vector with integer values.

"""

```
arr = np.array(values)
```

```
arr = (arr - np.min(arr)) / (np.max(arr) - np.min(arr))
```

```
x_norm = np.floor(arr * 5000000000)
```

```
return x_norm
```

```
def normalizationAux(values , mean):
```

"""

Normalize a vector with a specified mean.

Parameters:

values (ndarray): Input vector to be normalized.

mean (float): Mean value to add during normalization.

Returns:

ndarray: Normalized vector.

"""

```
arr = np.array(values)
```

```
arr = (arr - np.min(arr)) / (np.max(arr) - np.min(arr))
```

```
arr = arr + mean
```

```
arr = arr / (np.max(arr))
```

```
x_norm = np.floor(arr * 255)
```

```
return x_norm
```

```
def calculate_pixel_correlation(original_pixel , encrypted_pixel):
```

"""

Calculate the correlation coefficient between two sets of pixels.

Parameters:

original_pixel (ndarray): Original set of pixels.

encrypted_pixel (ndarray): Encrypted set of pixels.

Returns:

float: Correlation coefficient between the two sets of pixels.

"""

```

O_mean = np.mean(original_pixel)
E_mean = np.mean(encrypted_pixel)

O_diff = original_pixel - O_mean
E_diff = encrypted_pixel - E_mean
D_O = np.sum(O_diff ** 2)
D_E = np.sum(E_diff ** 2)
COE = np.sum(O_diff * E_diff) / (np.sqrt(D_O) * np.sqrt(D_E))

return COE

def evaluate_pixel_correlation_D(original_image, encrypted_image,
num_pairs):
    """
    Evaluate the correlation coefficient between
    adjacent pixels diagonally.

    Parameters:
    original_image (ndarray): Original image.
    encrypted_image (ndarray): Encrypted image.
    num_pairs (int): Number of pixel pairs to evaluate.

    Returns:
    tuple: Tuple containing the sets of adjacent pixels,
    encrypted adjacent pixels.
    """
    assert original_image.shape == encrypted_image.shape,
    "Images must have the same shape"

    N = original_image.size # Total number of adjacent pixel pairs

    indices = np.random.choice(N, size=num_pairs, replace=False)

    correlation_coefficients = []

    adjacent_pixels = []
    encrypted_adjacent_pixels = []

    for index in indices:

```

```

    i , j = np.unravel_index(index , original_image.shape)

    is_encrypted_image = False
    if is_encrypted_image == False :
        adjacent_pixels.append(original_image[i , j])
        if(i>510):
            i=500
        if(j>510):
            j=500
        encrypted_adjacent_pixels.append(
            original_image[i + 1, j + 1])
    else :
        adjacent_pixels.append(encrypted_image[i , j])
        if(i>510):
            i=500
        if(j>510):
            j=500
        encrypted_adjacent_pixels.append(
            encrypted_image[i + 1, j + 1])

    COE = calculate_pixel_correlation(adjacent_pixels ,
    encrypted_adjacent_pixels)
    correlation_coefficients.append(COE)

    return adjacent_pixels , encrypted_adjacent_pixels ,
    correlation_coefficients

def evaluate_pixel_correlation_V(original_image , encrypted_image ,
num_pairs):
    """
    Evaluate the correlation coefficient between adjacent
    pixels vertically.

    Parameters:
    original_image (ndarray): Original image.
    encrypted_image (ndarray): Encrypted image.
    num_pairs (int): Number of pixel pairs to evaluate.

    Returns:

```

```

tuple: Tuple containing the sets of adjacent pixels ,
encrypted adjacent pixels , and correlation coefficients .
"""

# Check if the images have the same shape
assert original_image.shape == encrypted_image.shape ,
"Images must have the same shape"

N = original_image.size # Total number of adjacent pixel pairs

indices = np.random.choice(N, size=num_pairs, replace=False)
correlation_coefficients = []
adjacent_pixels=[]
encrypted_adjacent_pixels = []

for index in indices:
    i , j = np.unravel_index(index , original_image.shape)

    is_encrypted_image = False
    if is_encrypted_image == False :
        adjacent_pixels.append(original_image[i , j])
        if(i>510):
            i=500
        if(j>510):
            j=500
        encrypted_adjacent_pixels.append(original_image[i + 1, j])
    else:
        adjacent_pixels.append(encrypted_image[i , j])
        if(i>510):
            i=500
        if(j>510):
            j=500
        encrypted_adjacent_pixels.append(encrypted_image[i + 1, j])

COE = calculate_pixel_correlation(adjacent_pixels ,
encrypted_adjacent_pixels)
correlation_coefficients.append(COE)

return adjacent_pixels , encrypted_adjacent_pixels ,
correlation_coefficients

```

```

def evaluate_pixel_correlation_H(original_image , encrypted_image ,
num_pairs):
    """
    Evaluate the correlation coefficient between adjacent
    pixels horizontally.

    Parameters:
    original_image (ndarray): Original image.
    encrypted_image (ndarray): Encrypted image.
    num_pairs (int): Number of pixel pairs to evaluate.

    Returns:
    tuple: Tuple containing the sets of adjacent pixels , encrypted
    adjacent pixels , and correlation coefficients.
    """
    # Check if the images have the same shape
    assert original_image.shape == encrypted_image.shape ,
    "Images must have the same shape"

    N = original_image.size # Total number of adjacent pixel pairs

    indices = np.random.choice(N, size=num_pairs, replace=False)

    correlation_coefficients = []

    adjacent_pixels=[]
    encrypted_adjacent_pixels = []

    for index in indices:
        i , j = np.unravel_index(index , original_image.shape)

        is_encrypted_image = False
        if is_encrypted_image == False :
            adjacent_pixels.append(original_image[i , j])
            if(i>510):
                i=500
            if(j>510):
                j=500

```

```

        encrypted_adjacent_pixels.append(original_image[i, j + 1])
    else:
        adjacent_pixels.append(encrypted_image[i, j])
        if(i>510):
            i=500
        if(j>510):
            j=500
        encrypted_adjacent_pixels.append(encrypted_image[i, j + 1])
COE = calculate_pixel_correlation(adjacent_pixels ,
encrypted_adjacent_pixels)
correlation_coefficients.append(COE)

return adjacent_pixels , encrypted_adjacent_pixels ,
correlation_coefficients

```

A.2 Algorithms for Image Processing

This section presents the implemented algorithm for generation image encryption and decryption. For more details, refer to Section 3.2.2.

```

def encryptImage(image , values ):
    """
    Algorithm 2 for data encryption based on XOR operation.

    Parameters:
    image (ndarray): Original image.
    values (list): Key values for encryption.

    Returns:
    ndarray: Encrypted image.
    """
    # Initialize an array to store transformed data
    data = np.zeros((image.shape[0] , image.shape[1]))

    # Convert RGB values of each pixel into a single integer
    # and store in data array
    for i in range(0, image.shape[0]):
        for j in range(0, image.shape[1]):
            data[i, j] = int(image[i, j, 0] * 255) * 1000000 +
                int(image[i, j, 1] * 255) * 1000 +

```

```

        int(image[i, j, 2] * 255)

# Reshape data into 1D array
data = np.reshape(data, (-1))

# XOR operation between data and key values
aux = [int(data[i]) ^ int(values[i]) for i in range(0, len(data))]

# Reconstruct the encrypted image from the XOR result
imageOut = np.zeros(image.shape, dtype=np.uint16)
cont = 0
for i in range(0, image.shape[0]):
    for j in range(0, image.shape[1]):
        imageOut[i, j, 0] = int(aux[cont] / 1000000)
        imageOut[i, j, 1] = int((aux[cont] % 1000000) / 1000)
        imageOut[i, j, 2] = int(aux[cont] % 1000)
        cont = cont + 1

return imageOut

def decryptImage(image, values):
    """
    Algorithm 3 for data decryption based on XOR operation.

    Parameters:
    image (ndarray): Encrypted image.
    values (list): Key values used for encryption.

    Returns:
    ndarray: Decrypted image.
    """
    # Initialize an array to store transformed data
    data = np.zeros((image.shape[0], image.shape[1]))

    # Convert RGB values of each pixel into a single integer
    # and store in data array
    for i in range(0, image.shape[0]):
        for j in range(0, image.shape[1]):

```

```

        data[i, j] = int(image[i, j, 0]) * 1000000 +
        int(image[i, j, 1]) * 1000 +
        int(image[i, j, 2])

# Reshape data into 1D array
data = np.reshape(data, (-1))

# XOR operation between data and key values for decryption
aux = [int(data[i]) ^ int(values[i]) for i in range(0, len(data))]

# Reconstruct the decrypted image from the XOR result
imageOut = np.zeros(image.shape, dtype=np.uint16)
cont = 0
for i in range(0, image.shape[0]):
    for j in range(0, image.shape[1]):
        imageOut[i, j, 0] = int(aux[cont] / 1000000)
        imageOut[i, j, 1] = int((aux[cont] % 1000000) / 1000)
        imageOut[i, j, 2] = int(aux[cont] % 1000)
        cont = cont + 1

return imageOut

def imageToVector(image, index):
    """
    Convert a 2D image to a 1D vector using a specified color channel.

    Parameters:
    image (ndarray): Input 2D image.
    index (int): Index of the color channel to extract.

    Returns:
    ndarray: 1D vector representing the specified color channel.
    """
    bandR = image[:, :, index] * 255
    bandR = bandR.astype(int)
    bandR = np.reshape(bandR, (-1))
    return bandR

def imageToVectorR(image):

```

```
"""
```

Convert a 2D image to a 1D vector using only the red color channel.

Parameters:

image (ndarray): Input 2D image.

Returns:

ndarray: 1D vector representing the red color channel.

```
"""
```

```
bandR = image[:, :, 0] * 255
bandR = bandR.astype(int)
bandR = np.reshape(bandR, (-1))
return bandR
```

```
def keyImage(image, values):
```

```
"""
```

Map a 1D vector back to an image.

Parameters:

image (ndarray): Original image (used to determine shape).

values (list): 1D vector to map back to the image.

Returns:

ndarray: Image reconstructed from the 1D vector.

```
"""
```

```
imageOut = np.zeros(image.shape, dtype=np.uint16)
cont = 0
for i in range(image.shape[0]):
    for j in range(image.shape[1]):
        imageOut[i, j, 0] = int(values[cont] / 1000000)
        imageOut[i, j, 1] = int((values[cont] % 1000000) / 1000)
        imageOut[i, j, 2] = int(values[cont] % 1000)
        cont += 1
return imageOut
```

```
def crop_random_part(image):
```

```
"""
```

Crop a random part from the image.

Parameters:

image (PIL.Image): Input image.

Returns:

ndarray: Cropped image.

```
"""
img_array = np.array(image)
height, width, channels = img_array.shape

max_crop_height = min(height // 1.5, height)
max_crop_width = min(width // 1.5, width)

top = np.random.randint(0, height - max_crop_height)
left = np.random.randint(0, width - max_crop_width)

crop_height = np.random.randint(1, max_crop_height + 1)
crop_width = np.random.randint(1, max_crop_width + 1)

cropped_image = img_array[top:top + crop_height,
left:left + crop_width]
img_array[top:top + crop_height, left:left + crop_width] = 0

return img_array
```

```
def crop_square(image, n):
```

```
    """
```

Crop a square from the middle of the image.

Parameters:

image (PIL.Image): Input image.

n (int): Size of the square to crop.

Returns:

ndarray: Modified image with the cropped square removed.

```
    """
```

```
img_array = np.array(image)

height, width, _ = img_array.shape
```

```

    top = (height - n) // 2
    left = (width - n) // 2
    bottom = top + n
    right = left + n

    cropped_image = img_array[top:bottom, left:right]
    modified_image = np.copy(img_array)

    modified_image[top:bottom, left:right] = 0

    return modified_image

def add_salt_and_pepper_noise(image, salt_vs_pepper=0.5, amount=0.004):
    """
    Add salt and pepper noise to the image.

    Parameters:
    image (PIL.Image): Input image.
    salt_vs_pepper (float): Proportion of salt vs pepper noise.
    amount (float): Amount of noise to add.

    Returns:
    ndarray: Image with salt and pepper noise added.
    """
    img_array = np.array(image)

    num_salt = np.ceil(amount * img_array.size * salt_vs_pepper)
    salt_coords = [np.random.randint(0, i - 1, int(num_salt))
                    for i in img_array.shape]
    img_array[salt_coords[0], salt_coords[1], :] = 255

    num_pepper = np.ceil(amount * img_array.size *
                          (1.0 - salt_vs_pepper))
    pepper_coords = [np.random.randint(0, i - 1, int(num_pepper))
                     for i in img_array.shape]
    img_array[pepper_coords[0], pepper_coords[1], :] = 0

    return img_array

```

Quantum Circuits Implementations

B.1 Quantum Circuits as PRNG

This section presents the implemented algorithm for generation of pseudo random numbers. For more details, refer to Section 3.1.2.

```
def createQuantumCircuit(n_qubits, rx, ry, rz):  
    """  
    Create a quantum circuit with specified parameters.  
    Parameters:  
  
    n_qubits (int): Number of qubits in the circuit.  
    rx (float): Angle for the RX gate.  
    ry (float): Angle for the RY gate.  
    rz (float): Angle for the RZ gate.  
  
    Returns:  
    QuantumCircuit: Quantum circuit with the specified gates.  
    """  
    circ = QuantumCircuit(n_qubits)  
    for i in range(n_qubits):  
        circ.h(i)  
        circ.rx(rx, i)  
        circ.ry(ry, i)  
        circ.rz(rz, i)  
        circ.h(i)  
  
    return circ
```

```

def generateVector(n_qubits, rx, ry, rz, vector_length):
    """
    Algorithm 1 Quantum Quasi-Probability Pseudo Random Generator

    Parameters:
    n_qubits (int): Number of qubits in the circuit.
    rx (float): Angle for the RX gate.
    ry (float): Angle for the RY gate.
    rz (float): Angle for the RZ gate.
    vector_length (int): Length of the quantum vector to generate.

    Returns:
    (QuantumCircuit, list): Tuple containing the quantum circuit used
    and the generated quantum vector.
    """
    values = []
    cont = 0
    theta = 1 / 100

    backend = Aer.get_backend('statevector_simulator')

    while len(values) < vector_length:
        r_x = cont * theta
        r_y = cont * theta
        r_z = cont * theta
        circ = createQuantumCircuit(n_qubits, rx, ry, rz)
        for i in range(n_qubits):
            circ.h(i)
            circ.rx(r_x, i)
            circ.ry(r_y, i)
            circ.rz(r_z, i)
            circ.h(i)
        counts = backend.run(circ, shots=512).result().get_counts()
        new_values = list(counts.values())
        values.extend(new_values)
        cont += 1
    return circ, values

```

B.2 Multigraphs on Quantum Circuits

This section presents the implemented algorithm for the representation and manipulation of multigraph structures within quantum computing frameworks. For more details, refer to Section 3.3.4.

```
def getMultiGraphLine(u_input='00', binary_parts=[], backend=None):
    """
    Algorithm 4 for obtaining line graph.

    Parameters:
    u_input (str): Binary string for input u.
    multiGraphLine (list): List of transitions forming
    the multi-graph line.

    Returns:
    list: List of vertexes in the adjacent set.
    """
    set_of_transitions = []

    transitions = ['00', '10', '11', '01'] # Example transitions
    v_res = u_input[:-1]

    for part in binary_parts:
        for transition in transitions:
            t_res, v_res_aux = executeMultiGraph(backend,
                                                    shots=25,
                                                    ht=2,
                                                    hv=2,
                                                    t_input=transition,
                                                    u_input=v_res)

            if v_res_aux == part:
                set_of_transitions.append(transition)
                v_res = v_res_aux
                break
    return set_of_transitions
```

```

def getMultiGraphAdjacentSet(u_input='00',
                             multiGraphLine=[],
                             backend=None):
    """
    Algorithm 5 for obtaining adjacent vertices from line graph.

    Parameters:
    u_input (str): Binary string for input u.
    multiGraphLine (list): List of transitions forming the multi-graph li

    Returns:
    list: List of vertexes in the adjacent set.
    """
    set_of_vertex = []

    v_res = u_input

    for trn in multiGraphLine:
        t_res, v_res_aux = executeMultiGraph(backend, shots=25,
        ht=2, hv=2, t_input=trn, u_input=v_res)
        set_of_vertex.append(v_res_aux)
        v_res = v_res_aux

    return set_of_vertex

def transformImageToRotations(image, lisOfDictionaries=[],
listRotations=[]):
    """
    Algorithm 6 for selecting an adecuate set of rotations.

    Parameters:
    image (numpy.ndarray): Input image data.
    lisOfDictionaries (list): List of dictionaries for image
    transformation.
    listRotations (list): List of rotations for image transformation.

    Returns:
    list: List of transformed image data.
    """

```

```

image_shape = image.shape
list_Row = []
countTotal = 0
count = 0
for i in range(0, image_shape[0]-1, 2):
    list_Col = []
    for j in range(0, image_shape[1]-1, 2):
        listStates = generateListOfStates(image, fil=i, col=j)
        listDict = filter_dictionaries_by_labels(lisOfDictionaries,
                                                listStates)

        if len(listDict) > 0:
            firstDict = listDict[0]
            rotationIndex = get_index_exact(lisOfDictionaries,
                                             firstDict)

        else:
            count += 1
            countTotal += 1
print("total_not_reached", str(count/countTotal))
return list_Row

def generate_binary_numbers(n, k):
    """
    Generate binary numbers with n values and k digits.

    Parameters:
    n (int): Number of binary numbers to generate.
    k (int): Number of digits in each binary number.

    Returns:
    list: List of binary numbers.
    """
    binary_numbers = []
    for num in range(n):
        binary_numbers.append(format(num, '0{}b'.format(k)))
    return binary_numbers

def create_circuit(t, v):

```

```

"""
Create a quantum circuit based on binary inputs t and v.

Parameters:
t (str): Binary string for input t.
v (str): Binary string for input v.

Returns:
QuantumCircuit: Quantum circuit with the specified gates.
"""
qc = QuantumCircuit(4)

if t[0] == '1':
    qc.x(0)
if t[1] == '1':
    qc.x(1)

if v[0] == '1':
    qc.x(2)
if v[1] == '1':
    qc.x(3)

qc.cx(0, 1)
qc.cx(0, 2)
qc.x(1)
qc.cx(1, 3)

return qc

def executeMultiGraph(backend=backend, shots=256, ht=2, hv=2,
t_input='00', u_input='00'):
    """
    Execute the multi-graph quantum circuit and get results.

    Parameters:
    backend (Backend): Backend for quantum execution.
    shots (int): Number of shots for each measurement.
    ht (int): Number of qubits for t measurement.
    hv (int): Number of qubits for v measurement.

```

t_input (str): Binary string for input t.
u_input (str): Binary string for input u.

Returns:

tuple: Tuple containing results for t and v measurements.
"""

```
qc = create_circuit(t_input, u_input)
qc.measure_all()
job = backend.run(qc, shots=shots)
result = job.result()
counts = result.get_counts()
result_as_str = list(counts.keys())[0]
result_as_str = result_as_str[::-1]

t_result = result_as_str[:ht]
v_result = result_as_str[hv:]

return t_result, v_result
```

```
def executeMultiGraph(backend=backend, cir=None, shots=256,
ht=2, hv=2, t_input='00', u_input='00'):
```

"""

Execute the multi-graph quantum circuit and get results.

Parameters:

backend (Backend): Backend for quantum execution.
shots (int): Number of shots for each measurement.
ht (int): Number of qubits for t measurement.
hv (int): Number of qubits for v measurement.
t_input (str): Binary string for input t.
u_input (str): Binary string for input u.

Returns:

tuple: Tuple containing results for t and v measurements.
"""

```
qc = create_circuit(t_input, u_input)
qc.measure_all()
job = backend.run(qc, shots=shots)
result = job.result()
```

```

counts = result.get_counts()
result_as_str = list(counts.keys())[0]
result_as_str = result_as_str[::-1]

t_result = result_as_str[:ht]
v_result = result_as_str[hv:]

return t_result, v_result

def circuitHRy(listRotation):
    """
    Create a quantum circuit with H and Ry gates.

    Parameters:
    listRotation (list): List of rotation angles for the Ry gates.

    Returns:
    QuantumCircuit: Quantum circuit with the specified gates.
    """
    qc = QuantumCircuit(4)
    for qubit in range(4):
        qc.h(qubit)

    for qubit in range(4):
        qc.ry(pi/listRotation[qubit], qubit)

    qc.cx(0, 1)
    qc.cx(0, 2)
    qc.x(1)
    qc.cx(1, 3)

    return qc

def generate_grid(elements, dimensions):
    """
    Generate a grid of elements with given dimensions.

    Parameters:
    elements (list): List of elements to form the grid.

```

dimensions (int): Number of dimensions for the grid.

Returns:

list: List representing the grid of elements.

```
"""
return list(product(*([elements] * dimensions)))
```

```
def getProbabilities(backend=backend, shots=256, circuit=None,
listRotation=[]):
```

```
"""
Get the probabilities from a quantum circuit.
```

Parameters:

backend (Backend): Backend for quantum execution.

shots (int): Number of shots for each measurement.

circuit (function): Function to generate the quantum circuit.

listRotation (list): List of rotation angles for the circuit.

Returns:

dict: Dictionary containing probabilities for different measurement outcomes.

```
"""
qc = circuit(listRotation)
qc.measure_all()
job = backend.run(qc, shots=shots)
result = job.result()
counts = result.get_counts()

sorted_counts = dict(sorted(counts.items()))

return sorted_counts
```

```
def getProbConfiList(backend=backend, configurationList=[]):
    """
```

Get the probability configuration list.

Parameters:

backend (Backend): Backend for quantum execution.
configurationList (list): List of configurations for probability calculation.

Returns:

list: List of probability configurations.
"""

```
listOutputs = []
```

```
for rotations in configurationList:
```

```
    results = getProbabilities(backend=backend,
                               shots=1024,
                               circuit=circuitHRy,
                               listRotation=list(rotations))
    listOutputs.append(results)
```

```
return listOutputs
```

Protocols for Proposed Algorithms

C.1 Protocol for Quantum Quasi-Probability Pseudo Random Generator (QQ-PRG)

This section describes the execution protocol for the Quantum Quasi-Probability Pseudo Random Generator (QQ-PRG). The QQ-PRG is designed to generate pseudo-random numbers leveraging quantum quasi-probability distributions to enhance randomness and security.

```
# Generate a quantum circuit and a vector of values based on
# the given parameters
# Parameters:
#     n_qubits (int): Number of qubits in the circuit.
#     rx (float): Angle for the RX gate.
#     ry (float): Angle for the RY gate.
#     rz (float): Angle for the RZ gate.
#     vector_length (int): Length of the input vector.
#
# Returns:
#     tuple: Tuple containing the quantum circuit and the
#           vector of values.
circ , values = generateVector(3, 0.3333, 0.3333, 0.3333, len(band))

# Normalize the generated vector of values to create a symmetric key
#
# Parameters:
#     values (numpy.ndarray): Vector of values.
# Returns:
```

```
# numpy.ndarray: Normalized vector representing the symmetric key.
symmetric_key = normalization(values)
```

C.2 Protocol for Image Encryption and Decryption

This protocol outlines the steps for securely encrypting and decrypting images. The protocol ensures that images can be transmitted or stored securely, protecting them from unauthorized access.

```
# Open the image file "peppers.png"
#
# Parameters:
# filename (str): Path to the image file.
#
# Returns:
# PIL.Image.Image: Opened image object.
image = Image.open("peppers.png")

# Convert the image to a vector, extracting the red band (channel)
#
# Parameters:
# image (PIL.Image.Image): Image object.
#
# Returns:
# numpy.ndarray: Vector representing the red band.
band = imageToVectorR(image)

# Generate a quantum circuit and a vector of values based
# on the given parameters
#
# Parameters:
# n_qubits (int): Number of qubits in the quantum circuit.
# rx (float): Rotation angle for the RX gate.
# ry (float): Rotation angle for the RY gate.
# rz (float): Rotation angle for the RZ gate.
# band_length (int): Length of the band (vector).
#
# Returns:
# QuantumCircuit: Quantum circuit with the specified gates.
# list: Vector of values.
```

```

circ , values = generateVector(  n_qubits=3,
                                rx=0.3333,
                                ry=0.3333,
                                rz=0.3333,
                                band_length=len(band))

# Convert the symmetric key into an image format, matching the
# dimensions of the original image
#
# Parameters:
#     image: Original image data.
#     values: Normalized vector of values.
#
# Returns:
#     numpy.ndarray: Symmetric key image.

symmetric_key_as_image = keyImage(image, normalization(values))

# Encrypt the original image using the symmetric key image
#
# Parameters:
#     image: Original image data.
#     symmetric_key_image: Symmetric key image data.
#
# Returns:
#     numpy.ndarray: Encrypted image.

encrypted_image = encryptImage(image, symmetric_key_as_image)

# Decrypt the encrypted image using the values generated
# by the 3-qubit quantum circuit
#
# Parameters:
#     encrypted_image: Encrypted image data.
#     values3qubits: Vector of values generated
#     by the 3-qubit quantum circuit.
#
# Returns:

```

```
# numpy.ndarray: Decrypted image.
```

```
decrypted_image = decryptImage(encrypted_image, values3qubits)
```

C.3 Protocol for Data Encoding with Multi-Graph Structures

This protocol outlines the steps for encoding and compressing data using multi-graph structures. Multi-graph structures.

```
# Open the Lena image
```

```
#
```

```
# Parameters:
```

```
# filename (str): Path to the image file.
```

```
#
```

```
# Returns:
```

```
# PIL.Image.Image: Opened image object.
```

```
image = Image.open("Lena.png")
```

```
# Resize the image to 64x64 pixels with anti-aliasing
```

```
#
```

```
# Parameters:
```

```
# image (PIL.Image.Image): Original image object.
```

```
# size (tuple): New size of the image.
```

```
# anti_aliasing (bool, optional): Whether to apply
```

```
# anti-aliasing. Defaults to True.
```

```
#
```

```
# Returns:
```

```
# numpy.ndarray: Rescaled image array.
```

```
scaled_image = resize(image, (64, 64), anti_aliasing=True)
```

```
# Select the red color band and scale its values for quantum processing
```

```
#
```

```
# Parameters:
```

```
# image_array (numpy.ndarray): Image array.
```

```
#
```

```
# Returns:
```

```
# numpy.ndarray: Scaled red color band.
```

```
selected_band = np.uint8(scaled_image[:, :, 0] * 4)
```

[illegible]

```

# Obtain the four most probable states from the list of dictionaries
#
# Parameters:
#     list_of_dicts (list): List of dictionaries containing
#     probability configurations.
#
# Returns:
#     list: List of the four most probable states.
four_most_probable_states = obtener_primeras_4_claves(
                                list_of_dictionaries)

# Transform the selected image band into rotations based
# on the four most probable states and grid
#
# Parameters:
#     image_array (numpy.ndarray): Selected image band.
#     list_of_dictionaries (list): List of dictionaries of quantum states
#     list_rotations (list): List of rotations.
#
# Returns:
#     list: Transformed list of rotations.
list_image_to_rotation = transformImageToRotations(selected_band,
lisOfDictionaries=four_most_probable_states, listRotations=grid)

```