

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Davi de Pontes Pasquini

**Geração de Números Pseudoaleatórios em
Sistemas Quânticos Caóticos Aplicada à
Criptografia de Imagens**

Uberlândia, Brasil

2026

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Davi de Pontes Pasquini

**Geração de Números Pseudoaleatórios em Sistemas
Quânticos Caóticos Aplicada à Criptografia de Imagens**

Trabalho de conclusão de curso apresentado
à Faculdade de Computação da Universidade
Federal de Uberlândia, como parte dos requi-
sitos exigidos para a obtenção título de Ba-
charel em Ciência da Computação.

Orientador: Luis Fernando Faina

Coorientador: João Henrique de Sousa Pereira

Universidade Federal de Uberlândia – UFU

Faculdade de Computação

Bacharelado em Ciência da Computação

Uberlândia, Brasil

2026

Davi de Pontes Pasquini

Geração de Números Pseudoaleatórios em Sistemas Quânticos Caóticos Aplicada à Criptografia de Imagens

Trabalho de conclusão de curso apresentado à Faculdade de Computação da Universidade Federal de Uberlândia, como parte dos requisitos exigidos para a obtenção título de Bacharel em Ciência da Computação.

Trabalho aprovado. Uberlândia, Brasil, 01 de novembro de 2016:

Luis Fernando Faina
Orientador

Professor

Professor

Uberlândia, Brasil
2026

Resumo

Este trabalho investiga a geração de números pseudoaleatórios a partir de circuitos quânticos, com base na replicação do algoritmo *Quantum Quasi-Probability Pseudo Random Generator* (QQ-PRG), e propõe duas melhorias ao método original: a inserção de entrelaçamento quântico por meio de portas C-NOT e a substituição dos ângulos de rotação fixos por uma sequência caótica gerada pelo mapa logístico. A qualidade estatística das sequências produzidas por ambas as versões do algoritmo foi avaliada por meio da bateria de testes NIST SP 800-22, e sua aplicabilidade prática foi verificada em um esquema de encriptação simétrica de imagens digitais por meio da operação XOR, submetido a testes de histograma, correlação entre pixels adjacentes, ataque por recorte e adição de ruído. Os resultados mostram que o algoritmo original aprova apenas 2 dos 15 testes da bateria NIST, ao passo que o algoritmo modificado aprova a totalidade dos testes, evidenciando a eliminação de estruturas estatísticas indesejadas na sequência gerada. Essa melhoria se reflete na aplicação de encriptação de imagens, em que o algoritmo modificado elimina a correlação residual entre pixels horizontalmente adjacentes observada no algoritmo original. Conclui-se que a combinação de entrelaçamento quântico com uma fonte complementar de aleatoriedade caótica constitui uma estratégia eficaz e de baixo custo computacional para aprimorar geradores pseudoaleatórios quânticos de baixa complexidade.

Palavras-chave: Computação quântica, Números pseudoaleatórios, Mapa logístico, Caos determinístico, Encriptação de imagens.

Lista de ilustrações

Figura 1 – Representação de um <i>qubit</i> na Esfera de Bloch (Fonte: (BARCELOS; ANDRADE; BOAVENTURA,))	15
Figura 2 – Representação geométrica das rotações R_X , R_Y e R_Z na esfera de Bloch. (Fonte: <i>PennyLane</i>)	17
Figura 3 – Exemplo de circuito quântico composto por quatro <i>qubits</i> e quinze portas quânticas. (Fonte: Autor)	19
Figura 4 – Processo de encriptação de uma imagem utilizando operação XOR. (Fonte: Autor)	30
Figura 5 – Processo de decriptação da imagem por meio da reaplicação da sequência pseudo-aleatória. (Fonte: Autor)	31
Figura 6 – Sequência de Portas que compõem a Configuração do Sistema Quântico (Fonte: (CRUZ, 2024))	37
Figura 7 – Representação de um <i>qubit</i> e seu caminho de giro através de esfera de Bloch (Fonte: (CRUZ, 2024))	38
Figura 8 – Exemplo de circuito gerado pelo código (Fonte: Autor)	45
Figura 9 – Circuito modificado, com camada de entrelaçamento via C-NOT entre os ângulos de preparação (θ) e os ângulos pós-entrelaçamento (θ'), ambos extraídos da sequência caótica do mapa logístico (Fonte: Autor)	47
Figura 10 – Plataforma do <i>IBM Quantum Composer</i> (Fonte: Autor)	57
Figura 11 – Imagens usadas nos experimentos. (Fonte: <i>Universidad de Granada</i>)	61
Figura 12 – Histogramas das Imagens Originais. (Fonte: Autor)	63
Figura 12 – Histogramas das Imagens Originais (cont.). (Fonte: Autor)	64
Figura 13 – Imagens Encriptadas pelos dois algoritmos. (Fonte: Autor)	64
Figura 13 – Imagens Encriptadas pelos dois algoritmos (cont.). (Fonte: Autor)	65
Figura 14 – Histogramas das Imagens Encriptadas pelos dois algoritmos. (Fonte: Autor)	66
Figura 15 – Imagens Originais Decriptadas. (Fonte: Autor)	67
Figura 16 – Resultados do recorte utilizando diferentes tamanhos de região cifrada. (Fonte: Autor)	68
Figura 17 – Imagens encriptadas com ruído <i>salt-and-pepper</i> e suas versões decriptadas. (Fonte: Autor)	69

Lista de tabelas

Tabela 1 – Parâmetros para o algoritmo de geração de probabilidades	44
Tabela 2 – Parâmetros para o algoritmo de geração de probabilidades com as me- lhorias propostas	49
Tabela 3 – Parâmetros utilizados na configuração do algoritmo QQ-PRG original .	58
Tabela 4 – Parâmetros utilizados na configuração do algoritmo QQ-PRG modificado	58
Tabela 5 – Resultado da aplicação da bateria de testes NIST SP 800-22 no algo- ritmo original	59
Tabela 6 – Resultado da aplicação da bateria de testes NIST SP 800-22 no algo- ritmo modificado	60
Tabela 7 – Parâmetros utilizados no algoritmo QQ-PRG original para os testes de criptação e decriptação	62
Tabela 8 – Parâmetros utilizados no algoritmo QQ-PRG modificado para os testes de criptação e decriptação	62
Tabela 9 – Distribuição da Correlação para as imagens RGB	71
Tabela 10 – Distribuição da Correlação para as imagens em Escala de Cinza	72

Lista de abreviaturas e siglas

QQ-PRG	<i>Quantum Quasi-Probability Pseudo Random Generator</i>
PRNG	<i>Pseudo Random Number Generator</i>
TRNG	<i>True Random Number Generator</i>
QPP	<i>Quantum Permutation Pad</i>
C-NOT	<i>Controlled-NOT</i>
IBM	<i>International Business Machines Corporation</i>
LFSR	<i>Linear Feedback Shift Register</i>

Sumário

1	INTRODUÇÃO	10
2	FUNDAMENTAÇÃO TEÓRICA	13
2.1	Fundamentos de Sistemas Quânticos	13
2.1.1	<i>Qubits</i>	13
2.1.2	Espaço de Hilbert	14
2.1.3	Esfera de Bloch	15
2.1.4	Superposição Quântica	16
2.1.5	Portas Quânticas	16
2.1.5.1	Portas Quânticas X, Y e Z	16
2.1.5.2	Porta de Hadamard	17
2.1.5.3	Portas de Rotação	17
2.1.5.4	Porta R_X	18
2.1.5.5	Porta R_Y	18
2.1.5.6	Porta R_Z	18
2.1.6	Circuitos Quânticos	18
2.1.7	Distribuições Quasi-Probabilísticas (QD)	19
2.1.7.1	Função de Wigner	19
2.1.8	Cálculo de Quasi-Probabilidades	20
2.2	Encriptação de Dados	20
2.2.1	Números Aleatórios	20
2.2.1.1	<i>True Random Number Generation</i>	20
2.2.1.2	<i>Pseudo Random Number Generation</i>	21
2.2.2	O Mapa Logístico e a Teoria do Caos	21
2.2.2.1	Sistemas Dinâmicos Não Lineares e a Noção de Caos	21
2.2.2.2	O Mapa Logístico	22
2.2.2.3	Bifurcações e a Rota para o Caos	22
2.2.2.4	Expoente de Lyapunov e Sensibilidade às Condições Iniciais	23
2.2.2.5	O Mapa Logístico como Fonte de Números Pseudoaleatórios	23
2.2.3	Bateria de Testes NIST SP 800-22	24
2.2.3.1	Teste de Frequência (<i>Monobit</i>)	24
2.2.3.2	Teste de Frequência em Blocos	25
2.2.3.3	Teste de <i>Runs</i>	25
2.2.3.4	Teste da Maior <i>Run</i> de Uns em um Bloco	25
2.2.3.5	Teste de <i>Rank</i> de Matriz Binária	26

2.2.3.6	Teste da Transformada Discreta de <i>Fourier</i> (Espectral)	26
2.2.3.7	Teste de Correspondência de Padrões Não Sobrepostos	26
2.2.3.8	Teste de Correspondência de Padrões Sobrepostos	27
2.2.3.9	Teste Estatístico Universal de Maurer	27
2.2.3.10	Teste de Complexidade Linear	27
2.2.3.11	Teste Serial	28
2.2.3.12	Teste de Entropia Aproximada	28
2.2.3.13	Teste de Soma Cumulativa	28
2.2.3.14	Teste de Desvios Aleatórios	29
2.2.3.15	Teste de Desvios Aleatórios Variante	29
2.2.4	Esquemas de Encriptação e Decriptação de Imagens	30
2.2.5	Avaliação da Encriptação e Decriptação de Imagens	31
2.2.5.1	Análise de Histograma	31
2.2.5.2	Correlação entre Pixels Adjacentes	32
2.2.5.3	Teste de Ataque de Recorte	32
2.2.5.4	Teste de Análise de Ruído	33
2.3	Revisão Bibliográfica	33
3	MÉTODO	35
3.1	<i>Quantum Quasi-Probability Pseudo Random Generator</i>	35
3.1.1	Etapas Propostas Para a Geração de Números Pseudo-Aleatórios	35
3.1.2	Algoritmo Proposto	38
3.1.3	Entrelaçamento Quântico via Portas C-NOT	39
3.1.4	Randomização dos Ângulos de Rotação com o Mapa Logístico	40
3.2	Esquemas de Encriptação e Decriptação de Imagens	42
3.2.1	Normalização dos Valores Quasi-Probabilísticos	42
3.2.2	Algoritmo de Encriptação	42
3.2.3	Algoritmo de Decriptação	43
3.3	Computação Quântica no Qiskit	43
4	RESULTADOS E ANÁLISE DO QQPRG	44
4.1	Implementação do Algoritmo QQPRG	44
4.1.1	Geração do Conjunto de Probabilidades	44
4.1.2	Implementação das Melhorias: Entrelaçamento Quântico e Randomização Caótica dos Ângulos	46
4.1.3	Normalização do Vetor de Valores	49
4.1.4	Encriptação e Decriptação de Imagem usando XOR	50
4.1.5	Funções Auxiliares para Avaliação de Robustez	51
4.1.6	Análise de Histograma	52
4.1.7	Análise de Correlação entre Pixels Adjacentes	53

5	ANÁLISE DOS RESULTADOS	56
5.1	Configuração dos Experimentos	56
5.2	Gerador Pseudoaleatório Quântico de Quasi-Probabilidade (QQ-PRG)	57
5.2.1	Configuração para os Testes do Algoritmo QQ-PRG	58
5.2.2	Teste Estatístico do QQ-PRG	58
5.3	Testes de Encriptação e Decriptação de Imagens	60
5.3.1	Imagens Utilizadas nos Testes	61
5.3.2	Parâmetros QQ-PRG para Encriptação e Decriptação	61
5.3.3	Testes de Encriptação e Decriptação	62
5.3.3.1	Análise de Histogramas	62
5.3.3.2	Teste de Ataque de Recorte	67
5.3.3.3	Teste de Análise de Ruído	68
5.3.3.4	Análise de Correlação	70
6	CONCLUSÃO	74
	REFERÊNCIAS	77

1 Introdução

O tema inicialmente proposto para o trabalho é "Geração de Números Pseudoaleatórios em Sistemas Quânticos Caóticos Aplicada à Criptografia de Imagens". Este tema está inserido no ambiente da computação quântica, cujo objetivo é criar uma contraparte da computação tradicional incorporando novas propriedades como superposição, entrelaçamento, compressão e coerência nos algoritmos. Além disso, temos a área da geração de números pseudo-aleatórios PRNG (*Pseudo Random Number Generator*) e verdadeiramente aleatórios TRNG (*True Random Number Generator*), um dos tópicos mais importantes para criptografia de dados, sendo significativos para sistemas usados diariamente e impactando vários aspectos de nossas vidas. Para dar um sentido prático do uso do PRNG, propõe-se o seu uso na criptografia de imagens (CRUZ, 2024). Como extensão desse tema, este trabalho incorpora ainda conceitos da teoria do caos determinístico, em particular, o mapa logístico, como fonte complementar de aleatoriedade ao processo de geração quântica, investigando se a combinação entre aleatoriedade de origem quântica e aleatoriedade caótica de origem clássica é capaz de aprimorar as propriedades estatísticas do gerador original.

Este tema é importante no sentido que tem o potencial de avançar nossa compreensão e utilização de sistemas quânticos, particularmente através da exploração de estados quasi-probabilísticos. Neste estudo busca-se descobrir novas aplicações e tecnologias na computação quântica, as quais podem impactar significativamente campos como criptografia, segurança de dados e teoria computacional. Algumas subáreas importantes da computação quântica incluem circuitos quânticos e software, hardware quântico, *quantum information science* e *quantum machine learning* (CRUZ, 2024). Esses métodos, técnicas e algoritmos estão sendo usados para resolver problemas regulares como *quantum pattern matching* e oferecem soluções inovativas para problemas computacionais tradicionais (NIROULA; NAM, 2021; ZIDAN et al., 2023). Paralelamente, sistemas caóticos determinísticos, como o mapa logístico, têm sido amplamente empregados na literatura como fonte de aleatoriedade para fins criptográficos (ANDRECUT, 2012), em razão de sua sensibilidade às condições iniciais e de seu baixo custo computacional; a investigação de sua combinação com geradores quânticos representa, portanto, uma direção relevante tanto para a área de computação quântica quanto para a de criptografia caótica.

Relacionados à este tema temos alguns trabalhos como a tese de mestrado intitulada "*Multigraphs: Emergence from Hilbert Space Subdivision in Superposed Quantum Systems and their Image Encoding Application*" de Luis José Mantilla Santa Cruz pela Faculdade de Computação da Universidade Federal de Uberlândia (CRUZ, 2024) na qual discute-se a estratégia de utilizar sistemas quânticos aplicados na geração de números

pseudo-aleatórios para o encriptamento de imagens, além de realizar uma análise multigrafos derivada dos sistemas quânticos. Ademais, podemos citar também o artigo "*Pseudo Quantum Random Number Generator with Quantum Permutation Pad*" de Randy Kuang; Dafu Lou; Alex He; Chris McKenzie e Michael Redding publicados na *2021 - IEEE International Conference on Quantum Computing and Engineering (QCE)* (KUANG et al., 2021) onde adentram mais profundamente no tópico de geração de números aleatórios em sistemas quânticos com o protocolo *Quantum Permutation Pad* (QPP). No âmbito da geração de números pseudoaleatórios baseada em sistemas caóticos clássicos, destaca-se ainda o trabalho de Andrecut (ANDRECUT, 2012), que propõe o uso do mapa logístico em regime de caos completo como gerador pseudoaleatório, combinado a uma transformação de uniformização capaz de corrigir a distribuição não uniforme nativa do mapa, abordagem que fundamenta a melhoria proposta neste trabalho ao algoritmo de Mantilla Santa Cruz (CRUZ, 2024).

Para o trabalho, no tocante aos objetivos gerais, destacam-se quantificar as propriedades estatísticas dos estados quasi-probabilísticos e seus efeitos na aleatoriedade da geração de números pseudo-aleatórios em sistemas quânticos. Para exemplificar as características distintas de geradores de números pseudo-aleatórios associados com estados quasi-probabilísticos em sistemas quânticos, os mesmos são utilizados na encriptação e desencriptação de imagens. Como objetivo específico, propõe-se a investigação das potenciais vantagens e limitações de utilizar estados quasi-probabilísticos em sistemas quânticos para encriptação de imagem, com um foco na resiliência e robustez na presença de ataques. Adicionalmente, propõe-se como objetivo deste trabalho aprimorar o algoritmo gerador replicado por meio da introdução de entrelaçamento quântico, via portas C-NOT, e da substituição dos ângulos de rotação fixos por uma sequência caótica obtida a partir do mapa logístico, comparando quantitativamente o desempenho estatístico e a robustez criptográfica entre o algoritmo original e o algoritmo modificado propostos neste trabalho.

Durante o desenvolvimento do trabalho, será necessário um entendimento dos circuitos quânticos básicos bem como combinações de portas básicas na geração de circuitos quânticos para geração de números pseudo-aleatórios PRNG. Usaremos o algoritmo de circuitos quânticos como PRNG (CRUZ, 2024), implementado por meio da biblioteca *Qiskit*, usado para a geração de números pseudo-aleatórios em circuitos quânticos. A esse algoritmo propõe-se a incorporação de uma camada de entrelaçamento quântico e de uma fonte clássica de aleatoriedade caótica baseada no mapa logístico (ANDRECUT, 2012), cuja qualidade estatística será avaliada por meio da bateria de testes NIST SP 800-22. Para exemplificação da aplicação do gerador de números aleatórios na encriptação e desencriptação de imagens, propõe-se o uso de quatro imagens padrões comumente utilizadas pela comunidade científica. Por serem imagens com características bem conhecidas, a comparação dos resultados obtidos pode se dar com trabalhos correlatos que utilizem o mesmo conjunto de imagens. No desenvolvimento serão utilizados alguns algoritmos,

sendo parte deles para processamento geral de dados, fundamental para lidar e manipular eficientemente as imagens. Por fim, temos os algoritmos para processamento de imagens, usados para a geração da encriptação e descriptação da imagem, cujos resultados são avaliados por meio de testes de análise de histograma, correlação entre pixels adjacentes, ataque por recorte e adição de ruído ([CRUZ, 2024](#)).

2 Fundamentação Teórica

Computação Quântica é um campo que está evoluindo rapidamente e aproveita dos princípios da mecânica quântica para processar informações em maneiras fundamentalmente diferentes da computação clássica. No núcleo da computação quântica estão os *bits* quânticos, ou *qubits*, que podem existir em estados de superposição, permitindo que eles representem tanto 0 quanto 1 simultaneamente (YANOFSKY; MANNUCCI, 2008).

Computadores Quânticos realizam computações usando portas quânticas, que manipulam os estados quânticos dos *qubits*. Essas portas incluem operações como as da porta de *Hadamard*, porta *Controlled-NOT* (CNOT) e porta *phase*. Algoritmos quânticos exploram fenômenos quânticos como a superposição e entrelaçamento para resolver certos problemas exponencialmente mais rápido do que algoritmos clássicos. No entanto, a computação quântica ainda está em seus estágios iniciais, enfrentando desafios significativos como a incoerência de *qubits*, erros em operações quânticas e o desenvolvimento de *hardware* quântico escalável. Apesar desses desafios, a computação quântica possui a promessa de revolucionar os campos como criptografia, otimização e mais (NIELSEN; CHUANG, 2001).

2.1 Fundamentos de Sistemas Quânticos

Um sistema quântico pode ser descrito como uma combinação linear de estados únicos chamados estados bases. Estados comuns incluem $|0\rangle$ e $|1\rangle$. Um estado de superposição do sistema é dado pela combinação linear de seus estados base, onde, n é o número de *qubits*, c_i é um número complexo, e $|i\rangle$ é um elemento base (CRUZ, 2024).

$$|\psi\rangle = \sum_{i=0}^{2^n-1} c_i |i\rangle \quad (2.1)$$

2.1.1 Qubits

Um *qubit* é a unidade de informação fundamental em um sistema quântico de duas dimensões. Diferente dos *bits* clássicos, *qubits* podem existir em múltiplos estados simultaneamente, incluindo estados representados como 0, 1, ou uma combinação de ambos. Essa combinação é representada normalmente como uma combinação linear de dois estados base ortogonais, denotados como $|0\rangle$ e $|1\rangle$, com coeficientes complexos correspondentes α e β que são números complexos (Eq. 2.2).

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \quad (2.2)$$

Um *qubit* é representado como uma matriz 2 por 1 com números complexos (Eq. 2.3), onde os componentes α e β obedecem a restrição $|\alpha|^2 + |\beta|^2 = 1$ (CRUZ, 2024).

$$\begin{pmatrix} \alpha \\ \beta \end{pmatrix} \quad (2.3)$$

2.1.2 Espaço de Hilbert

O formalismo matemático da mecânica quântica é construído sobre o conceito de espaço de Hilbert, uma estrutura algébrica que fornece a linguagem precisa para descrever os estados de sistemas quânticos. Um espaço de Hilbert $\mathcal{H}$ é um espaço vetorial complexo dotado de um produto interno $\langle \cdot, \cdot \rangle$ que associa a cada par de vetores um número complexo, satisfazendo as propriedades de linearidade, conjugação hermitiana e positividade definida. Além disso, exige-se que o espaço seja completo em relação à norma induzida por esse produto interno, garantindo que sequências convergentes de vetores possuam limite dentro do próprio espaço (CRUZ, 2024).

Na mecânica quântica, cada estado físico possível de um sistema é representado por um vetor unitário em um espaço de Hilbert, usualmente denotado pela notação de Dirac como $|\psi\rangle$ (*ket*). O produto interno entre dois estados $|\phi\rangle$ e $|\psi\rangle$ é denotado por $\langle \phi | \psi \rangle$ (*bra-ket*), e seu módulo ao quadrado fornece a probabilidade de, ao medir o sistema no estado $|\psi\rangle$, encontrá-lo no estado $|\phi\rangle$.

Para um sistema composto de n qubits, o espaço de Hilbert associado é construído pelo produto tensorial dos espaços individuais de cada qubit:

$$\mathcal{H} = \mathcal{H}_1 \otimes \mathcal{H}_2 \otimes \cdots \otimes \mathcal{H}_n, \quad (2.4)$$

em que cada $\mathcal{H}_i \cong \mathbb{C}^2$ é o espaço de Hilbert bidimensional associado ao i -ésimo qubit. O espaço resultante tem dimensão 2^n , o que significa que um sistema de n qubits possui 2^n estados computacionais de base, número que cresce exponencialmente com n e está na origem do potencial de paralelismo da computação quântica. No contexto deste trabalho, com $n = 3$ qubits, o espaço de Hilbert do sistema tem dimensão $2^3 = 8$, e os estados quânticos do circuito são vetores nesse espaço de oito dimensões.

2.1.3 Esfera de Bloch

A Esfera de Bloch é uma representação geométrica do espaço de estados de um único qubit que permite visualizar de forma intuitiva as transformações produzidas pelas portas quânticas. Todo estado puro de um qubit pode ser escrito na forma:

$$|\psi\rangle = \cos\left(\frac{\theta}{2}\right) |0\rangle + e^{i\phi} \sin\left(\frac{\theta}{2}\right) |1\rangle, \quad (2.5)$$

em que $\theta \in [0, \pi]$ é o ângulo polar e $\phi \in [0, 2\pi)$ é o ângulo azimutal. Esses dois parâmetros definem unicamente um ponto na superfície de uma esfera unitária, chamada Esfera de Bloch (Figura 1): o polo norte da esfera corresponde ao estado $|0\rangle$ ($\theta = 0$), o polo sul corresponde ao estado $|1\rangle$ ($\theta = \pi$), e todo ponto intermediário na superfície corresponde a um estado de superposição entre os dois.

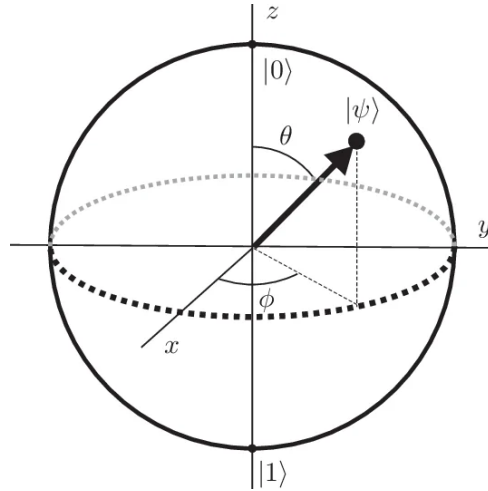


Figura 1 – Representação de um *qubit* na Esfera de Bloch (Fonte: (BARCELOS; ANDRADE; BOAVENTURA,))

A utilidade da Esfera de Bloch vai além da visualização: nela, qualquer porta quântica de um único qubit corresponde a uma rotação da esfera em torno de um de seus eixos. Em particular, as portas de rotação utilizadas neste trabalho ($R_x(\theta)$, $R_y(\theta)$ e $R_z(\theta)$) correspondem, respectivamente, a rotações pelo ângulo θ em torno dos eixos x , y e z da esfera. A porta de Hadamard, por sua vez, equivale a uma rotação que leva o polo norte ($|0\rangle$) até o equador da esfera, colocando o qubit em superposição uniforme entre $|0\rangle$ e $|1\rangle$.

É importante notar que a Esfera de Bloch representa adequadamente apenas o estado de um *único* qubit. Para sistemas com múltiplos qubits, o espaço de estados tem dimensão exponencialmente maior (conforme discutido na Seção 2.1.2) e não admite uma representação geométrica direta equivalente. Em particular, estados entrelaçados de múltiplos qubits, como os produzidos pela camada de portas C-NOT introduzida neste trabalho, não podem ser descritos como pontos individuais em esferas de Bloch separadas.

para cada qubit: eles são estados do espaço de Hilbert conjunto que não se decompõem em produtos tensoriais de estados individuais.

2.1.4 Superposição Quântica

Superposição quântica está entre um dos fenômenos mais cativantes da computação quântica. Ela permite a existência simultânea de múltiplos estados dentro de um sistema, concedendo acesso à um vasto conjunto de estados dentro do espaço de *Hilbert*. Um computador quântico possui a propriedade de que o número de potenciais estados aumenta exponencialmente com 2^n , onde n denota o número de *qubits* em superposição.

Por exemplo, um sistema composto de dois *qubits* em superposição pode ser descrito como uma combinação linear (Eq. 2.6). Aqui o conjunto de estados possíveis é denotado por $S = |00\rangle, |01\rangle, |10\rangle, |11\rangle$. Os coeficientes c_{ij} , que são números complexos, satisfazem a condição $|c_{00}|^2 + |c_{01}|^2 + |c_{10}|^2 + |c_{11}|^2 = 1$, com $|c_{ij}|^2$ representando a probabilidade de cada estado respectivo (CRUZ, 2024).

$$|\psi\rangle = c_{00}|00\rangle + c_{01}|01\rangle + c_{10}|10\rangle + c_{11}|11\rangle \quad (2.6)$$

2.1.5 Portas Quânticas

As portas quânticas são representadas por matrizes unitárias que possibilitam a manipulação de estados quânticos. Existem diversas portas quânticas utilizadas na computação quântica, cada uma com propósito específico e efeito distinto sobre o estado do *qubit*. A seguir, são detalhadas as portas quânticas X, Y, Z, *Hadamard* e as portas de rotação.

2.1.5.1 Portas Quânticas X, Y e Z

As portas quânticas X, Y e Z são operações fundamentais que atuam sobre *qubits*. Cada uma dessas portas corresponde a uma rotação de 180° em torno de um dos eixos da esfera de Bloch, que representa geometricamente o estado de um *qubit*.

A porta X realiza uma rotação de 180° (ou π radianos) em torno do eixo X da esfera de Bloch. Ela inverte os estados do *qubit*, transformando $|0\rangle$ em $|1\rangle$ e $|1\rangle$ em $|0\rangle$. Sua representação matricial é dada por:

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \quad (2.7)$$

A porta Y realiza uma rotação de 180° em torno do eixo Y. Essa operação combina inversão de estado com alteração de fase, transformando $|0\rangle$ em $i|1\rangle$ e $|1\rangle$ em $-i|0\rangle$, onde

i representa a unidade imaginária.

$$Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \quad (2.8)$$

A porta Z realiza uma rotação de 180° em torno do eixo Z. Ela preserva o estado $|0\rangle$ e introduz uma mudança de fase no estado $|1\rangle$, transformando-o em $-|1\rangle$.

$$Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \quad (2.9)$$

2.1.5.2 Porta de Hadamard

A porta *Hadamard* (H) aplica uma transformação que coloca o *qubit* em um estado de superposição. Ela transforma $|0\rangle$ em $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ e $|1\rangle$ em $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. Essa porta é fundamental para a criação de superposição, permitindo que o *qubit* seja descrito como combinação linear dos estados base.

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \quad (2.10)$$

2.1.5.3 Portas de Rotação

As portas quânticas R_X , R_Y e R_Z são operações de rotação contínua que atuam sobre um *qubit*, permitindo rotações arbitrárias em torno dos eixos X, Y e Z na esfera de Bloch (Figura 2). Essas portas generalizam as operações X, Y e Z, possibilitando o controle preciso do ângulo de rotação por meio de um parâmetro θ .

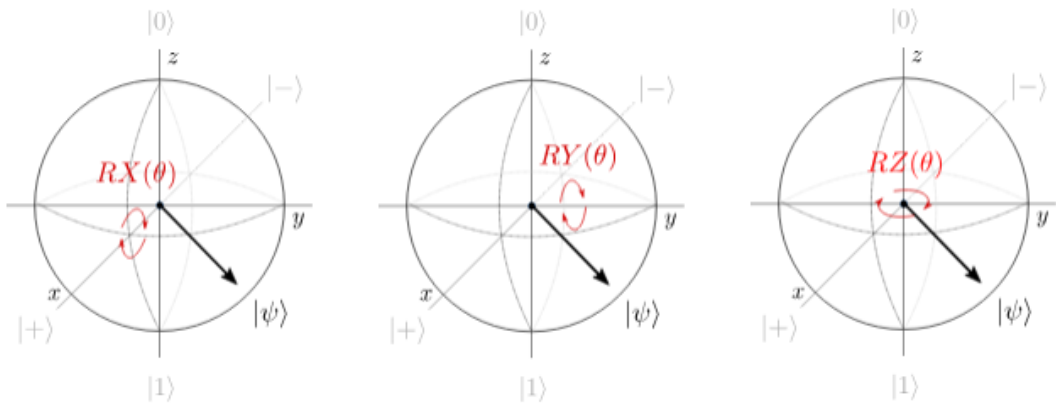


Figura 2 – Representação geométrica das rotações R_X , R_Y e R_Z na esfera de Bloch. (Fonte: [PennyLane](#))

2.1.5.4 Porta R_X

A porta $R_X(\theta)$ realiza uma rotação de θ radianos em torno do eixo X. Para $\theta = \pi$, obtém-se a porta X (até uma fase global). Sua representação matricial é:

$$R_X(\theta) = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & -i \sin\left(\frac{\theta}{2}\right) \\ -i \sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{pmatrix} \quad (2.11)$$

2.1.5.5 Porta R_Y

A porta $R_Y(\theta)$ realiza uma rotação de θ radianos em torno do eixo Y. Para $\theta = \pi$, obtém-se a porta Y (até uma fase global). Sua representação matricial é:

$$R_Y(\theta) = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & -\sin\left(\frac{\theta}{2}\right) \\ \sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{pmatrix} \quad (2.12)$$

2.1.5.6 Porta R_Z

A porta $R_Z(\theta)$ realiza uma rotação de θ radianos em torno do eixo Z. Diferentemente das portas R_X e R_Y , essa operação não altera as probabilidades de medição dos estados $|0\rangle$ e $|1\rangle$, mas modifica a fase relativa entre eles.

$$R_Z(\theta) = \begin{pmatrix} e^{-i\frac{\theta}{2}} & 0 \\ 0 & e^{i\frac{\theta}{2}} \end{pmatrix} \quad (2.13)$$

2.1.6 Circuitos Quânticos

Um circuito quântico é um conceito fundamental na computação quântica e no processamento de informação quântica. Ele serve como estrutura formal para a manipulação de *qubits* e para a execução de operações quânticas. Um circuito quântico é composto por um conjunto de *qubits* e uma sequência específica de portas quânticas aplicadas a esses estados.

Essas portas atuam individualmente ou em conjunto sobre os *qubits*, promovendo transformações unitárias que alteram a distribuição probabilística do sistema quântico. Dessa forma, circuitos quânticos constituem ferramentas essenciais para a exploração e análise de estados quânticos no espaço de Hilbert, sendo fundamentais para a resolução de diversos problemas na computação quântica.

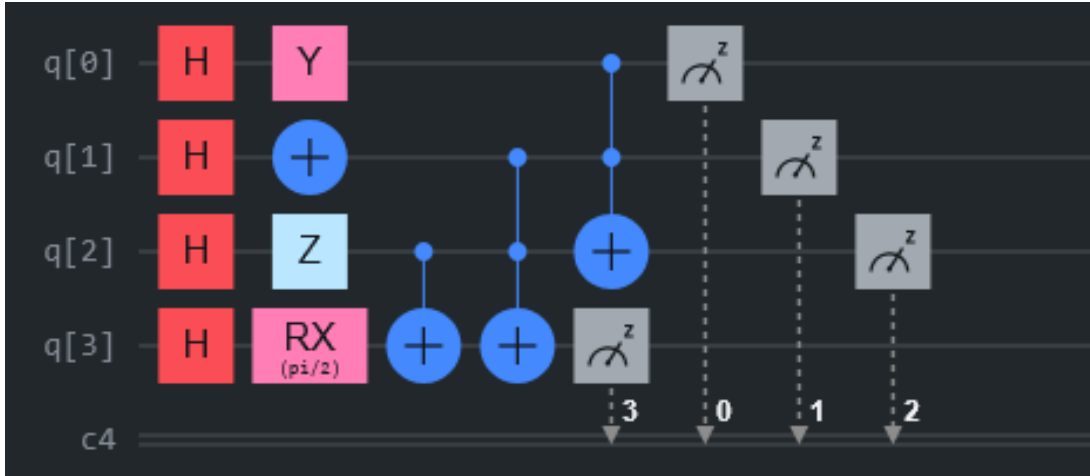


Figura 3 – Exemplo de circuito quântico composto por quatro *qubits* e quinze portas quânticas. (Fonte: Autor)

A execução de um circuito quântico envolve a aplicação sequencial das portas quânticas em uma ordem previamente definida. Ao longo dessa sequência, o estado dos *qubits* é progressivamente modificado, permitindo que computadores quânticos processem informações de maneira distinta dos computadores clássicos.

Os circuitos quânticos constituem a base formal dos algoritmos quânticos e desempenham papel central nas demonstrações de vantagem (ou supremacia) quântica. A Figura 3 ilustra um exemplo de circuito quântico simples (CRUZ, 2024).

2.1.7 Distribuições Quasi-Probabilísticas (QD)

Distribuições quasi-probabilísticas (QDs) fornecem um *framework* poderoso para descrever a distribuição conjunta de propriedades físicas que não podem ser simultaneamente conhecidas ou previstas com precisão arbitrária em sistemas quânticos. Um dos exemplos mais conhecidos e amplamente utilizados de QDs é a função de Wigner, que associa um estado quântico a uma distribuição quasi-probabilística (WIGNER, 1932; LEVY; LOSTAGLIO, 2020; CRUZ, 2024).

2.1.7.1 Função de Wigner

A função de Wigner, introduzida por Eugene Wigner em 1932, é uma QD fundamental na mecânica quântica. Ela fornece uma maneira de representar estados quânticos no espaço de fase, utilizando as variáveis de posição e momento. Diferentemente de distribuições probabilísticas clássicas, a função de Wigner pode assumir valores negativos, evidenciando a natureza não clássica inerente aos sistemas quânticos. Ela permite visualizar o estado quântico preservando analogias com distribuições clássicas, ao mesmo tempo em que mantém características fundamentais como superposição e princípio da incerteza (WIGNER, 1932; CRUZ, 2024).

A função de Wigner associada a um operador densidade ρ é definida como

$$W(x, p) = \frac{1}{\pi\hbar} \int_{-\infty}^{\infty} \langle x - y | \rho | x + y \rangle e^{\frac{2ipy}{\hbar}} dy, \quad (2.14)$$

onde x e p representam as coordenadas de posição e momento, respectivamente, e $\hbar$ é a constante de Planck reduzida.

2.1.8 Cálculo de Quasi-Probabilidades

O cálculo de distribuições quasi-probabilísticas, como a função de Wigner, geralmente envolve técnicas de reconstrução de estado baseadas em amostragem estatística. Esses métodos permitem estimar a distribuição quasi-probabilística associada a um estado quântico a partir de dados experimentais ou simulações numéricas (CRUZ, 2024).

2.2 Encriptação de Dados

A encriptação de dados é um componente fundamental da cibersegurança, sendo essencial para a proteção de informações sensíveis durante sua transmissão ou armazenamento. Números aleatórios desempenham um papel central em algoritmos criptográficos, fornecendo a imprevisibilidade necessária para a geração de chaves e para a construção de *ciphertexts* seguros.

2.2.1 Números Aleatórios

Números aleatórios são indispensáveis em diversas áreas da computação, particularmente na criptografia e em simulações computacionais, onde a imprevisibilidade é um requisito essencial. Existem dois métodos principais para geração de números aleatórios: a *True Random Number Generation* (TRNG) e a *Pseudo Random Number Generation* (PRNG).

2.2.1.1 True Random Number Generation

O objetivo de um TRNG é produzir números que sejam genuinamente imprevisíveis, mesmo sob conhecimento completo do sistema utilizado para sua geração. Em sistemas clássicos, isso geralmente envolve a exploração de fenômenos físicos intrinsecamente estocásticos, como ruído térmico, ruído atmosférico ou decaimento radioativo.

No contexto quântico, a geração de números verdadeiramente aleatórios explora a natureza intrinsecamente probabilística da mecânica quântica. Processos como a medição de estados em superposição, polarização de fótons individuais ou medições de *qubits*

preparados por portas de *Hadamard* permitem a obtenção de *bits* cuja distribuição é fundamentalmente imprevisível.

A aleatoriedade quântica decorre do colapso do estado quântico durante o processo de medição, o que garante, sob os postulados da mecânica quântica, alta entropia e ausência de determinismo clássico no resultado observado.

2.2.1.2 *Pseudo Random Number Generation*

A *Pseudo Random Number Generation* (PRNG) constitui um componente essencial de diversos sistemas criptográficos. Diferentemente de geradores verdadeiramente aleatórios, um PRNG é um algoritmo determinístico que produz sequências que aparentam ser aleatórias a partir de uma semente inicial.

Em aplicações criptográficas, PRNGs desempenham papel fundamental na geração de chaves, vetores de inicialização e demais parâmetros utilizados em processos de encriptação e decriptação. A segurança desses sistemas depende diretamente da qualidade estatística da sequência gerada e da imprevisibilidade da semente utilizada.

2.2.2 O Mapa Logístico e a Teoria do Caos

Esta subseção apresenta os fundamentos teóricos da dinâmica caótica necessários para compreender o uso do mapa logístico como fonte complementar de aleatoriedade neste trabalho. Parte-se da noção geral de caos determinístico em sistemas dinâmicos não lineares, passa-se pela definição formal do mapa logístico e pela rota de bifurcações que conduz ao regime caótico, e conclui-se discutindo as propriedades, sensibilidade às condições iniciais e distribuição estatística da órbita, que tornam esse mapa uma ferramenta adequada para a geração de números pseudoaleatórios.

2.2.2.1 Sistemas Dinâmicos Não Lineares e a Noção de Caos

Um sistema dinâmico descreve a evolução temporal do estado de um sistema a partir de uma regra determinística. Quando essa regra é não linear, mesmo sistemas extremamente simples (com poucas variáveis e equações de baixa ordem) podem apresentar comportamento aperiódico, sensível a perturbações mínimas e, na prática, imprevisível em horizontes longos de tempo. A esse fenômeno dá-se o nome de caos determinístico.

O caos não deve ser confundido com aleatoriedade no sentido estocástico: o sistema permanece inteiramente determinístico, isso é, conhecendo o estado atual e a regra de evolução, o estado seguinte é unicamente definido. O que caracteriza o caos é a combinação de três propriedades, formalizadas por (DEVANEY, 2021): (i) dependência sensível às condições iniciais, em que pequenas diferenças no estado inicial crescem exponencialmente ao longo das iterações; (ii) transitividade topológica, garantindo que o sistema explore de

forma densa o espaço de estados disponível; e (iii) existência de um conjunto denso de órbitas periódicas, ainda que a trajetória típica seja aperiódica. Essa combinação faz com que sequências geradas por sistemas caóticos exibam, na prática, propriedades estatísticas semelhantes às de sequências verdadeiramente aleatórias, o que motiva seu uso como fonte de números pseudoaleatórios e em esquemas de encriptação.

2.2.2.2 O Mapa Logístico

O mapa logístico é um dos exemplos mais estudados de sistema dinâmico caótico discreto e unidimensional. Originalmente derivado do modelo contínuo de crescimento populacional de Verhulst, sua versão discreta é definida pela relação de recorrência:

$$x_{n+1} = r \times x_n \times (1 - x_n) \quad (2.15)$$

em que $x_n \in [0, 1]$ representa o estado do sistema na n -ésima iteração e $r \in [0, 4]$ é o parâmetro de controle. Apesar de sua formulação algébrica elementar, trata-se de um simples polinômio quadrático, o mapa logístico exibe uma riqueza dinâmica surpreendente, que foi sistematizada no trabalho seminal de (MAY, 1976), amplamente reconhecido como um dos marcos fundadores da teoria do caos aplicada a sistemas biológicos, econômicos e físicos.

2.2.2.3 Bifurcações e a Rota para o Caos

O comportamento do mapa logístico depende criticamente do valor de r . Para $0 < r < 1$, todas as órbitas convergem para o ponto fixo trivial $x = 0$. Para $1 < r < 3$, o sistema converge para um único ponto fixo não trivial e estável. A partir de $r = 3$ esse ponto fixo perde estabilidade e o sistema passa a oscilar entre dois valores (uma órbita de período 2). Conforme r continua a crescer, ocorre uma sucessão infinita de bifurcações de duplicação de período (*period-doubling bifurcations*), em que o número de valores da órbita estável dobra a cada novo limiar de r (períodos 2, 4, 8, 16, ...).

Essa cascata de bifurcações se acumula em $r_\infty \approx 3.569946$, ponto a partir do qual o sistema entra no regime caótico, alternado por estreitas janelas de periodicidade. (FEIGENBAUM, 1978) demonstrou que a razão entre intervalos sucessivos de bifurcação converge para uma constante universal,

$$\delta \approx 4.6692016... \quad (2.16)$$

hoje conhecida como constante de Feigenbaum, que se repete em uma ampla classe de mapas unidimensionais com máximo quadrático, evidenciando que a rota para o caos por duplicação de período é uma propriedade universal, independente dos detalhes específicos do sistema.

Em $r = 4$, o mapa logístico atinge o regime de caos completo: toda a extensão do intervalo $[0, 1]$ é visitada de forma densa pela órbita típica, sem janelas de periodicidade remanescentes. É justamente esse regime ($r = 4$) o utilizado neste trabalho, pois garante a maior irregularidade estatística possível para o mapa.

2.2.2.4 Expoente de Lyapunov e Sensibilidade às Condições Iniciais

A característica mais marcante do regime caótico é a sensibilidade exponencial a perturbações no estado inicial, popularmente associada ao “efeito borboleta”. Essa propriedade é quantificada pelo expoente de Lyapunov, definido como:

$$\lambda = \lim_{N \rightarrow \infty} \frac{1}{N} \sum_{i=0}^{N-1} \ln |f'(x_i)| \quad (2.17)$$

em que f' é a derivada do mapa em relação a x . Valores de $\lambda > 0$ indicam divergência exponencial entre órbitas vizinhas e, portanto, regime caótico; $\lambda < 0$ indica convergência e estabilidade. Para o mapa logístico em $r = 4$, é possível mostrar analiticamente que $\lambda = \ln 2$, o valor máximo atingível por esse mapa, confirmando que esse ponto de operação corresponde ao regime de maior imprevisibilidade.

Essa sensibilidade tem uma consequência prática importante para a geração de números pseudoaleatórios: ainda que o mapa seja inteiramente determinístico e a fórmula geradora seja pública, pequenas diferenças na semente inicial (por exemplo, nos últimos bits de precisão de ponto flutuante) produzem sequências completamente distintas após poucas iterações, dificultando a antecipação do comportamento de longo prazo a partir de uma observação parcial do estado.

2.2.2.5 O Mapa Logístico como Fonte de Números Pseudoaleatórios

No regime $r = 4$, a densidade de probabilidade invariante do mapa logístico, isto é, distribuição estacionária de valores visitados por uma órbita típica não é uniforme, mas segue a chamada distribuição arco-seno:

$$f(x) = \frac{1}{\pi \sqrt{x(1-x)}} \quad (2.18)$$

Esse fato foi explorado por ([ANDRECUT, 2012](#)) ao propor o uso do mapa logístico como gerador de números pseudoaleatórios, introduzindo uma transformação de uniformização (baseada na transformação de Ulam-von Neumann) capaz de converter a saída não uniforme do mapa em uma sequência estatisticamente próxima de uma distribuição uniforme em $[0, 1]$. Essa abordagem inaugurou uma linha de pesquisa consolidada em geradores caóticos pseudoaleatórios e em esquemas de encriptação baseados em caos,

posteriormente estendida por trabalhos como os de (PATIDAR; SUD; PAREEK, 2009) e (Wang; Yang, 2012), que propuseram mecanismos de extração de bits e combinação de múltiplos mapas para mitigar fragilidades associadas à previsibilidade de sistemas caóticos de baixa dimensão quando usados de forma isolada.

É justamente essa propriedade, a capacidade de produzir, a partir de uma regra determinística simples e de baixo custo computacional, sequências com excelentes propriedades estatísticas de aleatoriedade, que motiva sua utilização neste trabalho como camada complementar ao gerador pseudoaleatório quântico proposto, reforçando a difusão estatística da sequência final por meio das propriedades egórdicas e de mistura (*mixing*) do caos determinístico.

2.2.3 Bateria de Testes NIST SP 800-22

A avaliação da confiabilidade e da qualidade estatística de sequências aleatórias é fundamental para sua utilização em aplicações criptográficas. Uma das metodologias mais reconhecidas para essa finalidade é a bateria de testes definida pelo NIST SP 800-22, composta por 15 testes estatísticos projetados para analisar propriedades essenciais de aleatoriedade em sequências binárias.

Esses testes incluem, entre outros, o teste de frequência (*Monobit*), o teste de *runs*, testes de complexidade linear e análises espectrais. O objetivo é verificar se a sequência apresenta uniformidade estatística, independência entre *bits* e ausência de padrões detectáveis que possam comprometer sua segurança.

A aprovação nessa bateria de testes não garante segurança criptográfica absoluta; contudo, fornece evidências estatísticas de que a sequência apresenta comportamento compatível com um processo aleatório ideal.

2.2.3.1 Teste de Frequência (*Monobit*)

O teste de frequência, também conhecido como *Monobit Test*, avalia a proporção de zeros e uns ao longo de toda a sequência binária. Seu objetivo é verificar se a quantidade de *bits* iguais a 1 é aproximadamente igual à quantidade de *bits* iguais a 0, como esperado em uma sequência verdadeiramente aleatória.

Mais especificamente, o teste analisa a proximidade da fração de *bits* iguais a 1 em relação ao valor esperado de $\frac{1}{2}$. Desvios significativos desse valor indicam a presença de viés na sequência, comprometendo sua aleatoriedade.

Este teste é considerado fundamental, pois muitos dos testes subsequentes assumem que a sequência não apresenta desequilíbrio significativo entre zeros e uns (RUKHIN et al., 2010).

2.2.3.2 Teste de Frequência em Blocos

O teste de frequência em blocos avalia a proporção de *bits* iguais a 1 em subsequências de tamanho fixo M (blocos de M *bits*). O objetivo é verificar se, em cada bloco, a quantidade de *bits* iguais a 1 é aproximadamente igual a $\frac{M}{2}$, conforme esperado para uma sequência verdadeiramente aleatória.

Diferentemente do teste de frequência global (*Monobit*), que considera a sequência completa, este teste permite identificar variações locais na distribuição de *bits*. Desvios significativos em blocos individuais podem indicar não-uniformidade na sequência.

Para o caso particular em que $M = 1$, o teste de frequência em blocos se reduz ao teste de frequência (*Monobit*) (RUKHIN et al., 2010).

2.2.3.3 Teste de *Runs*

O teste de *runs* avalia a quantidade de sequências consecutivas de bits idênticos presentes na sequência. Uma *run* é definida como uma subsequência contínua de bits iguais, limitada por bits de valor oposto em ambas as extremidades.

Uma *run* de comprimento k consiste em exatamente k bits idênticos consecutivos. O objetivo desse teste é verificar se o número total de *runs*, bem como a distribuição de seus comprimentos, está de acordo com o esperado para uma sequência verdadeiramente aleatória.

Em particular, o teste permite identificar se a alternância entre zeros e uns ocorre de forma excessivamente rápida ou lenta, o que pode indicar desvios do comportamento aleatório ideal (RUKHIN et al., 2010).

2.2.3.4 Teste da Maior *Run* de Uns em um Bloco

O teste da maior *run* de uns em um bloco tem como objetivo avaliar o comprimento máximo de sequências consecutivas de bits iguais a 1 dentro de blocos de tamanho fixo M .

Mais especificamente, o teste verifica se o comprimento da maior *run* de uns observado em cada bloco é consistente com o valor esperado para uma sequência verdadeiramente aleatória.

Desvios significativos nesse comprimento podem indicar a presença de padrões ou dependências na sequência, comprometendo sua aleatoriedade.

Ressalta-se que uma irregularidade no comprimento da maior *run* de uns implica, de forma equivalente, em irregularidades na maior *run* de zeros. Dessa forma, a análise considerando apenas as *runs* de uns é suficiente para a avaliação do comportamento da sequência (RUKHIN et al., 2010).

2.2.3.5 Teste de *Rank* de Matriz Binária

O teste de *rank* de matriz binária avalia o posto (*rank*) de submatrizes binárias disjuntas extraídas da sequência original. O objetivo é verificar a existência de dependência linear entre segmentos de tamanho fixo da sequência.

Para isso, a sequência binária é organizada em matrizes de dimensão predefinida, e o posto de cada matriz é calculado. Em uma sequência verdadeiramente aleatória, espera-se que a maioria dessas matrizes possua posto máximo ou próximo do máximo. Desvios desse comportamento indicam a presença de dependência linear entre os bits, o que compromete a aleatoriedade da sequência.

Este teste também está presente em outras baterias de avaliação estatística, como o conjunto *DIEHARD* (RUKHIN et al., 2010).

2.2.3.6 Teste da Transformada Discreta de *Fourier* (Espectral)

O teste da Transformada Discreta de *Fourier*, também conhecido como teste espectral, analisa a presença de picos no espectro de frequências da sequência binária.

Seu objetivo é detectar padrões periódicos ou estruturas repetitivas na sequência, os quais podem indicar desvios em relação ao comportamento esperado de uma sequência verdadeiramente aleatória.

O teste baseia-se na contagem do número de picos no espectro cuja magnitude excede um determinado limiar, correspondente a um nível de confiança de 95%. Para uma sequência aleatória ideal, espera-se que aproximadamente 5% dos picos ultrapassem esse limite. Desvios significativos dessa proporção indicam a presença de periodicidade ou correlação indesejada na sequência (RUKHIN et al., 2010).

2.2.3.7 Teste de Correspondência de Padrões Não Sobrepostos

O Teste de Correspondência de Padrões Não Sobrepostos tem como objetivo analisar a frequência de ocorrência de padrões binários pré-definidos em uma sequência. Esse teste é utilizado para identificar desvios de aleatoriedade associados à presença excessiva de estruturas específicas.

Para sua execução, a sequência binária é examinada por meio de uma janela deslizante de m bits, a qual busca a ocorrência de um padrão específico também de comprimento m . Quando o padrão não é encontrado, a janela é deslocada em um único bit. Por outro lado, quando ocorre uma correspondência, a janela é deslocada diretamente para a posição imediatamente após o final do padrão identificado, garantindo que as ocorrências contabilizadas não se sobreponham.

Esse procedimento diferencia-se do teste de Correspondência de Padrões Sobre-

postos (Seção 2.2.3.8), no qual a janela é sempre deslocada de apenas um *bit*, independentemente da ocorrência do padrão, permitindo sobreposição entre as correspondências (RUKHIN et al., 2010).

2.2.3.8 Teste de Correspondência de Padrões Sobrepostos

O Teste de Correspondência de Padrões Sobrepostos tem como objetivo analisar a frequência de ocorrência de um padrão binário pré-definido em uma sequência.

Assim como o Teste de Correspondência de Padrões Não Sobrepostos (Seção 2.2.3.7), este método utiliza uma janela deslizante de m bits para buscar um padrão específico de mesmo tamanho. Quando o padrão não é encontrado, a janela é deslocada em um único *bit*.

A principal diferença entre os dois testes reside no tratamento das ocorrências encontradas. No caso do teste sobreposto, mesmo após a identificação de uma correspondência, a janela é deslocada em apenas um *bit*, permitindo que ocorrências consecutivas compartilhem bits em comum. Dessa forma, o teste considera padrões sobrepostos ao longo da sequência, aumentando sua sensibilidade à presença de repetições estruturais (RUKHIN et al., 2010).

2.2.3.9 Teste Estatístico Universal de Maurer

O Teste Estatístico Universal de Maurer tem como objetivo avaliar a aleatoriedade de uma sequência por meio de sua compressibilidade.

Esse teste baseia-se na análise das distâncias entre ocorrências repetidas de padrões binários, interpretadas como uma medida indireta da capacidade de compressão da sequência. A ideia central é que sequências verdadeiramente aleatórias não apresentam redundâncias significativas e, portanto, não podem ser comprimidas de forma eficiente.

Caso a sequência apresente padrões recorrentes que permitam uma compressão significativa, isso indica a presença de estruturas não aleatórias. Assim, sequências altamente compressíveis são consideradas inadequadas do ponto de vista de aleatoriedade (RUKHIN et al., 2010).

2.2.3.10 Teste de Complexidade Linear

O Teste de Complexidade Linear tem como objetivo avaliar a complexidade de uma sequência binária por meio do comprimento do menor registro de deslocamento com realimentação linear (*Linear Feedback Shift Register* — LFSR) capaz de gerá-la.

A ideia central é que sequências verdadeiramente aleatórias apresentam alta complexidade linear, ou seja, requerem LFSRs de grande comprimento para sua reprodução. Por outro lado, sequências que podem ser geradas por LFSRs curtos possuem estrutura

interna significativa, indicando a presença de padrões e, conseqüentemente, desvios em relação ao comportamento aleatório esperado.

Assim, valores de complexidade linear significativamente baixos sugerem que a sequência não é adequada do ponto de vista de aleatoriedade (RUKHIN et al., 2010).

2.2.3.11 Teste Serial

O Teste Serial tem como objetivo analisar a frequência de ocorrência de todos os possíveis padrões binários de comprimento m ao longo da sequência.

Para isso, considera-se uma janela deslizante que percorre toda a sequência, permitindo a sobreposição entre os padrões analisados. O teste verifica se o número de ocorrências dos 2^m possíveis padrões de m bits é aproximadamente uniforme, como esperado em uma sequência verdadeiramente aleatória.

Em sequências ideais, cada padrão de comprimento m deve possuir a mesma probabilidade de ocorrência. Desvios significativos dessa uniformidade indicam a presença de estruturas ou dependências na sequência, comprometendo sua aleatoriedade.

Cabe destacar que, para $m = 1$, o Teste Serial se reduz ao Teste de Frequência (Monobit), apresentado na Seção 2.2.3.1 (RUKHIN et al., 2010).

2.2.3.12 Teste de Entropia Aproximada

O Teste de Entropia Aproximada tem como objetivo avaliar a regularidade e a previsibilidade de uma sequência binária por meio da análise da frequência de ocorrência de padrões sobrepostos.

De forma semelhante ao Teste Serial (Seção 2.2.3.11), este teste considera todos os padrões possíveis de comprimento m em uma janela deslizante com sobreposição ao longo da sequência. No entanto, sua principal característica é a comparação entre as frequências observadas de padrões de tamanho m e $m + 1$.

A ideia central é que, em uma sequência verdadeiramente aleatória, a introdução de um *bit* adicional não deve alterar significativamente a estrutura estatística dos padrões. Dessa forma, o teste mede a variação na entropia entre blocos consecutivos, sendo que desvios significativos indicam a presença de padrões ou dependências na sequência (RUKHIN et al., 2010).

2.2.3.13 Teste de Soma Cumulativa

O Teste de Soma Cumulativa (*Cumulative Sums Test*) tem como objetivo avaliar o desvio máximo de uma caminhada aleatória associada à sequência binária analisada.

Para isso, a sequência é transformada em uma sequência de valores $\{-1, +1\}$, na qual cada *bit* é mapeado para um desses valores. Em seguida, calcula-se a soma cumulativa desses termos, interpretando o resultado como uma caminhada aleatória unidimensional.

O teste verifica se o desvio máximo dessa caminhada em relação ao ponto de origem (zero) está de acordo com o comportamento esperado para uma sequência aleatória. Em sequências verdadeiramente aleatórias, espera-se que as flutuações ocorram de forma equilibrada em torno de zero. Por outro lado, desvios excessivamente grandes indicam a presença de tendências ou desequilíbrios na sequência, caracterizando não aleatoriedade (RUKHIN et al., 2010).

2.2.3.14 Teste de Desvios Aleatórios

O Teste de Desvios Aleatórios (*Random Excursions Test*) tem como objetivo analisar o comportamento de uma caminhada aleatória derivada da sequência binária.

Inicialmente, a sequência de *bits* é transformada em uma sequência de valores $\{-1, +1\}$. Em seguida, calcula-se a soma cumulativa desses valores, interpretando o resultado como uma caminhada aleatória unidimensional.

Um ciclo dessa caminhada é definido como o segmento que se inicia em zero e retorna a zero pela primeira vez. Dentro de cada ciclo, o teste contabiliza o número de visitas a determinados estados intermediários.

O objetivo é verificar se o número de visitas a cada estado específico está de acordo com o esperado para uma sequência verdadeiramente aleatória. O teste é composto por oito subtestes independentes, correspondentes aos estados $-4, -3, -2, -1, +1, +2, +3$ e $+4$. Desvios significativos nessas contagens indicam a presença de padrões ou dependências na sequência (RUKHIN et al., 2010).

2.2.3.15 Teste de Desvios Aleatórios Variante

O Teste de Desvios Aleatórios Variante (*Random Excursions Variant Test*) tem como objetivo analisar o número total de visitas a determinados estados em uma caminhada aleatória derivada da sequência binária.

Assim como no Teste de Desvios Aleatórios (Seção 2.2.3.14), a sequência de *bits* é inicialmente transformada em valores $\{-1, +1\}$, e a soma cumulativa é interpretada como uma caminhada aleatória unidimensional. No entanto, diferentemente do teste anterior, este não considera ciclos, mas sim o número total de vezes que cada estado é visitado ao longo de toda a sequência.

O teste verifica se o número de visitas a cada estado está de acordo com o esperado para uma sequência verdadeiramente aleatória. Ele é composto por dezoito subtestes independentes, correspondentes aos estados $-9, -8, \dots, -1$ e $+1, +2, \dots, +9$. Desvios

significativos nessas contagens indicam a presença de padrões ou dependências na sequência (RUKHIN et al., 2010).

2.2.4 Esquemas de Encriptação e Decriptação de Imagens

A encriptação de imagens é fundamental para proteger dados visuais contra acessos não autorizados, interceptação ou adulteração. A utilização de sequências pseudo-aleatórias em conjunto com operações criptográficas simples, como a operação XOR, constitui um método computacionalmente eficiente e amplamente empregado para esse propósito.

A operação XOR é uma operação binária definida bit a bit entre dois operandos. No contexto da encriptação de imagens digitais, cada pixel da imagem original é representado em formato binário e combinado com uma sequência de *bits* pseudo-aleatórios gerada por um PRNG. O processo pode ser descrito como

$$C(i, j) = I(i, j) \oplus K(i, j), \quad (2.19)$$

onde $I(i, j)$ representa o pixel original na posição (i, j) , $K(i, j)$ é o valor pseudo-aleatório correspondente e $C(i, j)$ é o pixel cifrado.

Esse procedimento (Figura 4) introduz aleatoriedade na estrutura da imagem, ocultando completamente seu conteúdo visual quando a sequência K apresenta propriedades adequadas de aleatoriedade.

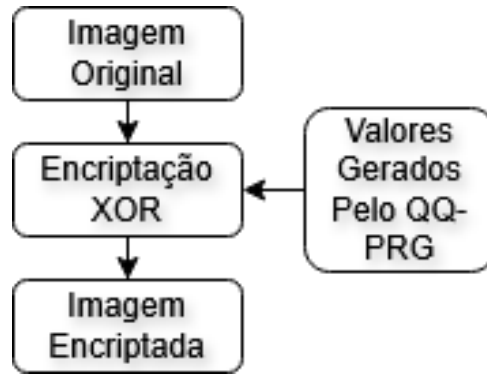


Figura 4 – Processo de encriptação de uma imagem utilizando operação XOR. (Fonte: Autor)

A decriptação (Figura 5) é realizada aplicando novamente a operação XOR com a mesma sequência pseudo-aleatória:

$$I(i, j) = C(i, j) \oplus K(i, j), \quad (2.20)$$

explorando a propriedade involutiva do operador XOR.

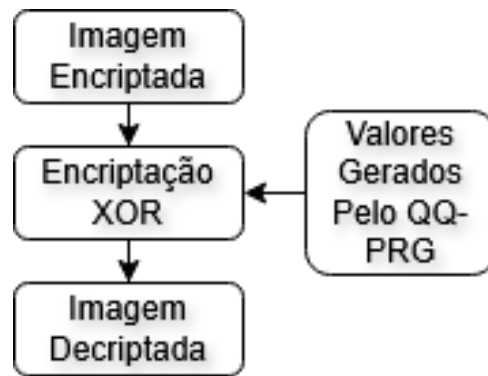


Figura 5 – Processo de decriptação da imagem por meio da reaplicação da sequência pseudo-aleatória. (Fonte: Autor)

2.2.5 Avaliação da Encriptação e Decriptação de Imagens

A avaliação da eficácia do esquema de encriptação é realizada por meio de métricas estatísticas amplamente utilizadas na área de criptografia de imagens.

A análise de histograma é empregada para verificar se a imagem cifrada apresenta distribuição aproximadamente uniforme das intensidades de pixel, eliminando padrões estatísticos visíveis presentes na imagem original.

A correlação entre pixels adjacentes é analisada nas direções horizontal, vertical e diagonal. Imagens naturais apresentam alta correlação entre pixels vizinhos; um esquema de encriptação eficaz deve reduzir significativamente essa correlação.

Além disso, testes de robustez, como ataques por recorte (*cropping attack*) e adição de ruído (*noise attack*), são utilizados para simular cenários de perda parcial de dados ou interferência durante a transmissão. A capacidade de recuperação da imagem original após tais perturbações fornece evidências adicionais da robustez do método proposto.

Essas métricas permitem avaliar quantitativamente a segurança estatística e a resistência do esquema de encriptação proposto (CRUZ; FAINA; PEREIRA, 2024; WANG et al., 2018).

2.2.5.1 Análise de Histograma

A análise de histograma é utilizada para avaliar a efetividade da encriptação de imagens por meio da comparação entre os histogramas da imagem original, encriptada e decriptada. Um histograma representa a distribuição das intensidades dos pixels em uma imagem. Em uma imagem adequadamente encriptada, o histograma deve apresentar uma distribuição aproximadamente uniforme, indicando que os valores dos pixels estão bem distribuídos e que a estrutura da imagem original foi devidamente ofuscada. Essa uniformidade sugere que a imagem encriptada não revela padrões visuais identificáveis da imagem original.

2.2.5.2 Correlação entre Pixels Adjacentes

Em uma imagem natural, cada píxel apresenta alta correlação com seus vizinhos. A análise de correlação é utilizada para estimar o relacionamento entre um píxel individual e seus pixels adjacentes. Essa análise é fundamental para compreender como o processo de encriptação afeta a estrutura estatística natural da imagem.

A Equação 2.21 apresenta uma forma de quantificar essa correlação, produzindo valores próximos de 1 para dados altamente correlacionados e valores próximos de 0 para dados com baixa correlação.

$$C_{OE} = \frac{\sum_{i=1}^N (O_i - \bar{O})(E_i - \bar{E})}{\sqrt{D(O) D(E)}} \quad (2.21)$$

Na Equação 2.21, C_{OE} representa o coeficiente de correlação entre duas variáveis, O e E , que podem ser interpretadas como conjuntos de valores observados e esperados, respectivamente. A somatória considera todos os N pares de pixels correspondentes, sendo O_i o valor observado e E_i o valor esperado.

Os termos $\bar{O}$ e $\bar{E}$ representam as médias dos respectivos conjuntos de dados, enquanto $D(O)$ e $D(E)$ correspondem às variâncias desses conjuntos. O coeficiente de correlação resultante fornece uma medida quantitativa do grau de similaridade entre os dois conjuntos de dados.

Uma alta correlação indica que os valores dos pixels preservam a estrutura original da imagem. Por outro lado, valores baixos de correlação indicam que as relações naturais entre os pixels foram significativamente alteradas, o que é desejável em imagens encriptadas, pois demonstra maior eficácia do processo criptográfico.

2.2.5.3 Teste de Ataque de Recorte

Para avaliar a robustez das imagens encriptadas frente à perda parcial de dados, foi realizado um teste de ataque de recorte. Esse procedimento consiste na remoção deliberada de regiões específicas da imagem encriptada, simulando cenários de perda de dados durante armazenamento ou transmissão. Em seguida, aplica-se o processo de deciptação sobre as imagens parcialmente corrompidas.

A análise das imagens resultantes permite avaliar a capacidade do método de encriptação em lidar com esse tipo de ataque, bem como verificar até que ponto a imagem original pode ser reconstruída. Um esquema de encriptação robusto deve manter, mesmo sob recorte, características suficientes que permitam a recuperação parcial das informações visuais.

2.2.5.4 Teste de Análise de Ruído

Este teste tem como objetivo avaliar a resistência das imagens encriptadas à degradação de informações durante transmissão e armazenamento. O procedimento consiste na adição deliberada de ruído do tipo *salt-and-pepper*, simulando a corrupção aleatória de pixels.

Esse tipo de ruído representa interferências comuns em sistemas digitais, nas quais determinados pixels são substituídos por valores extremos de intensidade. Após a inserção do ruído, realiza-se a deciptação da imagem para analisar o impacto da perturbação e avaliar a resiliência do esquema criptográfico proposto.

2.3 Revisão Bibliográfica

O trabalho intitulado "*Controlled alternate quantum walk-based pseudo-random number generator and its application to quantum color image encryption*" tem por objetivo a melhora o processo de encriptação de imagem usando um novo mecanismo de PRNG baseado em *quantum walk* alternado controlado com um pedestre agindo em um PRNG de duas dimensões. Seus resultados mostram que sua proposta tem boas características pseudo-aleatórias e qualidades vitais para aplicações com propósito de segurança, sendo seguro com alta eficácia contra a maioria dos ataques conhecidos. Neste artigo foi apresentado uma opção para a geração de números pseudo-aleatórios em encriptação de imagens em ambiente quântico, o que cai na temática deste projeto, citado anteriormente (EL-LATIF; ABD-EL-ATTY; VENEGAS-ANDRACA, 2019).

Já o artigo "*Quantum image encryption algorithm based on quantum key image*" busca apresentar um algoritmo de encriptação de imagem usando chave quântica de imagem, dizendo ser mais simples que os outros algoritmos de encriptação de imagem quânticos. A chave quântica de imagem é preparado com o *keystream* gerado usando um algoritmo criptográfico clássico chamado "*block cipher method*". Seus resultados mostram que o algoritmo de encriptação de imagem quântica proposto é eficiente e seguro. Este trabalho se relaciona à este projeto por compor a área de encriptação de imagens em sistemas quânticos, área a qual pretende-se explorar (WANG et al., 2018).

O trabalho "*Quantum-based PRNG for image encryption: a comprehensive study*" (CRUZ; FAINA; PEREIRA, 2024) realiza estudos na área de encriptação de imagens usando geradores de números pseudo-aleatórios em sistemas quânticos. Para o PRNG é proposto um algoritmo que usa um sistema quântico, seus parâmetros iniciais e um conjunto de portas ordenadas para gerar um conjunto de números baseados em quasi-probabilidade. O esquema para encriptação e deciptação de dados usa os valores quasi-probabilísticos, dados de imagem e operação binária reversível para gerar um esquema básico para segurança de dados. Apesar das limitações, o estudo conseguiu demonstrar

que explorando as propriedades únicas de estados quânticos e sua quasi-probabilidade associada, é possível desenvolver algoritmos de encriptação extremamente seguros. Esta monografia se dedica ao mesmo problema bem como a encriptação e desencriptação de imagens usando PRNG em sistemas quânticos.

Por fim a dissertação de mestrado "*Multigraphs: Emergence from Hilbert Space Subdivision in Superposed Quantum Systems and their Image Encoding Application*" (CRUZ, 2024), que busca contribuir 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.

Os métodos usados foram um algoritmo que aproveita as quasi-probabilidades inerentes à mecânica quântica para construir um PRNG a partir de um sistema quântico, depois foi desenvolvido um algoritmo de criptografia e descriptografia baseado no PRNG e por fim foi introduzido um novo *framework* para representação de multigrafos dentro de sistemas quânticos. Seus resultados conseguiram demonstrar esquemas de encriptação de imagens usando PRNG que oferecem segurança melhorada contra ataques clássicos, além disso foi apresentado uma nova perspectiva na utilização de multigrafos que pode levar a geração de múltiplos multigrafos com novas aplicações. Esta tese é onde se baseia este projeto, com exceção da parte de multigrafos, o foco do projeto será replicar e melhorar os resultados desse estudo.

3 Método

Neste capítulo são apresentados os métodos empregados na monografia. Inicialmente, na Seção 3.1, descrevem-se os geradores pseudoaleatórios baseados em quasi-probabilidades quânticas. Em seguida, na Seção 3.2, detalha-se o algoritmo utilizado para a geração da lista de valores com propriedades aleatórias. Posteriormente (Seção 3.3) são apresentados os esquemas de encriptação e decriptação de imagens. Por fim, apresenta-se o *Qiskit*, um *framework* desenvolvido pela *IBM (International Business Machines Corporation)*, fundamental para a implementação dos algoritmos apresentados neste trabalho.

3.1 Quantum Quasi-Probability Pseudo Random Generator

Este método utiliza os princípios da mecânica quântica, fazendo uso de circuitos quânticos, execução sequencial e quasi-probabilidades para gerar sequências de números com propriedades aleatórias. O algoritmo a ser discutido (CRUZ, 2024) utiliza um número pequeno de *qubits* e uma sequência de portas de rotação. Ele se destaca por exigir pelo menos 3 *qubits* para gerar sequências de números pseudo-aleatórios.

Baseado em um sistema quântico, um conjunto de portas de Hadamard e um conjunto de portas de rotação, o método proposto gera um conjunto de números binários com propriedades pseudo-aleatórias. O sistema consiste de um pequeno conjunto de *qubits* inicializados no estado $|0\rangle$. A inclusão das portas de Hadamard permitem que o algoritmo aproveite de mecanismos de superposição para explorar o espaço de Hilbert com eficiência. Adicionalmente, uma sequência de portas de rotação induz mudanças que modificam as probabilidades dos estados resultantes.

Devido às suas características, esse método tem potencial para encriptação e comunicação de dados. No entanto, a limitação principal do sistema está no tempo requerido para calcular o conjunto de números pseudo-aleatórios. Cada conjunto, possuindo 2^n elementos, depende de um sistema quântico e portas de rotação para computação, levando à um *overhead* computacional significativo.

3.1.1 Etapas Propostas Para a Geração de Números Pseudo-Aleatórios

Essa proposta traça um processo sequencial que contempla vários passos cujos objetivos são gerar números pseudo-aleatórios. Cada passo contribui para a geração de uma sequência grande de números por meio da criação, modificação e medição de sistemas quânticos.

Primeiramente, o algoritmo calcula o número de sistemas quânticos baseado em parâmetros de configuração conhecidos, número de *qubits* e o tamanho desejado da lista de números.

Em segundo lugar, o algoritmo entra em um processo iterativo onde sistemas quânticos são criados, modificados e medidos. Esse processo é essencial na geração de pseudo-aleatoriedade e assegura que os números resultantes possuam propriedades estatísticas desejáveis.

Finalmente, o algoritmo transforma as medidas do sistema quântico em números pseudo-aleatórios correspondentes.

Passo 1. Determinar a quantia necessária de Sistemas Quânticos

A quantidade necessária de sistemas quânticos, denominado k , para gerar uma sequência de $L \geq 0$ elementos é calculada conforme Eq. 3.1, onde n representa o número de *qubits* utilizados no sistema quântico.

$$k = \frac{L}{2^n} \quad (3.1)$$

Passo 2. Sistema Quântico

Essa proposta requer um sistema quântico composto de n *qubits* (Eq. 3.2). $\mathcal{H}_i$ representa o espaço de Hilbert associado ao i -ésimo *qubit*, e o produto tensorial simboliza a combinação desses espaços individuais em uma entidade quântica unificada.

$$\mathcal{H} = \mathcal{H}^{\otimes n} = \mathcal{H}_1 \otimes \mathcal{H}_2 \otimes \cdots \otimes \mathcal{H}_n, \quad (3.2)$$

O sistema é inicializado no estado $|0\rangle$ (Eq. 3.3). No entanto, não é uma restrição, já que o sistema admite qualquer estado para sua inicialização.

$$|\psi(0)\rangle = |0\rangle^{\otimes n} \quad (3.3)$$

Ademais, para obter um sistema superposicionado, aplica-se um conjunto de portas de Hadamard (Eq. 3.4).

$$|\psi(0)'\rangle = H^{\otimes n} |\psi(0)\rangle \quad (3.4)$$

Um componente essencial na definição do PRNG é o operador unitário. Esse operador funciona como um processo determinístico, aproveitando os princípios da mecânica quântica para induzir mudanças no sistema quântico (Eq. 3.5).

$$U = S(R_j^{\otimes n}) \quad (3.5)$$

A Eq. 3.4 $R_j(\theta)$ representa a rotação ao redor de um eixo específico do *qubit* j , com $\theta \in [0, \frac{\pi}{2}]$, enquanto S denota um operador de troca. O operador de troca introduz complexidade adicional no processo determinístico. Ele nos permita alterar dinamicamente o estado quântico (MANOUCHEHRI; WANG, 2013). No entanto, no contexto dessa proposta, o operador de troca simplifica a matriz de identidade. A definição formal do operador de troca é dada pela Eq. 3.6.

$$S = \sum_{i=0}^n |i\rangle \langle i| \quad (3.6)$$

Simplificar o operador unitário configurando o operador de troca como uma matriz de identidade nos permite definir um processo determinístico baseado primariamente nas rotações. Esse operador unitário simplificado é expresso pela Eq. 3.7.

$$U = I(R_j(\theta)^{\otimes n}) \quad (3.7)$$

A evolução de um sistema quântico ocorre discretamente, avançando por meio de diferentes *time steps* sob a influência de um operador unitário U (Eq. 3.7). A cada *time step* t , o operador unitário U age como um estado quântico $|\psi\rangle(t)$. Matematicamente, o estado quântico no tempo t , denotado como $|\psi\rangle(t)$, é derivado ao aplicar o operador unitário U t vezes ao estado inicial $|\psi\rangle(0)$, representado pela equação $|\psi\rangle(t) = U^t |\psi\rangle(0)$. A Figura 6 apresenta o circuito para esse processo.

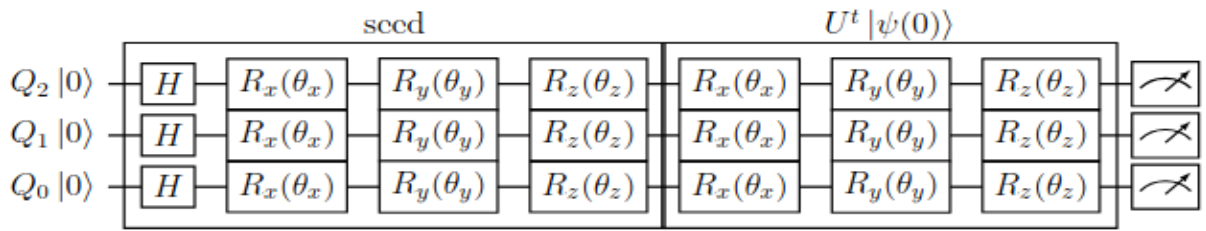


Figura 6 – Sequência de Portas que compõem a Configuração do Sistema Quântico (Fonte: (CRUZ, 2024))

Passo 3. Geração de Número Pseudo-Aleatório

Essa seção descreve as etapas de pré-processamento necessárias para gerar números pseudo-aleatórios a partir de uma distribuição da quasi-probabilidade de um sistema quântico. Inicialmente, valores são coletados da saída do sistema quântico. Esses valores passam por uma série de transformações para produzirem números pseudo-aleatórios. Isso inclui escalar cada valor por 10^9 para aumentar sua magnitude e depois pegar o

módulo por 10^3 para limitar a saída no espaço $[0, 1000]$. Esses números transformados são utilizados sequencialmente como números pseudo-aleatórios em várias aplicações que necessitam de aleatoriedade.

3.1.2 Algoritmo Proposto

Esse processo determinístico é elucidado na Figura 7, que serve como uma representação visual de uma potencial trajetória que um giro quântico pode atravessar pela esfera de Bloch (Seção 2.1.3). A figura ilustra as mecânicas escondidas do PRNG, dando luz à interlação de estados quânticos e transformações que resultam na geração dos números pseudo-aleatórios.

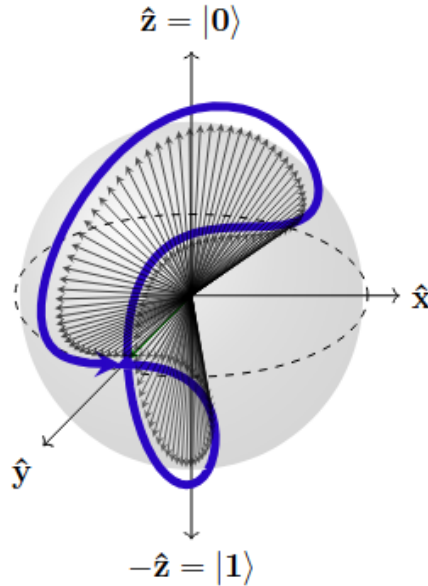


Figura 7 – Representação de um *qubit* e seu caminho de giro através de esfera de Bloch (Fonte: (CRUZ, 2024))

A estratégia envolve um sistema de n *qubit*, um operador unitário U , e parâmetros de rotação. Para conseguir isso, são utilizados os passos a seguir:

1. Comece com um sistema quântico denotado como $|\psi\rangle$.
2. Aplique um operador unitário U^t .
3. Receba os valores de quasi-probabilidades para os estados e os guarde em uma lista.
4. Itere pelos passos 1 a 3 até que o tamanho de lista desejado seja alcançado.

O Algoritmo 1 passa por todos os passos necessários para gerar um vetor adequado de dados. Os detalhes de implementação desse algoritmo podem ser encontrados no Capítulo 4.

Algoritmo 1 Algoritmo para obter um conjunto de probabilidades (CRUZ, 2024)

Require: n — Número de *qubits*, $\theta = f_s = \frac{1}{100}$ (Scale Factor), $\theta_x, \theta_y, \theta_z$ — Ângulos de Rotação, L — Quantidade de Números Aleatórios

Ensure: valores

Require: $backend \leftarrow Aer.get_backend('statevector_simulator')$

```

1:  $k \leftarrow \frac{L}{2^n}$ 
2:  $valores \leftarrow []$ 
3:  $cont = 0$ 
4: while  $k$  do
5:    $circ \leftarrow QuantumCircuit(n)$ 
6:   for  $i$  in  $range(n)$  do
7:      $circ.h(i)$ 
8:      $circ.rx(\theta_x, i)$ 
9:      $circ.ry(\theta_y, i)$ 
10:     $circ.rz(\theta_z, i)$ 
11:   end for
12:    $r\_x, r\_y, r\_z \leftarrow cont \times \theta$ 
13:   for  $i$  in  $newValues$  do
14:      $circ.rx(r\_x, i)$ 
15:      $circ.ry(r\_y, i)$ 
16:      $circ.rz(r\_z, i)$ 
17:   end for
18:    $counts \leftarrow backend.run(circ, shots=2048).result().get\_counts()$ 
19:    $newValues \leftarrow list(counts.values())$ 
20:   for  $i$  in  $newValues$  do
21:      $num = np.floor(i * 10^9) \% 10^3$ 
22:      $values.append(num)$ 
23:   end for
24:    $cont \leftarrow cont + 1$ 
25:    $k \leftarrow k - 1$ 
26: end while
   return  $values$ 

```

3.1.3 Entrelaçamento Quântico via Portas C-NOT

No Algoritmo 1, o laço **for** i **in** $range(n)$ aplica, a cada *qubit* i do registrador, isoladamente, uma porta de *Hadamard* seguida das rotações R_x , R_y e R_z , todas parametrizadas pelo mesmo ângulo escalar θ . Como nenhuma porta de duas *qubits* é utilizada nessa etapa, o estado global do sistema permanece sempre fatorável, isto é, redutível a um produto tensorial de estados de um único *qubit*:

$$|\psi(0)'\rangle = \bigotimes_{i=1}^n |\phi_i(\theta)\rangle, \quad (3.8)$$

em que cada $|\phi_i(\theta)\rangle$ é determinado pelo mesmo ângulo θ , aplicado de forma idêntica a todos os *qubits* do registrador nessa etapa. O estado resultante é, portanto, não apenas fatorável como também idêntico em cada subsistema, o que limita a diversidade estatística da distribuição de quasi-probabilidade obtida a partir das medições, já que a informação relevante do sistema completo se reduz, em essência, à de um único qubit repetido n vezes.

Propõe-se a inserção de uma camada de entrelaçamento quântico, por meio de portas C-NOT, imediatamente após o laço de preparação individual dos qubits e antes da segunda aplicação de rotações (descrita na Seção 3.1.4). A porta C-NOT entre o qubit de controle i e o qubit alvo $i + 1$ é definida como:

$$\text{CNOT}_{i,i+1} = |0\rangle\langle 0|_i \otimes I_{i+1} + |1\rangle\langle 1|_i \otimes X_{i+1}, \quad (3.9)$$

em que X_{i+1} é a porta NOT (Pauli-X) atuando sobre o qubit alvo. A camada completa de entrelaçamento, aplicada sequencialmente a todos os pares de qubits adjacentes do registrador, é dada pelo produto:

$$C = \prod_{i=0}^{n-2} \text{CNOT}_{i,i+1}, \quad (3.10)$$

formando uma cadeia que conecta todos os n qubits do circuito a partir de uma única camada de portas, escolha que favorece tanto a simplicidade de implementação no *Qiskit* (`circ.cx(i, i+1)`) quanto a propagação de correlações por todo o registrador.¹ O operador unitário simplificado definido na Eq. 3.7 é, portanto, redefinido como:

$$U' = C \cdot U = C \cdot I(R_j(\theta)^{\otimes n}). \quad (3.11)$$

Com essa modificação, o estado do sistema deixa de satisfazer a forma fatorável da Eq. 3.8: as correlações introduzidas pela camada C não podem ser reproduzidas por nenhuma distribuição de probabilidade clássica equivalente sobre os subsistemas individuais. Em particular, a distribuição de quasi-probabilidade resultante (Seção 2.1.7.1) passa a refletir essas correlações não locais, o que se espera traduzir em sequências de saída estatisticamente mais ricas, conforme avaliado pela bateria de testes NIST SP 800-22 no Capítulo 5.

3.1.4 Randomização dos Ângulos de Rotação com o Mapa Logístico

A segunda modificação proposta substitui a origem dos ângulos de rotação utilizados no Algoritmo 1. Na versão original, θ_x , θ_y e θ_z são parâmetros de entrada fixos, e o incremento $r_x, r_y, r_z = \text{cont} \times \theta$ cresce de maneira estritamente linear e determinística a cada iteração do laço externo. Ambas as escolhas introduzem regularidade ao processo de preparação dos qubits.

Propõe-se substituir essas fontes de parametrização por valores extraídos da sequência caótica gerada pelo mapa logístico em regime de caos completo (Seção 2.2.2):

$$x_{k+1} = 4x_k(1 - x_k), \quad x_0 \in (0, 1), \quad (3.12)$$

¹ Uma topologia alternativa em anel, fechando a cadeia entre o último e o primeiro qubit (`circ.cx(n-1, 0)` para $n \geq 3$), foi também considerada na implementação, mas não foi utilizada nos experimentos reportados neste trabalho.

cujos valores, após o período de *warm-up*, são uniformizados pela transformação de Ulam-von Neumann já apresentada na Seção 2.2.2, $y_k = \frac{2}{\pi} \arcsin \sqrt{x_k} \in [0, 1)$, e em seguida escalonados linearmente para o intervalo $[1, 4)$:

$$\theta_k = 1 + 3 y_k. \quad (3.13)$$

A cada iteração do laço principal do algoritmo, um único valor θ é consultado dessa sequência caótica e desempenha dois papéis: (i) é multiplicado pelo fator de escala $f_s = 1/100$ e aplicado, de forma idêntica a R_x , R_y e R_z , a todos os qubits do registrador, na etapa imediatamente posterior à camada de entrelaçamento; e (ii) o mesmo valor escalonado torna-se o ângulo de preparação (θ na Eq. 3.8) utilizado no bloco de *Hadamard* e rotações da próxima iteração do laço. Esse mecanismo encadeia todas as iterações do algoritmo por meio de uma única variável escalar evoluindo caoticamente: apenas a primeira execução do circuito utiliza um ângulo de preparação no intervalo bruto $[1, 4)$ radianos (definido fora do laço principal); a partir da segunda iteração, tanto o bloco de preparação quanto o bloco pós-entrelaçamento utilizam ângulos já escalonados por f_s , da ordem de $[0,01, 0,04)$ radianos.

O Algoritmo 2 resume as duas modificações propostas (entrelaçamento e randomização caótica dos ângulos) aplicadas em conjunto sobre o Algoritmo 1.

Algoritmo 2 Algoritmo modificado para obtenção do conjunto de probabilidades, com entrelaçamento quântico e ângulos de rotação caóticos

Require: n — Número de *qubits*, θ — Ângulo de preparação inicial (extraído da sequência caótica), f_s — Fator de escala, $size$ — Quantidade de Números Aleatórios desejada

Ensure: *values*

Require: *backend* $\leftarrow$ Aer.get_backend('statevector_simulator')

Require: *rng* $\leftarrow$ instância do gerador caótico (mapa logístico, Eq. 3.12)

```

1: values  $\leftarrow$  []
2: while len(values) < size do
3:   circ  $\leftarrow$  QuantumCircuit(n)
4:   for i in range(n) do
5:     circ.h(i)
6:     circ.rx( $\theta$ , i); circ.ry( $\theta$ , i); circ.rz( $\theta$ , i)
7:   end for
8:   for i in range(n - 1) do                                ▷ camada de entrelaçamento
9:     circ.cx(i, i + 1)
10:  end for
11:   $\theta \leftarrow \text{round}(\text{rng.uniform}(1, 4), 6) \times f_s$               ▷ novo ângulo, via Eq. 3.13
12:  for i in range(n) do
13:    circ.rx( $\theta$ , i); circ.ry( $\theta$ , i); circ.rz( $\theta$ , i)
14:  end for
15:  counts  $\leftarrow$  backend.run(circ, shots=2048).result().get_counts()
16:  newValues  $\leftarrow$  list(counts.values())
17:  for i in newValues do
18:    num  $\leftarrow \lfloor i \times 10^9 \rfloor \bmod 1024$ 
19:    values.append(num)
20:  end for
21: end while return values[: size]

```

A semente x_0 do mapa logístico passa a ser o parâmetro de controle responsável por toda a evolução posterior dos ângulos do circuito: sementes distintas produzem se-

quências de ângulos completamente descorrelacionadas entre si, em razão da sensibilidade às condições iniciais discutida na Seção 2.2.2.4, enquanto a repetição da mesma semente permite reproduzir exatamente a mesma sequência para fins de depuração e validação experimental.

3.2 Esquemas de Encriptação e Decriptação de Imagens

Os processos de encriptação e decriptação de imagens baseiam-se na aplicação de operações reversíveis sobre sequências binárias. Tipicamente, utiliza-se a operação XOR para essa finalidade. A operação XOR modifica a informação em nível de *bit*, combinando duas fontes de dados: a sequência original da imagem e uma sequência de números pseudo-aleatórios utilizada como chave criptográfica.

Para que o processo de encriptação e posterior decriptação ocorra corretamente, é fundamental que ambas as sequências possuam o mesmo comprimento. Dessa forma, a reversibilidade da operação é garantida, uma vez que aplicar XOR duas vezes com a mesma chave restaura os dados originais.

3.2.1 Normalização dos Valores Quasi-Probabilísticos

Para a utilização dos valores quasi-probabilísticos no processo criptográfico, é necessário realizar sua normalização. Neste trabalho, emprega-se a normalização do tipo *min-max* (PATRO; SAHU, 2015), conforme apresentada na Equação 3.14:

$$v_i = \frac{v_i - \min(V)}{\max(V) - \min(V)} \times c \quad (3.14)$$

em que V representa o conjunto de valores quasi-probabilísticos, $\min(V)$ e $\max(V)$ correspondem, respectivamente, ao menor e ao maior valor do conjunto, e c é uma constante de escala utilizada para ajustar a amplitude dos valores normalizados, possibilitando sua conversão para inteiros de maior magnitude quando necessário.

3.2.2 Algoritmo de Encriptação

O algoritmo de encriptação é baseado na operação $\oplus$ (XOR em *bit*) entre os valores da imagem e os valores normalizados quasi-probabilísticos. O Algoritmo 3 destaca o processo para transformar a imagem original em sua forma encriptada.

Algoritmo 3 Algoritmo para encriptação baseada em XOR (CRUZ, 2024)

Require: I, V
 1: $I' = \{\}$
 2: $N = |I|$
 3: **for** $i \leftarrow 0$ to N **do**
 4: $I'_i = I_i \oplus V_i$
 5: **end for**

3.2.3 Algoritmo de Decriptação

Como a operação binária do XOR ($\oplus$) é reversível, é possível aplicar o mesmo conjunto de operações para recuperar o conjunto de dados original. O Algoritmo 4 demonstra o processo de decriptação.

Algoritmo 4 Algoritmo para decriptação baseada em XOR (CRUZ, 2024)

Require: I', V
 1: $I' = \{\}$
 2: $N = |I'|$
 3: **for** $i \leftarrow 0$ to N **do**
 4: $I_i = I'_i \oplus V_i$
 5: **end for**

3.3 Computação Quântica no Qiskit

Qiskit foi fundado pela IBM para permitir o desenvolvimento de *software* para seu serviço de computação quântica em nuvem, IBM *Quantum Experience*. Contribuições também são feitas por apoiadores externos, geralmente de instituições acadêmicas.

O QisKit (*Quantum information software Kit*) é um framework open-source, que permite o desenvolvimento de softwares para computação quântica em nuvem. Ele fornece um conjunto de ferramentas para criar e manipular algoritmos quânticos e executá-los nos dispositivos da IBM bem como simulações na máquina do próprio usuário. O acesso pode ser diretamente na plataforma usando o IBM *Quantum Lab* ou instalando o pacote Qiskit (tendo o *Python* instalado). Ele segue o modelo de circuitos e pode ser usado para qualquer hardware quântico que siga esse modelo. Desta forma, qualquer pessoa pode usar esse framework para realizar simulações, tanto na sua máquina como também construir seus códigos e enviá-los para que sejam processados nos computadores quânticos da IBM (ALVES; FELIPE, 2022).

4 Resultados e Análise do QQPRG

Este capítulo descreve a implementação e os resultados obtidos com os algoritmos desenvolvidos neste trabalho. A Seção 4.1 apresenta os detalhes de implementação do algoritmo QQ-PRG original e do algoritmo modificado, incluindo o gerador de números pseudoaleatórios baseado no mapa logístico, a camada de entrelaçamento quântico, a normalização do vetor de valores e as funções auxiliares utilizadas nos testes de robustez. A Seção 4.1.4 descreve o esquema de encriptação e decriptação de imagens por operação XOR.

4.1 Implementação do Algoritmo QQPRG

Para a implementação e teste dos códigos foram usados os seguintes parâmetros.

Tabela 1 – Parâmetros para o algoritmo de geração de probabilidades

Parâmetro	Valor
<i>Qubits</i>	3
Fator de escala	$sf = \frac{1}{100}$
Comprimento	$L = 499 \times 499 = 249.001$
Número de sistemas quânticos	$k = \frac{499 \times 499}{2^3} = 31.125$
Rotações	$R_x = \frac{\pi}{3}, R_y = \frac{\pi}{3}, R_z = \frac{\pi}{3}$

4.1.1 Geração do Conjunto de Probabilidades

Algoritmo 5 descreve o Gerador de Números Pseudo-Aleatórios. A lógica do algoritmo é dividida em duas funções: i) criação do circuito quântico e ii) geração do vetor de valores aleatórios. Ademais, temos uma terceira função *imageToVectorR* que converte uma imagem em duas dimensões para um vetor unidimensional usando o canal da cor vermelha. É a partir desse vetor que obtemos *vector_length* usado para gerar o vetor de valores aleatórios.

Para a construção das chaves utilizadas no processo de encriptação e decriptação da imagem, emprega-se uma rotina baseada em circuitos quânticos, implementada com a biblioteca Qiskit. O objetivo é explorar a natureza probabilística dos estados quânticos para gerar números pseudoaleatórios (Algoritmo 5), que posteriormente servirão como vetores de chave para a operação XOR.

O processo inicia-se com a função `createQuantumCircuit` (Algoritmo 5 - linhas 5 a 14), que constrói um circuito quântico a partir de um número definido de *qubits*.

Sobre cada qubit são aplicadas portas de Hadamard, responsáveis por criar estados de superposição, seguidas por rotações nos eixos X, Y e Z do espaço de Bloch (portas R_x , R_y e R_z - Figura 8), que introduzem variações angulares parametrizadas. O resultado é um circuito capaz de gerar distribuições complexas e não triviais de probabilidades para os estados quânticos.

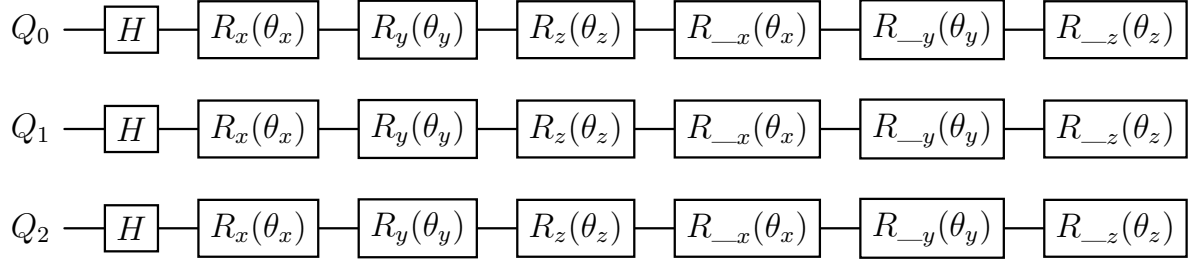


Figura 8 – Exemplo de circuito gerado pelo código (Fonte: Autor)

A função `generateVector` (Algoritmo 5 - linhas 16 a 44) é responsável por efetivamente gerar o vetor pseudoaleatório. Inicialmente, define-se um incremento angular $\theta = \frac{1}{100}$, utilizado para variar os parâmetros das rotações aplicadas aos *qubits* a cada iteração. Essa variação garante que o circuito não produza resultados determinísticos, mas sim distribuições distintas ao longo da execução. A cada ciclo, o circuito é criado com os parâmetros iniciais definidos pelo usuário (rx , ry , rz) e, em seguida, recebe incrementos proporcionais ao contador de iteração.

Após a construção do circuito, obtém-se seu vetor de estado utilizando o simulador de *statevector* da Qiskit Aer. Esse vetor contém as amplitudes complexas associadas a cada estado computacional possível. Como cada amplitude possui interpretação probabilística, calcula-se o módulo quadrado das amplitudes (Algoritmo 5 - linhas 37 e 38), resultando em um conjunto de probabilidades associado aos estados do sistema. Esses valores probabilísticos são então escalados e convertidos em números inteiros (linha 41), de forma a gerar um vetor utilizável como chave criptográfica. A cada iteração, novos valores são acrescentados até que o tamanho desejado do vetor seja alcançado.

Por fim, o código inclui a função `imageToVectorR` (Algoritmo 5 - linhas 46 a 50), que extrai apenas o canal vermelho de uma imagem, convertendo-o em um vetor unidimensional de inteiros. Essa rotina auxilia na integração entre os dados da imagem e os vetores pseudoaleatórios gerados, uma vez que ambos passam a ser representados em estruturas vetoriais compatíveis para a aplicação do XOR.

Algoritmo 5 Obtenção de um conjunto de probabilidades

```

1  from qiskit import QuantumCircuit
2  from qiskit_aer import Aer
3  from qiskit.quantum_info import Statevector
4
5  def createQuantumCircuit(n_qubits, rx, ry, rz):
6      circ = QuantumCircuit(n_qubits)
7      for i in range(n_qubits):
8          circ.h(i)
9          circ.rx(rx, i)
10         circ.ry(ry, i)
11         circ.rz(rz, i)
12         circ.h(i)
13
14     return circ
15
16 def generateVector(n_qubits, rx, ry, rz, vector_length):
17     values = []
18     cont = 0
19     theta = 1/100
20
21     backend = Aer.get_backend('statevector_simulator')
22
23     while len(values) < vector_length:
24         r_x = cont * theta
25         r_y = cont * theta
26         r_z = cont * theta
27         circ = createQuantumCircuit(n_qubits, rx, ry, rz)
28
29         for i in range(n_qubits):
30             circ.h(i)
31             circ.rx(r_x, i)
32             circ.ry(r_y, i)
33             circ.rz(r_z, i)
34             circ.h(i)
35
36         # Obtém o vetor de estado
37         state = Statevector.from_instruction(circ)
38         probs = np.abs(state.data) ** 2 # Probabilidades dos estados
39
40         # Converte as probabilidades em valores inteiros (por exemplo, escalando)
41         values.extend((probs * 1e10).astype(int)) # pode ajustar a escala conforme necessário
42         cont += 1
43
44     return circ, values[:vector_length]
45
46 def imageToVectorR(image):
47     bandR = image[:, :, 0]
48     bandR = bandR.astype(int)
49     bandR = np.reshape(bandR, (-1))
50     return bandR

```

4.1.2 Implementação das Melhorias: Entrelaçamento Quântico e Randomização Caótica dos Ângulos

Conforme proposto nas Seções 3.1.3 e 3.1.4 (Capítulo 3), duas modificações foram implementadas sobre o algoritmo de geração de probabilidades descrito na Seção 4.1.1: a inserção de uma camada de entrelaçamento quântico via portas C-NOT e a substituição dos ângulos de rotação fixos por valores extraídos de uma sequência caótica gerada pelo mapa logístico. O Algoritmo 6 apresenta a implementação completa dessas modificações, incluindo a classe `LogisticChaosPRNG`, responsável por gerar a sequência caótica, e a função `getProbabilities`, que substitui as funções `createQuantumCircuit` e

`generateVector` apresentadas na Seção 4.1.1.

A classe `LogisticChaosPRNG` (Algoritmo 6 — linhas 5 a 22) implementa o mapa logístico em regime de caos completo ($r = 4$), descartando um número configurável de iterações iniciais (*warm-up*, padrão de 1000 iterações) para eliminar o transiente decorrente da semente. O método `_uniform` (linhas 19 a 22) aplica a transformação de Ulam-von Neumann sobre cada novo valor gerado pelo mapa, convertendo a distribuição arco-seno nativa do mapa logístico em uma distribuição aproximadamente uniforme em $[0, 1)$; o método `uniform` (linhas 28 e 29) escala essa saída linearmente para um intervalo arbitrário $[a, b)$, sendo utilizado nesta implementação como `rng.uniform(1, 4)` para produzir valores no intervalo $[1, 4)$.

A função `getProbabilities` (Algoritmo 6 — linhas 32 a 56) substitui a construção estática do circuito por um processo iterativo no qual o ângulo de rotação `rotation` evolui caoticamente a cada execução do laço principal. Em cada iteração, o circuito é inicializado com portas de *Hadamard* seguidas de rotações R_x , R_y e R_z parametrizadas pelo valor corrente de `rotation`, aplicadas identicamente a todos os n qubits (linhas 38 a 42). Em seguida, insere-se a camada de entrelaçamento (linhas 44 e 45), na qual cada qubit i atua como controle de uma porta C-NOT sobre o qubit $i + 1$, formando uma cadeia que conecta todo o registrador. Um novo valor de `rotation` é então obtido a partir do gerador caótico, escalonado pelo fator $f_s = 1/100$ (linha 47), e aplicado a uma segunda camada de rotações *após* o entrelaçamento (linhas 48 a 51); esse mesmo valor escalonado é o que será utilizado como ângulo inicial da *próxima* iteração do laço, encadeando a evolução caótica ao longo de toda a execução do algoritmo. A Figura 9 ilustra o circuito resultante para $n = 3$ qubits.

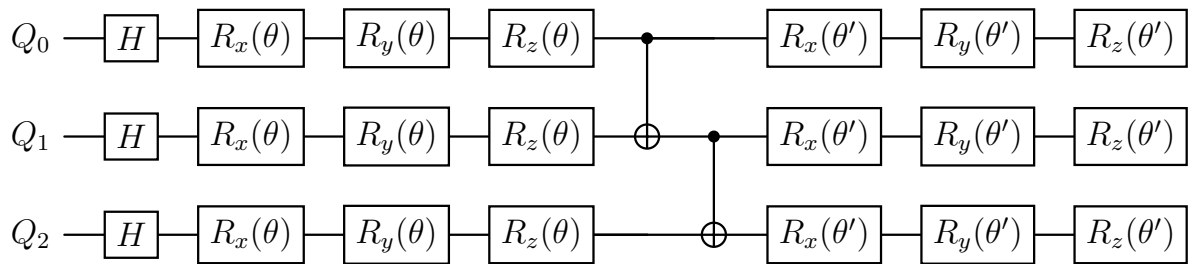


Figura 9 – Circuito modificado, com camada de entrelaçamento via C-NOT entre os ângulos de preparação (θ) e os ângulos pós-entrelaçamento (θ'), ambos extraídos da sequência caótica do mapa logístico (Fonte: Autor)

Por fim, a medição do circuito (linha 53) é realizada por meio do *backend* `statevector_simulator` da *Qiskit Aer*, executado com `shots=2048`, retornando contagens de medição (`counts.values()`) para cada estado computacional observado. Essa abordagem difere da utilizada na Seção 4.1.1, em que as probabilidades eram calculadas analiticamente a partir do vetor de estado (`Statevector.from_instruction`), sem ruído de amostragem; aqui, a gera-

ção de números pseudoaleatórios incorpora adicionalmente o ruído estatístico inerente ao processo de medição quântica simulado pelos *shots*. Os valores obtidos são então transformados (linhas 54 a 56) e acumulados até que o tamanho desejado do vetor seja atingido.

Algoritmo 6 Geração caótica e entrelaçada do conjunto de probabilidades (Parte 1)

```

1  import math
2  import time
3  import numpy as np
4  from qiskit import QuantumCircuit
5  from qiskit_aer import Aer
6
7  class LogisticChaosPRNG:
8      def __init__(self, seed=None, r=4.0, warmup=1000):
9          if seed is None:
10             seed = (time.time_ns() % (2 ** 53)) / 2 ** 53
11             if not (0.0 < seed < 1.0):
12                 raise ValueError("seed deve estar estritamente entre 0 e 1.")
13             for f in (0.0, 0.25, 0.5, 0.75, 1.0):
14                 if abs(seed - f) < 1e-12:
15                     seed += 1e-6
16             self.r = r
17             self.x = seed
18             for _ in range(warmup):
19                 self._step()
20
21         def _step(self):
22             self.x = self.r * self.x * (1.0 - self.x)
23             return self.x
24
25         def _uniform(self):
26             x = self._step()
27             x = min(max(x, 0.0), 1.0)
28             return (2.0 / math.pi) * math.asin(math.sqrt(x))
29
30         def random(self):
31             return self._uniform()
32
33         def uniform(self, a, b):
34             return a + (b - a) * self.random()

```

Algoritmo 6 Geração caótica e entrelaçada do conjunto de probabilidades (Parte 2)

```

35  def getProbabilities(n, theta, rotation, size):
36      values = []
37      cont = 0
38      rng = LogisticChaosPRNG(seed=None)
39      backend = Aer.get_backend('statevector_simulator')
40
41      while len(values) < size:
42          circ = QuantumCircuit(n)
43          for i in range(n):
44              circ.h(i)
45              circ.rx(rotation, i)
46              circ.ry(rotation, i)
47              circ.rz(rotation, i)
48
49          for i in range(n - 1):
50              circ.cx(i, i + 1)
51
52          rotation = round(rng.uniform(1, 4), 6) * theta
53
54          for i in range(n):
55              circ.rx(rotation, i)
56              circ.ry(rotation, i)
57              circ.rz(rotation, i)

```

Algoritmo 6 Geração caótica e entrelaçada do conjunto de probabilidades (Parte 3)

```

58     counts = backend.run(circ, shots=2048)
59     counts = counts.result()
60     counts = counts.get_counts()
61     newValues = list(counts.values())
62
63     for i in newValues:
64         num = np.floor(i * 1e9) % 1024
65         values.append(num)
66
67     cont += 1
68
69     return values[:size]

```

A Tabela 2 apresenta os parâmetros utilizados nos experimentos realizados com o algoritmo modificado, em substituição aos parâmetros fixos da Tabela 3.

Tabela 2 – Parâmetros para o algoritmo de geração de probabilidades com as melhorias propostas

Parâmetro	Valor
<i>Qubits</i>	$n = 3$
Fator de escala	$f_s = \frac{1}{100}$
Semente do mapa logístico	x_0 (derivada do relógio do sistema)
Período de <i>warm-up</i>	1000 iterações
Comprimento do vetor gerado	$size = 2048^2 = 4.194.304$
<i>Shots</i> por circuito	2048
Topologia de entrelaçamento	cadeia (CNOT _{$i, i+1$} , $i = 0, \dots, n - 2$)

4.1.3 Normalização do Vetor de Valores

Antes de ser utilizado como chave na operação XOR (Seção 4.1.4), o vetor de valores produzido pelo gerador de números pseudoaleatórios passa por uma etapa de normalização, implementada pela função `normalization` (Algoritmo 7). Essa etapa reescala linearmente os valores brutos para o intervalo $[0, 1]$ por meio da normalização min-max (linha 3),

$$v'_i = \frac{v_i - \min(v)}{\max(v) - \min(v)}, \quad (4.1)$$

e, em seguida, expande esse intervalo multiplicando cada valor por 5×10^9 e truncando para o inteiro mais próximo por baixo (linha 4), produzindo a chave final utilizada na Seção 4.1.4.

Algoritmo 7 Normalização do vetor de valores

```

1  def normalization(values):
2      arr = np.array(values)
3      arr = (arr - np.min(arr)) / (np.max(arr) - np.min(arr))
4      x_norm = np.floor(arr * 5000000000)
5      return x_norm

```

4.1.4 Encriptação e Decriptação de Imagem usando XOR

Essa implementação usa do XOR com os *bits* de uma imagem e o vetor de valores gerados pelo código demonstrado na subseção anterior e depois normalizados.

O método implementado baseia-se na operação lógica XOR aplicada a imagens digitais, permitindo tanto a encriptação quanto a decriptação de forma simétrica. A ideia central consiste em converter cada pixel da imagem em um número inteiro único, aplicar a operação XOR com uma chave de encriptação e, posteriormente, reverter o processo para recuperar a imagem original.

Algoritmo 8 Encriptação e decriptação da Imagem usando XOR

```

1  import numpy as np
2  from PIL import Image
3
4  def encrypt_decrypt_Image(image, values):
5      #Inicialize um vetor pra armazenar o dado transformado
6      data = np.zeros((image.shape[0], image.shape[1]), dtype=np.uint64)
7
8      #Converte os valores RGB de cada pixel em um unico inteiro e armazena num vetor de dados
9      for i in range(0, image.shape[0]):
10         for j in range(0, image.shape[1]):
11             data[i, j] = int(image[i, j, 0]) * 1000000 + int(image[i, j, 1]) * 1000 + int(image[i, j, 2])
12
13     #reforma os dados em um vetor de uma dimensao
14     data = np.reshape(data, (-1))
15
16     #operação XOR entre dados e valores chave
17     aux = [int(data[i]) ^ int(values[i]) for i in range(0, len(data))]
18
19     #reconstrua a imagem encriptada do resultado do XOR
20     imageOut = np.zeros(image.shape, dtype=np.uint16)
21     cont = 0
22     for i in range(0, image.shape[0]):
23         for j in range(0, image.shape[1]):
24             imageOut[i, j, 0] = int(aux[cont] / 1000000)
25             imageOut[i, j, 1] = int((aux[cont] % 1000000) / 1000)
26             imageOut[i, j, 2] = int(aux[cont] % 1000)
27             cont += 1
28
29     return imageOut

```

No processo de encriptação (Algoritmo 8), nas linhas 9 à 11, cada pixel da imagem, originalmente representado por três componentes de cor (R, G e B), é transformado em um único valor inteiro pela concatenação de seus canais. O canal vermelho é deslocado para a casa dos milhões, o canal verde para a casa dos milhares e o canal azul permanece nas unidades, formando um número da forma $R * 10^6 + G * 10^3 + B$. Essa representação

compacta facilita a aplicação da operação XOR em valores inteiros, em vez de tratar os três canais separadamente. Após essa etapa, na linha 14, a matriz bidimensional da imagem é linearizada em um vetor unidimensional, que é então submetido na linha 17 ao XOR com a chave de encriptação representada por um vetor de valores pseudoaleatórios. O resultado é um novo vetor que contém os pixels da imagem cifrada. Por fim, nas linhas 20 à 27, esse vetor é reconstruído novamente em formato RGB, recuperando os três canais a partir do número inteiro gerado, obtendo-se assim a imagem encriptada.

Já no processo de deciptação, aplica-se a mesma sequência de operações de forma inversa. A imagem cifrada é primeiramente convertida novamente em valores inteiros, seguindo o mesmo padrão de concatenação dos canais RGB. Essa matriz é linearizada e submetida à operação XOR utilizando-se a mesma chave empregada na fase de encriptação. Como o XOR possui a propriedade $(a \oplus b) \oplus b = a$, a aplicação repetida da mesma chave garante a recuperação exata dos valores originais dos pixels. Após essa etapa, os inteiros obtidos são decompostos em seus respectivos canais de cor, permitindo a reconstrução fiel da imagem original.

4.1.5 Funções Auxiliares para Avaliação de Robustez

Para avaliar a robustez do esquema de encriptação frente a perturbações na imagem cifrada (testes detalhados na Seção 2.2.5), foram implementadas duas funções auxiliares: `cut_image`, que simula um ataque de recorte, e `add_salt_and_pepper_noise`, que simula ruído do tipo sal-e-pimenta sobre a imagem cifrada. Ambas estão reproduzidas no Algoritmo 9.

A função `cut_image` (linhas 1 a 13) recebe a imagem e um parâmetro n correspondente ao lado, em pixels, de uma região quadrada centralizada. As coordenadas dessa região são calculadas de forma a centralizá-la na imagem (linhas 5 a 8), e os valores dos pixels contidos nela são zerados (linha 12), simulando a perda ou remoção de uma porção contígua da imagem cifrada, um cenário comum de ataque ou de corrupção parcial de dados durante a transmissão.

A função `add_salt_and_pepper_noise` (linhas 15 a 30) introduz ruído impulsivo na imagem, alternando pixels para branco (valor 255 - “sal”) e para preto (valor 0 - “pimenta”) em posições escolhidas aleatoriamente. Os parâmetros `svp` (*salt-vs-pepper ratio*) e `amount` controlam, respectivamente, a proporção entre pixels de sal e de pimenta e a densidade total de ruído aplicada à imagem, conforme as Eqs. 4.2 e 4.3:

$$\text{num_salt} = \lfloor \text{amount} \times h \times w \times \text{svp} \rfloor, \quad (4.2)$$

$$\text{num_pepper} = \lfloor \text{amount} \times h \times w \times (1 - \text{svp}) \rfloor, \quad (4.3)$$

em que h e w são, respectivamente, a altura e a largura da imagem. As coordenadas dos pixels afetados são sorteadas por meio do gerador de números pseudoaleatórios padrão do *NumPy* (`np.random.randint`, linhas 20, 21, 25 e 26) e não pelo gerador caótico *LogisticChaosPRNG*, uma escolha deliberada: o ruído introduzido aqui representa uma perturbação externa ao processo criptográfico, simulando um canal de transmissão ruidoso ou um ataque, e não deve depender da mesma fonte de aleatoriedade utilizada na geração da chave de encriptação.

Algoritmo 9 Funções auxiliares para os testes de ataque de recorte e de ruído

```

1  def cut_image(image, n):
2      img_array = np.array(image)
3      height, width, _ = img_array.shape
4
5      top = (height - n) // 2
6      left = (width - n) // 2
7      bottom = top + n
8      right = left + n
9
10     cropped_image = img_array[top:bottom, left:right]
11     modified_image = np.copy(img_array)
12
13     modified_image[top:bottom, left:right] = 0
14
15     return modified_image
16
17
18  def add_salt_and_pepper_noise(image, svp, amount):
19      img_array = np.array(image).copy()
20      h, w, c = img_array.shape
21
22      total_pixels = h * w
23      num_salt = int(amount * total_pixels * svp)
24      num_pepper = int(amount * total_pixels * (1 - svp))
25
26      # SALT
27      coords_row = np.random.randint(0, h, num_salt)
28      coords_col = np.random.randint(0, w, num_salt)
29      img_array[coords_row, coords_col] = 255
30
31      # PEPPER
32      coords_row = np.random.randint(0, h, num_pepper)
33      coords_col = np.random.randint(0, w, num_pepper)
34      img_array[coords_row, coords_col] = 0
35
36      return img_array

```

4.1.6 Análise de Histograma

A análise de histograma (Seção 2.2.5.1) consiste em examinar a distribuição de frequência dos níveis de intensidade de um canal de cor da imagem: em uma imagem original, essa distribuição tende a apresentar picos pronunciados, refletindo padrões visuais e regiões de cor predominante; já em uma imagem bem encriptada, espera-se uma distribuição aproximadamente uniforme entre os 256 níveis de intensidade possíveis, indicando ausência de padrões estatísticos exploráveis. O Algoritmo 10 apresenta a rotina utilizada para gerar esses histogramas.

A imagem é carregada por meio da biblioteca OpenCV (linha 5). É importante observar que o OpenCV representa imagens no formato BGR (e não RGB), de modo que o índice 2 no terceiro eixo da matriz (linha 6) corresponde ao canal vermelho, e não ao azul, como seria o caso em uma representação RGB convencional. O histograma propriamente dito é calculado e plotado pela função `plt.hist` (linha 9), que recebe o canal selecionado já linearizado em um vetor unidimensional (`.ravel()`) e o distribui em 256 *bins*, um para cada nível de intensidade possível no intervalo $[0, 256)$. As linhas seguintes (10 a 13) configuram apenas os elementos visuais do gráfico (título, rótulos dos eixos e tamanho da fonte dos valores nos eixos).

Na prática, essa rotina é executada separadamente para a imagem original e para a imagem encriptada de cada experimento (e pode ser repetida para os três canais de cor), permitindo comparar visualmente o achatamento da distribuição de intensidades introduzido pelo processo de encriptação (Seção 5.3.3.1).

Algoritmo 10 Geração do histograma de um canal de cor da imagem

```

1  import cv2
2  import numpy as np
3  import matplotlib.pyplot as plt
4
5  img = cv2.imread('Images/tree_256.png')
6  red_channel = img[:, :, 2]
7
8  plt.figure()
9  plt.hist(red_channel.ravel(), bins=256, range=[0, 256])
10 plt.title("Histograma - Árvore", fontsize=18)
11 plt.xlabel("Intensidade do Pixel", fontsize=14)
12 plt.ylabel("Frequência", fontsize=14)
13 plt.tick_params(axis='both', labelsize=12)
14 plt.show()

```

4.1.7 Análise de Correlação entre Pixels Adjacentes

A correlação entre pixels adjacentes (Seção 2.2.5.2) é uma métrica amplamente utilizada para avaliar a qualidade de esquemas de encriptação de imagens: em imagens naturais, pixels vizinhos tendem a apresentar valores muito semelhantes, resultando em coeficientes de correlação próximos de 1; já em uma imagem bem encriptada, essa estrutura espacial deve ser destruída, levando a coeficientes próximos de 0. O Algoritmo 11 apresenta as três funções implementadas para o cálculo dessa métrica.

A função `correlation_coefficient` (linhas 4 a 14) implementa o coeficiente de correlação de Pearson entre dois vetores x e y ,

$$r_{xy} = \frac{\sum_k (x_k - \bar{x})(y_k - \bar{y})}{\sqrt{\sum_k (x_k - \bar{x})^2 \sum_k (y_k - \bar{y})^2}}, \quad (4.4)$$

em que $\bar{x}$ e $\bar{y}$ são as médias amostrais de x e y (linhas 7 e 8), o numerador corresponde à covariância amostral entre os dois vetores (linha 9) e o denominador, ao produto de seus desvios-padrão amostrais, a menos do fator de normalização (linhas 10 a 13).

A função `get_adjacent_pixels` (linhas 17 a 36) constrói os vetores x e y a partir de um canal de cor da imagem (uma matriz $h \times w$), pareando cada píxel com um vizinho específico, de acordo com a direção solicitada: horizontal, em que cada píxel é pareado com o seu vizinho imediatamente à direita (linhas 22 a 25); vertical, em que cada píxel é pareado com o vizinho imediatamente abaixo (linhas 27 a 30); e diagonal, em que cada píxel é pareado com o vizinho na diagonal inferior-direita (linhas 32 a 35).

Por fim, a função `image_correlations` (linhas 39 a 54) integra as duas anteriores: carrega a imagem a partir de um caminho de arquivo, separa seus três canais de cor (R, G e B, linhas 42 a 46) e, para cada canal, calcula o coeficiente de correlação nas três direções consideradas, retornando um dicionário aninhado com os resultados por canal e por direção. Na prática, essa função é aplicada separadamente à imagem original e à imagem encriptada do mesmo experimento, permitindo comparar a queda na correlação espacial introduzida pelo processo de encriptação (Seção 5.3.3.4).

Algoritmo 11 Cálculo da correlação entre pixels adjacentes (Parte 1)

```

1  import numpy as np
2  from PIL import Image
3
4  def correlation_coefficient(x, y):
5      x = np.array(x, dtype=np.float64)
6      y = np.array(y, dtype=np.float64)
7      mean_x = np.mean(x)
8      mean_y = np.mean(y)
9      numerator = np.sum((x - mean_x) * (y - mean_y))
10     denominator = np.sqrt(
11         np.sum((x - mean_x) ** 2) *
12         np.sum((y - mean_y) ** 2)
13     )
14     return numerator / denominator
15
16
17 def get_adjacent_pixels(channel, direction="horizontal"):
18     x = []
19     y = []
20     h, w = channel.shape
21     if direction == "horizontal":
22         for i in range(h):
23             for j in range(w - 1):
24                 x.append(channel[i, j])
25                 y.append(channel[i, j + 1])
26     elif direction == "vertical":
27         for i in range(h - 1):
28             for j in range(w):
29                 x.append(channel[i, j])
30                 y.append(channel[i + 1, j])
31     elif direction == "diagonal":
32         for i in range(h - 1):
33             for j in range(w - 1):
34                 x.append(channel[i, j])
35                 y.append(channel[i + 1, j + 1])
36     return x, y

```

Algoritmo 11 Cálculo da correlação entre pixels adjacentes (cont.)

```
39 def image_correlations(image_path):
40     img = Image.open(image_path).convert("RGB")
41     img = np.array(img)
42     channels = {
43         "R": img[:, :, 0],
44         "G": img[:, :, 1],
45         "B": img[:, :, 2]
46     }
47     results = {}
48     for cname, channel in channels.items():
49         results[cname] = {}
50         for direction in ["horizontal", "vertical", "diagonal"]:
51             x, y = get_adjacent_pixels(channel, direction)
52             corr = correlation_coefficient(x, y)
53             results[cname][direction] = corr
54     return results
```

5 Análise dos Resultados

Este capítulo apresenta os resultados empíricos obtidos a partir das estruturas e algoritmos desenvolvidos ao longo deste trabalho, com o objetivo de avaliar, de forma sistemática, o uso de circuitos quânticos na geração de números pseudoaleatórios aplicados à encriptação e decriptação de imagens. Além da replicação do algoritmo QQ-PRG original (Algoritmo 1), avaliam-se também as duas melhorias propostas neste trabalho — a inserção de entrelaçamento quântico via portas C-NOT e a randomização dos ângulos de rotação por meio do mapa logístico (Algoritmo 2), de modo a verificar se essas modificações de fato aprimoram a qualidade estatística da sequência gerada e, por consequência, a robustez do esquema de encriptação resultante.

As análises experimentais realizadas buscam verificar a eficiência, o desempenho e a viabilidade prática dos métodos propostos, tanto no contexto da geração de sequências pseudoaleatórias quanto em sua aplicação criptográfica. Os resultados obtidos fornecem evidências concretas do potencial da computação quântica, combinada a uma fonte complementar de aleatoriedade caótica, não apenas no âmbito teórico, mas também em aplicações computacionais e de segurança da informação.

A Seção 5.2 dedica-se à avaliação estatística do esquema de geração de números pseudoaleatórios, comparando o algoritmo original (Algoritmo 1) ao algoritmo modificado com entrelaçamento e randomização caótica (Algoritmo 2). Por meio da aplicação de testes estatísticos apropriados, são analisadas a aleatoriedade e a confiabilidade das sequências geradas por ambas as versões, características fundamentais para sua utilização em sistemas criptográficos.

Por fim, a Seção 5.3 apresenta uma análise detalhada dos testes aplicados aos esquemas de encriptação e decriptação descritos no Algoritmo 8, por meio de quatro métricas de avaliação: análise de histograma, correlação entre pixels adjacentes, teste de ataque de recorte e teste de análise de ruído. A partir da investigação dessas métricas de segurança e eficiência, busca-se validar a robustez e a força criptográfica dos métodos propostos, especialmente no contexto da proteção de imagens digitais, com particular atenção ao impacto das melhorias propostas sobre essa robustez.

5.1 Configuração dos Experimentos

Os experimentos apresentados neste capítulo foram conduzidos com foco na precisão, reprodutibilidade e confiabilidade dos resultados obtidos. Para a execução e simulação dos circuitos quânticos, foi utilizado o simulador *IBM Qiskit Aer*, escolhido devido à sua

robustez e à fidelidade na simulação de sistemas quânticos.

O *Qiskit Aer* fornece um *framework* adequado às etapas de desenvolvimento, teste e validação dos algoritmos propostos, permitindo a análise controlada do comportamento dos circuitos quânticos sem as limitações inerentes aos dispositivos quânticos físicos atualmente disponíveis.

Adicionalmente, com o objetivo de complementar a análise experimental e facilitar a visualização dos circuitos quânticos, a Figura 10 apresenta a plataforma *IBM Quantum Composer*. Essa ferramenta oferece uma interface gráfica intuitiva, contribuindo para o entendimento da estrutura e do funcionamento dos algoritmos quânticos empregados neste trabalho.

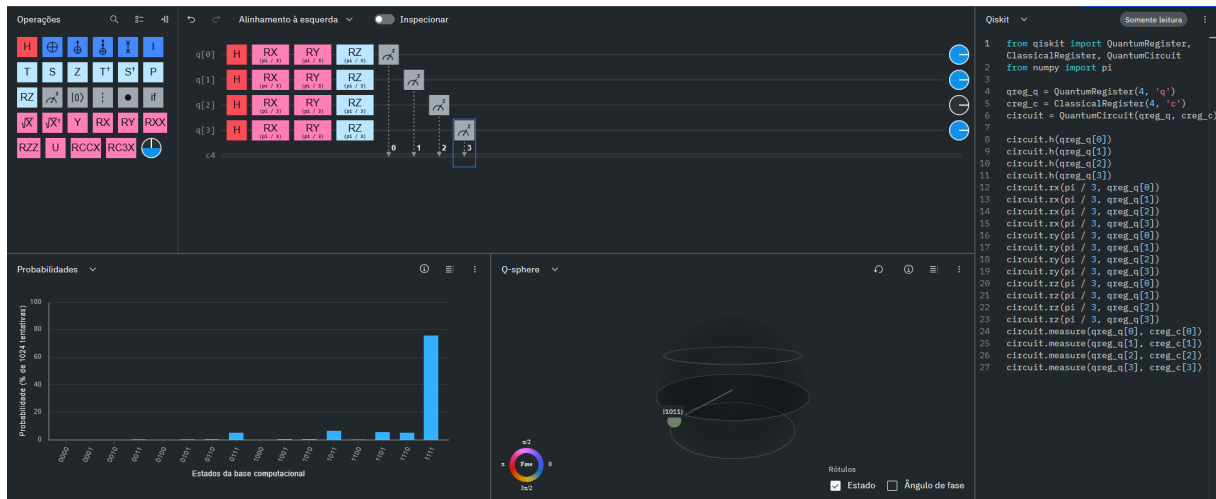


Figura 10 – Plataforma do *IBM Quantum Composer* (Fonte: Autor)

Além disso, a flexibilidade e a escalabilidade do simulador *Qiskit Aer* facilita a experimentação sob diferentes paradigmas da computação quântica. Seu conjunto de ferramentas e funcionalidades possibilita a implementação e a avaliação de algoritmos e estruturas mais complexas, enriquecendo o processo experimental e contribuindo para a obtenção de resultados com maior fidelidade.

5.2 Gerador Pseudoaleatório Quântico de Quasi-Probabilidade (QQ-PRG)

Esta seção apresenta os resultados experimentais obtidos a partir do Algoritmo 1 (algoritmo original) e do Algoritmo 2 (algoritmo modificado com entrelaçamento quântico e randomização caótica dos ângulos de rotação), com o objetivo de avaliar e comparar a aleatoriedade das sequências geradas por suas saídas. A implementação detalhada de ambos os algoritmos encontra-se descrita nas Seções 4.1.1 e 4.1.2, respectivamente.

A análise experimental é organizada em duas partes. Inicialmente, é apresentada a configuração adotada para a execução dos experimentos de cada uma das duas versões do algoritmo QQ-PRG. Em seguida, são discutidos os testes estatísticos (detalhados na Seção 2.2.3) aplicados aos resultados obtidos por ambas as versões, de modo a avaliar o impacto das melhorias propostas sobre o grau de aleatoriedade dos valores produzidos.

5.2.1 Configuração para os Testes do Algoritmo QQ-PRG

Para a avaliação do algoritmo QQ-PRG, é necessária a definição de uma configuração inicial composta por diversos parâmetros fundamentais ao seu funcionamento. A Tabela 3 apresenta os parâmetros adotados para o algoritmo original, enquanto a Tabela 4 apresenta os parâmetros adotados para o algoritmo modificado. Em ambos os casos, a configuração foi escolhida de modo a possibilitar a geração de uma sequência com dimensões adequadas às aplicações de encriptação e decriptação de imagens consideradas neste trabalho, mantendo o mesmo comprimento de sequência L e o mesmo número k de sistemas quânticos para permitir uma comparação direta entre as duas versões.

Tabela 3 – Parâmetros utilizados na configuração do algoritmo QQ-PRG original

Parâmetro	Valor
Número de <i>qubits</i>	$n = 3$
Fator de escala	$sf = \frac{1}{100}$
Comprimento da sequência	$L = 2048 \times 2048 = 4.194.304$
Número de sistemas quânticos	$k = \frac{2048 \times 2048}{2^n} = 524.288$
Ângulos de rotação	$R_x = \frac{\pi}{3}, R_y = \frac{\pi}{3}, R_z = \frac{\pi}{3}$

Tabela 4 – Parâmetros utilizados na configuração do algoritmo QQ-PRG modificado

Parâmetro	Valor
Número de <i>qubits</i>	$n = 3$
Fator de escala	$f_s = \frac{1}{100}$
Comprimento da sequência	$L = 2048 \times 2048 = 4.194.304$
Número de sistemas quânticos	$k = \frac{2048 \times 2048}{2^n} = 524.288$
Semente do mapa logístico	x_0 (derivada do relógio do sistema)
Período de <i>warm-up</i>	1000 iterações
<i>Shots</i> por circuito	2048
Topologia de entrelaçamento	cadeia ($\text{CNOT}_{i,i+1}, i = 0, \dots, n-2$)

5.2.2 Teste Estatístico do QQ-PRG

As sequências de números pseudoaleatórios obtidas a partir da execução de cada uma das duas versões do algoritmo (Algoritmos 1 e 2) foram avaliadas por meio da bateria de testes estatísticos NIST SP 800-22. Para cada um dos 15 testes que compõem

essa bateria, foi adotado um nível de significância $\alpha = 0.01$, o que implica que, para ser considerada adequada, a sequência deve apresentar uma probabilidade de pelo menos 99% de satisfazer os critérios de aleatoriedade estabelecidos. Cada sequência completa, composta por 4.194.304 números decimais de 10 bits cada (totalizando 41.943.040 bits), foi subdividida em 41 subsequências para fins de teste, com taxa mínima de aprovação de aproximadamente 38/41 para a maioria dos testes e de aproximadamente 18/20 para o teste de Desvios Aleatórios Variante.

A Tabela 5 apresenta os resultados obtidos pelo algoritmo original. Observa-se que o gerador obteve sucesso em apenas 2 dos 15 testes da bateria (*Rank* e *LinearComplexity*), enquanto todos os demais testes falharam de forma acentuada, com praticamente a totalidade das 41 subsequências reprovadas (proporção 0/41 na maioria dos casos). Chama atenção, em particular, o resultado dos testes de *Random Excursions* e *Random Excursions Variant*: para nenhuma das subsequências foi possível computar essas estatísticas, uma vez que elas exigem um número mínimo de cruzamentos por zero (*cycles*) ao longo do passeio aleatório associado à sequência, uma indicação adicional do alto grau de estrutura e repetição presente na sequência gerada pelo algoritmo original, consistente com a ausência de entrelaçamento e com o uso de ângulos de rotação fixos.

Tabela 5 – Resultado da aplicação da bateria de testes NIST SP 800-22 no algoritmo original

C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	P-Value	Proporção	Teste Estatístico
41	0	0	0	0	0	0	0	0	0	0.000000*	0/41*	<i>Frequency</i>
41	0	0	0	0	0	0	0	0	0	0.000000*	0/41*	<i>BlockFrequency</i>
41	0	0	0	0	0	0	0	0	0	0.000000*	0/41*	<i>CumulativeSums</i>
41	0	0	0	0	0	0	0	0	0	0.000000*	0/41*	<i>CumulativeSums</i>
41	0	0	0	0	0	0	0	0	0	0.000000*	0/41*	<i>Runs</i>
41	0	0	0	0	0	0	0	0	0	0.000000*	0/41*	<i>LongestRun</i>
3	8	2	7	3	5	5	2	1	5	0.227773	41/41	<i>Rank</i>
41	0	0	0	0	0	0	0	0	0	0.000000*	0/41*	<i>FFT</i>
—	—	—	—	—	—	—	—	—	—	—	0/148*	<i>NonOverlappingTemplate</i>
41	0	0	0	0	0	0	0	0	0	0.000000*	0/41*	<i>OverlappingTemplate</i>
30	2	2	3	1	2	0	1	0	0	0.000000*	22/41*	<i>Universal</i>
41	0	0	0	0	0	0	0	0	0	0.000000*	0/41*	<i>ApproximateEntropy</i>
—	—	—	—	—	—	—	—	—	—	—	—*	<i>RandomExcursions</i>
—	—	—	—	—	—	—	—	—	—	—	—*	<i>RandomExcursionsVariant</i>
41	0	0	0	0	0	0	0	0	0	0.000000*	0/41*	<i>Serial</i>
41	0	0	0	0	0	0	0	0	0	0.000000*	0/41*	<i>Serial</i>
3	3	9	1	5	5	7	4	2	2	0.131500	41/41	<i>LinearComplexity</i>

A Tabela 6 apresenta os resultados obtidos pelo algoritmo modificado, incorporando o entrelaçamento via portas C-NOT e a randomização dos ângulos de rotação via mapa logístico (Seções 3.1.3 e 3.1.4). Em contraste direto com o algoritmo original, o gerador modificado obteve sucesso nos 15 testes da bateria, incluindo a totalidade das 148 variantes do teste *NonOverlappingTemplate* e das 8 e 18 variantes, respectivamente, dos testes *Random Excursions* e *Random Excursions Variant*, ambos inconclusivos no

algoritmo original por ausência de dados suficientes, e agora plenamente computáveis e aprovados, o que por si só já indica uma sequência com uma estrutura de flutuação muito mais rica e menos enviesada.

Tabela 6 – Resultado da aplicação da bateria de testes NIST SP 800-22 no algoritmo modificado

C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	<i>P-Value</i>	Proporção	Teste Estatístico
8	5	6	5	3	1	6	2	1	4	0.199580	40/41	<i>Frequency</i>
2	10	2	5	2	4	5	2	6	3	0.098036	41/41	<i>BlockFrequency</i>
7	3	5	2	7	5	3	6	2	1	0.293235	40/41	<i>CumulativeSums</i>
7	6	3	6	5	2	2	5	2	3	0.509162	40/41	<i>CumulativeSums</i>
5	2	5	3	7	4	9	0	3	3	0.098036	41/41	<i>Runs</i>
2	3	5	5	9	5	5	2	0	5	0.131500	41/41	<i>LongestRun</i>
5	6	5	2	5	5	2	3	3	5	0.855534	41/41	<i>Rank</i>
6	3	2	6	1	4	4	5	5	5	0.714660	41/41	<i>FFT</i>
—	—	—	—	—	—	—	—	—	—	—	148/148	<i>NonOverlappingTemplate</i>
6	1	5	4	7	1	7	1	3	6	0.131500	39/41	<i>OverlappingTemplate</i>
3	2	2	4	5	6	3	8	2	6	0.371101	41/41	<i>Universal</i>
2	9	6	3	3	4	5	1	7	1	0.072289	40/41	<i>ApproximateEntropy</i>
—	—	—	—	—	—	—	—	—	—	—	8/8	<i>RandomExcursions</i>
—	—	—	—	—	—	—	—	—	—	—	18/18	<i>RandomExcursionsVariant</i>
4	10	2	5	3	4	3	4	6	0	0.072289	41/41	<i>Serial</i>
3	9	5	6	3	3	2	4	4	2	0.330628	41/41	<i>Serial</i>
3	3	3	3	2	5	3	6	5	8	0.559523	41/41	<i>LinearComplexity</i>

Os resultados confirmam, de forma direta e quantitativa, a hipótese que motivou as modificações propostas neste trabalho: a ausência de entrelaçamento e o uso de ângulos de rotação fixos no algoritmo original produzem uma sequência de saída altamente estruturada e previsível, que falha em praticamente todos os critérios de aleatoriedade da bateria NIST SP 800-22 (aprovação em apenas 2 dos 15 testes). A introdução do entrelaçamento quântico via portas C-NOT, combinada à randomização contínua dos ângulos de rotação por meio do mapa logístico, elimina essa estrutura: o algoritmo modificado aprova a totalidade dos 15 testes da bateria, incluindo todas as variantes dos testes *NonOverlappingTemplate*, *Random Excursions* e *Random Excursions Variant*.

5.3 Testes de Encriptação e Decriptação de Imagens

Esta seção apresenta uma análise detalhada do desempenho dos esquemas de encriptação e decriptação descritos na Seção 2.2.4. O objetivo é avaliar a robustez estatística e a segurança do método proposto quando aplicado a imagens digitais, comparando o desempenho obtido com a sequência de chave gerada pelo algoritmo QQ-PRG original (Algoritmo 5) e com a sequência gerada pelo algoritmo QQ-PRG modificado (Algoritmo 6), incorporando o entrelaçamento quântico via portas C-NOT e a randomização caótica dos ângulos de rotação.

Os processos de encriptação e decriptação são implementados por meio do Algoritmo 8, conforme detalhado na Seção 4.1.4. Esse esquema opera a partir das sequências geradas por qualquer uma das duas versões do QQ-PRG, que atuam como geradoras da chave pseudoaleatória utilizada no esquema simétrico, permitindo, assim, isolar o efeito das melhorias propostas sobre a qualidade da encriptação resultante, mantendo fixos os demais elementos do esquema (operação XOR, normalização, formato das imagens).

A avaliação experimental é conduzida por meio de quatro testes principais: análise de histogramas, teste de recorte (ataque por corte), teste de adição de ruído e análise de correlação entre pixels adjacentes (implementados conforme as Seções 4.1.5, 4.1.6 e 4.1.7). Esses testes permitem investigar propriedades fundamentais de um sistema criptográfico aplicado a imagens, tais como difusão, resistência a perdas parciais de dados, robustez a perturbações externas e eliminação de dependências estatísticas espaciais, para cada uma das duas versões do algoritmo.

5.3.1 Imagens Utilizadas nos Testes

Os dados utilizados nos experimentos de encriptação e decriptação de imagens são apresentados na Figura 11. Esse conjunto é composto por quatro imagens no formato RGB, amplamente empregadas em estudos de processamento e criptografia de imagens, permitindo uma análise consistente do desempenho dos métodos de encriptação e decriptação propostos, sob ambas as versões do algoritmo QQ-PRG.

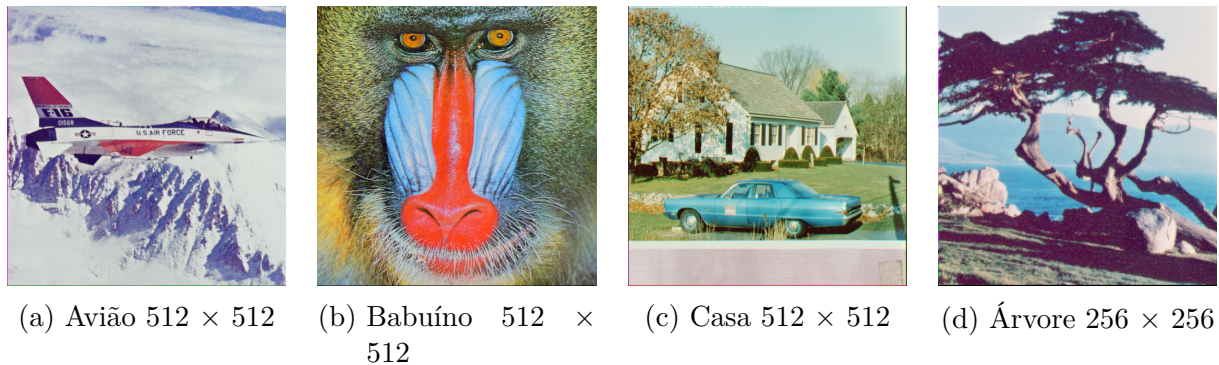


Figura 11 – Imagens usadas nos experimentos. (Fonte: [Universidad de Granada](#))

5.3.2 Parâmetros QQ-PRG para Encriptação e Decriptação

A avaliação dos algoritmos de encriptação e decriptação requer a utilização de uma chave simétrica composta por uma sequência de valores pseudoaleatórios, gerada por meio da configuração apropriada de cada uma das duas versões do algoritmo QQ-PRG. Em ambos os casos, o comprimento da sequência é determinado pela quantidade de pixels das imagens de teste de 512×512 (Figura 11), garantindo que a chave gerada seja compatível com o esquema de encriptação por XOR (Algoritmo 8). Para a imagem de 256×256 ,

utiliza-se o subconjunto inicial da mesma sequência gerada, ou uma nova execução com L ajustado, conforme a estratégia de experimentação adotada.

A Tabela 7 apresenta os parâmetros adotados na geração da sequência pelo algoritmo original, e a Tabela 8 apresenta os parâmetros adotados na geração da sequência pelo algoritmo modificado.

Tabela 7 – Parâmetros utilizados no algoritmo QQ-PRG original para os testes de encriptação e deciptação

Parâmetro	Valor
Número de <i>qubits</i>	$n = 3$
Fator de escala	$sf = \frac{1}{100}$
Comprimento da sequência	$L = 512 \times 512 = 262.144$
Número de sistemas quânticos	$k = \frac{512 \times 512}{2^n} = 32.768$
Ângulos de rotação	$R_x = \frac{\pi}{3}, R_y = \frac{\pi}{3}, R_z = \frac{\pi}{3}$

Tabela 8 – Parâmetros utilizados no algoritmo QQ-PRG modificado para os testes de encriptação e deciptação

Parâmetro	Valor
Número de <i>qubits</i>	$n = 3$
Fator de escala	$f_s = \frac{1}{100}$
Comprimento da sequência	$L = 512 \times 512 = 262.144$
Número de sistemas quânticos	$k = \frac{512 \times 512}{2^n} = 32.768$
Semente do mapa logístico	x_0 (derivada do relógio do sistema)
Período de <i>warm-up</i>	1000 iterações
<i>Shots</i> por circuito	2048
Topologia de entrelaçamento	cadeia ($\text{CNOT}_{i,i+1}, i = 0, \dots, n-2$)

5.3.3 Testes de Encriptação e Deciptação

Para verificar a eficácia e a robustez dos algoritmos de criptografia propostos, são conduzidos diversos testes com o objetivo de avaliar seu desempenho sob diferentes condições operacionais. Entre os testes selecionados estão as análises de histograma, recorte (ataque por corte), adição de ruído e correlação entre pixels adjacentes (CRUZ, 2024). Cada um desses testes é projetado para investigar propriedades específicas dos dados encriptados e analisar o comportamento estatístico das imagens resultantes. Em todos os testes, os resultados obtidos pelo algoritmo original (Algoritmo 1) e pelo algoritmo modificado (Algoritmo 2) são apresentados lado a lado, permitindo uma avaliação comparativa direta do impacto das melhorias propostas sobre a qualidade da encriptação.

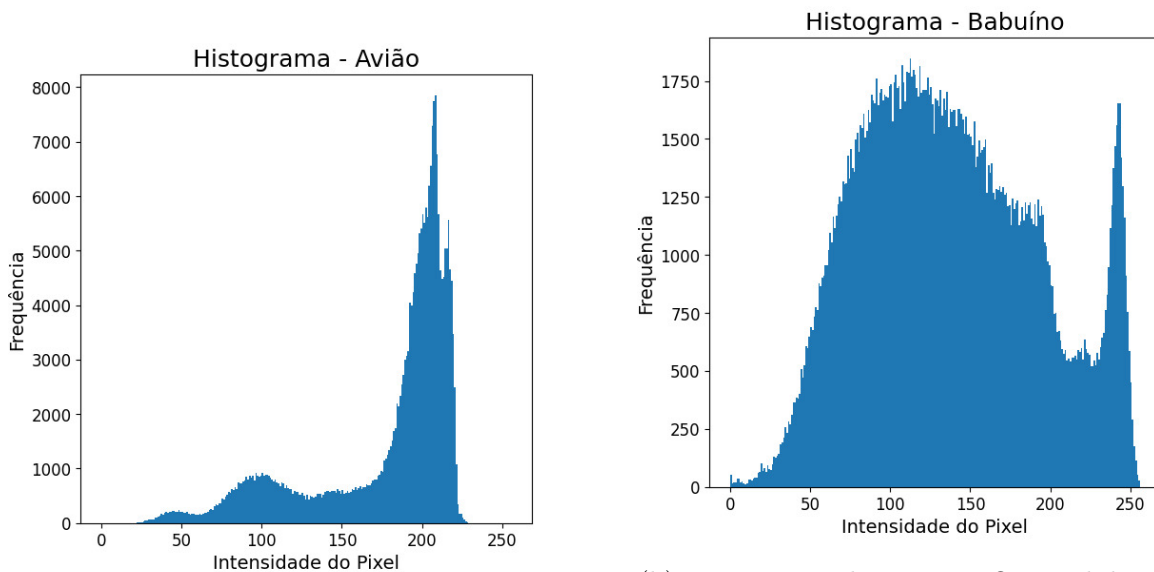
5.3.3.1 Análise de Histogramas

A análise de histograma fornece informações sobre a distribuição de intensidade dos pixels, permitindo avaliar se o processo de encriptação elimina padrões estruturais

presentes na imagem original. Uma distribuição aproximadamente uniforme é desejável, pois indica ausência de vazamento estatístico de informação.

Em histogramas de imagens digitais, o eixo horizontal representa os níveis de intensidade dos pixels, normalmente variando entre 0 e 255 em imagens de 8 *bits*, enquanto o eixo vertical representa a frequência de ocorrência desses valores. Dessa forma, o histograma descreve quantas vezes cada intensidade aparece na imagem, permitindo analisar propriedades estatísticas e padrões visuais presentes nos dados.

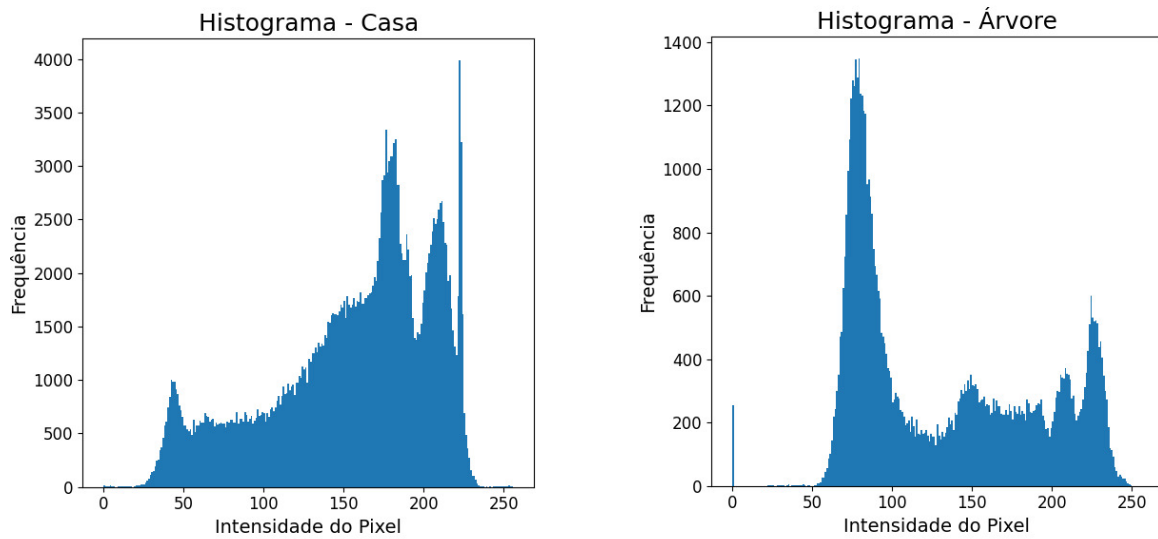
As imagens originais apresentam histogramas característicos que refletem propriedades estruturais específicas de cada cena. A Figura 11 exibe o conjunto de imagens utilizadas (Avião, Babuíno, Casa e Árvore), enquanto seus respectivos histogramas do canal vermelho são apresentados nos painéis (a), (b), (c) e (d) da Figura 12. Observa-se que os histogramas das imagens originais não seguem uma distribuição uniforme, evidenciando padrões estatísticos inerentes ao conteúdo visual de cada imagem.



(a) Histograma da Imagem Original do Avião

(b) Histograma da Imagem Original do Babuíno

Figura 12 – Histogramas das Imagens Originais. (Fonte: Autor)



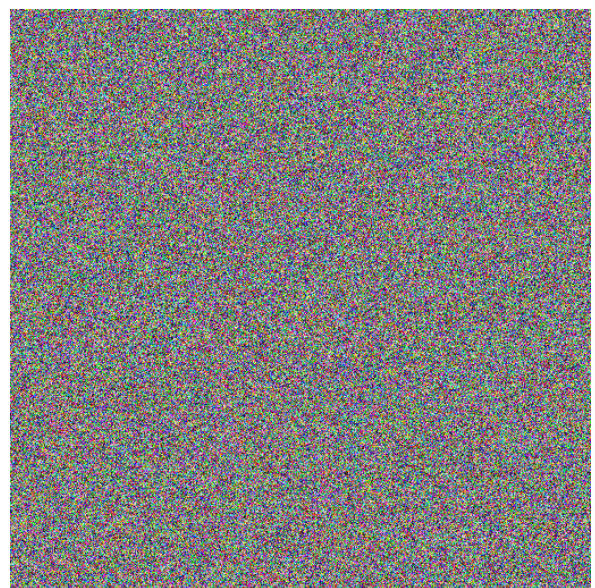
(c) Histograma da Imagem Original da Casa (d) Histograma da Imagem Original da Árvore

Figura 12 – Histogramas das Imagens Originais (cont.). (Fonte: Autor)

A Figura 13 apresenta as versões encriptadas das imagens utilizando os dois algoritmos avaliados. Em cada par, a imagem da esquerda foi gerada pelo algoritmo original e a da direita pelo algoritmo modificado, ambas produzidas a partir da mesma imagem de entrada por meio do Algoritmo 8 utilizando, respectivamente, a sequência pseudoaleatória de cada versão do QQ-PRG como chave simétrica. Visualmente, ambas as versões eliminam completamente os padrões estruturais da imagem original, produzindo saídas com aparência de ruído aleatório indistinguíveis a olho nu, resultado esperado para qualquer esquema de encriptação com boa difusão.



(a) Babuíno Encriptado — Algoritmo Original

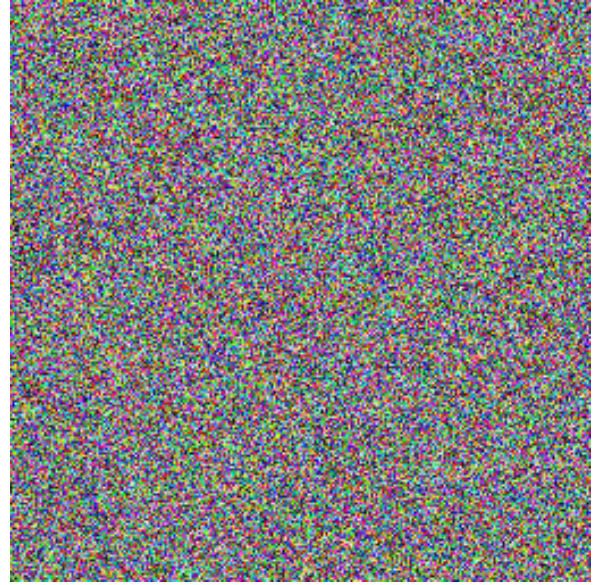


(b) Babuíno Encriptado — Algoritmo Modificado

Figura 13 – Imagens Encriptadas pelos dois algoritmos. (Fonte: Autor)



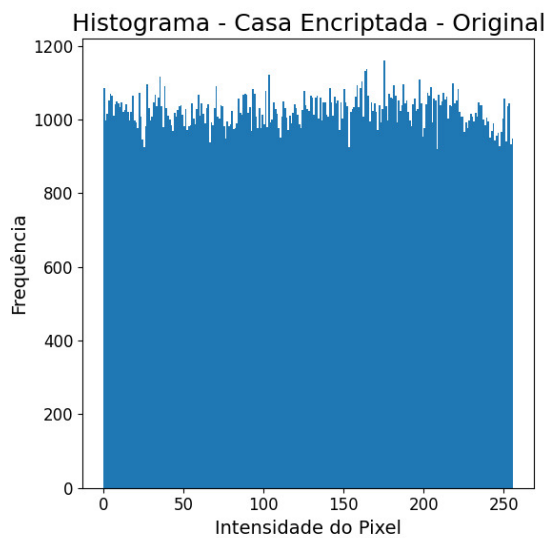
(c) Árvore Encriptada — Algoritmo Original



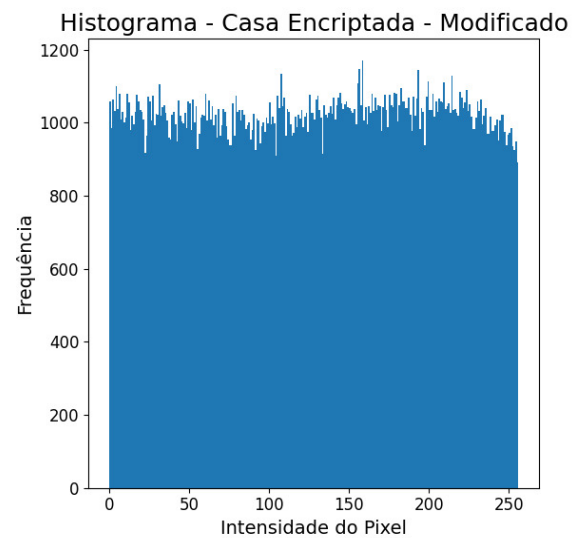
(d) Árvore Encriptada — Algoritmo Modificado

Figura 13 – Imagens Encriptadas pelos dois algoritmos (cont.). (Fonte: Autor)

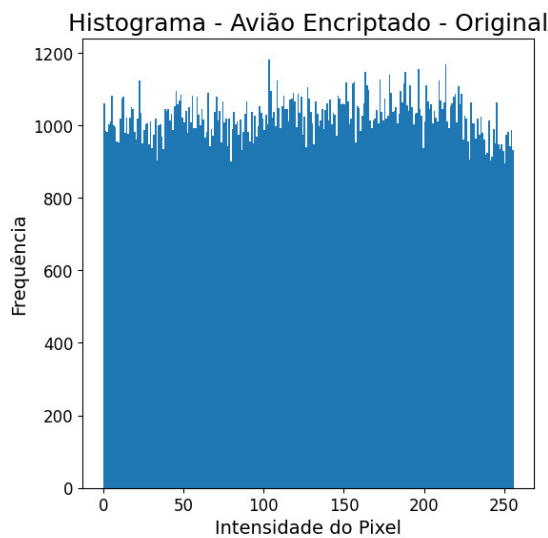
A Figura 14 mostra os histogramas correspondentes às imagens encriptadas (canal vermelho), apresentados em pares de algoritmo original e modificado para a imagem da Casa e do Babuíno. Em ambos os casos, observa-se uma distribuição aproximadamente uniforme das intensidades de píxel no intervalo $[0, 255]$, com frequências oscilando em torno de ~ 1000 ocorrências por nível de intensidade para a imagem de 512×512 pixels — valor próximo ao esperado de $\frac{512 \times 512}{256} \approx 1024$ para uma distribuição perfeitamente uniforme. Não se observa, visual ou numericamente, diferença significativa entre os histogramas do algoritmo original e do modificado para este teste, resultado que é esperado e, na verdade, positivo: significa que ambos os algoritmos promovem difusão estatística suficiente para eliminar os padrões da imagem original, e que a análise de histograma, por si só, não é sensível o suficiente para discriminar os dois geradores. A distinção qualitativa entre os algoritmos foi capturada de forma mais precisa pelos testes estatísticos NIST SP 800-22 apresentados na Seção 5.2, onde o algoritmo modificado demonstrou superioridade significativa.



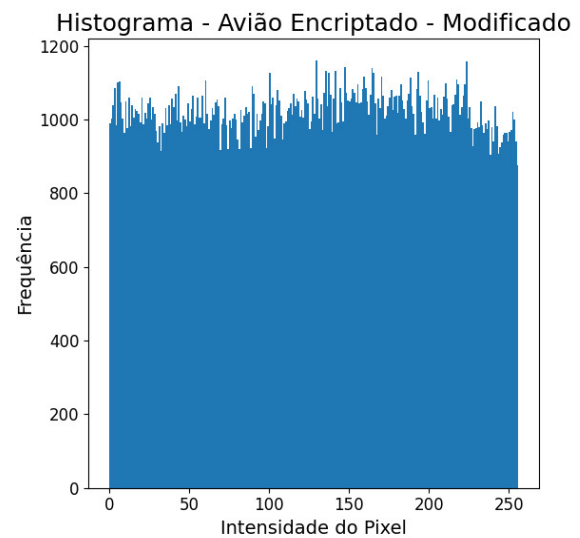
(a) Histograma da Casa Encriptada - Algoritmo Original



(b) Histograma da Casa Encriptada - Algoritmo Modificado



(c) Histograma do Avião Encriptado - Algoritmo Original



(d) Histograma do Avião Encriptado - Algoritmo Modificado

Figura 14 – Histogramas das Imagens Encriptadas pelos dois algoritmos. (Fonte: Autor)

Por fim, as imagens decriptadas são obtidas aplicando novamente o Algoritmo 8 com a mesma chave pseudoaleatória utilizada na encriptação. A Figura 15 apresenta os resultados da reconstrução, evidenciando a correta recuperação das imagens originais em ambas as versões do algoritmo e, conseqüentemente, a consistência do esquema simétrico adotado.

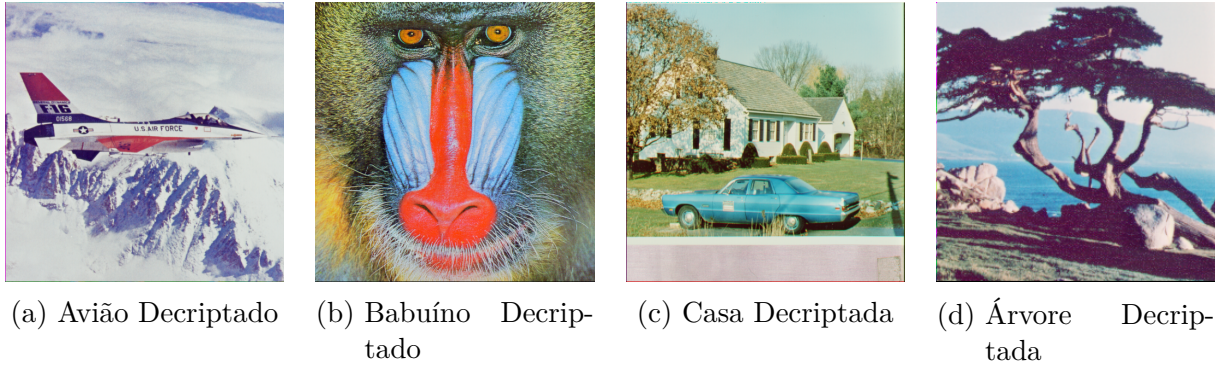


Figura 15 – Imagens Originais Decriptadas. (Fonte: Autor)

5.3.3.2 Teste de Ataque de Recorte

O teste de recorte consiste na remoção deliberada de regiões da imagem encriptada, com o objetivo de avaliar a capacidade do algoritmo de recuperar informações após perda parcial de dados. Esse teste está relacionado à propriedade de difusão do sistema criptográfico.

Com o objetivo de avaliar a resiliência do esquema proposto frente à perda parcial de dados, foi realizado um teste de ataque por recorte (*cropping attack*). Esse teste consiste na remoção deliberada de regiões específicas da imagem encriptada, simulando situações de degradação que podem ocorrer durante a transmissão ou armazenamento da informação.

Diferentemente dos testes de histograma e de correlação, o resultado do teste de recorte não depende de qual dos dois algoritmos QQ-PRG gerou a chave de encriptação. Isso decorre diretamente da natureza do esquema adotado: o Algoritmo 8 opera por XOR pixel a pixel, sem interdependência entre posições distintas da imagem. Assim, zerar uma região da imagem encriptada equivale a substituir os valores naquela região pelos valores da chave (pois $0 \oplus k_i = k_i$), e a deciptação de cada pixel fora da região afetada permanece intacta independentemente do gerador utilizado. O padrão de degradação na imagem deciptada é, portanto, determinado pela geometria da região removida e pela operação XOR, e não pelas propriedades estatísticas da sequência pseudoaleatória. Por essa razão, os resultados são apresentados uma única vez, sendo válidos para ambas as versões do algoritmo.

A robustez do sistema é analisada por meio da deciptação das imagens recortadas, verificando-se o impacto da perda de dados na reconstrução da imagem original. A Figura 16 apresenta os resultados obtidos: as imagens encriptadas foram submetidas à remoção de regiões quadradas de tamanhos 60×60 , 180×180 e 360×360 pixels, respectivamente (painéis a, b e c), e as correspondentes imagens deciptadas são mostradas nos painéis (d), (e) e (f).

Observa-se que a degradação na imagem decriptada é localizada e proporcional à área removida, sem propagação para regiões adjacentes, comportamento esperado em um cifrador de fluxo baseado em XOR. A imagem original é fielmente recuperada em todas as regiões não afetadas pelo recorte, independentemente do tamanho da área removida.

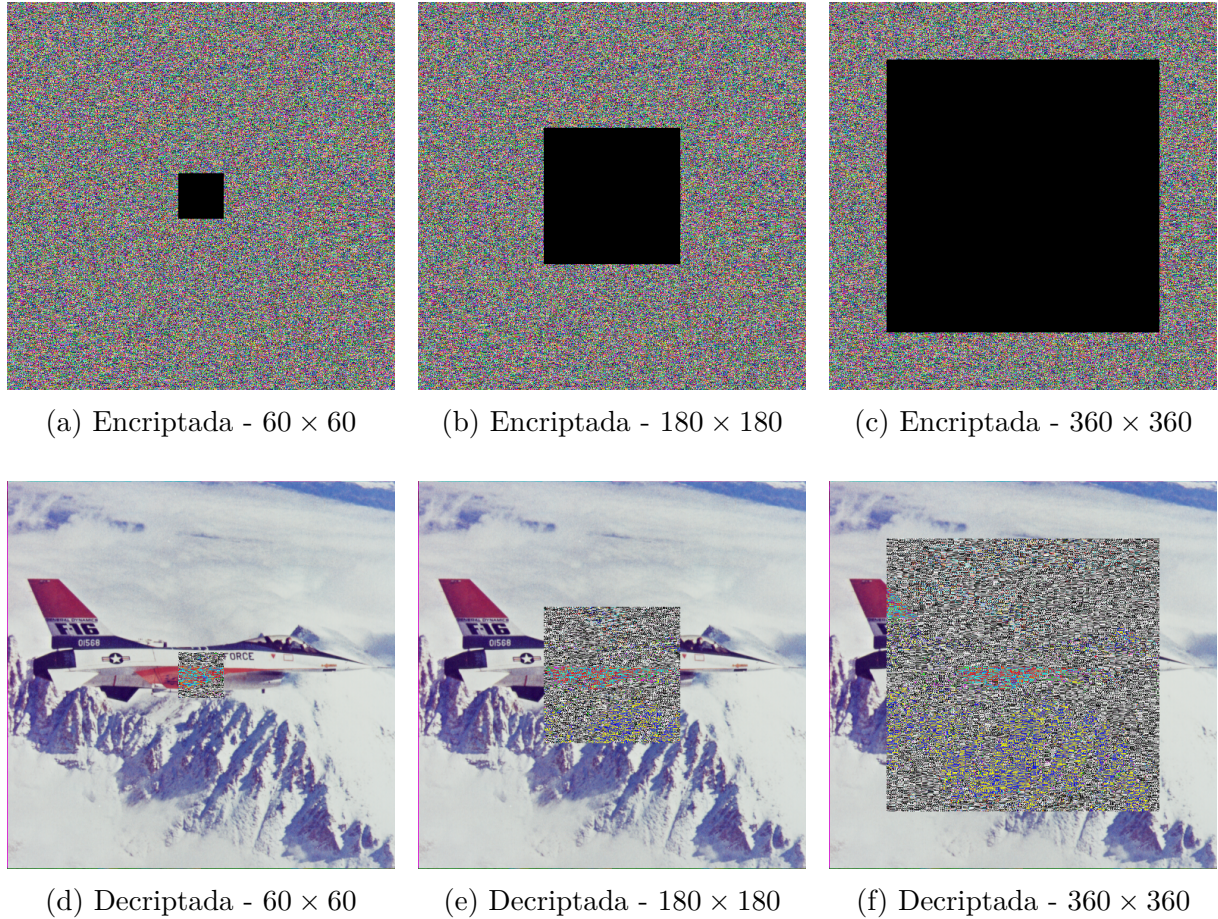


Figura 16 – Resultados do recorte utilizando diferentes tamanhos de região cifrada.
(Fonte: Autor)

5.3.3.3 Teste de Análise de Ruído

O teste de adição de ruído introduz perturbações aleatórias na imagem encriptada, analisando a resiliência do algoritmo frente a interferências externas ou degradação do canal de transmissão.

Para avaliar a resiliência das imagens encriptadas frente a possíveis degradações durante processos de transmissão e armazenamento, foi conduzido um teste de análise de ruído. Esse teste consiste na introdução deliberada de ruído do tipo *salt-and-pepper*, implementado conforme descrito na Seção 4.1.5, simulando a corrupção aleatória de pixels semelhante às interferências comumente observadas em canais de comunicação digital e dispositivos de armazenamento.

Pelo mesmo argumento apresentado na subseção anterior, o resultado desse teste

também é independente do PRNG utilizado na geração da chave: o ruído corrompe pixels individuais da imagem encriptada, e a decriptação de cada posição afetada produz um valor incorreto de forma estritamente localizada, sem propagação para posições vizinhas — consequência direta da independência espacial do XOR. Os pixels não afetados pelo ruído são decriptados corretamente em ambos os algoritmos. Por essa razão, os resultados são apresentados uma única vez, sendo válidos para ambas as versões do QQ-PRG.

A Figura 17 apresenta os experimentos realizados, exibindo as imagens cifradas afetadas por três níveis de ruído, 0.4%, 4% e 40% de pixels corrompidos, bem como suas respectivas versões decriptadas. Esses valores foram definidos para representar cenários de interferência leve, moderada e severa.

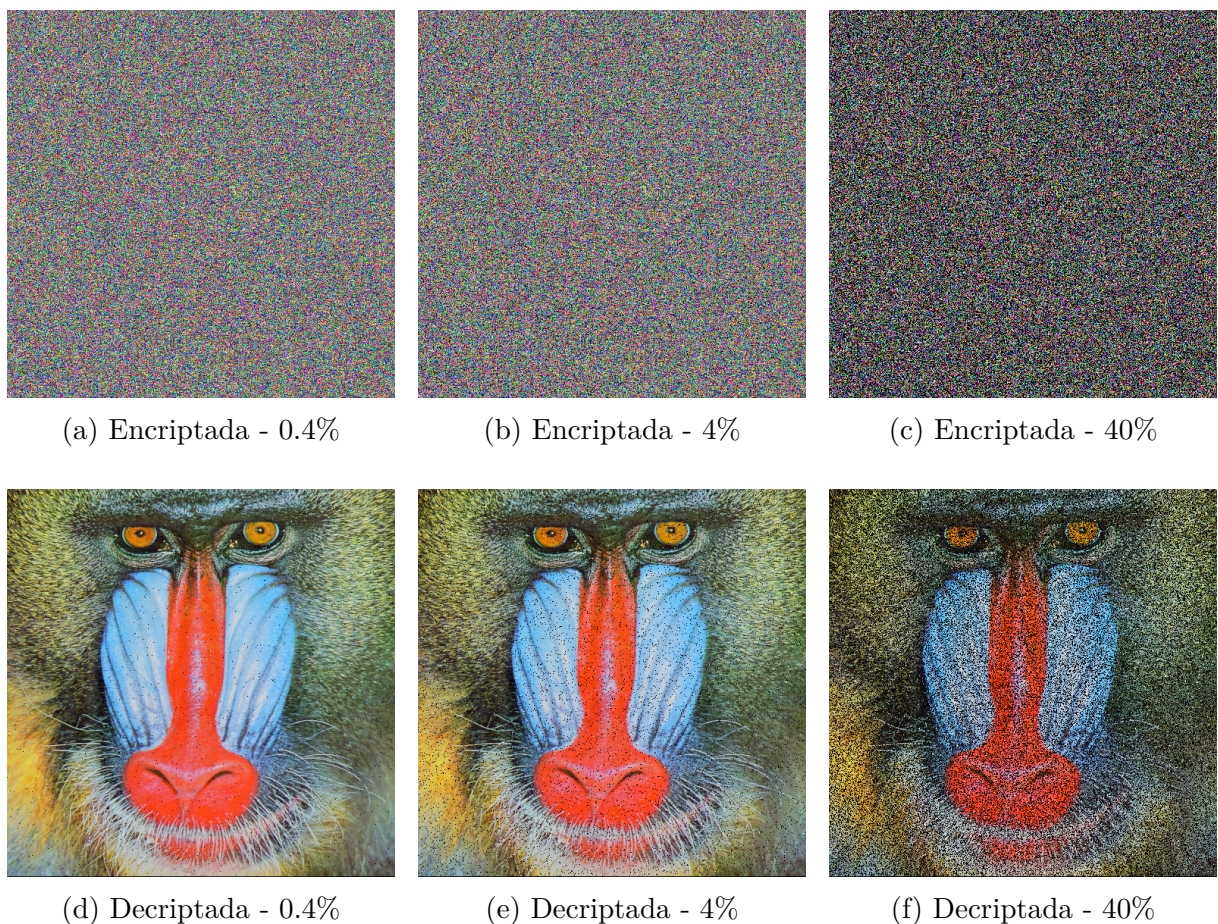


Figura 17 – Imagens encriptadas com ruído *salt-and-pepper* e suas versões decriptadas. (Fonte: Autor)

As imagens decriptadas (d), (e) e (f) demonstram que a degradação é estritamente proporcional ao nível de ruído aplicado e localizada nos pixels individualmente corrompidos: para 0.4%, a imagem recuperada permanece praticamente intacta; para 4%, surgem artefatos visuais perceptíveis mas a imagem original ainda é claramente reconhecível; para 40%, a corrupção é severa o suficiente para comprometer a identificabilidade visual da cena. Em todos os casos, os pixels não afetados pelo ruído são corretamente decriptados,

confirmando a consistência do esquema simétrico adotado.

5.3.3.4 Análise de Correlação

Por fim, a análise de correlação entre pixels mede o grau de dependência estatística entre pixels adjacentes nas imagens original e encriptada. Um esquema criptográfico eficiente deve reduzir significativamente essa correlação na imagem encriptada, dificultando a identificação de padrões estruturais.

Para garantir uma avaliação abrangente, os coeficientes de correlação de Pearson (Eq. 4.4) foram calculados nas direções horizontal, vertical e diagonal para as quatro imagens de teste (Avião, Babuíno, Casa e Árvore), tanto no formato RGB quanto em escala de cinza, considerando separadamente as versões encriptadas pelos dois algoritmos avaliados. Os resultados são apresentados nas Tabelas 9 e 10.

Tabela 9 – Distribuição da Correlação para as imagens RGB

Imagem	Direção	R	G	B
Avião	Horizontal	0.972	0.957	0.963
	Vertical	0.956	0.967	0.935
	Diagonal	0.934	0.932	0.914
Avião Encriptado Algoritmo Original	Horizontal	0.191	0.038	0.013
	Vertical	0.002	−0.001	0.001
	Diagonal	−0.002	0.000	−0.002
Avião Encriptado Algoritmo Modificado	Horizontal	0.000	−0.002	0.002
	Vertical	0.002	−0.002	0.000
	Diagonal	−0.003	−0.003	−0.000
Árvore	Horizontal	0.958	0.968	0.961
	Vertical	0.936	0.945	0.940
	Diagonal	0.915	0.931	0.926
Árvore Encriptada Algoritmo Original	Horizontal	0.172	0.026	0.008
	Vertical	−0.005	−0.006	−0.000
	Diagonal	−0.006	0.007	0.001
Árvore Encriptada Algoritmo Modificado	Horizontal	−0.005	−0.003	−0.000
	Vertical	−0.002	0.004	0.003
	Diagonal	−0.003	0.000	0.002
Babuíno	Horizontal	0.923	0.865	0.907
	Vertical	0.865	0.765	0.880
	Diagonal	0.854	0.734	0.839
Babuíno Encriptado Algoritmo Original	Horizontal	0.132	0.011	0.000
	Vertical	0.000	0.000	0.000
	Diagonal	−0.001	−0.000	0.000
Babuíno Encriptado Algoritmo Modificado	Horizontal	−0.002	−0.001	−0.000
	Vertical	0.000	0.005	0.004
	Diagonal	−0.002	0.003	0.000
Casa	Horizontal	0.953	0.939	0.972
	Vertical	0.957	0.942	0.968
	Diagonal	0.922	0.890	0.944
Casa Encriptada Algoritmo Original	Horizontal	0.175	0.034	0.012
	Vertical	0.000	−0.002	−0.001
	Diagonal	−0.000	−0.001	0.001
Casa Encriptada Algoritmo Modificado	Horizontal	0.002	−0.002	−0.000
	Vertical	0.001	−0.001	0.001
	Diagonal	0.001	−0.002	−0.001

Tabela 10 – Distribuição da Correlação para as imagens em Escala de Cinza

Imagem	Horizontal	Vertical	Diagonal
Avião	0.966	0.964	0.937
Avião Encriptado Algoritmo Original	0.224	0.000	0.003
Avião Encriptado Algoritmo Modificado	−0.001	−0.002	−0.002
Árvore	0.968	0.945	0.929
Árvore Encriptada Algoritmo Original	0.210	0.012	0.011
Árvore Encriptada Algoritmo Modificado	−0.005	0.002	−0.000
Babuíno	0.866	0.758	0.726
Babuíno Encriptado Algoritmo Original	0.157	−0.000	0.004
Babuíno Encriptado Algoritmo Modificado	0.000	0.005	0.003
Casa	0.948	0.957	0.913
Casa Encriptada Algoritmo Original	0.206	−0.001	0.002
Casa Encriptada Algoritmo Modificado	−0.002	0.000	−0.002

Como esperado, as imagens originais apresentam coeficientes de correlação elevados em todas as direções e canais analisados, com valores entre 0.726 e 0.972, refletindo a forte dependência estatística existente entre pixels vizinhos em imagens naturais. Após a encriptação, ambos os algoritmos reduzem esses coeficientes de forma substancial — resultado consistente com o comportamento observado na análise de histograma da Seção 5.3.3.1.

No entanto, diferentemente do histograma, a análise de correlação revela uma distinção quantitativamente significativa entre os dois algoritmos. No algoritmo original, observa-se uma correlação horizontal residual sistemática no canal vermelho (R) e, de forma ainda mais pronunciada, na escala de cinza: os valores horizontais variam entre 0.132 (Babuíno) e 0.224 (Avião em escala de cinza), enquanto as direções vertical e diagonal se aproximam de zero. Esse padrão é consistente com a estrutura da sequência de saída do algoritmo original identificada nos testes NIST SP 800-22 (Seção 5.2): a presença de regularidades na componente horizontal da chave traduz-se diretamente em correlação residual entre pixels horizontalmente adjacentes na imagem encriptada.

No algoritmo modificado, essa assimetria é completamente eliminada: os coeficientes de correlação em todas as direções, canais e imagens testadas situam-se no intervalo $[-0.005, 0.005]$, valores estatisticamente indistinguíveis de zero e sem distinção sistemática entre as direções analisadas. Esse resultado confirma, no domínio da encriptação de imagens, a melhoria na qualidade estatística da sequência pseudoaleatória já verificada pelos testes NIST: a randomização caótica dos ângulos de rotação e o entrelaçamento quântico eliminam as regularidades que se manifestavam como correlação horizontal residual no algoritmo original, produzindo uma chave cuja estrutura não favorece nenhuma direção espacial.

6 Conclusão

Este trabalho teve como objetivo investigar o uso de circuitos quânticos como fonte de números pseudoaleatórios, por meio da replicação do algoritmo *Quantum Quasi-Probability Pseudo Random Generator* (QQ-PRG) (CRUZ, 2024), e propor melhorias a esse algoritmo com base em uma fonte complementar de aleatoriedade de natureza clássica: o mapa logístico em regime de caos completo. A motivação central foi verificar se a introdução de entrelaçamento quântico, por meio de portas C-NOT, e a substituição de parâmetros de rotação fixos por uma sequência caótica seriam capazes de corrigir as limitações estatísticas do gerador original, e em que medida essas melhorias se refletiriam em uma aplicação prática de segurança da informação, a encriptação simétrica de imagens digitais por XOR.

A fundamentação teórica reunida no Capítulo 2 estabeleceu as duas bases conceituais necessárias ao desenvolvimento deste trabalho: de um lado, os fundamentos de sistemas quânticos, portas e circuitos relevantes à construção do QQ-PRG; de outro, a teoria do caos determinístico e, em particular, o mapa logístico, cuja sensibilidade às condições iniciais e cuja densidade invariante arco-seno o tornam uma fonte adequada de aleatoriedade quando combinada à transformação de Ulam-von Neumann. A integração dessas duas linhas (aleatoriedade de origem quântica e aleatoriedade caótica de origem clássica) constituiu a proposta central deste trabalho.

Os resultados obtidos confirmam, de forma consistente em diferentes níveis de análise, que as melhorias propostas produzem um ganho substancial na qualidade estatística do gerador. Na avaliação direta da sequência pseudoaleatória por meio da bateria de testes NIST SP 800-22 (Seção 5.2), o algoritmo original obteve sucesso em apenas 2 dos 15 testes da bateria, com os testes de *Random Excursions* e sua variante sequer sendo computáveis devido à ausência de ciclos suficientes na sequência, um indicativo claro do alto grau de estrutura presente nas saídas do algoritmo original, decorrente da ausência de entrelaçamento entre os qubits e do uso de ângulos de rotação fixos e repetidos. O algoritmo modificado, por sua vez, obteve sucesso na totalidade dos 15 testes da bateria, incluindo todas as variantes dos testes *NonOverlappingTemplate*, *Random Excursions* e *Random Excursions Variant*, evidenciando que a combinação de entrelaçamento quântico com a randomização caótica dos ângulos elimina de forma eficaz as regularidades estatísticas do gerador original.

Essa melhoria se confirmou, ainda que de forma mais sutil, na aplicação prática de encriptação de imagens (Seção 5.3). A análise de histograma mostrou-se incapaz de distinguir as duas versões do algoritmo: ambas produziram distribuições de intensidade

aproximadamente uniformes nas imagens encriptadas, resultado esperado e desejável, mas que evidencia os limites desse teste como ferramenta de discriminação entre geradores de qualidade distinta. Os testes de ataque por recorte e de adição de ruído, por sua vez, mostraram-se, por construção, independentes do gerador pseudoaleatório utilizado, uma consequência direta da natureza pixel-independente do esquema de encriptação por XOR adotado. Foi na análise de correlação entre pixels adjacentes (Seção 5.3.3.4) que a diferença entre os dois algoritmos se manifestou de maneira clara e consistente: o algoritmo original apresentou um resíduo de correlação horizontal sistemático, entre 0.132 e 0.224, presente em todas as quatro imagens testadas tanto no canal vermelho quanto na escala de cinza, ao passo que o algoritmo modificado eliminou essa assimetria, restringindo todos os coeficientes ao intervalo $[-0.005, 0.005]$, sem favorecer nenhuma direção espacial específica. Esse resultado conecta diretamente a estrutura identificada pelos testes NIST à sua consequência prática em um esquema de encriptação real, e constitui a principal evidência empírica deste trabalho a favor das melhorias propostas.

Como principal contribuição, este trabalho demonstra que limitações de geradores pseudoaleatórios quânticos de baixa complexidade, como a ausência de correlações entre qubits e a repetição de parâmetros de preparação, podem ser mitigadas com baixo custo computacional adicional por meio de mecanismos relativamente simples: uma única camada de portas C-NOT e uma fonte clássica de números caóticos, de implementação trivial e custo computacional desprezível frente à simulação do circuito quântico. Isso sugere que, em aplicações práticas de criptografia caótico-quântica, parte considerável do ganho estatístico pode ser obtida por meio de mecanismos clássicos relativamente simples, sem a necessidade de aumentar a complexidade do circuito quântico em si.

Uma limitação deste trabalho deve ser destacada: a avaliação foi conduzida inteiramente em simulação (`statevector_simulator` da *Qiskit Aer*), não havendo, no escopo deste trabalho, validação em hardware quântico real, onde efeitos de ruído e descoerência poderiam alterar os resultados obtidos. Ademais, os testes de robustez a recorte e a ruído, conforme discutido, não permitiram diferenciar os dois algoritmos em função da natureza pixel-independente do esquema de encriptação por XOR utilizado; métricas de difusão interespaical, como o NPCR (*Number of Pixels Change Rate*) e o UACI (*Unified Average Changing Intensity*), não foram exploradas neste trabalho e poderiam revelar diferenças adicionais entre as duas versões do gerador sob esse aspecto.

Como sugestões para trabalhos futuros, destacam-se: a validação dos resultados em hardware quântico real, de modo a avaliar o impacto de ruído físico sobre as propriedades estatísticas aqui reportadas em simulação; a exploração de topologias alternativas de entrelaçamento (por exemplo, em anel ou totalmente conectada) e sua comparação com a topologia em cadeia adotada neste trabalho; a investigação de outras fontes de aleatoriedade caótica além do mapa logístico, como mapas multidimensionais ou sistemas

de Lorenz discretizados, e sua combinação com circuitos quânticos de maior número de qubits; a aplicação de métricas adicionais de avaliação criptográfica, como NPCR, UACI e análise de sensibilidade à chave (*key sensitivity analysis*).

Em síntese, os resultados obtidos neste trabalho indicam que a combinação de circuitos quânticos com fontes complementares de aleatoriedade caótica constitui uma direção promissora para a geração de números pseudoaleatórios de qualidade estatística superior, com aplicações diretas e mensuráveis em esquemas de encriptação de imagens digitais.

Referências

- ALVES, W. M. S.; FELIPE, J. C. C. Quantum Algorithms using the Qiskit: An Approach to teach Quantum Computation and Information. **Revista Brasileira de Ensino de Física**, 2022. Citado na página 43.
- ANDRECUT, M. Logistic map as a random number generator. **International Journal of Modern Physics B**, v. 12, 01 2012. Citado 3 vezes nas páginas 10, 11 e 23.
- BARCELOS, C.; ANDRADE, E.; BOAVENTURA, M. Notas em matemática aplicada. Citado 2 vezes nas páginas 4 e 15.
- CRUZ, L. J. M. S. **Multigraphs: Emergence from Hilbert Space Subdivision in Superposed Quantum Systems and their Image Encoding Application**. Dissertação (Mestrado) — Universidade Federal de Uberlândia, 2024. Citado 17 vezes nas páginas 4, 10, 11, 12, 13, 14, 16, 19, 20, 34, 35, 37, 38, 39, 43, 62 e 74.
- CRUZ, L. J. M. S.; FAINA, L. F.; PEREIRA, J. H. de S. Quantum-based PRNG for Image Encryption: a Comprehensive Study. **Taylor & Francis**, 2024. Citado 2 vezes nas páginas 31 e 33.
- DEVANEY, R. **An Introduction to Chaotic Dynamical Systems**. [S.l.: s.n.], 2021. ISBN 9780429280801. Citado na página 21.
- 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**, 2019. Citado na página 33.
- FEIGENBAUM, M. Quantitative universality for a class of nonlinear transformations. **J. Stat. Phys.**, v. 19, p. 25–52, 07 1978. Citado na página 22.
- KUANG, R.; LOU, D.; HE, A.; MCKENZIE, C.; REDDING, M. Pseudo quantum random number generator with quantum permutation pad. **IEEE**, 2021. Citado na página 11.
- LEVY, A.; LOSTAGLIO, M. Quasiprobability distribution for heat fluctuations in the quantum regime. **PRX Quantum**, American Physical Society (APS), v. 1, n. 1, set. 2020. ISSN 2691-3399. Disponível em: <<http://dx.doi.org/10.1103/PRXQuantum.1.010309>>. Citado na página 19.
- MANOUCHEHRI, K.; WANG, J. **Physical Implementation of Quantum Walks**. [S.l.]: Quantum Science and Technology, 2013. Citado na página 37.
- MAY, R. Simple mathematical models with very complicated dynamics. **Nature**, v. 26, p. 457, 07 1976. Citado na página 22.
- NIELSEN, M. A.; CHUANG, I. L. **Quantum Computation and Quantum Information**. [S.l.]: Cambridge University Press, 2001. Citado na página 13.

NIROULA, P.; NAM, Y. A quantum algorithm for string matching. **Nature Publishing Group**, 2021. Citado na página 10.

PATIDAR, V.; SUD, K.; PAREEK, N. A pseudo random bit generator based on chaotic logistic map and its statistical testing. **Informatica**, v. 33, p. 441–452, 01 2009. Citado na página 24.

PATRO, S. G. K.; SAHU, K. K. **Normalization: A Preprocessing Stage**. 2015. Disponível em: <<https://arxiv.org/abs/1503.06462>>. Citado na página 42.

RUKHIN, A.; SOTO, J.; NECHVATAL, J.; SMID, M.; BARKER, E.; LEIGH, S.; LEVENSON, M.; VANGEL, M.; BANKS, D.; HECKERT, N.; DRAY, J.; VO, S.; BASSHAM, L. A statistical test suite for random and pseudorandom number generators for cryptographic applications. 4 2010. Disponível em: <<https://csrc.nist.gov/pubs/sp/800/22/r1/upd1/final>>. Citado 7 vezes nas páginas 24, 25, 26, 27, 28, 29 e 30.

WANG, J.; GENG, Y.-C.; HAN, L.; LIU, J.-Q. Quantum image encryption algorithm based on quantum key image. **International Journal of Theoretical Physics**, 2018. Citado 2 vezes nas páginas 31 e 33.

Wang, X.-Y.; Yang, L. Design of Pseudo-Random Bit Generator Based on Chaotic Maps. **International Journal of Modern Physics B**, v. 26, n. 32, p. 1250208, dez. 2012. Citado na página 24.

WIGNER, E. On the quantum correction for thermodynamic equilibrium. **Phys. Rev.**, American Physical Society, v. 40, p. 749–759, Jun 1932. Disponível em: <<https://link.aps.org/doi/10.1103/PhysRev.40.749>>. Citado na página 19.

YANOFSKY, N. S.; MANNUCCI, M. A. **Quantum Computing for Computer Scientists**. [S.l.]: Cambridge University Press, 2008. Citado na página 13.

ZIDAN, M.; HEGAZY, S. F.; ABDEL-ATY, M.; OBAYYA, S. S. Rapid solution of logical equivalence problems by quantum computation algorithm. **Applied Soft Computing**, 2023. Citado na página 10.