ANÁLISE COMPARATIVA ENTRE CRIVO QUADRÁTICO E ALGORITMO QUÂNTICO DE SHOR APLICADO À CRIPTOGRAFIA

COMPARATIVE ANALYSIS BETWEEN QUADRATIC SIEVE AND SHOR'S QUANTUM ALGORITHM APPLIED TO CRYPTOGRAPHY

Erick Galvão da Silva, Faculdade de Tecnologia de Americana, erick.silva82@fatec.sp.gov.br
Leonardo Rodrigues Ribeiro, Faculdade de Tecnologia de Americana, leonardo.ribeiro16@fatec.sp.gov.br
Stheffani Gonçalves Rocha Emboava, Faculdade de Tecnologia de Americana, stheffani.emboava@fatec.sp.gov.br
Mariana Godoy Vazquez Miano, Faculdade de Tecnologia de Americana, mariana.miano@fatec.sp.gov.br

Resumo

Este artigo apresenta uma análise técnica focada na iminente transição da computação clássica para a computação quântica e seus desafios. Ele explora o ambiente quântico da Microsoft e as linguagens Q# (quântica) e Python (clássica), conduzindo uma comparação de eficiência entre a computação clássica e a quântica em uma parte essencial do processo de decifragem de dados: a fatoração de números compostos com dezenas de dígitos. O estudo envolve uma análise comparativa dos resultados ao utilizar o Algoritmo Quântico de Shor e o Algoritmo Clássico do Crivo Quadrático, destacando o profundo impacto da computação quântica na segurança de sistemas criptográficos frequentemente utilizados em bancos de dados e aplicações atuais. O estudo contribui significativamente para a busca de soluções que assegurem uma transição segura e de baixa complexidade para as tecnologias quânticas, fornecendo *insights* sobre os desafios e vantagens da utilização de algoritmos quânticos, com forte foco na segurança da informação.

Palavras-chave: Computação Quântica, Criptografia, Segurança da Informação.

Abstract

This article presents a technical analysis focused on the imminent transition from classical computing to quantum computing and its challenges. It explores the quantum environment of Microsoft and the programming languages Q# (a Microsoft quantum language) and Python (classical language), conducting an efficiency comparison between classical and quantum computing in an essential part of the data decryption process: the factorization of composite numbers with a large number of digits. The study entails a comparative analysis of the results when employing the Quantum Shor Algorithm and the Classical Quadratic Sieve Algorithm, highlighting the profound impact of quantum computing on the security of cryptographic systems, frequently used for current applications and databases. This contributes significantly to the quest for solutions that ensure a secure and low-complexity transition to quantum technologies. Consequently, this study provides vital insights about the challenges and advantages of utilizing quantum algorithms, with a strong emphasis on information security.

Keywords*: Quantum Computing, Cryptography, Information Security.*

1. Introdução

A chegada da computação quântica traz promessas e incertezas, especialmente em relação ao seu impacto na segurança da informação. À medida que nos aproximamos dessa mudança transformadora, torna-se fundamental compreender as complexidades e implicações que essa transição envolve. No presente artigo, é explorado o processo de fatoração, fundamental para a garantia da segurança de algoritmos criptográficos usados em larga escala atualmente.

A ascenção da computação quântica representa uma grande ameaça para a Criptografia moderna, tornando necessária a busca por novas soluções, que suportem possíveis ataques quânticos. Para demonstrar tal fator, utilizou-se o ambiente quântico desenvolvido pela Microsoft e a linguagem quântica Q#, juntamente com a linguagem clássica multiparadigma Python, para demonstrar os efeitos da computação quântica sobre a clássica, destacando seu impacto na segurança proporcionada pelos algoritmos criptográficos comumente utilizados em aplicações e bancos de dados, além de contribuir com a busca por plataformas e linguagens que possibilitem uma migração segura e de baixa complexidade.

Os códigos utilizados para obtenção dos resultados a serem apresentados contaram com a implementação de bibliotecas e métodos de criptografia utilizados em larga escala, destacando especialmente o RSA (Rivest-Shamir-Adleman), desenvolvendo assim uma análise comparativa dos resultados de decifragem dos dados por meio do Algoritmo Clássico do Crivo Quadrático, além do Algoritmo Quântico de Shor.

2. Referencial Teórico

O referencial teórico do artigo aborda uma ampla gama de tópicos, começando com uma exploração da computação quântica e o impacto do Algoritmo de Shor na quebra de sistemas de criptografia. Também analisa os conceitos de criptografia simétrica e assimétrica, incluindo os algoritmos Blowfish e RSA. A implementação é considerada através das linguagens Python e Q# (Q Sharp), enquanto o Microsoft Azure Quantum é apresentado como uma plataforma de computação quântica em nuvem. O papel da criptografia na segurança de dados e sua relação com bancos de dados também são discutidos. Em conjunto, o referencial oferece uma base sólida para entender a interseção

entre computação quântica, criptografia, linguagens de programação e aplicações práticas.

2.1. Computação Quântica

A Computação Quântica é uma tecnologia emergente da Ciência da Computação, que utiliza de conceitos e fundamentos da Física, Matemática e Mecânica Quântica, sendo esses o tripé da Computação Quântica, e se destaca por sua capacidade de processamento de dados exponencialmente maior quando comparado à computação clássica (PRESKILL, 2018, tradução nossa). Enquanto os computadores clássicos operam com bits clássicos, de 0 e 1, os bits quânticos, ou qubits, operam com a superposição de estados, onde o bit pode estar em zero, um, ou simultaneamente entre os dois estados. Os qubits também podem estar entrelaçados, e através disso, e possível realizar as correlações quânticas, que não são possíveis em sistemas clássicos. (LACAVA & MIANO, 2018).

Existem diversas vantagens potenciais no uso da Computação Quântica em relação à Computação Clássica, como a velocidade e eficiência na resolução de determinados problemas, por exemplo a fatoração de números grandes, em que na computação clássica esse resultado pode levar anos para serem obtidos (e por conta disso se tornou a base da criptografia clássica), já com a criptografia quântica, essa fatoração leva segundos, minutos ou dias, dependendo do tamanho do número e sua complexidade (NIELSEN & CHUANG, 2010, tradução nossa). Outra vantagem, seria a simulação de sistemas quânticos complexos e a otimização de problemas em grande escala, como o processamento de dados através do uso de tecnologias como algoritmos de Machine Learning para o processamento de uma massa de dados extensa, que demanda de velocidade e grandes recursos. (MIANO, 2020).

2.1.1. Algoritmo de Shor

O algoritmo de Shor é um algoritmo quântico desenvolvido por Peter Shor em 1994. É conhecido por sua capacidade de fatorar números inteiros grandes de forma significativamente mais rápida do que qualquer algoritmo clássico conhecido.

A fatoração de números inteiros com centenas de algarismos em seus fatores primos é um problema desafiador para a computação clássica, especialmente quando os números envolvidos têm centenas ou milhares de dígitos (MIANO & OLIVEIRA, 2021). A segurança de muitos sistemas criptográficos modernos, como o RSA, depende da dificuldade de

fatoração.

O algoritmo de Shor utiliza princípios da computação quântica, aproveitando a capacidade dos qubits de existirem em estados de superposição e entrelaçamento. Em um computador quântico com capacidade suficiente, o algoritmo de Shor pode ser usado para fatorar um número inteiro N em seus fatores primos em tempo polinomial, o que representa um avanço significativo em relação aos métodos clássicos exponenciais. É um algoritmo quântico revolucionário que pode fatorar números inteiros grandes de forma eficiente. Sua descoberta levantou questões sobre a segurança dos sistemas criptográficos atuais e impulsionou a pesquisa em criptografia pós-quântica (MIANO & OLIVEIRA, 2021).

Segundo MIANO (2020), o objetivo do algoritmo é achar o período de uma função, e na sequência, encontrar os fatores do valor solicitado.

O algoritmo de Shor tem uma aplicação fundamentalmente importante na área de criptografia, especificamente na quebra de criptografia baseada no problema de fatoração de números inteiros grandes. O RSA (Rivest-Shamir-Adleman), um dos algoritmos de criptografia mais utilizados, é vulnerável à fatoração de números primos grandes, e o algoritmo de Shor pode ser utilizado para realizar essa fatoração de forma muito mais eficiente em um computador quântico (CHAVES, 2018).

A implicação direta disso é que, se um computador quântico com poder de processamento adequado for construído, ele poderá fatorar rapidamente os números envolvidos no RSA, comprometendo a segurança de comunicações e transações protegidas por esse algoritmo.

2.2. Criptografia

A criptografia é uma ciência que se dedica a proteger a comunicação e os dados em face de adversários. Com base em algoritmos matemáticos, ela possui diversas aplicações práticas, desde transações comerciais online até segurança militar. Algoritmos clássicos, como DES, AES, RSA e ECC, são amplamente utilizados para garantir a confidencialidade e a integridade das informações. A criptografia desperta grande interesse entre pesquisadores, agências governamentais, defensores dos direitos civis e usuários comuns que valorizam sua privacidade e buscam a proteção de seus dados (ALENEZI *et al.*, 2020, tradução nossa).

A criptografia moderna é um campo em constante evolução, adaptando-se aos avanços tecnológicos e às novas ameaças digitais. Seu papel é crucial na proteção de informações sensíveis e na manutenção da segurança dos sistemas digitais. Com a crescente quantidade de dados pessoais circulando pela internet, a necessidade de técnicas criptográficas eficazes e confiáveis é cada vez mais evidente. A pesquisa e o desenvolvimento contínuos nesse campo são essenciais para garantir a privacidade e a confiança nas comunicações e transações online. Desempenha um papel fundamental na proteção da privacidade e segurança das informações em um mundo digital. É uma disciplina em constante avanço, impulsionada pelas demandas da sociedade moderna e pela necessidade de proteger dados sensíveis de adversários mal-intencionados.

Através de algoritmos criptográficos robustos, a criptografia oferece meios eficazes de proteger a confidencialidade, a integridade e a autenticidade dos dados. A compreensão e o avanço contínuo nesse campo são essenciais para enfrentar assegurando a integridade e confidencialidade dos sistemas e das comunicações digitais.

2.2.1. Criptografia Assimétrica x Criptografia Simétrica

A criptografia pode ser separada em dois tipos, simétrica e assimétrica. Na criptografia simétrica, a mesma chave é usada tanto para criptografar quanto para descriptografar os dados. A chave é compartilhada entre as partes autorizadas de antemão. A criptografia simétrica é rápida e eficiente, adequada para grandes volumes de dados. Exemplos de algoritmos simétricos são o AES (Advanced Encryption Standard) e o DES (Data Encryption Standard) (CHAVES, 2018).

Na criptografia assimétrica ou criptografia de chave pública, são usadas duas chaves diferentes: uma chave pública e uma chave privada. A chave pública é usada para criptografar os dados, enquanto a chave privada correspondente é usada para descriptografá-los. As chaves são geradas em pares e são matematicamente relacionadas. A criptografia assimétrica é mais segura em termos de compartilhamento de chaves, mas é computacionalmente mais intensiva do que a criptografia simétrica (CHAVES, 2018). O RSA e o ECC (Elliptic Curve Cryptography) são exemplos de algoritmos assimétricos amplamente utilizados (ESWARAN *et al.*, 2022, tradução nossa).

2.2.2. RSA (Rivest-Shamir-Adleman) – criptografia assimétrica

A criptografia RSA, como citado anteriormente, é um modelo baseado em chaves públicas que, conforme observação de A. Budiman, é composto por três principais estágios: geração de chave, criptografia e descriptografia. Esse tipo de criptografia é baseado na fatoração de números primos extensos, e desde sua criação na década de 70, por parte dos estudiosos Ron Rivest, Adi Shamir e Leonard Adleman, mostrou-se um método de segurança eficiente.

Este algoritmo é amplamente utilizado em aplicações práticas, como a troca segura de chaves para estabelecer comunicações seguras, a criação de assinaturas digitais para verificar a autenticidade e integridade de mensagens, bem como em sistemas de autenticação de dois fatores, como tokens de segurança e certificados digitais. Sua segurança é baseada na dificuldade de fatorar números inteiros grandes, o que torna o algoritmo eficaz na proteção de informações sensíveis em bancos de dados e em comunicações online, quando consideramos apenas vulnerabilidades clássicas (CHAVES, 2018).

O algoritmo pode ser dividido em 4 etapas: geração de chaves, criptografia, transmissão da mensagem e descriptografia. Não há dúvidas de que a etapa elementar do algoritmo é a criação das chaves pública, pois sem os resultados deste processo não é possível realizar nenhuma outra tarefa criptográfica, por isso é necessário compreender os cálculos envolvidos nessa parte do algoritmo.

O primeiro passo na criação de chaves RSA é escolher dois números primos distintos, normalmente números com cerca de 1024 bits, denotados como "p" e "q", e multiplicá-los para gerar um valor "N", portanto temos que:

$$N = p \times q$$

Uma vez que possuímos o valor de N, é necessário calcular a função totiente de Euler (φ), que é o número de inteiros positivos menores que "N", e co-primos a "N". Podemos escrever essa etapa da seguinte forma:

$$\varphi(N) = (p - 1) \times (q - 1)$$

Agora, um expoente público "e" é escolhido. Este valor deve ser um número inteiro positivo menor que φ(N) e co-primos a φ(N). Geralmente, o valor "e" é escolhido como um

número primo para simplificar os cálculos, para que seja possível calcular o expoente privado "d", que é o inverso multiplicativo de "e" módulo φ(n), aplicando a seguinte fórmula:

$$d \equiv e^{-1} (mod\ \varphi(n))$$

Os valores de "e" e "d" geram então a chave pública, que consiste em (N, e), e a chave privada, que consiste em (N, e). A chave pública é compartilhada amplamente e usada para criptografar mensagens, enquanto a chave privada é mantida em segredo e usada para descriptografar mensagens. A criação cuidadosa dessas chaves, baseada nas propriedades matemáticas dos números primos e no relacionamento entre "e" e "d," é o cerne da segurança do algoritmo RSA.

Levando em conta as informações apresentadas, temos que o RSA é mais seguro e flexível, pois permite o uso de assinaturas digitais e a troca de chaves sem a necessidade de um canal seguro, sendo, portanto, o algoritmo priorizado pelo fator de segurança, mostrando-se mais inerente ao projeto desenvolvido.

2.2.3. Blowfish

Criado por Bruce Schneier em 1993, o Blowfish é um algoritmo de criptografia de bloco assimétrico que tem sido amplamente utilizado em várias aplicações e sistemas. Uma das principais características do Blowfish é a sua operação em blocos fixos de dados de 64 bits, utilizando uma chave de criptografia variável de 32 a 448 bits. Essa flexibilidade no tamanho da chave permite adaptar a segurança do algoritmo de acordo com as necessidades específicas. Além disso, o Blowfish utiliza uma estrutura de rondas de transformações matemáticas para processar os dados de entrada. Ele é conhecido por seu desempenho rápido e eficiente, sendo otimizado para diferentes plataformas, incluindo implementações em hardware dedicado. Sua velocidade é um fator importante em muitos sistemas que exigem criptografia eficiente em tempo real (ALENZI *et al.*, 2020, p. 256, tradução nossa).

O Blowfish é uma opção viável para aplicações que exigem uma camada adicional de segurança, como a proteção de dados em trânsito ou a geração de assinaturas digitais. No entanto, antes de implementar o algoritmo, é importante considerar cuidadosamente o contexto de uso, as necessidades de segurança e as melhores práticas atuais em relação à criptografia. Neste caso, o Blowfish é útil quando tratamos de uma quantidade massiva de

dados, pois é capaz de criptografá-los rapidamente, contudo, quando priorizamos o fator de segurança, é um método inferior ao RSA.

2.2.4. Algoritmo do Crivo Quadrático

Os crivos em geral, são técnicas utilizadas para estimar números primos dentro de um conjunto numérico, desempenhando um papel fundamental na teoria dos números, com aplicações na matemática e na ciência da computação. O Crivo Quadrático é um método de fatoração de inteiros com pouco mais de 20 anos, considerado um dos mais eficientes métodos de fatoração de números inteiros, tendo como principal objetivo a fatoração de inteiros grandes (TERENCIANI, 2015). Pelo fato de demonstrar-se superior a outros métodos de fatoração, o Crivo tornou-se relevante para a área da criptografia.

O crivo quadrático funciona da seguinte forma: dado um número composto n, escolhe-se um inteiro b tal que b^2 seja próximo de n. Em seguida, procura-se por números x e y tais que $x^2 - y^2$ seja divisível por n, ou seja:

$$x^2 \equiv y^2 \pmod{n}$$

Isso implica que (x + y) (x - y) seja divisível por n, e que o máximo divisor comum de x + y e n, ou de x - y e n, seja um fator não trivial de n. Para encontrar esses números x e y, o crivo quadrático usa uma função quadrática $q(x) = (x + b)^2 - n$ e testa se os valores de q(x) são divisíveis por primos pequenos. Se forem, eles são chamados de números lisos. O objetivo é encontrar um conjunto de números lisos cujo produto seja um quadrado perfeito, pois isso garante a existência de x e y com a propriedade desejada.

Conforme dito anteriormente, esse algoritmo é mais rápido do que qualquer outro algoritmo clássico quando tratamos da fatoração de grandes números inteiros, e por esse motivo é utilizado em métodos de criptografia, tanto para encontrar números primos quanto para calcular inversos multiplicativos, duas operações essenciais para garantir a segurança das comunicações baseadas em métodos clássicos de criptografia.

2.3. Linguagens de Programação

Considerando os objetivos do trabalho e a área de conhecimento enfatizada nesse artigo (Tecnologia da Informação), mostrou-se imprescindível a construção de um programa, demonstrando na prática os resultados almejados.

Foi dedicada atenção especial às linguagens de programação empregadas nesse contexto, abrangendo tanto a linguagem clássica, como o Python e a linguagem quântica Q#, e identificou-se a oportunidade de conceber o algoritmo de Shor que, conforme o item 2.2.1 deste mesmo artigo, pode ser usado para fatorar um número inteiro N em seus fatores primos em tempo polinomial. Essa análise comparativa permitiu avaliar as diferenças de tempo e eficiência entre essas duas abordagens.

Era essencial para o desenvolvimento do código que ele possuísse uma linguagem de programação quântica e outra clássica, e isso era claro desde o início de sua programação. Python mostrou-se uma linguagem muito versátil, de modo que possibilitou a interação das diferentes partes do código. Uma das vantagens de usar o Q# para programar algoritmos quânticos é que ele pode se comunicar facilmente com outras linguagens populares, como o Python. O Python é uma linguagem de alto nível, amplamente usada para ciência de dados, aprendizado de máquina e desenvolvimento web. O Q# é uma linguagem específica de domínio, projetada para expressar operações quânticas de forma clara e concisa. A integração dessas duas linguagens é um diferencial que facilita a criação de aplicações híbridas, que combinam o poder do processamento quântico com a flexibilidade do Python.

2.3.1. Python

Python é uma linguagem de programação de alto nível, interpretada e orientada a objetos, conhecida por sua sintaxe clara e legível. Sua semântica dinâmica permite que os desenvolvedores escrevam código de maneira rápida e eficiente, sem a necessidade de declarações explícitas de tipos de variáveis. A digitação dinâmica do Python facilita a prototipagem e o desenvolvimento ágil de aplicativos, tornando-o uma escolha popular para uma ampla gama de projetos (PYTHON, 2023, tradução nossa).

Uma das características distintivas do Python é a presença de estruturas de dados incorporadas, como listas, dicionários e conjuntos, que facilitam a manipulação e gerenciamento de informações complexas. Além disso, a ligação dinâmica do Python permite que as chamadas de função e os métodos sejam resolvidos em tempo de execução, proporcionando flexibilidade aos desenvolvedores (PYTHON, 2023, tradução nossa).

A popularidade do Python também se deve à sua vasta coleção de bibliotecas e frameworks, que abrangem desde desenvolvimento web até aprendizado de máquina e

processamento de dados. Essas bibliotecas fornecem funcionalidades adicionais e permitem aos desenvolvedores aproveitarem soluções pré-existentes, acelerando o tempo de desenvolvimento e aumentando a eficiência. Com sua comunidade ativa e dedicada, o Python continua a evoluir e se adaptar às demandas da indústria de software, consolidando sua posição como uma linguagem poderosa e versátil (PYTHON, 2023, tradução nossa).

2.3.2. Q# (Q Sharp)

Q# é uma linguagem de programação desenvolvida pela Microsoft, projetada especialmente para computação quântica. Foi lançada ao público como parte do Kit de Desenvolvimento Quântico (QDK). É uma linguagem de alto nível, independente de hardware e otimizada para lidar com algoritmos e operações quânticas, além do desenvolvimento e simulação de programas quânticos. Q# também introduz novas estruturas e operações específicas para quântica, como repetir até sucesso e estimativa de fase adaptativa, que permitem a integração de computações quânticas e clássicas. (MICROSOFT, 2023c, tradução nossa).

O Q# é uma linguagem de programação quântica que apresenta várias características essenciais, como o uso de qubits, a contrapartida quântica dos bits clássicos, no qual o Q# é capaz de representar e manipular informações quânticas de maneira eficaz. Além disso, o Q# permite que os programadores definam operações quânticas, que são as instruções fundamentais necessárias para a manipulação desses qubits.

Uma característica notável é a capacidade de criar algoritmos quânticos complexos, incluindo o algoritmo de Shor. Outra vantagem, que favorece no desenvolvimento e depuração de códigos, o Q# oferece a opção de testar programas em simuladores quânticos, fornecendo a possibilidade de testar e depurar códigos em ambientes controlados fornecidos pelo Kit de Desenvolvimento da Microsoft, sem a necessidade de acesso a hardware quântico real.

Por fim, o Q# é complementado por bibliotecas e operações padrão que simplificam a implementação de algoritmos quânticos comuns, o que torna a programação quântica mais acessível e eficiente para aqueles que desejam explorar o mundo da computação quântica.

2.4. Plataformas e Simuladores quânticos

As plataformas e simuladores quânticos são ambientes que contém ferramentas desenvolvidas para simular e estudar sistemas quânticos, através das máquinas clássicas. Elas fornecem a possibilidade de investigar o comportamento de sistemas quânticos, realizar experimentos e testes de algoritmos quânticos virtualmente, oferecendo a possibilidade de trabalhar com tecnologias híbridas, o que auxilia na realização dos estudos e experimentos para desenvolvimento deste trabalho, pois podemos, através delas, simular ambientes quânticos através das máquinas virtuais (MICROSOFT, 2023a).

2.4.1. Microsoft Azure Quantum

O Microsoft Azure Quantum foi a plataforma de simulação escolhida para o desenvolvimento do projeto. Devido à maturidade da Microsoft em serviços baseados em nuvem, seu simulador quântico mostrou-se muito eficiente para o desenvolvimento do projeto. Dentro da plataforma contamos com um conjunto de ferramentas de desenvolvimento QDK (Microsoft Quantum Development Kit) e a linguagem de programação Q# (Q Sharp), nativos da plataforma, oferecendo suporte à simulação, programação e integração com hardware quântico (MICROSOFT, 2023b).

3. Metodologia

Este trabalho descreve uma pesquisa teórica de caráter experimental na área da computação, cujo intuito é demonstrar a vantagem de soluções quânticas sobre códigos clássicos, aproximando esta tecnologia das soluções computacionais vigentes. Assim sendo, foi desenvolvido um código que demonstrasse, de maneira comparativa, a superioridade da computação quântica, explorando as vulnerabilidades resultantes do aumento da capacidade de processamento de um computador.

3.1. Código

O código desenvolvido para este projeto apresenta uma abordagem híbrida, combinando duas linguagens distintas: uma quântica, o Q#, e outra clássica, o Python. Essa combinação permite aproveitar as vantagens da computação quântica para as operações matemáticas envolvidas na criptografia, enquanto a manipulação dos dados em suas demais particularidades é realizada por meio do Python.

No contexto da criptografia, foram empregados dois algoritmos amplamente utilizados: Blowfish e RSA. Esses algoritmos foram implementados em Q# para realizar a criptografia dos dados de forma segura. No entanto, para testar a segurança do sistema, foram empregados dois métodos diferentes de quebra de criptografia. Primeiramente, o algoritmo do Crivo Quadrático, programado em Python, foi utilizado para testar a segurança dos dados. Em seguida, o poderoso algoritmo quântico de Shor, implementado em Q#, foi empregado para realizar o mesmo processo.

Para uma análise completa e comparativa do desempenho entre a computação quântica e a computação clássica, o foco foi direcionado especificamente para o processo de descriptografia dos dados criptografados. Ao analisar os resultados obtidos, foi possível avaliar as vantagens e limitações de cada abordagem, bem como compreender o potencial disruptivo da computação quântica no campo da criptografia.

Essa integração entre as linguagens Q# e Python, aliada ao uso do algoritmo de Shor e dos métodos de descriptografia empregados, proporcionou uma análise abrangente e aprofundada do desempenho da computação quântica, permitindo mensurar sua ameaça à segurança em relação à computação clássica no contexto da criptografia de dados.

3.2. Simulador Quântico

O Microsoft Azure Quantum foi o ambiente escolhido para o processamento quântico, utilizando os créditos fornecidos pela Microsoft para credenciais acadêmicas.

A plataforma permitiu a exploração de todo o potencial da computação quântica, indo além das capacidades dos computadores clássicos. Com acesso à infraestrutura quântica do Azure Quantum, foram implementados algoritmos, executado simulações e até mesmo realizado experimentos em hardware quântico real.

4. Resultados e Discussões

Os resultados e discussões giram em torno da análise comparativa proporcionada pelo código desenvolvido para este artigo. Serão apresentadas a seguir conclusões e discussões teóricas sobre o estudo desenvolvido, e os principais resultados obtidos nesta análise, a partir da fatoração de números extraídos de chaves criptográficas.

4.1. Fatoração com o Crivo Quadrático

A execução do algoritmo do crivo quadrático foi realizada em uma máquina virtual hospedada na Azure, com recursos computacionais medidos para que não afetassem seu desempenho. Essa medida foi necessária pois esperava-se que o código levaria mais tempo para ser executado do que o tempo que os recursos disponíveis fisicamente poderiam continuar ligados.

Inicialmente, é necessário compreender a eficiência do Crivo Quadrático e a maneira que foi aplicado, conforme as figuras 1, 2 e 3:

Figura 1 – Código de fatoração por divisão sucessiva

```python
import math
import time

def fatoracao_em_primos(numero):
    fatores_primos = []
    divisor = 2

    while divisor * divisor <= numero:
        if numero % divisor == 0:
            fatores_primos.append(divisor)
            numero //= divisor
        else:
            divisor += 1

    if numero > 1:
        fatores_primos.append(numero)

    return fatores_primos

numero_para_fatorar = int(input("Digite um número para fatorar em primos: "))

inicio = time.time()
fatores_primos = fatoracao_em_primos(numero_para_fatorar)
fim = time.time()

tempo_decorrido = fim - inicio

print(f"Os fatores primos de {numero_para_fatorar} são: {fatores_primos}")
print(f"Tempo decorrido para fatoração: {tempo_decorrido} segundos")

```

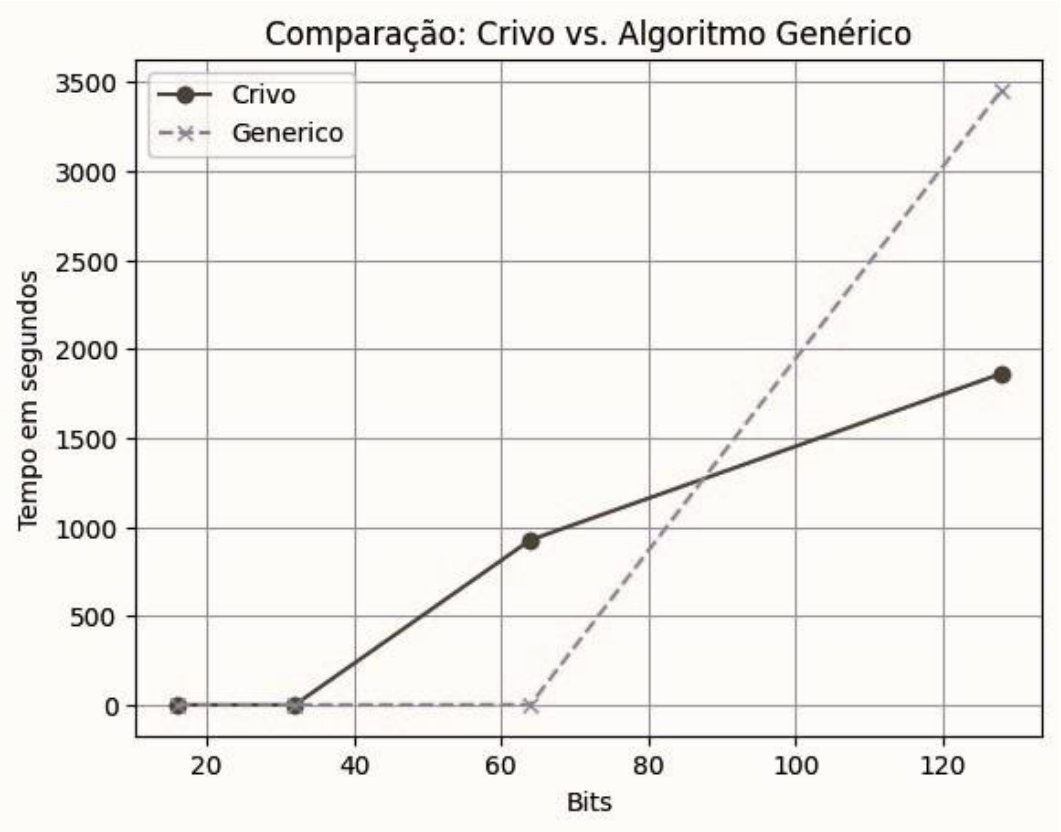
Fonte: Autores

Figura 2 – Código de fatoração usando crivo quadrático

```python
import math
import time
import secrets

num_bytes = 128 // 8
chave_hexadecimal = secrets.token_hex(num_bytes)

print(f"Chave de 128 bits (hexadecimal): {chave_hexadecimal}")

chave_numero = int(chave_hexadecimal, 16)

print(f"Chave de 128 bits (decimal): {chave_numero}")

def calcular_mdc(a, b):
    while b:
        a, b = b, a % b
    return a

def fatoracao_quadratica(numero):
    if numero < 2:
        return [numero]

    fatores_primos = []
    while numero % 2 == 0:
        fatores_primos.append(2)
        numero //= 2

    if numero == 1:
        return fatores_primos

    for a in range(3, int(math.sqrt(numero)) + 1, 2):
        while numero % a == 0:
            fatores_primos.append(a)
            numero //= a

    if numero > 1:
        fatores_primos.append(numero)

    return fatores_primos
print("Fatorando!")
numero_para_fatorar = chave_numero

inicio = time.time()
fatores_primos = fatoracao_quadratica(numero_para_fatorar)
fim = time.time()

tempo_decorrido = fim - inicio

print(f"Os fatores primos de {numero_para_fatorar} são: {fatores_primos}")
print(f"Tempo decorrido: {tempo_decorrido} segundos")

```

Fonte: Autores

Figura 3 - Gráfico de comparação do tempo de fatoração usando Crivo Quadrático e um algoritmo de fatoração genérico



Fonte: Autores

Como é possível observar, o tempo de execução passa a ser menor à medida que a a quantidade de bits do número a ser fatorado aumenta, e isso acontece porque o algoritmo conta com muitos cálculos e operações modulares, que podem ser facilmente evitados com métodos mais simples. Na tabela 1, é possível verificar com mais detalhes os resultados da execução do algoritmo:

Tabela 1 – Tempo de execução da fatoração de uma chave usando Crivo Quadrático

Fatoração de chaves usando algoritmo de Crivo Quadrático				
Bits	16	32	64	128
Tempo	0.001001	0.0029964	926.05079	1320.0054

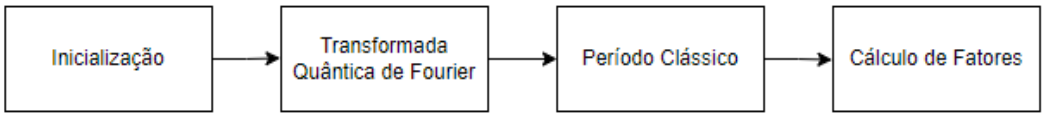
Fonte: Autores

Com isso, constata-se a eficiência do Crivo, mas quando se trata de fatoração de chaves e quebra de criptografia, ele não é suficiente para ameaçar um sistema criptográfico como o RSA, que utiliza chaves de 2048 bits, embora seja muito útil para funções de criptografia mais simples e específicas que envolvam chaves um pouco menores.

4.2. Fatoração com o Algoritmo de Shor

O Algoritmo de Shor possui uma sequência lógica de etapas para realizar os cálculos, o processo do algoritmo de Shor para fatorar N e encontrar seus fatores primos p e q acontece da seguinte maneira, conforme a figura 4:

Figura 4 - Etapas do algoritmo de Shor



Fonte: Autores

Dentre essas etapas, podemos dizer que o essencial está contido na Transformada Quântica de Fourier, que faz uso de um circuito quântico para executar a Transformada Quântica de Fourier em um registrador quântico que representa os números x que são potenciais candidatos a p e q. Por padrão, foram utilizadas chaves de 2048 bits, buscando um experimento ainda mais próximo da realidade dos algoritmos criptográficos. Observe os resultados da implementação do algoritmo nas figuras 5, 6 e 7, e na tabela 2:

Figura 5 - Geração do número composto e fatoração através do algoritmo de Shor

```python
import qsharp
import time
import math
from sympy import randprime

a = 5

n = randprime(1000000000, 10000000000)
print(f"Número primo gerado: {n}")
print("-" * 40)

periodCandidate = period_candidate
print("Periodo candidato encontrado: ",period_candidate)

start_time = time.time()

if periodCandidate % 2 == 0 and pow(a, periodCandidate // 2, n) != n - 1:
    factor1 = math.gcd(pow(a, periodCandidate // 2, n) - 1, n)
    factor2 = n // factor1
    print(f"Fatores encontrados: {factor1}, {factor2}")
else:
    print("Período não válido para fatorização.")

end_time = time.time()
execution_time = end_time - start_time

print(f"Tempo de execução: {execution_time:.6f} segundos")
print("-" * 40)
```

Fonte: Autores

Figura 6 - Resultados da execução do algoritmo de Shor

```
----------------------------------------
Numero a ser fatorado:  9621358634681131337
Periodo candidato encontrado: 4
O valor de A, é:  8738736391927395474
Fatores encontrados: 1, 9621358634681131337
Tempo de execução: 0.000176 segundos
----------------------------------------
```

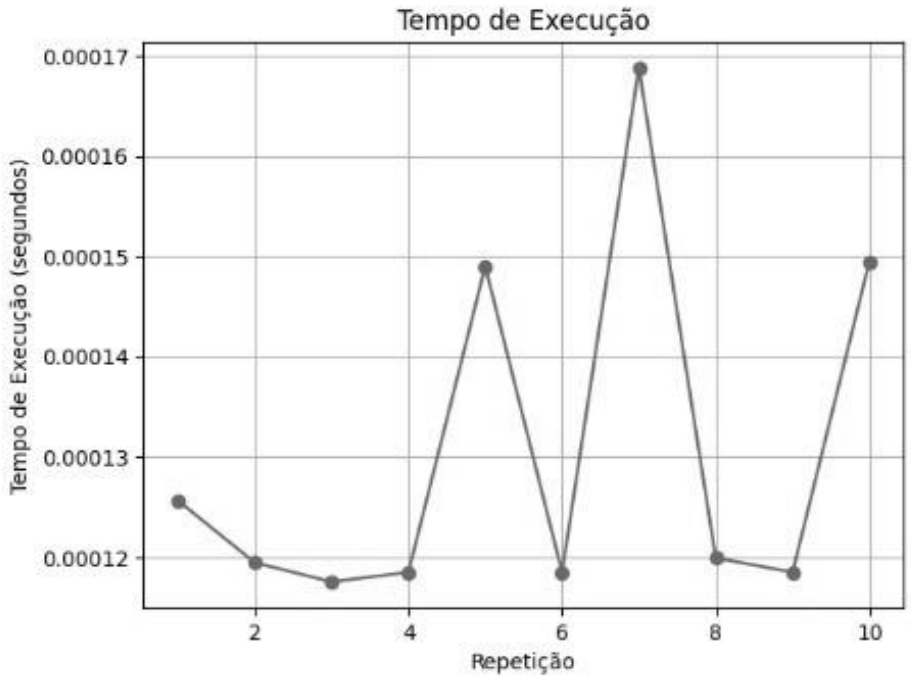
Fonte: Autores

Tabela 2 – Tempo de execução da fatoração de uma chave de 2048 bits usando Shor

Chave de 2048 bits fatorada					
Repetições	2	4	6	8	10
Tempo	0. 0.000131	0.000120	0.000148	0.000119	0.000186

Fonte: Autores

Figura 7 - Tempo de execução da fatoração de uma chave de 2048 bits usando Shor



Fonte: Autores

É essencial notar que, em comparação ao Crivo Quadrático, algoritmo clássico de fatoração, o Algoritmo de Shor mostrou-se extremamente veloz na busca pelos fatores primos de N, evidenciando a eficiência notavelmente superior da utilização de um recurso quântico sobre operações e processos matemáticos e computacionais clássicos.

5. Considerações Finais

Tendo em vista a pesquisa apresentada ao longo deste artigo, conclui-se que a computação quântica apresenta potencial superior em termos de eficiência e processamento em comparação à computação clássica, o que pôde ser apresentado de maneira precisa por

meio da utilização do algoritmo de Shor na fatoração de inteiros.

Desse modo, o Algoritmo mostra-se promissor não somente para um ataque, mas também para a mitigação de incidentes de segurança, pois pode ser utilizado para desbloquear arquivos criptografados em ataques de ransomware, por exemplo. Nesse sentido, espera-se que esse avanço tecnológico favoreça, assim como qualquer ferramenta comum à sociedade, aqueles que agem dentro e fora dos limites éticos.

Contudo, é importante ressaltar que a superioridade da computação quântica em termos de eficiência e processamento se aplica especificamente a determinadas tarefas, como a descriptografia de dados, pois possui em sua base propriedades matemáticas complexas. Para outras aplicações e desafios computacionais, a computação clássica ainda possui sua relevância e utilidade. Portanto, é fundamental considerar cuidadosamente o contexto e a natureza dos problemas a serem abordados ao avaliar a eficácia das abordagens computacionais, a fim de obter os resultados mais próximos dos esperados.

Em suma, com base nas análises realizadas, constatou-se que a computação quântica apresenta uma vantagem notável em relação à eficiência e ao processamento de dados, destacando-se na fatoração de inteiros, e desse modo impactando a segurança de sistemas criptográficos como o RSA, que tem como garantia de segurança a dificuldade de fatoração de grandes números. Quanto ao Crivo Quadrático, é importante ressaltar que sua eficiência ainda é superior a outros algoritmos de fatoração clássicos por possuir menos operações aritméticas, além de uma complexidade subexponencial, mas por estar limitado aos recursos matemáticos e computacionais clássicos, seus resultados ainda ficam muito distantes do algoritmo de Shor.

Essa descoberta ressalta a importância contínua da pesquisa e do desenvolvimento nesse campo promissor, com o intuito de explorar todo o potencial da computação quântica para aplicações futuras, pois apesar de oferecer vantagens significativas em relação à computação clássica, como maior velocidade, menor consumo de energia e maior segurança, ainda existem muitos desafios técnicos e científicos a serem superados para tornar essa tecnologia acessível e confiável. Por isso, é fundamental investir em projetos de inovação e colaboração que possam contribuir para o avanço do conhecimento e da prática nessa área estratégica.

Referências

ALENEZI, Mohammed N.; ALABDULRAZZAQ, Haneen; MOHAMMAD, Nada Q. Symmetric encryption algorithms: Review and evaluation study. International Journal of Communication Networks and Information Security, v. 12, n. 2, p. 256-272, 2020.

CHAVES, R. O. G. Táticas de ataque e protocolos quânticos de distribuição de chaves criptográficas utilizando estratégia de discriminação de estados. Dissertação (mestrado) – Universidade Federal de Minas Gerais – Departamento de Física. 2018.

LACAVA, Lucas; MIANO, Mariana V. Implementação do algoritmo quântico Deutsch-Jozsa em linguagem funcional e no simulador IBM Q Experience. 2018. Revista Tecnológica da Fatec Americana, vol. 06 n. 2, abr/set de 2018.

MIANO, Mariana V. Aplicação de protocolos quânticos e algoritmo de Shor para a segurança da informação. 2020. Revista Tecnológica da Fatec Americana, vol. 8 n. 01, 2020.

MIANO, Mariana V. OLIVEIRA, Aleccheevina S. Desempenho de algoritmos quânticos e clássicos em treinamento de Machine Learning supervisionado. 2021. Revista Tecnológica da Fatec Americana, v. 9 n. 02.

MICROSOFT. O que é computação em nuvem. MICROSOFT, 2023a. Disponível em: https://azure.microsoft.com/pt-br/resources/cloud-computing-dictionary/what-is-cloud-computing/. Acesso em: 19 abr 2023.

MICROSOFT. O que é o Azure. MICROSOFT, 2023b. Disponível em: https://azure.microsoft.com/pt-br/resources/cloud-computing-dictionary/what-is-azure/. Acesso em: 19 abr 2023

MICROSOFT. What are Q# and the Quantum Development Kit? MICROSOFT, 2023c. Disponível em: https://learn.microsoft.com/en-us/azure/quantum/overview-what-is-qsharp-and-qdk. Acesso em: 1 mai 2023

MySQL Documentation, MySQL, 2023. Disponível em: https://dev.mysql.com/doc/. Acesso em: 21 abr 2023.

NIELSEN, Michael A.; CHUANG, Isaac L. Quantum Computation and Quantum Information. 10th Edition. Editora: Cambridge University Press, 2010. p. 1–161.

PRESKILL, J. (2018). Quantum Computing in the NISQ era and beyond. Quantum, 2, 79.

PYTHON. What is Python? PYTHON, 2023. Disponível em: https://docs.python.org/3/faq/general.html#what-is-python. Acesso em: 19 mai 2023.

TERENCIANI, M. F. Crivo Quadrático: Implementação do Processo de Obtenção de um Quadrado Perfeito. Anais do Enic, v. 1, n. 4, 2015.