

Compartilhamento de informações sensíveis da cadeia de suprimentos: Um estudo sobre vulnerabilidades nos sistemas de criptografia RSA a partir do uso do algoritmo de *shor*

Recebido
Received
Recibido
Jun. 2024

Aceito
Accepted
Aceptado
Dez. 2024

Publicado
Published
Publicado
Jan./Mar. 2025
Ene./Mar. 2025

<https://git.fateczl.edu.br>

e-ISSN
2965-3339

DOI
10.29327/2384439.3.2-3

São Paulo
v. 3 | n. 2
v. 3 | i. 2
e32015
Janeiro-Março
January-March
Enero-Marzo
2025



*Sharing sensitive supply chain information:
A study on vulnerabilities in RSA encryption systems from the use of
the shor algorithm*
*Compartir información confidencial de la cadena de suministro:
Un estudio sobre las vulnerabilidades en los sistemas de cifrado
RSA por el uso del algoritmo shor*

Rodrigo de Souza¹

rodrigo.souza119@fatec.sp.gov.br

Antônio Cesar Galhardi²

antonio.galhardi@cpspos.sp.gov.br

1 – Fatec de Franco da Rocha

2 - Unidade de Pós-Graduação, Extensão e Pesquisa - Centro Estadual de Educação
Tecnológica Paula Souza

Resumo:

Este trabalho propõe uma análise comparativa de algoritmos, considerando-se como base o Algoritmo de Shor. No trabalho emprega-se a ferramenta de inteligência artificial ChatGPT, solicitando-a que elabore códigos na linguagem e computação quântica Qiskit em dois momentos: agosto de 2023 e fevereiro de 2024. Os resultados mostraram que houve um aprendizado considerável da ferramenta ChatGPT em relação às melhores estratégias de implementação do algoritmo, de tal forma que, considerando a última sugestão, de fevereiro de 2024, tem-se um risco considerável à criptografia RSA, caso este tipo de código seja implementado sem supervisão nos ambientes de acesso remoto a computadores quânticos. Destaca-se que o risco à criptografia RSA tem implicações diretas na logística contemporânea, especialmente no compartilhamento de informações sensíveis na cadeia de suprimentos, aumentando a necessidade de desenvolvimento de soluções em segurança da informação que possibilitem prevenir ou atenuar as consequências de tal avanço.

Palavras-chave: criptografia, aprendizado de máquina, algoritmo de Shor, cibersegurança.

Abstract:

This work proposes a comparative analysis of algorithms, considering Shor's Algorithm as a basis. The ChatGPT artificial intelligence tool is used in the work, asking it to create codes in the Qiskit language and quantum computing at two

moments: August 2023 and February 2024. The results showed that there was considerable learning from the ChatGPT tool in relation to better strategies for implementing the algorithm, in such a way that, considering the last suggestion, from February 2024, there is a considerable risk to RSA encryption, if this type of code is implemented without supervision in remote access environments to quantum computers. It is noteworthy that the risk to RSA encryption has direct implications for contemporary logistics, especially in the sharing of sensitive information in the supply chain, increasing the need to develop information security solutions that make it possible to prevent or mitigate the consequences of such advancement.

Keywords: *cryptography, machine learning, Shor's algorithm, cybersecurity.*

Resumen:

Este trabajo propone un análisis comparativo de algoritmos, basado en el algoritmo de Shor. En la obra se utiliza la herramienta de inteligencia artificial ChatGPT, pidiéndole que escriba códigos en el lenguaje Qiskit y la computación cuántica en dos momentos: agosto de 2023 y febrero de 2024. Los resultados mostraron que hubo un aprendizaje considerable de la herramienta ChatGPT en relación con las mejores estrategias para implementar el algoritmo, de tal manera que, considerando la última sugerencia, a partir de febrero de 2024, existe un riesgo considerable para el cifrado RSA, si este tipo de código se implementa sin supervisión en entornos de acceso remoto a computadoras cuánticas. Cabe destacar que el riesgo para el cifrado RSA tiene implicaciones directas en la logística contemporánea, especialmente en el intercambio de información sensible en la cadena de suministro, aumentando la necesidad de desarrollar soluciones de seguridad de la información que permitan prevenir o mitigar las consecuencias de dicho avance.

Palabras clave: *criptografía, machine learning, algoritmo Shor, ciberseguridad.*

1. INTRODUÇÃO

A criptografia é a técnica de encriptação (codificação) e deciptação (decodificação) de uma mensagem por meio de uma chave. Seu uso remonta à períodos anteriores ao advento da computação contemporânea, mas foi com o uso de computadores eletromecânicos durante a segunda guerra mundial, em especial a máquina alemã denominada Enigma, que a criptografia alcançou novo patamar. Com o surgimento de semicondutores e a arquitetura de circuitos integrados, por volta dos anos 1960, a computação passou a contar com maior capacidade de processamento, armazenamento e agilidade.

A criptografia RSA (*Rivest-Shamir-Adleman*) surgiu em 1978 como uma tecnologia assimétrica, ou seja, baseada em duas chaves, uma delas é pública e se constitui do produto de duas chaves privadas, as quais concedem acesso aos dados. O benefício do modelo RSA é a dificuldade de fatoração de números primos grandes e é esta técnica a empregada na geração da chave pública. A quebra deste tipo de criptografia exigiria a fatoração de um número em seus constituintes primos e esta operação demanda um nível de processamento extremamente elevado, em função do tamanho do número empregado na chave pública (Uzeda et al., 2022).

A tecnologia RSA esteve livre de ameaças até 1994, quando o matemático americano Peter Shor propôs um modelo de algoritmo que demandaria sua implementação em computadores quânticos e que seria capaz de romper criptografias RSA com facilidade, uma vez que se baseia no poder da computação quântica para a fatoração (Shor, 1994).

A crescente interconectividade das cadeias de suprimentos contemporâneas, impulsionada pelas tecnologias web e a necessidade de eficiência operacional, trouxe à tona uma série de desafios em relação à segurança da informação. O compartilhamento de informações sensíveis entre diferentes entidades, como fornecedores, fabricantes, distribuidores e varejistas, tem sido uma prática comum, uma vez que se visa a otimização do fluxo de mercadorias. No entanto, esta prática traz consigo um risco substancial, especialmente em um cenário no qual a segurança cibernética é uma preocupação crescente. Vulnerabilidades nos sistemas de TI, falta de criptografia robusta e métodos de autenticação ineficientes podem expor as informações a ciberataques, resultando em vazamento de dados, interrupções na cadeia de suprimentos e perda de confiança dos clientes. Este risco é amplificado quando se considera a possibilidade de ataques oriundos de computadores quânticos, que poderiam romper criptografias tradicionais, como a RSA, de forma muito mais rápida do que os computadores clássicos. Nesse contexto, é essencial que as organizações adotem medidas preventivas para garantir a integridade das informações sensíveis em suas cadeias de suprimentos. Além disso, o uso de tecnologias emergentes, como blockchain e inteligência artificial, pode ajudar a fortalecer a segurança e a resiliência das cadeias de suprimentos, garantindo a integridade e a confidencialidade das informações compartilhadas.

Até meados dos anos 2000, o Algoritmo de *Shor* era visto muito mais como uma curiosidade acadêmica do que como um instrumento prático, no entanto, dados

os avanços no desenvolvimento de computadores quânticos, sua aplicabilidade passou de uma situação hipotética à uma possibilidade (VIEIRA; ALBUQUERQUE, 2020). Além disso, com o surgimento de sistemas de inteligência artificial generativos, dos quais o mais famoso e talvez um dos mais avançados, é o ChatGPT, lançado no final de 2022, a possibilidade de tradução do Algoritmo de *Shor* em uma linguagem quântica como o *Qiskit*, passível de implementação no ambiente quântico de acesso remoto da IBM denominado IBM Quantum Experience, abre possibilidades pouco exploradas até então, de uso inadequado destas ferramentas na tentativa de romper sistemas de cibersegurança baseados em criptografia RSA. Neste estudo, a questão básica de pesquisa é: Qual o nível de assertividade do código *Qiskit* proposto pelo ChatGPT para o Algoritmo de *Shor*? Para a resposta desta pergunta, este artigo, baseou-se inicialmente em uma pesquisa bibliográfica do tipo estado da arte, sobre computação quântica e IA generativa. Em seguida, elaborou-se uma pesquisa empírica, na qual solicitou-se ao ChatGPT a proposição de um código *Qiskit* para o Algoritmo de *Shor*, e que foi submetida à uma análise comparativa entre o código proposto e o algoritmo original, de forma a se verificar qual o nível de assertividade. Em um terceiro momento, realizou-se a mesma solicitação ao ChatGPT, após 6 meses, de forma a se verificar o nível de aprendizado da plataforma em relação à construção do algoritmo.

2. EMBASAMENTO TEÓRICO

A computação atual é baseada na tecnologia do semicondutor, empregada na construção de transistores, que são dispositivos com capacidade de conduzir e não conduzir eletricidade; ou seja: o nome semicondutor vem do fato de que o tipo de material empregado apresenta propriedades condutores e isolantes. A existência desta propriedade semicondutora é imprescindível para o funcionamento dos computadores, já que o sistema computacional é baseado em bits, que são os valores de 0 (zero) e 1 (um), empregados no código binário.

Os semicondutores são formados a partir de uma estrutura cristalina de silício, um material isolante. Sua estrutura cristalina é então dopada (adição de pequenas quantidades) com elementos que a tornarão semicondutora. O processo de dopagem pode ser executado de duas formas distintas (Weste; Harris, 2009):

- a) Dopagem negativa: insere-se átomos de fósforo com um elétron a mais;
- b) Dopagem positiva: insere-se átomos de índio, com um elétron a menos.

A ausência de um elétron ou a presença de um elétron a mais tornarão a estrutura cristalina semicondutora, possibilitando o processo de codificação binária. Note-se que o funcionamento do transistor apresenta apenas um estado por vez, ou apenas um bit por vez pode ser lido. De forma geral, na dopagem negativa, com o fluxo de corrente, tem-se o bit 1 e na ausência de corrente, tem-se o bit 0. Na dopagem positiva, ocorre o contrário (Weste; Harris, 2009).

Na computação quântica não se faz processamento por transistores. Os elementos empregados são subpartículas atômicas, como elétrons e fótons,

manipulados individualmente. Como a agitação de moléculas é um fenômeno termodinâmico, um aspecto relevante é a necessidade de se manter tais computadores no zero absoluto, 0°K (-273°C), já que em temperaturas superiores, haverá agitação das partículas subatômicas. No zero absoluto, apenas as partículas selecionadas sofrem agitação (Portugal; Marquezino, 2019).

O termo “quântico” refere-se ao uso dos preceitos da mecânica quântica na determinação e explicação dos processos computacionais. Em relação a estes conceitos, a ideia de entrelaçamento quântico é especialmente importante, já que é a forma pela qual se troca informações em um computador quântico. O entrelaçamento quântico é a paridade de spin (momento angular) de duas partículas subatômicas que, uma vez juntas, mantem tal paridade mesmo sendo afastadas posteriormente (Toneli, 2022).

Outro aspecto relevante é o princípio da incerteza de *Heisenberg*, segundo o qual, de forma simplificada, não se pode definir posição e velocidade de uma partícula subatômica com precisão. A escolha de uma grandeza implica na determinação probabilística da segunda. Na perspectiva do computador quântico, isso se aplica na possibilidade de que um bit quântico (ou *qubit*) poder assumir o valor 0, o valor 1 ou ambos simultaneamente. Estima-se que um computador de cerca de 100 *qubit* tenha a mesma capacidade de processamento da soma de todos os computadores do mundo, atualmente (Parada, 2019).

2.1 O algoritmo de *Shor*

O Algoritmo de *Shor* foi desenvolvido por Peter *Shor* em 1994 e possibilita a fatoração de um número N em tempo polinomial, empregando-se um computador quântico. Suas etapas são (Shor, 1994):

- a) Preparação de Estado Quântico: deve-se preparar um estado quântico em uma superposição de todos os possíveis valores para a entrada do algoritmo. Para isso, são utilizados *qubits* em dois registradores, um para a entrada e outro para a saída. O registrador de entrada é inicializado em um estado superposto com todos os possíveis valores da entrada, enquanto o registrador de saída é inicializado em um estado com valor 1.
- b) Transformada de Fourier Quântica: deve-se aplicar uma transformada de Fourier quântica ao registrador de entrada. Essa transformada converte a superposição de todos os possíveis valores em um conjunto de amplitudes que correspondem aos fatores primos do número a ser fatorado.
- c) Medição e pós-processamento: deve-se medir o registrador de saída e realizar um pós-processamento clássico para encontrar os fatores primos. A medição do registrador de saída colapsa a superposição de estados, resultando em um valor único que representa um dos fatores primos do número a ser fatorado. Depois que um fator é encontrado, um algoritmo clássico de fatoração é usado para encontrar o outro fator.

O pseudocódigo do algoritmo, baseado no trabalho de Shor, (1994), é demonstrado a seguir:

Início

1. Escolha um inteiro aleatório a entre 1 e $N-1$.
2. Calcule o máximo divisor comum de a e N . Se o m.d.c. for diferente de 1, então a e N têm um fator em comum e a fatoração de N é encontrada.
3. Crie um circuito quântico que atua na base $\{|0\rangle, |1\rangle, \dots, |N-1\rangle\}$ com uma porta unitária $U(x)$ que calcula $a^x \bmod N$.
4. Prepare um registrador quântico em um estado superposto usando um número suficiente de *qubits* para armazenar N .
5. Aplique a porta $U(x)$ ao estado superposto. Isso cria uma superposição de estados que representam todas as possíveis potências de $a \bmod N$.
6. Aplique a transformada de Fourier quântica ao estado superposto.
7. Meça o registrador quântico. O resultado da medição é um número r que é um múltiplo inteiro do período da função $a^x \bmod N$.
8. Calcule o máximo divisor comum de N e $a^{(r/2)} + 1$. Se o m.d.c. for diferente de 1, então a fatoração de N é encontrada. Caso contrário, volte ao passo 1 e escolha um novo valor de a .

Fim

O pseudocódigo acima pode ser compreendido da seguinte forma: O algoritmo começa escolhendo um número inteiro aleatório “ a ”, que é usado para calcular as possíveis potências de “ a ” $\bmod N$. Em seguida, é calculado o máximo divisor comum entre “ a ” e N . Se o m.d.c. for diferente de 1, então a fatoração de N é encontrada. O próximo passo é criar um circuito quântico que atue na base $\{|0\rangle, |1\rangle, \dots, |N-1\rangle\}$ e incluir uma porta unitária $U(x)$ que calcula $a^x \bmod N$ para cada x da base. Em seguida, um registrador quântico é preparado em um estado superposto e a porta $U(x)$ é aplicada ao estado superposto. Isso cria uma superposição de estados que representam todas as possíveis potências de $a \bmod N$. A transformada de Fourier quântica é aplicada ao estado superposto, e em seguida é feita uma medição do registrador quântico. O resultado da medição é um número r que é um múltiplo inteiro do período da função $a^x \bmod N$. O período é o menor número inteiro positivo r para o qual $a^r \bmod N = 1$. Por fim, é calculado o máximo divisor comum de N e $a^{(r/2)} + 1$. Se o m.d.c. for diferente de 1 e diferente de N , então a fatoração de N é encontrada. Caso contrário, o algoritmo deve ser repetido com um novo valor aleatório de “ a ” (Shor, 1994). Cabe ressaltar que o objetivo da função “ \bmod ” é retornar o resto de uma divisão. O resto da divisão é um elemento chave para os processos de criptografia e decifração.

2.2 Criptografia RSA

O sistema de criptografias RSA baseia-se no seguinte pressuposto: dados os fatores primos p e q , com valores grandes em cada um, tais valores formam uma chave privada de uma chave pública $n = p \cdot q$ (CASTRO, 2019). Isso significa que, tanto a codificação, quanto a decodificação dependem de p e q . Esta abordagem criptográfica obedece ao Teorema Fundamental da Aritmética em N , segundo o qual: considerando-se um número inteiro positivo $n = 1$, pode-se escrevê-lo da seguinte forma $n = p_1^{e_1} \cdot \dots \cdot p_k^{e_k}$, sendo $1 < p_1 < \dots < p_k$ e $p =$ primo. Ou seja, em síntese, sabemos que qualquer número natural pode ser decomposto em

constituintes primos.

A aplicação da sistemática RSA demanda a conversão de letras em números e a definição de um número para os espaços entre as palavras. Feito isso, escolhe-se um “expoente de codificação” e eleva-se o número correspondente à mensagem por este expoente. Logo após, divide-se o resultado por um “módulo de criptografia”. O processo de decodificação segue a lógica inversa (Teixeira, 2020).

2.3 ChatGPT

O ChatGPT é um *chatbot*, ou seja, um robô para conversas em chat, baseado em Inteligência Artificial generativa. A IA generativa normalmente é baseada em uma Rede Neural Artificial, e atua por meio de pesquisas livres sobre determinado assunto, realizando uma síntese na resposta, a qual também é armazenada como um aprendizado. Esta abordagem de inteligência artificial é baseada em processamento de linguagem natural e isso é relevante dada a necessidade de se aproximar a interface da IA ao usuário final (Ramesh et al., 2017).

3. DESENVOLVIMENTO DA TEMÁTICA

A pesquisa baseia-se em uma comparação entre a proposta de Algoritmo de *Shor* na linguagem quântica *Qiskit*, desenvolvida pelo ChatGPT e o algoritmo original, de forma se verificar se há correspondência entre ambos. Tal comparação ocorreu em dois momentos: o primeiro em agosto de 2023 e o segundo, em fevereiro de 2024. O uso de dois momentos deve-se ao objetivo de se investigar o nível de aprendizado e aperfeiçoamento da plataforma em relação ao algoritmo proposto. Na plataforma ChatGPT, o texto provocador foi “Faça o Algoritmo de *Shor* na linguagem *Qiskit*”. O resultado do primeiro código gerado (agosto de 2023) pode ser visto na Figura 1.

Ao comparar-se linha a linha a proposta do ChatGPT com o algoritmo original de Peter Shor, em pseudocódigo, observa-se um nível de fidelidade elevado. Deve-se considerar que as primeiras três linhas do código da figura 1 referem-se às bibliotecas importadas. O código proposto também empregou um número para fatoração (15), como forma de exemplificar sua operacionalidade. A análise linha a linha é afeita a seguir:

1. Shor: Escolha um inteiro aleatório a entre 1 e $N-1$;
ChatGPT: $N = 15$;
2. Shor: Calcule o máximo divisor comum de a e N . Se o m.d.c. for diferente de 1, então a e N têm um fator em comum e a fatoração de N é encontrada;
ChatGPT: Não realizado;
3. Shor: Crie um circuito quântico que atua na base $\{|0\rangle, |1\rangle, \dots, |N-1\rangle\}$ com uma porta unitária $U(x)$ que calcula $a^x \bmod N$;
ChatGPT: $qc = \text{QuantumCircuit}(4, 4)$;
4. Shor: Prepare um registrador quântico em um estado superposto usando um número suficiente de qubits para armazenar N ;

- ChatGPT: qc.h(range(3))
qc.x(3);
5. Shor: Aplique a porta $U(x)$ ao estado superposto. Isso cria uma superposição de estados que representam todas as possíveis potências de a mod N ;
ChatGPT: shor = Shor(N, a=2,
quantum_instance=Aer.get_backend('qasm_simulator'))
qc.append(shor.circuit, qc.qubits[:3]);
 6. Shor: Aplique a transformada de Fourier quântica ao estado superposto;
ChatGPT: qc.append(QFT(3, inverse=True), qc.qubits[:3]);
 7. Shor: Meça o registrador quântico. O resultado da medição é um número r que é um múltiplo inteiro do período da função $a^x \text{ mod } N$;
ChatGPT: qc.measure(range(3), range(3));
 8. Shor: Calcule o máximo divisor comum de N e $a^{(r/2)} + 1$. Se o m.d.c. for diferente de 1, então a fatoração de N é encontrada. Caso contrário, volte ao passo 1 e escolha um novo valor de a ;
ChatGPT: backend = Aer.get_backend('qasm_simulator')
job = execute(qc, backend, shots=1024)

Figura 1 – Algoritmo de Shor em Qiskit proposto pelo ChatGPT em agosto de 2023

```
from qiskit import QuantumCircuit, Aer, execute
from qiskit.algorithms import Shor
from qiskit.circuit.library import QFT

# Define o número a ser fatorado
N = 15

# Cria um circuito quântico para o algoritmo de Shor
qc = QuantumCircuit(4, 4)

# Aplica as portas necessárias no primeiro registrador
qc.h(range(3))
qc.x(3)

# Aplica o algoritmo de Shor
shor = Shor(N, a=2, quantum_instance=Aer.get_backend('qasm_simulator'))
qc.append(shor.circuit, qc.qubits[:3])

# Aplica a transformada de Fourier inversa no primeiro registrador
qc.append(QFT(3, inverse=True), qc.qubits[:3])
```

Fonte: Os autores (2024)

Com exceção do item 2, os demais pontos do Algoritmo de Shor foram atendidos. O item 2, embora não tenha sido proposto pelo ChatGPT, não é imprescindível para o resultado, já que se trata de uma verificação a priori da fatoração. Quanto aos demais, na perspectiva de funcionalidades, foram atendidos na íntegra.

O segundo código gerado (fevereiro de 2024) é apresentado na figura 2.

Figura 2 – Algoritmo de *Shor* em *Qiskit* proposto pelo *ChatGPT* em fevereiro de 2024

```

from math import gcd
from random import randint
from qiskit import Aer, QuantumCircuit, transpile, assemble, execute
from qiskit.visualization import plot_histogram

def modular_exponentiation(a, exponent, N):
    """Perform modular exponentiation a^exponent mod N."""
    return pow(a, exponent, N)

def quantum_modular_exponentiation(qc, a, exponent, N, n):
    """Implement modular exponentiation a^exponent mod N using a quantum circuit."""
    for i in range(n):
        qc.x(i) if (exponent & (1 << i)) else None
    for i in range(n):
        quantum_multiply_by_a_mod_N(qc, a, 2 ** i, N, n)
    return qc

def quantum_multiply_by_a_mod_N(qc, a, exponent, N, n):
    """Implement the controlled modular multiplication by a."""
    for i in range(n):
        qc.swap(i, n + i)
    for i in range(n):
        qc.cx(i, n + i) if (exponent & (1 << i)) else None
    for i in range(n - 1, -1, -1):
        qc.cx(i, n + i) if (exponent & (1 << i)) else None
    return qc

def find_period(N, a, n):
    """Find the period r of the function f(x) = a^x mod N."""
    qc = QuantumCircuit(n * 2, n)

    quantum_modular_exponentiation(qc, a, 1, N, n)

    for q in range(n):
        qc.measure(q, q)

    backend = Aer.get_backend('qasm_simulator')
    result = execute(qc, backend, shots=1).result()
    counts = result.get_counts(qc)

    measured_period = int(list(counts.keys())[0], 2)
    measured_period = measured_period if measured_period else 2 ** n

    return measured_period

def shor(N, n_trials=20):
    """Shor's Algorithm to find prime factors of N."""
    if N % 2 == 0:
        return 2

    for _ in range(n_trials):
        a = randint(2, N - 1)
        if gcd(a, N) != 1:
            return gcd(a, N)
        else:
            measured_period = find_period(N, a, n)
            if measured_period % 2 != 0:
                continue

            if modular_exponentiation(a, measured_period // 2, N) != N - 1:
                p = gcd(modular_exponentiation(a, measured_period // 2, N) + 1, N)
                q = gcd(modular_exponentiation(a, measured_period // 2, N) - 1, N)
                return p, q

# Example usage
N = 15
n = 4 # Number of qubits needed to represent N
factors = shor(N)
print("Factors of", N, "are:", factors)

```

Fonte: os autores, 2024.

Após a proposição do novo código, pelo *ChatGPT*, procedeu-se à sua análise em relação ao algoritmo original. O novo código (fevereiro de 2024) proposto pelo *ChatGPT*, além dos itens já atendidos no algoritmo anterior (agosto de 2023), também atendeu o item 2 (máximo divisor comum), que não havia sido atendido no primeiro código gerado. O próximo passo foi a análise comparativa dos algoritmos, a fim de se verificar qual dos dois códigos faz uso mais racional e otimizado da ferramenta *Qiskit*. Os critérios para esta análise foram (Johnston; Harrigan; Segovia, 2019):

a) Controle e Flexibilidade: Trata-se da capacidade de controlar cada aspecto do algoritmo e da implementação, bem como de sua flexibilidade para ajustar e otimizar diferentes partes do código de acordo com as necessidades específicas do problema.

b) Implementação da Transformada de Fourier Quântica (QFT): Verifica a forma como a QFT é implementada no código. Considera-se que, uma implementação manual pode oferecer maior controle sobre cada detalhe da QFT, possibilitando ajustes específicos conforme necessário. Por outro lado, optando-se pelo uso de uma classe predefinida, há a simplificação do código, no entanto, pode-se limitar a acurácia dos cálculos.

c) Personalização e Otimização: Trata-se da capacidade que o modelo oferece de se personalizar o algoritmo como um todo, de forma a atender às necessidades específicas do problema em questão, ou seja, tal personalização inclui ajuste de parâmetros, otimização de partes do código e adaptação do algoritmo para diferentes cenários de uso.

4. RESULTADOS E DISCUSSÃO

Com relação ao controle de código e sua flexibilidade, observa-se que, no código de fevereiro de 2024, o algoritmo de *Shor* é implementado manualmente, o que oferece maior controle sobre cada passo do processo, incluindo valores de referência para as variáveis e eventuais aproximações. Desta forma, é possível otimizar cada etapa do algoritmo de acordo com as características específicas do problema, como por exemplo, ajustando-se a escolha de 'a' dinamicamente, maximizando a probabilidade de encontrar o período rapidamente.

No código de agosto de 2023, o algoritmo de *Shor* é encapsulado em uma classe padronizada fornecida pela biblioteca *Qiskit*. Embora esta estratégia ofereça conveniência, pode-se limitar o controle sobre o processo, já que, algumas otimizações ou ajustes específicos podem não ser possíveis sem modificar diretamente o código-fonte da classe "*Shor*".

Com relação à implementação da Transformada de Fourier Quântica (QFT), observa-se que, no código de fevereiro de 2024, a QFT é implementada manualmente usando uma função definida no próprio código. Tal solução possibilita maior controle sobre a implementação da QFT e a capacidade de ajustá-la para diferentes tamanhos de registro ou requisitos de precisão.

No código de agosto de 2023, a QFT é aplicada usando uma classe padronizada

fornecida pela biblioteca *Qiskit*. Novamente, tem-se a facilidade e a agilidade de trabalhar com uma solução definida a priori, no entanto, esta opção pode limitar a capacidade de personalização da QFT, especialmente em situações nas quais haver a necessidade de uma implementação específica.

Com relação à personalização e otimização do código, observa-se que os recursos oferecidos pela implementação manual do algoritmo de *Shor* e da QFT no código de fevereiro de 2024, permitem maior personalização e otimização para o problema específico em questão. Por exemplo, é possível ajustar os parâmetros do algoritmo de *Shor*, como o número de tentativas para encontrar fatores primos ou a precisão da QFT, para melhorar o desempenho ou a eficiência do algoritmo.

No código de agosto de 2023, as opções de personalização podem ser mais limitadas devido à encapsulação do algoritmo de *Shor* e da QFT em classes fornecidas pela biblioteca *Qiskit*. O uso destes recursos padronizados pode dificultar ou impossibilitar ajustes específicos para melhorar o desempenho ou a eficiência do algoritmo.

5. CONSIDERAÇÕES FINAIS

Os resultados evidenciaram que, em ambas as situações, houve atendimentos do algoritmo de *Shor* nos códigos propostos pelo ChatGPT, no entanto, parece ter ocorrido um aprendizado por parte da plataforma no período de 6 meses entre as solicitações. Este aprendizado parece ter melhorado a capacidade do código em relação à tratativa de situações específicas que fatoração, ou seja, tem-se na primeira proposta (agosto de 2023) um uso massivo de recursos padronizados, já disponibilizados para usuários da linguagem *Qiskit*, no entanto, na proposta de fevereiro de 2023, observa-se a implementação “manual” da maior parte destes recursos.

A opção pela implementação “manual” dos recursos é positiva quando se considera que a estrutura dos algoritmos RSA não é padrão em sua totalidade, portanto, para cada algoritmo RSA, há uma necessidade específica de fatoração dos números primos, dado o nível de complexidade da técnica empregada na criptografia. Quando se emprega um código com muitos elementos padronizados, como ocorreu no código proposto em agosto de 2023, pode-se esbarrar em problemas que não sejam passíveis de decomposição com tais ferramentas. Por outro lado, optando-se por uma solução customizada, como se observou no código de fevereiro de 2024, a possibilidade de não se romper a criptografia é menor, uma vez que o código é ajustável às especificações do problema em questão.

O que se verifica, é que a ferramenta ChatGPT aparentemente “aprendeu” que o algoritmo de *Shor* é melhor implementado quando se deixa de lado as ferramentas padrão da linguagem *Qiskit* e se emprega códigos personalizados e ajustáveis. Este aprendizado, do ponto de vista da inteligência artificial, é muito positivo, uma vez que as inúmeras iterações possibilitaram uma melhoria contundente na construção do algoritmo, por outro lado, considerando-se

aspectos da cibersegurança, tem-se um risco maior, já que a implementação deste código otimizado em uma plataforma de acesso remoto a computadores quânticos, poderia abrir eventuais brechas nos sistemas de criptografia o que, em última análise, poderia comprometer operações sensíveis tais como blockchain e transações financeiras tradicionais, portanto, há uma necessidade latente de aprimoramento dos sistemas de criptografia em uso atualmente.

Por fim, considera-se que a crescente dependência dos sistemas logísticos na troca de informações sensíveis, demanda a implementação de medidas preventivas para garantir a integridade de dados e informações. Como possibilidades a serem exploradas, tem-se a adoção de criptografia robusta, políticas de segurança rigorosas e o investimento em tecnologias emergentes, como blockchain e inteligência artificial, de forma a se fortalecer a segurança e a resiliência das cadeias de suprimentos.

REFERÊNCIAS

CASTRO, C. C. **Criptografia RSA**. Trabalho de Conclusão de Curso - Universidade Federal de Santa Catarina, Departamento de Matemática (MAT), Centro de Blumenau, Curso de Licenciatura em Matemática, Blumenau, 2019.

JOHNSTON, E.R.; HARIGAN, N.; SEGOVIA, M.G. **Programming quantum computers**. O'Reilly Media, Inc., Sebastopol, 2029.

PARADA, I.U. **Aplicação de processamento quântico para aceleração da resolução de autômatos probabilísticos**. Trabalho de conclusão de curso (Bacharelado em Engenharia Mecatrônica) — Universidade de Brasília, Brasília, 2019.

PORTUGAL, R.; MARQUEZINO, F. **Introdução à Programação de Computadores Quânticos**. Sociedade Brasileira de Computação, 2019.

RAMESH, K.; RAVISHANKARAN, S.; JOSHI, A.; CHANDRASEKARAN, K. **A survey of design techniques for conversational agents**. International Conference on Information, Communication and Computing Technology, Springer, Singapore, 2017.

SHOR, P.W. **Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer**. 35th Annual Symposium on Foundations of Computer Science, Santa Fe, NM, Nov. 20–22, 1994, IEEE Computer Society Press, pp. 124–134, 1994.

TONELI, D.A. **Desenvolvimento de um tutorial para o ensino de computação quântica**. Trabalho de Conclusão de Curso – Unifesp, 2022.

TEIXEIRA, M.A.F. **Números inteiros e criptografia RSA**. Dissertação (mestrado) - Universidade Estadual Paulista (Unesp), Instituto de Geociências e Ciências Exatas, Rio Claro, 2020.

UZEDA, E. E. F.; DALLASEN, R. V.; SANTOS, M. C.; ROLLWAGEN, A. F.; RIBEIRO, D. D.; FIGUEIREDO, J. A. O. **Utilização do algoritmo de Shor para quebra de criptografia RSA em computadores quânticos**. Salão do Conhecimento, Unijuí, Ijuí, 2022.

VIEIRA, L.A.; ALBUQUERQUE, C.D. **Um estudo passo a passo do Algoritmo de Shor**.
Proceeding Series of the Brazilian Society of Computational and Applied Mathematics,
v. 7, n. 1, 2020.
WESTE, N.H.E., HARRIS, D. M. **CMOS VLSI Design A Circuits and Systems Perspective**.
Addison – Wesley, New York, 2009.

"Os conteúdos expressos no trabalho, assim como os direitos autorais de figuras e dados,
bem como sua revisão ortográfica e das normas ABNT são de inteira responsabilidade
do(s) autor(es)."