UNIVERSIDADE FEDERAL DE SANTA CATARINA
CAMPUS TRINDADE
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Luiz Filipe Anderson de Sousa Moura

Ameaças quânticas ao TLS 1.3

Florianópolis
2024

Luiz Filipe Anderson de Sousa Moura

Ameaças quânticas ao TLS 1.3

Dissertação submetida ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Santa Catarina para a obtenção do título de Mestre em Ciência da Computação.
Orientador: Prof. Ricardo Felipe Custódio, Dr.

Florianópolis
2024

Ficha catalográfica gerada por meio de sistema automatizado gerenciado pela BU/UFSC.
Dados inseridos pelo próprio autor.

Moura, Luiz Filipe Anderson de Sousa
Ameaças quânticas ao TLS 1.3 / Luiz Filipe Anderson de Sousa Moura ; orientador, Ricardo Felipe Custódio, 2024.
103 p.

Dissertação (mestrado) - Universidade Federal de Santa Catarina, Centro Tecnológico, Programa de Pós-Graduação em Ciência da Computação, Florianópolis, 2024.

Inclui referências.

1. Ciência da Computação. 2. Computação quântica. 3. Segurança da informação. I. Custódio, Ricardo Felipe. II. Universidade Federal de Santa Catarina. Programa de Pós Graduação em Ciência da Computação. III. Título.

Luiz Filipe Anderson de Sousa Moura

Ameaças quânticas ao TLS 1.3

O presente trabalho em nível de mestrado foi avaliado e aprovado por banca examinadora composta pelos seguintes membros:

Prof. Ricardo Felipe Custódio, Dr.
Instituição UFSC

Prof. Jean Everson Martina, Dr.
Instituição UFSC

Profa. Carla Merkle Westphall, Dra.
Instituição UFSC

Prof. Alexandre Augusto Giron, Dr.
Instituição UTFPR

Certificamos que esta é a **versão original e final** do trabalho de conclusão que foi julgado adequado para obtenção do título de Mestre em Ciência da Computação.

Documento assinado digitalmente
Marcio Bastos Castro
Data: 17/06/2024 15:53:08-0300
CPF: ***.707.720-**
Verifique as assinaturas em https://v.ufsc.br

Coordenação do Programa de
Pós-Graduação

Documento assinado digitalmente
Ricardo Felipe Custodio
Data: 18/06/2024 15:23:05-0300
CPF: ***.981.609-**
Verifique as assinaturas em https://v.ufsc.br

Prof. Ricardo Felipe Custódio, Dr.
Orientador

Florianópolis, 2024.

Dedicado aos novos estudantes, aos
quais este trabalho possa interessar.

AGRADECIMENTOS

Agradeço aos professores, pelo conhecimento. Agradeço aos membros da banca, titulares e suplentes, pelas sugestões ao trabalho. Um agradecimento especial ao professor Alexandre Augusto Giron, pela larga ajuda a este trabalho. Agradeço à SETEC, pela bolsa no ano de 2022, sob a qual participei do grupo de desenvolvimento do projeto SAAS - Sistema de Acompanhamento e Avaliações de Cursos. E agradeço à CAPES, pela bolsa acadêmica no restante do curso.

"I think I can safely say that
nobody understands quantum mechanics"
(Feynman, 1965)
"Despite the great difficulty of constructing a truly general-purpose
quantum computer, it might be relatively easy to construct a
special-purpose quantum factoring machine which could be used
for code-breaking. History does have a tendency to repeat itself;
were not the first digital computers used for code-breaking?"
(Song Y. Yan, 2000, "Number Theory for Computing")

RESUMO

O termo, computação quântica, foi primeiramente pronunciado há mais de 40 anos, mas esta ainda é uma área de vanguarda da computação. Esse novo paradigma ameaça destruir a segurança da informação como a conhecemos por ser capaz de resolver uma determinada classe de problemas considerados difíceis para os computadores clássicos, problemas como o da fatoração e o do logaritmo discreto, os quais são o cerne da criptografia assimétrica moderna. Este trabalho expõe essas ameaças, principalmente ao *Transport Layer Security* (TLS) 1.3, modela a ameaça, detalha cenários de ataque aos diferentes modos do TLS 1.3, levanta requisitos de armazenamento para um ataque *Store-Now-Decrypt-Later* e apresenta formas de mitigar essas ameaças. Depois, o trabalho mapeia sistematicamente ferramentas de computação quântica e experimenta com uma delas para entender melhor os desafios que um atacante enfrentaria em ataques reais ao TLS 1.3. Ao que se concluiu que a ameaça quântica se tornará real aos poucos em vez de imediatamente. À medida que as tecnologias quânticas avançam, as tecnologias de defesa pós-quânticas também avançam. Uma preocupação adicional por parte do atacante é o uso de engenharia social para filtrar melhor quais pacotes deverão ser capturados para um ataque *Store-Now-Decrypt-Later*; em reflexo a esta preocupação, as companhias também devem estar conscientes desses ataques.

Palavras-chave: Computação quântica; Criptologia; Transport Layer Security; Store-Now-Decrypt-Later; Algoritmo de Shor.

ABSTRACT

The term, quantum computing, was first coined more than 40 years ago, but this is still a cutting-edge area of computing. This new paradigm threatens to destroy information security as we know it by being able to solve a certain class of problems considered difficult for classical computers, problems such as factorization and discrete logarithm, which are the core of modern asymmetric cryptography. This work exposes these threats, mainly to Transport Layer Security (TLS) 1.3, models the threat, details attack scenarios for the different modes of TLS 1.3, raises storage requirements for a Store-Now-Decrypt-Later attack and presents ways to mitigate these threats. Next, the work systematically maps quantum computing tools and experiments with one of them to better understand the challenges an attacker would face in real attacks on TLS 1.3. It was concluded that the quantum threat will become real gradually rather than immediately. As quantum technologies advance, so do post-quantum defense technologies. An additional concern for the attacker is the use of social engineering to better filter which packets should be captured for a Store-Now-Decrypt-Later attack; In response to this concern, companies must also be aware of these attacks.

Keywords: Quantum computing; Cryptology; Transport Layer Security; Store-Now-Decrypt-Later; Shor's algorithm.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

0-RTT	Zero Round Trip Time
3DES	Triple Data Encryption Standard
AES	Advanced Encryption Standard
BB84	Bennett-Brassard 1984
BB92	Bennett-Brassard 1992
BBM92	Bennett-Brassard-Mermin 1992
BQP	Bounded Error, Quantum, Polinomial Time
CBC-MAC	Cipher Block Chaining Message Authentication Code
COW	Coherent One-Way
DES	Data Encryption Standard
DH	Diffie-Hellman
DHE	Diffie-Hellman Ephemeral
DPS	Differential Phase Shift
E91	Ekert 1991
EC	Elliptic-Curve
ECDLP	Elliptic-Curve Discrete Logarithm Problem
EdDSA	Edwards-curve Digital Signature Algorithm
GCM	Galois/Counter Mode
GMAC	Galois Message Authentication Code
HKDF	HMAC-based Extract-and-Expand Key Distribution Function
HTTPS	Hyper Text Transfer Protocol Secure
NISQ	Noisy Intermediate Scale Quantum
NIST	National Institute os Standardization and Technology
NP	Non-deterministic Polynomial Time
OCB	Offset Codebook mode
P	Polynomial Time
PCS	Post Compromise Security
PFS	Perfect Forward Secrecy
PMAC	Parallelizable Message Authentication Code
PQC	Post-Quantum Cryptography
PQKEM	Post-Quantum Key-Encapsulation Mechanism
PSK	Pre Shared Key
QFT	Quantum Fourier Transform
QKD	Quantum Key Distribution
RSA	Rivest-Shamir-Adleman
RSASSA-PSS	RSA Probabilistic Signature Scheme
SARG02	Scarani-Acin-Ribordy-Gisim 2002
SARG04	Scarani-Acin-Ribordy-Gisim 2004

SNDL	Store-Now-Decrypt-Layer
SSP	Six-State Protocol
TLS	Transport Layer Security
VPN	Virtual Private Network
X3DH	Extended Triple Diffie-Hellman

LISTA DE SÍMBOLOS

$\|\psi\rangle$	Equação de estado de um qubit ou registrador
a, b	Amplitudes de probabilidade
e	Número neperiano
i	Unidade imaginária
S	Quantidade de estados
G	Porta Lógica
$\|q\rangle$	Equação de estado de um registrador
ns	Nanosegundos
μs	Microssegundos
epk	Chave pública extraída
MB	Megabytes
GB	Gigabytes
TB	Terabytes

SUMÁRIO

1 INTRODUÇÃO

1.1 CONTEXTUALIZAÇÃO DO PROBLEMA

O início do século XX introduziu a física quântica, que revolucionou a compreensão de partículas subatômicas com sua natureza probabilística (SHOR, 1994). Na virada do século, em 1900, Max Planck sugeriu a ideia de que corpos subatômicos têm energia quantizada em pequenos pacotes de energia chamados *quanta*. Einstein ganhou o Prêmio Nobel pelo estudo do efeito fotoelétrico, onde um fóton apresenta comportamento dual entre onda e partícula, adicionando momento a um elétron (FREITAS; FERREIRA; SILVA FILHO, 2019). DeBroglie também estudou a dualidade onda-partícula, mas desta vez mostrando que um corpúsculo também poderia apresentar comportamento de onda. Schroedinger e Heisenberg também evidenciaram comportamentos estranhos no mundo subatômico, como a incerteza de medir a exata posição de uma partícula em um determinado tempo. A noção central que governa a mecânica das partículas subatômicas é que, em vez de medir a posição x de uma partícula em um determinado instante, só podemos medir a probabilidade da partícula se encontrar em x (KHAN, T. M.; ROBLES-KELLY, 2020).

Computação é o processo de receber uma entrada e produzir uma saída, que depende dos valores de entrada. Esses valores de entrada e saída são valores abstratos (a entrada e a saída são objetos abstratos) que podem, ou não, representar algo concreto. Contudo, ao falar de uma máquina de computar (um computador), fala-se de um objeto com extensão física, as entradas e saídas também possuem caráter concreto. As propriedades físicas das máquinas de computar ditam a velocidade de operação, as entradas são um conjunto de valores iniciais possíveis para uma máquina e as saídas são um conjunto de valores possíveis de serem lidos num estado de parada da máquina (DEUTSCH, D. E., 1989).

A computação quântica é um novo paradigma da computação, cuja ideia conceitual foi introduzida em 1982 pelo físico Richard Feynman, que permite a resolução de uma certa classe de problemas complexos em tempo polinomial. Devido a características do mundo quântico, como emaranhamento e superposição, essa nova classe de computadores conseguiria resolver, em tempo hábil, problemas matemáticos difíceis, que durariam uma infinidade de tempo em computadores comuns. O tema da computação quântica é um dos maiores desafios para a ciência da computação do século XXI e está sendo estudado de maneira intensa atualmente (MAVROEIDIS *et al.*, 2018; HAGOUEL; KARAFYLLIDIS, 2012).

A segurança da criptografia de chaves públicas está baseada em problemas "*Non-deterministic Polynomial Time*" (NP), os quais levam um tempo de ordem exponencial para serem solucionados (ou não há conhecimento sobre algoritmos que resolvam algum desses problemas em tempo polinomial), mas podem ser verificados

em tempo polinomial (SHOR, 1994; YUNAKOVSKY *et al.*, 2021). Neste aspecto, os computadores quânticos apresentam alguma vantagem sobre computadores clássicos e põem em risco a segurança de sistemas computacionais atuais ao resolverem, em tempo hábil, alguns problemas matemáticos considerados difíceis ou impossíveis para computadores clássicos. No caso da criptografia de chaves assimétricas, o algoritmo de Shor (SHOR, 1994) apresenta uma enorme ameaça, podendo quebrar a força criptográfica de alguns algoritmos. Como, por exemplo, a troca de chaves do "*Transport Layer Security*" (TLS) (RFC 8446), usado em comunicação "*Hyper Text Transfer Protocol Secure*" (HTTPS), essas poderiam ser facilmente decriptadas por um computador quântico, quebrando o sistema de proteção do protocolo (RESCORLA, 2018; SIKERIDIS; KAMPANAKIS; DEVETSIKIOTIS, 2020b, 2020a). Além da criptografia de chaves públicas, a criptografia de chaves simétricas também sofre ameaça. O algoritmo de busca definido por Grover (GROVER, 1996) encontra um valor em uma lista desordenada em tempo potencialmente menor do que os melhores algoritmos de computação clássica, possibilitando ataques de força bruta que seriam impossíveis com computadores convencionais. A boa notícia, no segundo caso, é que o algoritmo de Grover não causa um impacto tão forte à segurança da criptografia simétrica como o algoritmo de Shor faz com a criptografia assimétrica, sendo que uma chave maior pode ser suficiente para manter o mesmo nível de segurança (YUNAKOVSKY *et al.*, 2021; MAVROEIDIS *et al.*, 2018; GILL *et al.*, 2022).

Ainda que em termos práticos a computação quântica não tenha chegado a um nível de ameaçar a segurança dos sistemas criptográficos, o melhor palpite é de que a ameaça se torne real dentro das próximas duas décadas (MOSCA; PIANI, 2022a; VOGT; FUNKE, 2021) e, por este motivo, estudar os impactos da computação quântica na criptografia atual é de suma importância. É estimado que mais de 60 porcento da internet use HTTPS baseado em TLS (CHAN *et al.*, 2018; SIKERIDIS; KAMPANAKIS; DEVETSIKIOTIS, 2020b). O estudo do tema agora permite que ações sejam tomadas mais cedo e ajuda a evitar que informações interceptadas hoje sejam decodificadas por agentes maliciosos daqui a alguns anos (MOSCA; PIANI, 2022a; VOGT; FUNKE, 2021).

1.2 OBJETIVOS

1.2.1 Objetivo Geral

Estudar a computação quântica como um novo paradigma da computação, suas implicações à segurança da informação, com enfoque nas ameaças ao TLS 1.3 e meios de mitigar as ameaças. Após isso, mapear as ferramentas existentes para realizar um ataque e experimentar com uma das ferramentas.

1.2.2 Objetivos Específicos

Para melhor realizar os objetivos propostos, este trabalho procura:

- apresentar uma revisão bibliográfica acerca da computação quântica e dos efeitos dela na segurança da informação;
- desenvolver um modelo de ameaça detalhado e delinear cenários específicos de ataque ao TLS 1.3;
- realizar um mapeamento sistemático da literatura para identificar ferramentas disponíveis para simular ou operar um computador quântico;
- realizar experimentos com a ferramenta escolhida, aplicando o algoritmo de Shor;
- avaliar os desafios técnicos e limitações práticas de um ataque realista utilizando ferramentas de computação quântica.

1.3 PROPOSTA

Para atingir os objetivos descritos na seção anterior, a proposta deste trabalho é segmentada nas seguintes etapas:

- realizar uma revisão narrativa da literatura sobre a computação quântica e suas ameaças à segurança da informação;
- propor um modelo de ameaça ao TLS 1.3 e então cenários de ataque ao protocolo;
- realizar um mapeamento sistemático da literatura à procura das ferramentas plausíveis de serem usadas para os fins deste trabalho;
- dedicar uma seção para explicar e experimentar uma ferramenta escolhida, executando o algoritmo de Shor nela;
- compilar os resultados obtidos e extrair deles um panorama geral sobre as ameaças da computação quântica à segurança da informação, especialmente ao TLS 1.3.

1.4 METODOLOGIA

Começando pela revisão narrativa da literatura, ela tem especificações bastante diferentes e bastante menos rigorosas do que uma revisão sistemática ou um mapeamento sistemático. Este tipo de revisão não segue um protocolo definido e a inclusão de artigos é baseada na experiência ou intuição dos autores, mas serve para uma revisão rápida e abrangente sobre um determinado assunto (PAE, 2015). O trabalho visa fazer uma revisão do tipo narrativa para entender e apresentar a computação

quântica e a ameaça que esse novo paradigma da computação traz à segurança de sistemas.

Com as informações e o conhecimento adquirido da revisão bibliográfica, um modelo de ameaça ao TLS 1.3 é elaborado, com cenários de ataque aos diversos modos de comunicação existentes no protocolo. Então uma nova pesquisa bibliográfica é feita buscando métodos existentes e pensando novos métodos de mitigar o problema.

Um trabalho sistemático segue regras mais rígidas do que uma revisão narrativa. O trabalho propõe um mapeamento sistemático da literatura com a intenção de encontrar as ferramentas que podem ser usadas para os objetivos deste trabalho. O mapeamento a se realizar estará baseado nas orientações de Kitchenham 2007 (KITCHENHAM; CHARTERS, 2007).

Após selecionar a ferramenta mais adequada, uma análise detalhada da ferramenta será apresentada e testes iniciais utilizando uma implementação prática do algoritmo de Shor serão realizados.

Ao final, trabalhos relacionados serão apresentados e os resultados dos capítulos anteriores serão revisitados na conclusão.

1.5 CONTRIBUIÇÃO PARA A CIÊNCIA DA COMPUTAÇÃO

O trabalho apresenta uma revisão bibliográfica sobre o tema. Então propõe vias pelas quais o TLS 1.3, um dos protocolos de segurança mais importantes da atualidade, pode ser vulnerável a atacantes quânticos, incluindo: um modelo de ameaça com possíveis atores e época esperada para que cada camada da ameaça seja realizada, visão detalhada de como os quatro modos de *handshake* do TLS 1.3 podem ser vulneráveis, requisitos necessários para que um ataque *Store-Now-Decrypt-Later* (SNDL) seja realizado, e métodos de mitigação do problema. Após isso, um mapeamento sistemático acerca das diversas ferramentas usadas para operar ou simular um computador quântico é feito com a intenção de complementar outros mapeamentos existentes e melhor escolher uma ferramenta para estudo. Então, o algoritmo de Shor é experimentado na ferramenta escolhida.

1.6 ESTRUTURA DA DISSERTAÇÃO

O capítulo 2 mostra uma revisão da literatura acerca dos temas relacionados à pesquisa. O capítulo 3 apresenta um mapeamento sistemático da literatura a fim de listar as ferramentas mais usadas e que são úteis ao estudo da criptologia. O capítulo 4 experimenta com a ferramenta selecionada usando o algoritmo de Shor. O capítulo 5 apresenta um modelo de ameaça e cenários de ataque ao TLS 1.3, então mostra métodos de mitigação das ameaças quânticas. O capítulo 6 é a conclusão do trabalho.

2 REVISÃO DA LITERATURA

2.1 ALGORITMOS CRIPTOGRÁFICOS

A comunicação entre dispositivos eletrônicos é uma marca da nossa época. Tecnologias como internet, mensagens instantâneas, compras online, bancos online, assinaturas digitais, videoconferências, são exemplos de tecnologias que nunca foram possíveis em eras anteriores. Para assegurar que essas comunicações sejam seguras, a confidencialidade, integridade e autenticidade, a criptografia, ou seja, o processo de proteger dados armazenados ou em trânsito, se faz extremamente necessária para o mundo moderno. Atualmente existem dois tipos básicos de criptosistemas: simétrico e assimétrico (MAVROEIDIS *et al.*, 2018). À luz da computação quântica, a criptografia assimétrica é a que corre maior risco, enquanto que a criptografia simétrica corre um risco menor, podendo ser evitado usando uma chave maior (MAVROEIDIS *et al.*, 2018; BERNSTEIN, D. J., 2009).

2.1.1 Criptografia simétrica

No tipo de criptografia simétrica, tanto quem envia quanto quem recebe a mensagem deve possuir a mesma chave e usar o mesmo algoritmo para encriptar e decriptar a mensagem. Esta chave deve ser mantida em segredo e, portanto, deve também haver um método seguro para transmitir esta chave pela rede. Dentre as vantagens desse modelo de criptografia estão a eficiência e a simplicidade de implementação. Dentre as desvantagens, a dificuldade de gerenciar e compartilhar as chaves são os principais exemplos.

Para exemplificar seu funcionamento, imagine que Bob encripta uma mensagem usando uma chave e envia a mensagem encriptada para Alice. Para que Alice consiga ler a mensagem, ela precisa usar o mesmo algoritmo e a mesma chave que Bob usou.

Alguns algoritmos de criptografia simétrica conhecidos são o "*Advanced Encryption Standard*" (AES), um dos algoritmos criptográficos mais utilizados atualmente, usado em diversas aplicações, de mensagens seguras a armazenamento de dados; o "*Data Encryption Standard*" (DES), um algoritmo muito importante no passado, mas atualmente é considerado inseguro devido ao tamanho curto de suas chaves; e o "*Triple Data Encryption Standard*" (3DES), uma versão mais segura do DES, aplicando-o triplamente, deixando-o mais seguro, mas ainda é lento em comparação com algoritmos mais modernos (MAVROEIDIS *et al.*, 2018; YUNAKOVSKY *et al.*, 2021).

2.1.2 Criptografia assimétrica

Esse tipo de criptografia, também conhecido como criptografia de chaves públicas, recebe esse nome, porque a encriptação ou decriptação ocorre diferente em

cada um dos lados. Geralmente, cada um dos integrantes tem um par de chaves, uma chave pública, que pode ser conhecida por outros, e uma chave privada, que deve ser mantida em segredo.

Caso Alice e Bob queiram trocar mensagens, as mensagens encriptadas com a chave pública de alguém, só podem ser decifradas com a chave privada da mesma pessoa. O contrário também funciona: as mensagens encriptadas com a chave privada de alguém, só podem ser decifradas com a chave pública da mesma pessoa. Por exemplo, se Bob quer mandar uma mensagem à Alice, ele primeiro pede a chave pública da Alice e envia a mensagem encriptada com a chave pública da Alice, então a mensagem só pode ser decifrada usando-se a chave privada da Alice.

A criptografia assimétrica é comumente usada para resolver o problema da troca de chaves, criando um canal seguro, permitindo que seja possível a comunicação em redes inseguras — como a internet. Porém, devido a seu maior custo computacional, a criptografia de chaves públicas não é comumente usada para trocas de mensagens, atuando principalmente como um canal seguro para a troca de chaves.

Além disso, este conceito de chaves públicas e privadas também são empregados nas assinaturas digitais. Neste caso, Alice assina digitalmente um documento com a chave privada dela e Bob pode verificar a assinatura com a chave pública de Alice.

A segurança de chaves assimétricas pertence à classe de problemas NP, que são verificáveis em tempo polinomial. Isso quer dizer que assinar documentos e verificar a assinatura usando a chave pública são problemas computacionalmente fáceis, mas descobrir a chave privada de alguém a partir de sua chave pública pode levar milhares de anos em computadores clássicos.

Problemas de fatoração de inteiros e de logaritmos discretos são problemas NP usados em criptossistemas modernos de chaves públicas, como o Rivest-Shamir-Adleman (RSA) e o Diffie-Hellman (DH).

O RSA explora o fato de ser difícil fatorar números compostos por primos, de modo que é computacionalmente inviável encontrar a chave privada a partir da chave pública de alguém.

DH e criptografia por cuvas elípticas (EC) exploram o problema de logaritmo discreto, o qual se refere a encontrar $r = \log_g x \mod p$ de $g^r = x \mod p$. Se os parâmetros forem grandes o suficiente, esse problema se torna muito difícil computacionalmente (MAVROEIDIS *et al.*, 2018; YUNAKOVSKY *et al.*, 2021).

2.1.3 TLS 1.3

O TLS 1.3 é um protocolo que assegura a comunicação segura entre pares, publicado na RFC 8446 (RESCORLA, 2018) de 2018 como uma versão melhorada em relação ao TLS 1.2 (RFC 5246). Essa nova versão não é apenas mais segura, mas também é mais rápida e simples do que a versão prévia. Algumas das atualizações

dessa versão em relação à 1.2, segundo o RFC 8446 (RESCORLA, 2018), são:

- algoritmos de criptografia simétrica considerados desatualizados foram eliminados;
- introduz o "*Zero Round Trip Time*" (0-RTT) que torna mais rápidas as conexões a páginas recentemente visitadas;
- remove *cipher suites* com Diffie-Hellman ou RSA estáticos (reutilizam chaves);
- após o ServerHello, todas as mensagens de *handshake* são encriptadas;
- derivação de chaves com o "*HMAC-based Extract-and-Expand Key Distribution Function*" (HKDF);
- algoritmos de curvas elípticas estão agora nas configurações básicas;
- novos algoritmos de assinatura digital (por exemplo, "*Edwards-curve Digital Signature Algorithm*" (EdDSA)) foram incluídos;
- o "*RSA Probabilistic Signature Scheme*" (RSASSA-PSS) substituiu o *RSA Padding*.

O *handshake* do TLS 1.3 garante que as partes estabeleçam a chave, negociem a versão do protocolo, selecionem os algoritmos criptográficos e, opcionalmente, autentiquem um ao outro. O *handshake* é dividido em três partes:

- **Troca de chaves:** estabelecimento de chaves e seleção de parâmetros criptográficos;
- **Parâmetros do servidor:** estabelecimento de parâmetros do *handshake*;
- **Autenticação:** autenticação do servidor (opcionalmente do cliente também), confirmação das chaves e integridade do *handshake*.

Detalhando um pouco melhor a troca de chaves, esta etapa é composta por uma mensagem `ClientHello` e uma mensagem `ServerHello`. A comunicação TLS 1.3 sempre começa com um `ClientHello`, o qual contém:

- a versão do protocolo;
- um número aleatório de 32 bytes;
- uma lista de cifras simétricas suportadas junto com algoritmos de hash para HKDF;
- a extensão `supported_groups` com os grupos (EC)DHEphemeral suportados e uma extensão `key_shares` com parâmetros criptográficos para cada um desses grupos;
- a extensão `signature_algorithms` com os algoritmos de assinatura aceitos pelo cliente e também pode ser acrescentada uma extensão chamada `signature_algorithms_cert` com algoritmos para certificados específicos;

- a extensão `pre_shared_key` com uma lista de chaves simétricas conhecidas pelo cliente e a extensão *psk_key_exchange_modes* que indica os modos de troca de chaves que devem ser usados com as "*Pre Shared Keys*" (PSKs).

Caso o servidor selecione uma PSK, ele também deve selecionar um dos modos de troca de chaves listado em `psk_key_exchange_modes`. Caso o servidor não selecione, o servidor é capaz de escolher uma cifra simétrica, um grupo (EC)DHE com a `key share` e um algoritmo de assinatura. Se, por algum motivo, o servidor escolher um grupo (EC)DHE e o cliente não tiver enviado uma extensão `key_share` compatível, o servidor responde com uma mensagem `HelloRetryRequest`.

Uma visão mais detalhada sobre curvas elípticas e Diffie-Hellman se encontra nos apêndices A, B e C.

O `ServerHello` define os parâmetros criptográficos adequados de acordo com as informações contidas no `ClientHello`. Se o servidor for bem sucedido em selecionar todos os parâmetros, o `ServerHello` deve conter:

- se estiver usando PSK, acrescenta uma extensão `pre_shared_key` indicando o valor selecionado. Se não estiver usando PSK, (EC)DHE e autenticação baseada em certificado devem ser utilizadas;
- se usando (EC)DHE, acrescenta uma extensão `key_share`;
- se autenticando via certificado, envia ambas as mensagens: `Certificate` e `CertificateVerify`. — No TLS 1.3, sempre um PSK ou um certificado devem ser usados, mas não ambos.

Se o servidor não conseguir negociar um conjunto de parâmetros, ele deve abortar o *handshake* com os alertas `handshake_failure` ou `insufficient_security`.

2.1.3.1 Modos de *handshake* do TLS 1.3

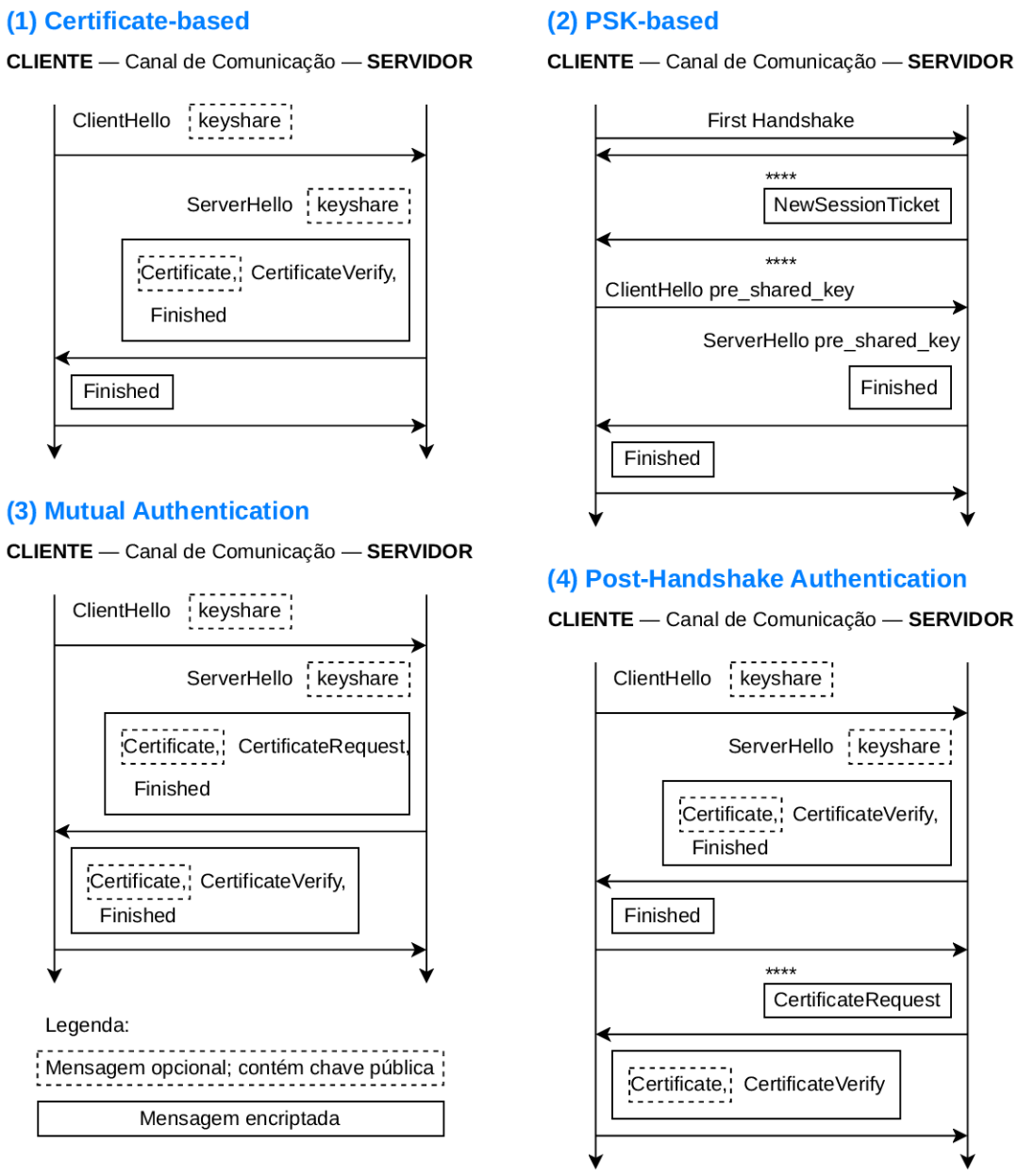
Há quatro modos de *handshake* básicos no TLS 1.3 exemplificados na Figura 1. Eles são os seguintes:

- `Certificate-based mode` (1): a autenticação do servidor acontece com a mensagem de `ServerHello` e não é requisitada a autenticação por parte do cliente. Este é o mais comum dentre os modos;
- `PSK mode` (2): o TLS aproveita as informações de um *handshake* anterior, `First Handshake`, para uma autenticação mais rápida. Neste caso, o TLS reutiliza as chaves trocadas anteriormente e o servidor gera um novo *ticket* de seção, `NewSessionTicket`, contendo uma PSK que o cliente deverá usar em futuros *handshakes*;
- `Mutual authentication` (3): o processo do *handshake* é similar ao do modo (1), com a diferença de que o cliente é convidado a se autenticar após o

servidor. Durante a autenticação do servidor, ele envia uma requisição de autenticação ao cliente por meio da mensagem `CertificateRequest`;

- `Post-handshake authentication` (4): o servidor se autentica durante o *handshake*, de maneira similar à do modo (1) e, apenas após a mensagem de `Finished`, o servidor pede para que o cliente se autentique também, enviando uma mensagem de `CertificateRequest`.

Figura 1 – Diferentes modos de *handshake* do TLS 1.3.



Fonte – (MOURA; GIRON; CUSTODIO, 2023)

2.2 O COMPUTADOR QUÂNTICO

Ao que tudo indica, Feynman, em 1982, foi o primeiro a imaginar a computação quântica, argumentando que a natureza é quântica, e que simulá-la na arquitetura de von Neumann seria extremamente custoso (FEYNMAN, 1982). Outros trabalhos pioneiros importantes no assunto que relaciona mecânica quântica e computação, são: Benioff, 1982, que mostrou que é possível simular uma máquina de Turing por meio de processos quânticos (BENIOFF, 1982); e Deutsch, 1985 e 1989, que deu início a modelos de computação quântica, estudando máquinas de Turing quânticas e circuitos quânticos (DEUTSCH, D., 1985; DEUTSCH, D. E., 1989).

Mecânica quântica diz respeito a objetos microscópicos, de escala subatômica, que apresentam um comportamento diferente daquele esperado pela física clássica. Computadores quânticos utilizam qubits, que podem estar no estado 0, 1, ou em uma superposição de ambos, ao contrário dos bits clássicos que são binários (MAVROEIDIS *et al.*, 2018).

Bits são a menor porção de informação não probabilística, representando qualquer sistema de dois estados. Portanto, bits possuem um caráter discreto, mesmo que a natureza da física clássica seja contínua. *Quantum* diz respeito a algo com característica discreta. Desse modo, a computação (de caráter essencialmente discreto) é mais naturalmente explicada por uma teoria quântica da informação do que por uma teoria clássica (DEUTSCH, D. E., 1989).

O qubit é a unidade de informação quântica e pode ser representado por $|\psi\rangle$[1]. O estado de um determinado qubit é um vetor em um espaço de Hilbert bidimensional. Um qubit pode estar em um dos estados $|0\rangle$, $|1\rangle$ ou em uma superposição destes estados, ou seja, em uma combinação linear destes estados, representada pela equação (1).

$$|\psi\rangle = a\,|0\rangle + b\,|1\rangle \tag{1}$$

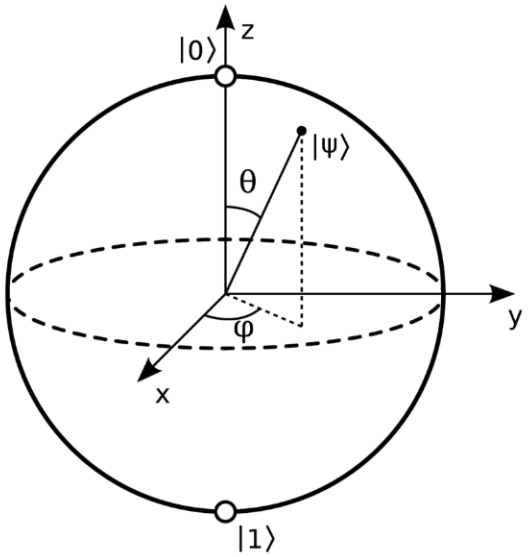
Quando um qubit é mensurado, temos uma probabilidade $|a|^2$ de encontrá-lo no estado $|0\rangle$ e $|b|^2$ de encontrá-lo no estado $|1\rangle$. Ao que chamamos a, b de amplitudes de probabilidade. E as probabilidades são normalizadas como mostra a equação (2).

$$1 = |a|^2 + |b|^2 \tag{2}$$

Um qubit também pode ser representado por um vetor na esfera de Bloch (a Figura 2 mostra um qubit representado na esfera de Bloch) após certas manipulações matemáticas que resultam na equação (3). A demonstração se encontra no apêndice I.

[1] Sendo **v** um vetor num espaço vetorial complexo, o ket $|v\rangle$ representa este vetor e o bra $\langle v|$ representa seu complexo conjugado. Esta notação é conhecida como bra-ket ou notação de Dirac, útil em álgebra linear. Nela também vale o produto interno $\langle v|v\rangle$ ou $\langle u|v\rangle$.

Figura 2 – Representação de um vetor (qubit) na esfera de Bloch.



Fonte – (HAGOUEL; KARAFYLLIDIS, 2012)

$$|\psi\rangle = \cos\frac{\varphi}{2}|0\rangle + e^{i\theta}\sin\frac{\varphi}{2}|1\rangle \tag{3}$$

Fisicamente, um qubit pode representar um sistema de dois estados quânticos. Os mais estudados são o spin de um elétron ou a polarização de um fóton (HAGOUEL; KARAFYLLIDIS, 2012).

Após ser mensurado, o qubit (que pode ser interpretado como o spin de um elétron ou um fóton polarizado) colapsa para um dos estados 0 ou 1 (ou seja, para o estado base $|0\rangle$ ou para o estado excitado $|1\rangle$). Além disso, dois qubits podem estar emaranhados. Isso quer dizer que o estado de um deles não pode ser interpretado de maneira independente do outro, mas ambos podem ser interpretados como um único objeto de quatro estados. Este efeito de emaranhamento não depende da distância, podendo afastar ambas partículas anos-luz de distância e elas continuariam emaranhadas. Estas propriedades tornam o computador quântico uma verdadeira máquina de processamento paralelo, sendo que n qubits podem processar 2^n operações paralelamente (MAVROEIDIS *et al.*, 2018). Em outras palavras, uma superposição de n qubits pode existir em todas as combinações de $|000...0\rangle$ a $|111...1\rangle$, ou seja, o número S de estados de superposição é representado pela equação (4).

$$S = 2.2.2.....2 = 2^n \tag{4}$$

Portanto, a superposição de estados quânticos cresce exponencialmente (KHAN, T. M.; ROBLES-KELLY, 2020).

Um registrador quântico é um sistema físico que contém dois ou mais qubits e um computador quântico armazena seus dados em um ou mais registradores quânticos. O estado de um registrador quântico é um vetor em um espaço de Hilbert multidimensional, representado pelo produto tensorial dos estados dos qubits que o compõem. Os registradores clássicos são conceitualmente diferentes dos registradores quânticos. Um registrador clássico de tamanho n é um vetor de n 0s ou 1s; um registrador quântico de tamanho n é um vetor de n qubits (DEUTSCH, D. E., 1989; KHAN, T. M.; ROBLES-KELLY, 2020). O Quadro 1 compara registradores clássicos e quânticos.

Quadro 1 – Registradores clássicos vs registradores quânticos.

Registrador de n bits	**Registrador de n qubits**
2^n estados possíveis (por vez)	2^n estados possíveis (em paralelo)
Avaliável	Parcialmente avaliável
Cópias independentes	Impossível copiar independentemente
Individualmente apagável	Impossível apagar individualmente
Leitura não destrutível	Leitura modifica o valor atual
Determinístico	Probabilístico

Fonte – (KHAN, T. M.; ROBLES-KELLY, 2020)

Uma porta lógica é uma máquina de computar que recebe um número fixo de bits, realiza uma computação e tem um número fixo de bits como saída (DEUTSCH, D. E., 1989). Na ciência da computação clássica, um conjunto de portas lógicas são agrupadas para formar um circuito digital. Portas lógicas quânticas também computam uma saída que depende de uma entrada, mas essa entrada é geralmente um estado de superposição. Portas lógicas quânticas são a transformação de um ou mais qubits; são operadores do espaço de Hilbert que rotacionam o vetor de estados de qubits ou registradores. Matematicamente são uma matriz unitária e aplica-se uma porta lógica a um qubit multiplicando-se a matriz pelo vetor de estado (HAGOUEL; KARAFYLLIDIS, 2012; KHAN, T. M.; ROBLES-KELLY, 2020; ROY, 2020). Seja $|\psi_0\rangle$ o estado inicial de um qubit ou registrador quântico e $|\psi_1\rangle$ seu estado final, uma porta lógica quântica atua de acordo com a equação (5).

$$G|\psi_0\rangle = |\psi_1\rangle \tag{5}$$

No modelo de circuito quântico, o estado inicial dos registradores são a entrada da computação, uma sequência de passos computacionais — sendo que cada passo computacional é um número de portas quânticas operando sobre os registradores — é executada e o estado final dos registradores é mensurado, revelando o resultado da computação (HAGOUEL; KARAFYLLIDIS, 2012). De modo que, ao final do n-ésimo passo, o estado do registrador $|q\rangle$ será representado de acordo com a equação (6).

$$|q_n\rangle = G_n|q_{n-1}\rangle \tag{6}$$

Por estar fundamentado na física quântica, a computação ocorre percorrendo todas as possibilidades de caminhos simultaneamente e cada caminho tem uma amplitude complexa associada. Somando as amplitudes dos caminhos que convergem a um final e então tirando a raiz quadrada do valor absoluto, podemos calcular a probabilidade de que este seja um estado final da máquina (SHOR, 1994).

Os computadores quânticos podem ser de uso universal ou de uso não-universal. A diferença entre estes dois tipos é que computadores universais resolvem qualquer tipo de problema, enquanto que computadores não-universais resolvem apenas problemas de uma determinada natureza (MAVROEIDIS *et al.*, 2018).

O rápido desenvolvimento da computação quântica fará com que, dentro de pouco tempo, mesmo o melhor supercomputador clássico seja superado em poder computacional. Aumentando linearmente o número de qubits, o espaço computacional de um computador quântico cresce exponencialmente (SOEKEN; HAENER; ROETTELER, 2018).

Contudo, no estado atual, os computadores quânticos reias ainda apresentam alguns desafios. Por exemplo, eles trabalham de forma probabilística. Isso quer dizer que levantam várias soluções para um problema onde apenas uma é a solução ótima. Um problema importante é que são muito sensíveis ao calor e a perturbações no ambiente, tendo que operar a temperaturas de quase zero Kelvin e qualquer perturbação pode causar um *bit-flip* ou erro de fase. Também, seus qubits permanecem em seus estados quânticos por um curto período de tempo (MAVROEIDIS *et al.*, 2018). À perda de coerência dos estados quânticos dá-se o nome de decoerência e este é um dos maiores desafios ao construir computadores quânticos confiáveis. Em geral, a decoerência não causa imprecisão dos resultados de um computador clássico, mas, em computadores quânticos, a decoerência pode causar uma perda de informação irreversível nos estados de superposição e os erros em uma porta quântica se acumulam em um circuito quântico maior. A implicação disso é que qubits em superposição iriam se comportar como bits clássicos, perdendo qualquer vantagem quântica. Computadores que levam em conta as perturbações externas são chamados "*Noisy Intermediate Scale Quantum*" (NISQ) (PRESKILL, 2012; GILL *et al.*, 2022). Nesta definição, *noisy* se refere aos erros em suas portas lógicas e problemas de decoerência de seus estados, e *intermediate scale* se refere ao número de qubits físicos (BERTELS; SARKAR; ASHRAF, 2021).

2.3 A AMEAÇA QUÂNTICA

Como recém discutido, a segurança da informação, falando, neste caso, principalmente dos modelos de criptossistemas que envolvem chaves públicas e chaves privadas, baseia-se na classe de problemas NP, assumindo que não há algoritmos para resolver esses problemas matemáticos em um tempo razoável. No entanto, a

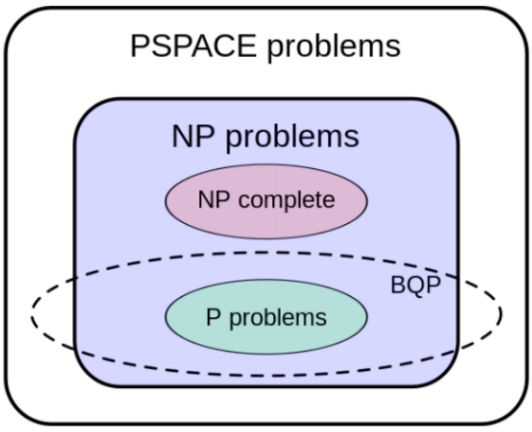
verificação desses problemas pode ser feita em tempo polinomial.

Contudo, hoje fala-se de uma nova geração de computadores e de um novo paradigma que não envolve mais diodos e transistores, mas que está baseada em conceitos da física moderna para fazer computação. Esta nova geração de computadores põe em risco diversos aspectos da criptografia moderna, elevando o poder de criptoanálise a um segundo patamar, no qual alguns problemas NP relevantes podem se resolvidos em tempo polinomial (YUNAKOVSKY *et al.*, 2021; VOGT; FUNKE, 2021).

Acredita-se que computadores quânticos podem resolver em tempo polinomial qualquer problema que um computador clássico leve um tempo polinomial para resolver (AARONSON, 2008). Isso quer dizer que, teoricamente, computadores quânticos podem resolver qualquer problema da classe "*Polinomial Time*" (P) em tempo polinomial. Contudo, apesar de computadores quânticos possuírem a capacidade teórica de resolver alguns problemas NP em tempo polinomial, isso não se aplica a qualquer problema da classe NP (AARONSON, 2008). Por conseguinte, os computadores quânticos não conseguem resolver problemas NP-Completo em tempo polinomial, pois essa classe de problemas é formada por problemas $x \in NP$ tal que qualquer problema NP possa ser reduzido a x em tempo polinomial.

A classe "*Bounded Error, Quantum, Polinomial Time*" (BQP), que corresponde à classe de problemas que um computador quântico pode decidir em tempo polinomial e com uma probabilidade pequena de erros (BERNSTEIN, E.; VAZIRANI, 1997), engloba a classe P, alguns problemas NP e alguns poucos problemas PSPACE. Esta última classe, PSPACE, refere-se a problemas que requerem uma quantidade polinomial de memória, mas uma quantidade exponencial de passos de computação para um computador clássico (AARONSON, 2008). A Figura 3 mostra a classe BQP em comparação às classes anteriormente citadas.

Figura 3 – A classe BQP.



Fonte – (AARONSON, 2008)

A ciência da criptografia ocupa um papel fundamental em toda comunicação ou transação eletrônica, desde trocas de e-mails, mensagens, movimentações financeiras, trocas de serviços, voto eletrônico ou até mesmo armazenamento em nuvem, para citar alguns. Em todos estes exemplos é importante que as informações estejam asseguradas pela confidencialidade, integridade, autenticidade e pelo não-repúdio. Respectivamente, confidencialidade quer dizer que as informações só devem ser vistas por usuários com os devidos privilégios, integridade quer dizer que as informações não devem ser alteradas ou corrompidas, autenticidade quer dizer que em ambas as partes os usuários devem ser exatamente quem dizem ser e, o não repúdio quer dizer que, caso haja falha na segurança, o responsável pelo problema deve ser identificado. Para garantir estas propriedades, a criptografia deve garantir que a troca de chaves seja feita de maneira segura e que apenas usuários que participaram da troca de chaves sejam capazes de ler a mensagem encriptada (MAVROEIDIS *et al.*, 2018).

Computadores quânticos são capazes de uma agilidade matemática que não é possível em computadores clássicos. Os sistemas criptográficos atuais, em grande maioria, não preveem esse poder computacional, consequentemente, computadores quânticos podem quebrar chaves criptográficas mais rapidamente, seja por cálculo ou busca exaustiva (*brute force*) dentre diversas possibilidades. Assim, permitindo que um terceiro indivíduo se coloque entre o remetente e o destinatário (*man-in-the-middle*) e seja capaz de interceptar a comunicação (MAVROEIDIS *et al.*, 2018) ou permitindo que dados coletados hoje sejam decifrados e lidos em um futuro breve (VOGT; FUNKE, 2021), ataque conhecido por SNDL. De acordo com (CHEN *et al.*, 2016), os computadores quânticos trarão o fim dos esquemas atuais de criptografia de chave pública. O Quadro 2 traz o impacto da computação quântica sobre alguns algoritmos de segurança mais conhecidos.

Quadro 2 – Impacto da computação quântica em alguns dos algoritmos criptográficos mais comuns.

Algoritmo	**Tipo**	**Propósito**	**Impacto**
AES-256	Simétrico	Criptografia	Chave maior
SHA-256, SHA-3	-	Hash	Saída maior
RSA	Assimétrico	Assin., troca chaves	Inseguro
ECDSA, ECDH	Assimétrico	Assin., troca chaves	Inseguro

Fonte – (CHEN *et al.*, 2016)

De acordo com (VOGT; FUNKE, 2021), o algoritmo de Shor quebra completamente a criptografia assimétrica, enquanto que o algoritmo de Grover, que apresenta alguma ameaça à criptografia simétrica, faz necessário apenas uma chave com o dobro do tamanho. A Tabela 1 compara os bits de segurança de diferentes algoritmos criptográficos para computadores clássicos com os bits de segurança aparentes para

computadores quânticos.

Tabela 1 – Impacto da computação quântica nos bits de segurança.

Tipo	Algoritmo	Bits	Bits aparentes	Ataque
Assimétrico	RSA 2048	112	0	Shor
Assimétrico	RSA 3072	128	0	Shor
Assimétrico	seqp256rl	128	0	Shor
Assimétrico	seqp521rl	256	0	Shor
Simétrico	AES 128	128	64	Grover
Simétrico	AES 256	256	128	Grover

Fonte – (VOGT; FUNKE, 2021)

Mais adiante serão discutidos alguns algoritmos quânticos mais a fundo.

2.3.1 Algoritmo de Shor

O algoritmo de Shor foi desenvolvido em 1994 pelo matemático Peter Shor, no artigo "*Algorithms for quantum computation: discrete logarithms and factoring*" (SHOR, 1994). Este é o algoritmo de criptoanálise quântica mais discutido atualmente. Neste artigo, é demonstrado que a fatoração de números grandes mudará profundamente com os computadores quânticos (MAVROEIDIS *et al.*, 2018).

Como apresentado na seção anterior, os sistemas criptográficos de chave pública (criptografia assimétrica) baseiam-se fundamentalmente na fatoração de números grandes e logaritmos discretos. Portanto, o algoritmo de Shor pode causar o colapso da criptografia assimétrica como a conhecemos. Em especial, o RSA e o DH. Na prática, um agente malicioso em poder de um computador quântico poderia extrair a chave privada de um usuário a partir de sua chave pública, por exemplo (MAVROEIDIS *et al.*, 2018; YUNAKOVSKY *et al.*, 2021).

Dentre os protocolos mais ameaçados pela computação quântica está o TLS, padronizado na RFC 8446 (RESCORLA, 2018). É estimado que mais de 60 porcento da internet use HTTPS baseado em TLS (CHAN *et al.*, 2018; SIKERIDIS; KAMPANAKIS; DEVETSIKIOTIS, 2020b). No TLS 1.3 (atual versão), o *handshake* usa o Diffie-Helman efêmero para trocas de chaves e promove autenticação com o uso de certificados X.509 (SCHWABE; STEBILA; WIGGERS, 2020; SIKERIDIS; KAMPANAKIS; DEVETSIKIOTIS, 2020a, 2020b).

O algoritmo de Shor oferece uma vantagem exponencial na fatoração de inteiros e no cálculo de logaritmos discretos em comparação aos algoritmos clássicos, tornando-se uma grande ameaça à criptografia de chaves públicas. No entanto, há ainda alguns desafios para a sua perfeita operação. A Tabela 2 mostra algumas implementações do algoritmo de Shor e seus respectivos requerimentos de *hardware*.

Tabela 2 – Requerimentos de *hardware* quântico para diferentes implementações do algoritmo de Shor.

Implementação	**Número de qubits**	**Número de portas**
(SHOR, 1994)	$\Theta(n)$	$\Theta(n^3 log(n))$
(BECKMAN *et al.*, 1996)	$5n+1$	$\Theta(n^3)$
(VEDRAL; BARENCO; EKERT, A., 1996)	$4n+3$	$\Theta(n^3)$
(BEAUREGARD, 2002)	$2n+3$	$\Theta(n^3 log(\frac{n}{\xi}) log(\frac{1}{\xi}))$
(TAKAHASHI; KUNIHIRO, 2006)	$2n+2$	$\Theta(n^3 log(\frac{n}{\xi}) log(\frac{1}{\xi}))$
(HÄNER; ROETTELER; SVORE, Krysta M, 2016)	$2n+2$	$\Theta(n^3 log(n))$
(GIDNEY, 2017)	$2n+1$	$\Theta(n^3 log(n))$

Fonte – (SUO *et al.*, 2020)

Se, por exemplo, o objetivo for quebrar o RSA-2048 usando a implementação mais recente do algoritmo dentre as apresentadas, seriam necessários $2n+1 = 4097$ qubits e aproximadamente $n^3 log(n) \approx 9 * 10^{10}$ portas lógicas. Mas os qubits no estado atual não são perfeitos e a computação apresenta erros que devem ser contornados para obter um resultado confiável. Segundo (GIDNEY; EKERÅ, 2021), esses erros podem ser contornados executando os cálculos diversas vezes ou combinando cerca de 1548 qubits para simular um único qubit lógico perfeito. Além disso, há o problema do tempo de coerência dos qubits e o tempo de cada porta lógica, que depende da tecnologia utilizada. Segundo (SUCHARA *et al.*, 2013), computadores quânticos utilizando a tecnologia de supercondutores têm um tempo de porta lógica médio de 25 *ns*, computadores construídos sobre a tecnologia de átomos neutros possuem tempo médio de 19 μs, e computadores utilizando íons presos têm tempo médio de 32 μs. É importante que o tempo de computação não supere o tempo de coerência.

Algumas novas implementações do algoritmo de Shor ampliam sua usabilidade para problemas de logaritmos discretos com curvas elípticas, do inglês "*Elliptic-Curve Discrete Logarithm Problem*" (ECDLP). Problema bastante utilizado em criptografia de chaves públicas, em especial, no TLS 1.3. Mas também há implementações em computação adiabática para os problemas de fatoração de inteiros e de logaritmos discretos que vêm mostrando ótimos resultados. Para o problema da fatoração de inteiros, podemos citar as implementações (LI, Z. *et al.*, 2017; JIANG *et al.*, 2018; PENG *et al.*, 2019); já para o problema de logaritmos discretos, podemos citar (WROŃSKI, 2022). A Tabela 3 apresenta os melhores resultados na fatoração de inteiros alcançados pelo algoritmo de Shor e por implementações adiabáticas.

Tabela 3 – Melhores realizações na fatoração de inteiros.

Ano	Largura da chave	Algoritmo
2001	4 bits	Shor
2012	5 bits	Shor
2012	16 bits	Adiabático
2016	18 bits	Adiabático
2018	19 bits	Adiabático
2019	20 bits	Adiabático
2020	41 bits	Adiabático

Fonte – (SUO *et al.*, 2020; PETRENKO, 2023; RUNGE, 2023)

Diferentemente do modelo de computação quântica baseado em circuitos, o modelo de computação quântica adiabática, apesar de ser mais fácil de ser construído, não é um modelo universal. Em vez disso, a computação quântica adiabática é especializada em resolver problemas específicos, como o problema de minimização. Ela faz isso preparando um registrador quântico inicial e evoluindo seus microestados de forma adiabaticamente lenta, chegando o mais próximo possível do estado fundamental do sistema (LAUMANN *et al.*, 2015; SOUZA *et al.*, 2021; YARKONI *et al.*, 2022).

2.3.2 Algoritmo de Grover

O algoritmo de Grover foi criado em 1996 por Lov Grover (GROVER, 1996). Este algoritmo usa um computador quântico para procurar em uma lista desordenada de tamanho N e faz isso em $\sqrt{N}$ buscas (um computador convencional poderia levar o tempo de N/2 buscas) (MAVROEIDIS *et al.*, 2018; BONE; CASTRO, 1997). Segundo (MINA-ZICU; SIMION, 2020), o algoritmo de Grover pode ser usado para busca num espaço de chaves e acrescenta que, uma vez que o AES é vulnerável a ataques de força bruta (*brute-force*), é também vulnerável ao algoritmo de Grover. Considerando uma chave AES de n bits, o espaço de chaves é $N = 2^n$ e o algoritmo de Grover necessitaria de $\sqrt{N} = 2^{n/2}$ buscas até encontrar a chave válida (um computador clássico necessitaria $N/2 = 2^{n-1}$ buscas).

O algoritmo de Grover também pode procurar por colisões em uma função de hash, uma vez que sua segurança depende de uma saída de tamanho fixo (MAVROEIDIS *et al.*, 2018).

Também é possível rodar variantes desse algoritmo em computadores adiabáticos, como em (HEN, 2017).

2.3.3 Algoritmo de Simon

O algoritmo de Simon é um outro algoritmo para encontrar período de funções, assim como o algoritmo de Shor. Ele é importante para a história da computação

quântica por ser o primeiro algoritmo quântico a apresentar um ganho exponencial em relação a algoritmos clássicos projetados para resolver o mesmo problema e também por servir de inspiração a outros algoritmos quânticos, assim como o próprio algoritmo de Shor, baseados na Transformada de Fourier Quântica, do inglês "*Quantum Fourier Transform*" (QFT). Pode ser poderoso para a realização de *slide attacks*, uma ameaça a redes Feistel e a modos de encriptação como "*Cipher Block Chaining Message Authentication Code*" (CBC-MAC), "*Galois/Counter Mode*" (GCM), "*Galois Message Authentication Code*" (GMAC), "*Parallelizable Message Authentication Code*" (PMAC) e "*Offset Codebook mode*" (OCB) (KUWAKADO; MORII, 2010; KAPLAN *et al.*, 2016; SANTOLI; SCHAFFNER, 2016; DONG; WANG, X., 2018; SHI *et al.*, 2019; SUO *et al.*, 2020; PETRENKO, 2023).

Uma outra implementação para atacar cifras de bloco, desta vez usando um computador adiabático, pode ser encontrada em (BUREK *et al.*, 2022).

2.4 OUTROS TRABALHOS RELACIONADOS

No que tange a exploração da ameaça da computação quântica à segurança de sistemas de informação, dois outros trabalhos foram profundamente analisados para a produção deste. O primeiro (MAVROEIDIS *et al.*, 2018) apresenta um panorama da computação quântica, focado nos algoritmos de Shor e Grover, apresentando suas ameaças às criptografias simétrica e assimétrica. Por fim, mostra interessantes soluções ao problema na era pós-quântica e conclui que a computação ameaça um colapso à segurança eletrônica, por isso deve-se tomar de artifícios como a "*Quantum Key Distribution*" (QKD) e a "*Post-Quantum Cryptography*" (PQC).

O segundo trabalho (RUNGE, 2023) traz um panorama ainda maior da computação quântica e algoritmos de criptoanálise ao incluir os feitos da computação adiabática e curvas de evolução da tecnologia. Mais adiante ele propõe atores de ameaça, explora a ameaça a veículos elétricos e apresenta formas de mitigação do problema. Por fim, conclui que a ameaça da computação quântica não será realizada de um dia pro outro, mas de maneira gradual.

A respeito do mapeamento sistemático acerca das ferramentas de computação quântica, também dois estudos trabalharam de forma parecida, mas com objetivos diferentes. O primeiro (JUNIOR; CAMARGO, 2021) é um mapeamento sistemático também baseado em (KITCHENHAM; CHARTERS, 2007), mas com o objetivo de revelar as diferenças na engenharia de *software* de um *hardware* clássico para um *hardware* quântico. As principais diferenças são (1) o foco na engenharia de *software*, (2) não incluir *preprints* ou patentes, (3) a quantidade de ferramentas mapeadas e materiais selecionados são muito inferior.

No segundo caso (GILL *et al.*, 2022), o trabalho não é um mapeamento sistemático, pois não segue a estrutura de um, mas é um *survey* listando diversas ferramentas

de computação quântica. O objetivo geral do artigo é explorar desafios em aberto e possibilidades de trabalhos futuros. As principais diferenças entre aquele trabalho e o mapeamento deste são (1) a falta de regras claras que possam ser replicadas, (2) foco muito amplo, tornando difícil saber quais das ferramentas seriam úteis para a área de pesquisa deste trabalho.

3 FERRAMENTAS DE COMPUTAÇÃO QUÂNTICA

3.1 OBJETIVO E QUESTÕES DE PESQUISA

3.1.1 Objetivo

Além de um modelo de ameaça e cenários de ataque ao TLS 1.3, esse trabalho tem um segundo objetivo de explorar ferramentas com as quais um atacante poderia operar um computador quântico e experimentar uma dessas ferramentas para entender melhor o estado da arte, assim como os desafios que um atacante quântico enfrentaria durante o processo.

3.1.2 Questões de pesquisa

- QP1. Quais são os *softwares* ou ferramentas disponíveis para simular ou operar um computador quântico real no contexto da criptologia e quais são mencionados mais vezes?
- QP2. Considerando os materiais selecionados, é possível levantar quais características de cada ferramenta?
- QP3. Considerando os materiais incluídos, é possível concluir que houve um aumento de materiais publicados na área desde 2020?
- QP4. Há um padrão observável com respeito aos locais evolvidos nas publicações?

3.2 FONTES DE PESQUISA E *STRING* DE BUSCA

3.2.1 Fontes de pesquisa

A área de computação quântica é essencialmente interdisciplinar. Por este motivo, foi escolhida uma base de dados que represente a ciência da computação, uma base de dados que represente a física, uma base de dados para as engenharias e uma base de dados ampla:

- **ArXiv**: Esta base representa bem a população de materiais das áreas de física, matemática, computação e engenharia elétrica. Outro motivo para a seleção é a busca por *preprints*.
- **ACM Digital Library**: Esta base é uma das mais usadas para ciência da computação e também usada para física computacional.
- **IEEE Xplore**: Base de dados bastante usada em engenharia e computação.
- **Google *Scholar***: Uma base bastante ampla de dados, onde também é possível pesquisar por patentes, artigos de conferência e trabalhos acadêmicos. Inclui resultados de praticamente qualquer área do conhecimento.

3.2.2 *String* de busca

A *string* de busca foi feita com base nos termos usados para se referir a ferramentas de *software* utilizadas para simular ou operar computadores quânticos e um termo de filtro, *"quantum computer"*: **("quantum computer software"OR "quantum computing software"OR "quantum simulation software"OR "quantum simulation tool"OR "quantum computing simulator"OR "quantum computer simulator"OR "software for quantum"OR "quantum software"OR "quantum tool"OR "tool for quantum") AND "quantum computer"**.

3.3 CRITÉRIOS DE INCLUSÃO E EXCLUSÃO

O objetivo do trabalho é encontrar e classificar os *softwares* ou ferramentas para computação quântica. É esperado que parte das ferramentas existentes venham na forma de *preprints*, patentes e nem sempre na forma de artigos. Desde 1982, alguns conceitos e ferramentas foram atualizados, por isso é importante saber o que é tendência atualmente, então apenas materiais publicados de 2020 em diante serão aceitos. Para ser incluído na pesquisa:

- CI1. O material deve apontar, comentar ou discutir sobre uma ferramenta que sirva para simular ou operar um computador quântico universal e também apresentar algum detalhe da ferramenta.
- CI2. O material deve ser um documento oficial, como artigo, patente, conferência, livro, tese, dissertação, trabalho de conclusão e até *preprints*.
- CI3. O material deve ter sido escrito de 2020 em diante.
- CI4. Qualquer país de publicação será permitido, mas apenas textos em inglês, português, espanhol ou italiano. Apesar de ser possível a tradução, perde-se conteúdo.
- CI5. Material não repetido.
- CI6. A ferramenta é útil à área da criptologia e o texto traz indícios para isto, deixando evidente que pode auxiliar ou rodar um algoritmo útil à criptologia.

3.4 PROCEDIMENTO DE SELEÇÃO

O procedimento de seleção seguirá alguns passos, listados a seguir:

1. Aplicar a *string* de busca em cada uma das bases.

   Levantar o número inicial de trabalhos.

2. Retirar materiais repetidos (CI5).

   Levantar o número de trabalhos não repetidos.

3. Primeira iteração: Aplicar os critérios de inclusão e exclusão no título, resumo e palavras-chave de maneira preliminar. Caso o material não tenha resumo, apenas o título será examinado.

   Levantar o número de trabalhos até este ponto.

4. Segunda iteração: Examinar o texto integralmente, aplicando os critérios de inclusão e exclusão.

   Levantar o número final de trabalhos.

5. Extrair as informações dos trabalhos selecionados.

3.5 APLICANDO A *STRING* DE BUSCA

No mecanismo de busca da ArXiv a *string* de busca foi aplicada em todos os campos do texto (*all fields*). As áreas selecionadas (*subjects*) foram física (todas), ciência da computação, engenharia elétrica e de sistemas, matemática. A data foi selecionada respeitando o critério CI3, retornando textos de 2020 ou mais recentes.

Na ACM, a *string* de busca também foi aplicada em qualquer lugar do texto (*anywhere*), obedece CI3 e nenhum filtro extra foi aplicado.

Usando o motor de busca da IEEE Xplore, a *string* foi aplicada no texto completo (*full text only*) e a data dos resultados também obedecem o CI3.

O Google *Scholar* não tem uma ferramenta de busca tão precisa quanto as três anteriores, no entanto, tem uma busca mais ampla. A ferramenta tem um limite de 256 caracteres em suas buscas e por esta razão a *string* de busca precisou ser alterada especialmente para este *site*. A *string* usada foi: **quantum AND ("computing simulator"OR "computer simulator"OR software OR tool)**. 174 mil resultados retornaram, então a *string* foi aplicada apenas no título para filtrar melhor esse número. Patentes também foram aceitas, a data dos materiais obedecem o CI3 e não foram incluídas citações.

A Tabela 4 mostra o levantamento inicial de resultados por base de dados e a soma total destes.

Tabela 4 – Número inicial de artigos por base de dados.

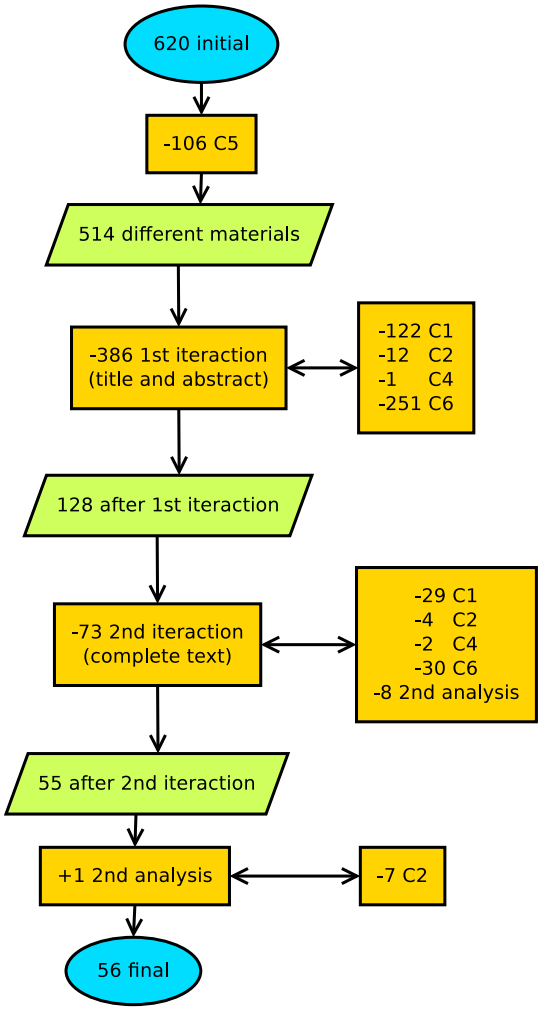
Base	**Número**
ArXiv	135
ACM	60
IEEE Xplore	71
Google *Scholar*	354
	620

Fonte – Elaborado pelo autor.

3.6 APLICANDO OS CRITÉRIOS DE SELEÇÃO

O critério CI3 já foi aplicado nos mecanismos de busca. Com os 620 materiais iniciais, 106 eram repetidos (CI5), restando 514 materiais. Após isso, 386 foram excluídos na primeira iteração, restando 128 materiais. 73 foram excluídos na segunda iteração, restando 55 materiais. Haviam 8 trabalhos que não estavam acessíveis na primeira análise e foram analisados separadamente. Nesta segunda análise, um foi aceito e 7 foram excluídos. O processo detalhado pode ser verificado na Figura 4.

Figura 4 – Fluxograma dos trabalhos selecionados.



Fonte – Elaborado pelo autor.

Para que fique melhor detalhado e não haja prejuízo de informação, o quadro 3 apresenta a lista dos 56 materiais selecionados e seus respectivos índices (ID).

Quadro 3 – Trabalhos incluídos.

ID	Referência
M1	(VEMULA *et al.*, 2022)
M2	(METWALLI; VAN METER, 2022)
M3	(HIETALA, 2022)
M4	(KHAN, A. A. *et al.*, 2022)
M5	(SMITH *et al.*, 2020)
M6	(ALYAMI *et al.*, 2022)
M7	(WU *et al.*, 2021)
M8	(ZAMBONI *et al.*, 2020)
M9	(BRANDHOFER; DEVITT; POLIAN, 2021)
M10	(AOUN *et al.*, 2022)
M11	(PALTENGHI; PRADEL, 2022)
M12	(CÓRCOLES *et al.*, 2020)
M13	(SCEKIC; YAKARYILMAZ, 2022)
M14	(KIM; CHO; SEO, 2021)
M15	(ANTIPOV; KIKTENKO; FEDOROV, 2022)
M16	(PALER; BASMADJIAN, 2022)
M17	(OLIVIERI; ASKARPOUR; NITTO, 2021)
M18	(YU, 2022)
M19	(MOHAMMADBAGHERPOOR *et al.*, 2021)
M20	(NGUYEN, T.; MCCASKEY, Alexander J., 2022)
M21	(CHARETON *et al.*, 2022)
M22	(ANGARA *et al.*, 2020)
M23	(MANDRA *et al.*, 2021)
M24	(ZHAO, P.; ZHAO, J.; MA, 2021)
M25	(GUO *et al.*, 2022)
M26	(DA ROSA; DE SANTIAGO, 2022)
M27	(GUSTAFSON *et al.*, 2021)
M28	(LAROSE *et al.*, 2022)
M29	(ALI; YUE, 2020)
M30	(MARTINA *et al.*, 2022)
M31	(MIRANSKYY; ZHANG, L.; DOLISKANI, 2021)
M32	(WANG, S.; ADAMS; BROADBENT, 2021)
M33	(ALGHADEER *et al.*, 2022)
M34	(DEY *et al.*, 2020)
M35	(NGUYEN, H. T.; USMAN; BUYYA, 2022)
M36	(DONKERS *et al.*, 2022)
M37	(PEDURI; BHAT; GROSSER, 2022)
M38	(J. *et al.*, 2022)
M39	(WANG, S. P.; SAKK, 2021)
M40	(MIRANSKYY *et al.*, 2022)

Continua na próxima página

Continuação do quadro

ID	Referência
M41	(SODHI; KAPUR, 2021)
M42	(NAGARAJAN; LOCKWOOD; COFFRIN, 2021)
M43	(HEVIA; PETERSSEN; PIATTINI, 2022)
M44	(ESCANEZ-EXPOSITO; CABALLERO-GIL; MARTIN-FERNANDEZ, 2022)
M45	(MOUEDDENE *et al.*, 2020)
M46	(HEESE; BICKERT; NIEDERLE, 2022)
M47	(OLIVER, 2020)
M48	(SANCHEZ-RAMIREZ *et al.*, 2021)
M49	(KIEFL; HAGEL, 2020)
M50	(DE STEFANO *et al.*, 2022)
M51	(MYKHAILOVA; SVORE, Krysta M., 2020)
M52	(ANGARA *et al.*, 2022)
M53	(COBB; SCHNEIDER; LEE, 2022)
M54	(SCHMITT; DE MICHELI, 2022)
M55	(MCCASKEY, Alexander J *et al.*, 2020)
M56	(VALENCIA *et al.*, 2021)

Fonte – Elaborado pelo autor.

Fim do quadro

3.7 AMEAÇAS À VALIDADE

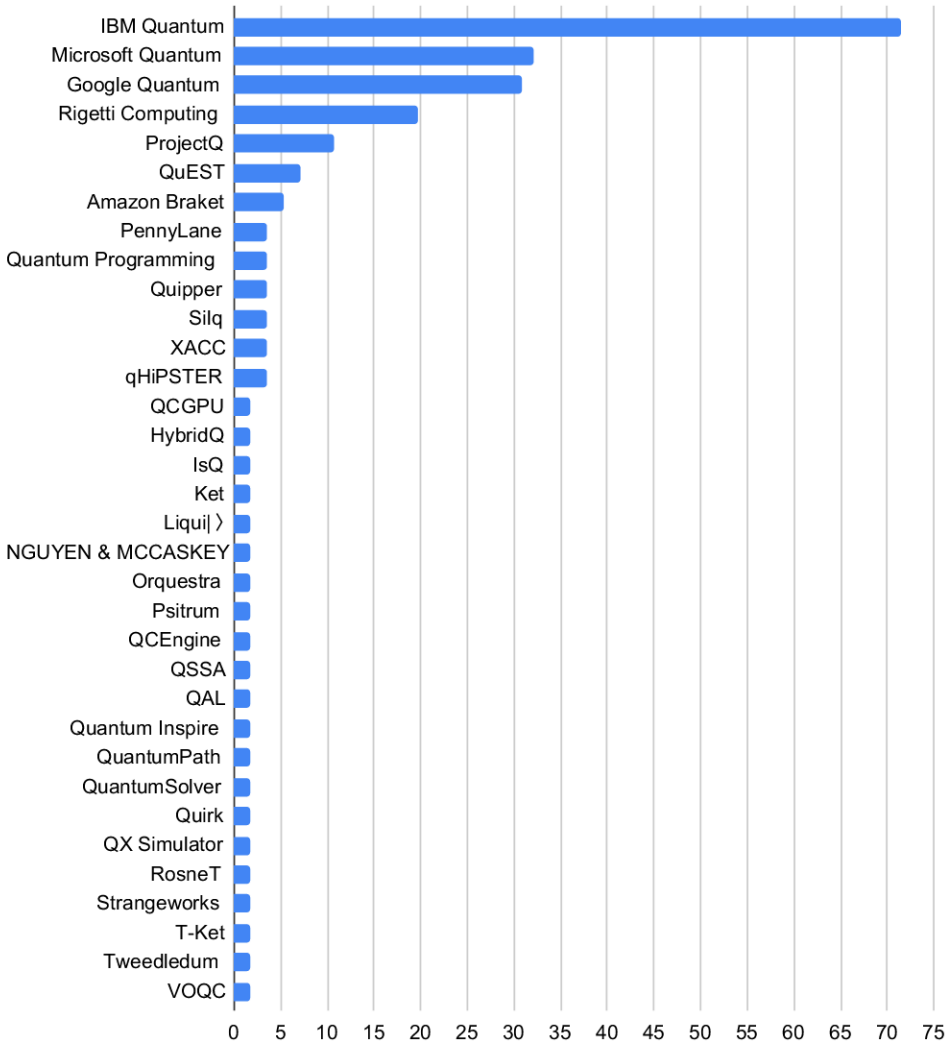
Nenhum estudo sistemático está livre de ameaças à validade, ainda que usem um protocolo bem estruturado e sejam executados de forma perfeccionista. Esta seção aponta as ameaças que o presente mapeamento apresenta, seguindo os cinco tipos de validade identificadas por Maxwell (MAXWELL, 1992).

- **Validade descritiva.** Apesar do trabalho ter sido revisado, a maior parte do trabalho foi feito por apenas uma pessoa. Este fato pode aumentar as chances de que alguma palavra-chave tenha sido ignorada ou que algum outro erro manual tenha ocorrido.
- **Validade interpretativa.** Neste tópico, também é válido mencionar o fato de o trabalho ter sido feito em maior parte por apenas uma pessoa. Mas o critério CI4 foi pensado para evitar falhas na interpretação.
- **Validade teórica.** Os critérios CI1 e CI6 dependem da experiência do pesquisador, portanto, é possível que alguém com uma experiência maior encontre resultados diferentes.
- **Generalização da validade.** O critério CI4, apesar de melhorar a interpretação correta, pode ter excluído materiais que deveriam ser incluídos. A falta de termos como *toolkit*, *framework*, ou SDK pode ter o número de materiais incluídos, mas é pouco provável.

3.8 RESULTADOS

3.8.1 QP1. Quais são os softwares ou ferramentas disponíveis para simular ou operar um computador quântico real no contexto da criptologia e quais são mencionados mais vezes?

Figura 5 – Ferramentas mapeadas e frequência.



Fonte – Elaborado pelo autor.

A plataforma da IBM apareceu em mais de 70% dos 56 materiais selecionados. As plataformas da Microsoft, Google e Rigetti também se mostraram presentes numa boa quantidade dos materiais selecionados, mas muito menos frequentes do que a plataforma da IBM. A Figura 5 detalha a frequência com que cada ferramenta apareceu nos trabalhos selecionados.

3.8.2 QP2. Considerando os materiais selecionados, é possível levantar quais características de cada ferramenta?

IBM Quantum

Qiskit é um SDK Python que permite projetar aplicações em circuitos quânticos (M4, M7, M10, M11, M16, M19, M24, M28, M29, M30, M35, M43), podendo ser facilmente incorporado em Jupyter Notebook (M35, M52) e foi desenvolvido em colaboração open-source com a comunidade (M12). Quantum Composer é um editor gráfico que permite criar circuitos quânticos, abstraindo a programação do Qiskit (M35, M41, M43), sem que seja necessário conhecimentos em programação (M52). Quantum Lab é um Jupyter Notebook baseado em nuvem que permite rodar Qiskit sem precisar instalar nada (M43).

A IBM foi a primeira a trabalhar na construção de um computador universal e facilitar o acesso através de uma API, o Qiskit (M34). Atualmente disponibiliza documentação e tutoriais online (M38), além de uma variedade de bibliotecas e ferramentas adicionais (M35).

A tecnologia usa o modelo de computação quântica baseado em circuitos, um dos mais usados e de propósito geral (M33, M35, M47, M50). Permite vantagem teórica sobre computadores clássicos no que tange computação paralela em larga escala (M50).

O pacote é bastante acessível e capaz (M53), possui os benefícios de um serviço de computação quântica baseado em nuvem (M35, M41, M46) e disponibiliza acesso gratuito (M38, M52) a computadores quânticos e simuladores (M35, M49, M53) desde 2016 (M14). O simulador remoto tem performance semelhante ao da IonQ, perdendo em *runtime* e ganhando em capacidade (M36). Circuitos e resultados são facilmente visualizados online, mas há um limite de 10 mil segundos de execução (+- 2h e 46m) (M53). Também vale lembrar que além das opções em nuvem, existe um simulador local (M13).

O hardware físico dos computadores quânticos opera em qubits transmon supercondutores (M19, M46). O simulador possui alguns módulos:

- TERRA, módulo básico, importa funcionalidades de tradução para a linguagem OpenQASM para interação com computadores quânticos (M12, M14);
- IGNIS, disponibiliza ferramentas para melhor caracterização de ruídos, melhorar portas lógicas e processar na presença de ruídos (M12, M14);
- AER, implementa um simulador com modelagem de ruído (M14) e dá acesso a simuladores de alta performance que ajudam a entender até que ponto os processadores clássicos podem simular um quântico (M12);
- IBMQ PROVIDER, permite acesso aos computadores e simuladores remotos (M14);

- AQUA, disponibiliza uma camada de abstração de gates quânticos, permite importação de algoritmos a aplicações de mais alto nível (M12, M14).

A ferramenta pode ser aplicada em criptografia (M6, M21). Sendo capaz de rodar os algoritmos de Shor (M31, M38, M39) e Grover (M1, M2, M8, M9, M22, M38, M39, M40). Além disso, a biblioteca Qiskit Acqua já tem o algoritmo de Shor embutido, então o programador não precisa conhecer os detalhes, apenas chama uma classe Python com o algoritmo de Shor (M31).

Quanto a algumas aplicações práticas nos trabalhos incluídos, tem-se as seguintes considerações:

- Executou esquema de autenticação de senha num computador IBM de 27 qubits (ibmq-sydney) (M32);
- Foi utilizado computador de 27 qubits para o estudo (M19);
- Usou máquina real e simulador livre de ruídos (M18);
- Usou Qiskit pela facilidade de uso e disponibilidade de computadores quânticos (M17).

Microsoft Quantum

Q# é uma linguagem de alto nível específica para circuitos quânticos (M3, M4, M10, M11, M14, M29, M31, M35, M43, M51). A linguagem se parece com C# e F# (M53) e pode ser usada como extensão do Visual Studio (M43, M53). Uma curiosidade é que o Github classifica essa linguagem como Q# (M50).

QDK (Quantum Development Kit) é um SDK open-source que inclui um compilador, simulador de propósito geral (M33, M35, M47), *debugger*, estimador de recursos e diversas ferramentas para produzir algoritmos quânticos (M35, M51). Os programas podem ser criados em aplicações *standalone*, em Jupyter Notebooks, *command line*, ou integrados em C# ou Python (M35).

O QDK pode ser instalado em Windows, Mac ou Linux com ambiente Visual Studio. Possui boa documentação e suporte, contudo, a programação exige uma curva de aprendizagem adicional para profissionais não acostumados com linguagens de baixo nível (M41).

Azure Quantum é a plataforma online da Microsoft que permite rodar programas Q# em simuladores ou em computadores quânticos de diferentes vendedores (IonQ e Honeywell são suportados por padrão), em adição, problemas de otimização podem ser rodados no próprio hardware Azure (M43, M53). Contudo, apenas a primeira hora de simulação é gratuita (M53).

Google Quantum

Framework da Google para criar e editar circuitos NISQ (M34) de computação quântica universal (M33, M47). Cirq é um SDK Python open-source para escrever circuitos quânticos (M4, M10, M28, M31, M35, M50). A ferramenta é relativamente nova, sendo que em 2020 ainda estava em fase alfa (M14), mas tem o poder de rodar programas em simuladores ou processadores quânticos da Google (M35). Cirq usa o simulador open-source qsim, o qual tem aceleração por GPU utilizando CUDA (M27).

Os programas usam apenas qubits, sem registradores clássicos explicitamente. Em computadores quânticos reais, os qubits formam uma topologia 2D. O Cirq permite definir os qubits em linha ou em grade (M13).

Rigetti Computing

A empresa Rigetti fornece ferramentas open-source através do SDK Forest, a linguagem de alto nível pyQuil, a biblioteca de programas quânticos Grove, a linguagem a nível de instruções quânticas Quill (comparado ao assembly), o compilador Quilc e a máquina virtual QVM (M28, M30, M31, M41, M43).

Para rodar um script, deve-se rodar o Quilc e a QVM. Assim um código pyQuil é compilado para Quil nativo pela Quilc e rodado pela QVM (M5, M13, M41), esta última escrita em C (M14). A computação segue o modelo de portas lógicas universais (M47). Os qubits são definidos como lista e o programa permite criar bits clássicos para armazenar medidas (M13).

Forest tem serviços em nuvem que consistem na linguagem de instruções quânticas Quil, na biblioteca Python pyQuil para construir programas, na biblioteca de programas quânticos Grove e no ambiente de simulação QVM (M34).

ProjectQ

Framework open-source em Python desenvolvido pela ETH Zurich (M14, M43). Permite criar programas e traduzí-los para qualquer tipo de *backend*, seja simulador ou computador quântico real. Atualmente está integrado na IBM Quantum Experience e suportará outras plataformas. (M43)

Sua sintaxe é mais enxuta e mais próxima da notação matemática da física quântica. “Engines” são os objetos principais e o programa, nelas inserem-se uma série de qubits. Os operadores são aplicados diretamente aos qubits. Os circuitos são desenhados usando Matplotlib. (M13)

QuEST

Simulador de circuitos quânticos universais (M33, M45) de Oxford (M33). Usa *multithread*, distribuição e aceleração por GPU (M33). É rápido e com baixo erro. Seu

simulador local tem ótimo *runtime* e capacidade (M36).

Amazon Braket

Amazon Braket é um serviço de computação quântica oferecido como parte da AWS (M56). Possui um SDK e uma máquina real na nuvem. Usa o modelo de preços *pay-per-use* para sua nuvem. O Python SDK faz interface com o serviço de computação quântica da Amazon para executar aplicativos em simuladores ou em computadores quânticos de terceiros (Rigetti, D-Wave, IonQ) (M28, M35).

É possível executar algoritmos como o de Shor, mas não é possível medir e redefinir qubits, como é possível em vários outros simuladores e como é feito em diversas implementações do algoritmo de Shor. Este fato implica que é mais difícil executar o algoritmo de Shor na plataforma. Ao executar um algoritmo, o circuito permanece o mesmo, independentemente do simulador ou hardware utilizado. Porém, a forma como o circuito é chamado muda dependendo da plataforma em que será executado (M56).

Xanadu PennyLane

É uma biblioteca Python da empresa Xanadu para aprender programação quântica e otimizar computação híbrida. Tem um Add-on que permite trabalhar com outras plataformas, como o Qiskit (M43). Apresenta as bases para executar o algoritmo de Shor (M15).

Quantum Programming Studio

Ambiente web para construir circuitos quânticos baseados em gates através de um editor *drap-and-drop* ou de código OpenQASM. Permite que os circuitos sejam exportados para diversas linguagens e rodados em diversos simuladores ou computadores quânticos, como Rigetti, IBM, Google, Microsoft (M43).

Quipper

É uma linguagem para programar computadores quânticos (M11) que pode ser aplicada em criptografia (M21).

Silq

Linguagm de programação para computadores quânticos (M11, M53) similar ao Q#. Pode ser usada via Visual Studio Code ou VSCodium da Silq. Linguagem pouco mais simplificada do que o Q#. Código deve ser rodado na máquina local, não possui serviços online (M53).

XACC

É um framework híbrido clássico-quântico que permite a programação, compilação e execução de aplicações científicas. Implementado em C++, mas permite a extensão para linguagens de mais alto nível, como Python, Julia, etc (M22, M55).

qHiPSTER

O nome significa *Quantum High Performance Software Testing Environment*, mas também é conhecido como Intel *Quantum Simulator* (Intel-QS ou IQS). É um simulador de computação quântica universal (M45) que aproveita ao máximo as arquiteturas multicore (M33).

CGPU

Ele aproveita a aceleração OpenCL e GPU para construir um Simulador de Computação Quântica universal (M45).

HybridQ

HybridQ é uma plataforma que contém um *framework* para integrar diversas técnicas de simulação e rodar numa variedade de *hardwares*. A plataforma foi escrita inteiramente em Python e usa algumas linguagens compiladas em partes essenciais para melhorar a performance da simulação. Com a plataforma é possível projetar um circuito *noisy* ou *noiseless* e simulá-lo em diversas tecnologias sem ter que reescrever o código (M23).

IsQ

IsQ é uma linguagem (compilador incluso) que contém diversas funcionalidades, incluindo algumas funcionalidades que não são comuns em outras linguagens. É uma linguagem flexível, por poder ser compilada para diferentes representações intermediárias (IRs), como OpenQASM 3, QIR e QCIS. Códigos IsQ podem ser compilados para QCIS assembly e executados no hardware quântico supercondutor da USTC (universidade de ciência e tecnologia da China) (M25).

Ket

Ket é uma linguagem de programação clássico-quântica que mantém códigos clássicos e quânticos bem separados e tem uma interação dinâmica entre computadores clássicos e computadores quânticos baseados em nuvem. Os códigos da linguagem são baseados em Python, mas com a extensão da biblioteca Libket, em C++ (M26).

LIQUi|>

Language-Integrated Quantum Operations é uma plataforma de simulação da Microsoft. O *software* pode ser aplicado em criptografia (M21).

NGUYEN & MCCASKEY

Uma extensão para a linguagem Python que permite computação quântica-clássica heterogênea por meio de uma infraestrutura C++ robusta para compilação quântica *just-in-time* (QJIT). Construído sobre QCOR, uma extensão e compilador de linguagem C++, para permitir uma abordagem independente de hardware quântico para a computação quântica-clássica e manter o desempenho necessário para modelos de computação acoplados CPU-QPU (M20).

Orquestra

Desenvolvido por Zapta Computing, permite trabalhar com *frameworks* e com *hardwares* dos principais vendedores através de um conjunto de ferramentas modulares baseadas em *workflow* (M43).

Psitrum

Simulador de computação quântica universal em *hardware* clássico. Possui GUI dividida em quatro seções: *circuit designer*, *quantum state tracer*, *visualization graphs* e representação numérica (M33).

QCEngine

Simulador de computador quântico executado localmente. Ele lê códigos em JavaScript e permite converter seus programas em Qiskit ou Q# (M29).

QSSA

Quantum IR (Intermediate Representation) que promete ser um modelo otimizado por levar em consideração o que se conhece em otimização de compiladores na última década (M37).

Quantum Arithmetic Library (QAL)

Contruído por diversos pacotes que permitem gerar arquivos OpenQASM de componentes aritméticos, como soma, multiplicação, divisão, subtração. Pode rodar Grover (M8).

Quantum Inspire

Desenvolvido pela QuTech (fundada em colaboração entre a TU Delf e a organização holandesa para pesquisa em ciência aplicada). Possui uma variedade de funcionalidades para programar, executar e examinar algoritmos. Possui uma GUI para programar em quantum assembly language (QASM) e visualizar operações no diagrama de circuito (M43).

QuantumPath

É um ecossistema de ferramentas, serviços e processos que simplificam o desenvolvimento de algoritmos quânticos em sistemas de informação híbridos. Suporta ambos modelos baseado em gates quânticos e *annealing* (M43).

QuantumSolver

Ferramenta open-source baseada no Qiskit (portanto, baseada em Python3) orientada a quem não tem experiência em programação. Possuí CLI ou *Web Interface* (Python3 e Flask, no backend, React, HTML5, CSS3, no front), ambos com os mesmos recursos. No menu principal, o usuário pode se autenticar como visitante ou com uma conta IBM, então uma lita de *backends* e algoritmos disponíveis aparece para seleção. Escolhido o algoritmo a se rodar no *backends* e definidos os parâmetros, há a opção de rodar uma vez ou múltiplas (M44).

Quirk

Simulador de computação quântica universal que permite desenvolver circuitos graficamente, sem código (M33).

QX Simulator

Simulador de computação quântica universal com API C++ (M45).

RosneT

Biblioteca Python que segue modelo de programação *task-based* para estender operações com tensores em sistemas distribuídos. É baseada no sistema de simulação *Tensor Network*. Usa memória secundária, tornando a simulação tolerante a falhas e ultrapassando o limite de memória RAM (simular computadores quânticos usa muita memória RAM). Oferece uma interface de programação simples. Possibilidade de especificar os recursos necessários, gerenciamento de memória que permite aplicações com larga quantidade de dados, polimorfismo das *task* permitindo diferentes versões da *task* para diferentes hardwares (M48).

Strangeworks Quantum Computing Platform

Ambiente web que suporta diferentes *frameworks* (Qiskit, Q#, Cirq, Forest, etc), também suportando os simuladores e computadores quânticos das diversas empresas (M43).

T-Ket

SDK Python para computadores quânticos baseados em gate e que pode executar Grover (M8).

Tweedledum

Uma biblioteca complementar para síntese e compilação de circuitos quânticos que busca aprimorar outros compiladores e *frameworks*. Implementado em C++17 e com ligações em Python para facilitar a integração de diferentes *frameworks*. Algumas ferramentas já o utilizam, como qiskit-terra, quilc e staq (M54).

Verified Optimizer for Quantum Circuit (VOQC)

VOQC contém otimizações para *Small Quantum Intermediate Representation* (SQIR) e SQIR pode rodar Grover e Shor (M3).

Mais considerações extraídas dos materiais

- O material M14 afirma que as plataformas da IBM, Google e Rigetti são as mais indicadas para se usar com um *hardware* quântico real, pois as empresas por trás desses *frameworks* construíram seus próprios computadores quânticos e constantemente testam e melhoram a interação entre as necessidades do mundo real e o *hardware* quântico.
- O material M36 traz um *benchmark* das plataformas mais importantes de computação quântica usando a ferramenta QPack. Em resumo, o simulador QuEST da universidade de Oxford foi o melhor geral (183,2 pontos), o simulador da Google ficou em segundo lugar (157,6 pontos), o simulador IBM Qiskit aer ficou em terceiro (147,2 pontos) e o QVM da Rigetti ficou em último (104,8 pontos).

- Segundo M50, 95% dos projetos open-source que envolvem computação quântica no Github foram feitos com o IBM Qiskit, Google Cirq ou Microsoft Q#. Essas três tecnologias são conhecidas por serem mais maduras e estáveis, tendo funcionalidades peculiares e permitindo rodar códigos em simuladores ou em computadores quânticos reais. A pesquisa encontrou 442 (60,5%) repositórios Qiskit no Github, 217 (29,7%) Cirq e 72 (9,8%) Q#. A pesquisa também descobriu que Qiskit é o mais usado entre estudantes de graduação e pós-graduação, pesquisadores e desenvolvedores open-source, Q# é o mais utilizado na indústria e Cirq é o menos utilizado em todas as categorias. Por fim, a pesquisa também encontrou que 27% dos desenvolvedores são dos EUA, 12% da Índia e 9% da Itália.

3.8.3 QP3. Considerando os materiais incluídos, é possível concluir que houve um aumento de materiais publicados na área desde 2020?

Dos 56 materiais selecionados, 50% deles eram dos dez primeiros meses de 2022, 28,6% eram de 2021 e 21,4% eram de 2020. Houve aumento de 2020 para 2021, mas um aumento ainda maior de 2021 para 2022.

3.8.4 QP4. Há um padrão observável com respeito aos locais evolvidos nas publicações?

A lista de locais foi construída considerando-se os locais indicados pelos autores dos trabalhos, não apenas os autores principais. 98 diferentes locais foram encontrados nos 56 materiais selecionados, abrangendo todos os continentes. Contudo, alguns locais não podem ser correlacionados com alguma localização física, como exemplo: IBM, Google, Intel.

O destaque aqui vai para a IBM. Cinco dos 56 materiais escolhidos tinham pelo menos um dos autores trabalhando na IBM. Três dos 56 materiais tinham pelo menos um dos autores do Laboratório Nacional de Los Alamos (EUA). Os demais locais apareceram apenas em um ou dois materiais, não sendo possível perceber qualquer padrão observável, visto que o número de trabalhos selecionados é pequeno em relação ao número de trabalhos nos diversos campos da computação quântica.

3.9 DISCUSSÃO DOS RESULTADOS DO MAPEAMENTO

Ao que se pode perceber com a análise dos resultados e das características das ferramentas encontradas, o ambiente da IBM é o mais utilizado. Em concordância com a pesquisa de M50, o ambiente da IBM é o mais utilizado entre estudantes, pesquisadores e desenvolvedores open-source. Ainda segundo M50, o Quiskit da IBM domina em número de repositórios no Github. A ferramenta também foi mencionada

em mais de 70% dos materiais selecionados por este mapeamento, o que faz com que uma maior quantidade de características pudessem ser levantadas.

Outros ambientes bastante mencionados foram os da Google, Microsoft e Rigetti. O material M14 afirma que as plataformas da IBM, Google e Rigetti são as melhores para se usar com um *hardware* quântico real e o material M50 afirma que o Q# da Microsoft é a linguagem de computação quântica mais usada na indústria. Além do Quiskit, M50 levanta que 95% dos projetos open-source que envolvem computação quântica no Github são escritos em Qiskit, Cirq ou Q#.

Em termos de desempenho de simuladores, o material M36 realiza uma série de *benchmarks* e conclui que o simulador QuEST é o melhor no geral, mais rápido e com menos erros, seguido pelo simulador usado no Google Cirq, pelo Quiskit aer e o QVM da Rigetti aparece em último com uma desvantagem considerável.

Ao que diz respeito das ferramentas e *softwares* encontrados pela QP1, a maior parte dos resultados foram mencionados em apenas um ou dois materiais. Apesar de menos conhecidas ou menos usadas, algumas dessas ferramentas apresentam funcionalidades interessantes. Como uma maneira de explicar o baixo número de aparecimento da maior parte das ferramentas pode-se atribuir o fato de que são recém criadas. Como visto nos resultados da QP3, metade dos materiais selecionados foram publicados em 2022.

No que tange a seleção de uma ferramenta para experimentar o algoritmo de Shor no capítulo seguinte, a melhor escolha é o ambiente da IBM com Qiskit. Além de ser o mais popular e o mais usado, o ambiente possui vasta documentação, vasta comunidade, uma linguagem simples em Python que pode ser usada em um *notebook* e há ainda um método com o algoritmo de Shor implementado que pode ser chamado através do Qiskit Acqua (M31).

4 EXPERIMENTAÇÃO COM A FERRAMENTA

4.1 PRIMEIROS PASSOS

Como visto no Capítulo 3, a ferramenta escolhida para experimentação foi o Qiskit da IBM. Bastante sobre a ferramenta já foi discutido no Capítulo 3. Agora será feita uma introdução prática da ferramenta e, em seguida, ela será usada para executar o algoritmo de Shor. Ressaltando que este não é um tutorial de como usar a ferramenta, mas sim noções básicas para instruir o leitor sobre os aspectos gerais de seu funcionamento.

Na Figura 6 há um código Python utilizando a biblioteca Qiskit. Como pode-se observar, a sintaxe e palavras chaves utilizadas são simplesmente Python, a maneira como são feitas importações, o uso de comandos como o *print*, a atribuição de variáveis e até o uso de laços de repetição são permitidos. Nas duas primeiras linhas de código foram feitas as importações necessárias. Neste caso, foram importados os registradores, o circuito e o QasmSimulator. Este é o simulador de computação quântica que executa na própria máquina. Em seguida, é criado um registrador quântico de dois qubits; um registrador clássico de dois bits, usado para armazenar os resultados das medidas; um circuito quântico contendo os registradores recém criados; e uma porta Hadamard (H) é aplicada ao primeiro qubit do registrador (o Apêndice D explica a porta H). A medida dos qubits foi feita dentro do laço de repetição, sendo aplicada a cada qubit e armazenada em seus respectivos bits clássicos. Ao fim do código, a simulação foi executada num *backend* que, neste caso, foi o QasmSimulator. A simulação foi executada 100 vezes e então o resultado é impresso na tela.

Figura 6 – Introdução à biblioteca Qiskit.

```
01: from qiskit import QuantumCircuit,QuantumRegister,ClassicalRegister
02: from qiskit.providers.aer import QasmSimulator
03:
04: q = QuantumRegister(2)
05: c = ClassicalRegister(2)
06: qc = QuantumCircuit(q,c)
07: qc.h(0)
08: for i in range(2):
09:     qc.measure(q[i],c[i])
10:
11: backend = QasmSimulator()
12: job = backend.run(qc, shots=100)
13: result = job.result()
14: counts = result.get_counts()
15: print(counts)
```

Fonte – Elaborado pelo autor.

A Figura 7 é a saída da simulação anterior. O código da Figura 6 foi executado cinco vezes para que seja possível ver os diferentes resultados obtidos, visto que um computador quântico trabalha sobre probabilidades. Lê-se a saída da seguinte maneira: na primeira linha, das 100 vezes que o cálculo foi realizado, 42 vezes resultou na leitura 01 e 58 vezes na leitura 00. Explicação: no Qiskit, todos os qubits inicializados num registrador estão no estado 0, ao aplicar a porta H em um dos qubits, ele entrou num estado de superposição com 50% de chance de se encontrar em cada um dos estados. Assim, há 100% de chance de um dos qubits se encontrar no estado 0 e 50% de chance do segundo qubit se encontrar no mesmo estado, resultando em 50% de chance do estado final da máquina se encontrar no estado 00 e 50% de se encontrar no estado 01. Ao executar o código cinco vezes, percebemos que os resultados giram em torno isso. As linhas pares mostram os resultados.

Figura 7 – Primeiros resultados com o Qiskit.

```
01: cmd: python3 introducao.py
02: {'01': 42, '00': 58}
03: cmd: python3 introducao.py
04: {'01': 45, '00': 55}
05: cmd: python3 introducao.py
06: {'01': 56, '00': 44}
07: cmd: python3 introducao.py
08: {'01': 46, '00': 54}
09: cmd: python3 introducao.py
10: {'01': 49, '00': 51}
```

Fonte – Elaborado pelo autor.

4.2 O ALGORITMO DE SHOR

Para entender melhor sobre o algoritmo de Shor, é recomendado ler os Apêndices D, E, F, G e H. O apêndice H explica o algoritmo de Shor e os demais explicam o funcionamento de portas lógicas quânticas relevantes e a transformada de Fourier quântica. Nesta seção será executada a implementação do algoritmo de Shor contida no pacote de algoritmos do Qiskit. Desta vez, a importação é diferente. Não iremos montar um circuito quântico, mas importaremos uma instância de um circuito quântico para o algoritmo de Shor. Para chamar a implementação oficial do algoritmo de Shor do Qiskit devemos importar o pacote algoritmos, uma coleção de diversos outros algoritmos há neste pacote.

Na Figura 8, o código também é executado na máquina local, por meio do QasmSimulator. Então definimos uma instância quântica e iniciamos o algoritmo de Shor com ela. A instância foi executada 100 vezes no *backend* escolhido. Por fim, o

algoritmo de Shor é chamado a fatorar o número 15 e o resultado é mostrado na tela por meio do comando *print*.

Figura 8 – Algoritmo de Shor com o Qiskit.

```
01: from qiskit.utils import QuantumInstance
02: from qiskit.algorithms import Shor
03: from qiskit.providers.aer import QasmSimulator
04:
05: backend = QasmSimulator()
06: shor = Shor(QuantumInstance(backend, shots=100))
07: result = shor.factor(15)
08: print(result)
```

Fonte – Elaborado pelo autor.

Na Figura 9 está a saída da Figura 8, mostrando que o algoritmo encontrou os fatores 3 e 5 corretamente e levou 7 segundos para isso em um processador com 12 *threads*. O tempo de execução foi medido com o comando *time* no sistema operacional Linux.

Figura 9 – Resultado da fatoração do número 15.

```
01: {'factors': [[3, 5]], 'successful_counts': 2, 'total_counts': 4}
02:
03: real   0m7,623s
04: user   0m11,896s
05: sys    0m1,751s
```

Fonte – Elaborado pelo autor.

Em seguida o código da Figura 8 foi executado para diferentes números em largura de bits crescente. Para o número 21, o código encontrou o resultado correto em 31 segundos. O resultado está na Figura 10.

Figura 10 – Resultado da fatoração do número 21.

```
01: {'factors': [[3, 7]], 'successful_counts': 6, 'total_counts': 23}
02:
03: real   0m31,325s
04: user   3m50,491s
05: sys    0m2,094s
```

Fonte – Elaborado pelo autor.

Para o número 39, o código da Figura 8 encontrou o resultado em 580 segundos (9 minutos + 40 segundos). O resultado está na Figura 11.

Figura 11 – Resultado da fatoração do número 39.

```
01: {'factors': [[3, 13]], 'successful_counts': 13, 'total_counts': 29}
02:
03: real   9m40,604s
04: user   109m51,949s
05: sys    0m7,731s
```

Fonte – Elaborado pelo autor.

O número 15 = 2^3 + 7 tem 4 bits, o número 21 = 2^4 + 5 tem 5 bits e o número 39 = 2^5 + 7 tem 6 bits. Números de 7 bits de largura não puderam ser fatorados em tempo hábil. Comparando os tempos de compilação, $T(5) \approx T(4)^{1,76}$ e $T(6) \approx T(5)^{1,85}$. Considerando a equação (7), podemos considerar que f(n) é uma monótona crescente ou uma constante, caso outros aspectos computacionais tenham influenciado no tempo. Mesmo para a opção mais otimista em que $f(n) \approx 1{,}76$, temos que $T(7) \approx 580^{1,76}$ = 73052 segundos (aproximadamente 20 horas).

$$log_{T(n)} T(n+1) = f(n) \tag{7}$$

Para estimar o tempo necessário para decifrar uma chave de 2048 bits (comum no RSA), pode-se considerar que para um valor de 6 bits o tempo foi de 580 segundos ($\approx$ 10 minutos) e, usando o mesmo computador, o tempo seria algo $\approx 580^{1,76^{2042}}$ segundos mesmo na perspectiva mais otimista (muito maior do que a idade estimada do universo conhecido).

4.3 EXECUTANDO NUM SIMULADOR NA NUVEM

A IBM disponibiliza o uso de serviços em nuvem para a execução de códigos em simuladores ou em computadores quânticos reais. A conta gratuita garante acesso limitado a esses recursos. Com essa conta, há a possibilidade de usar remotamente computadores quânticos de 7 qubits e até um computador de 127 qubits, com uma fila de espera e o uso desses computadores é limitado a 10 minutos por mês. A versão paga apresenta mais opções de computadores de 127 qubits e alguns computadores de 27 qubits, sem mais limite de tempo. A implementação do algoritmo de Shor do Qiskit utilizada requer 4n + 2 qubits para a execução, onde n é a largura de bits do número a fatorar. Trabalhando com o número 39, que possui largura de 6 bits, precisamos de 26 qubits. O QasmSimulator dos servidores da IBM simula 32 qubits, então é o suficiente para fatorar números de até 7 bits. Os computadores quânticos

disponíveis em nuvem de 7 qubits não têm capacidade de fatorar nenhum número com esse algoritmo (não há números compostos de dois primos para n = 1), mas o computador de 127 qubits tem capacidade teórica de fatorar números de até 31 bits de largura (até 2147483648).

O código da Figura 12. executa o algoritmo no QasmSimulator dos servidores da IBM, então, em vez de importar o simulador no código (*import QasmSimulator*), o ambiente de execução remota é importado (*import IBMQ, providers*). Tendo uma conta na IBM, o usuário deve se autenticar por meio de um token (linhas 05, 07 e 08), então escolher um *backend* disponível para executar o código (linha 09).

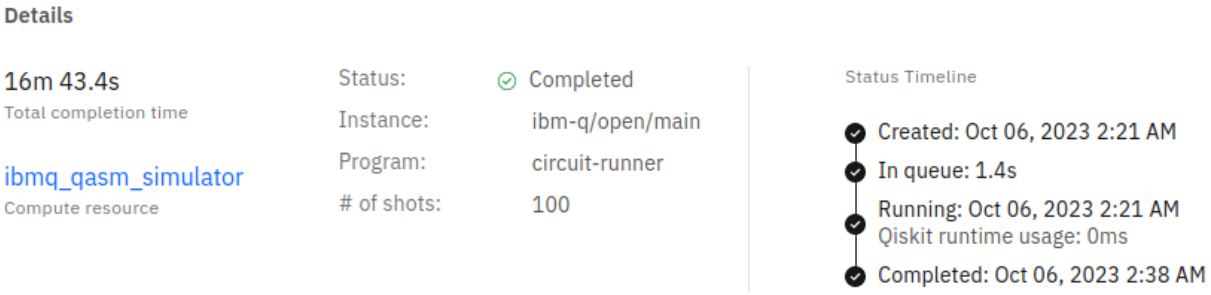
Figura 12 – Algoritmo de Shor na nuvem.

```
01: from qiskit.utils import QuantumInstance
02: from qiskit.algorithms import Shor
03: from qiskit import IBMQ,providers
04:
05: token = input('Insert IBM API TOKEN: ')
06:
07: IBMQ.save_account(token)
08: provider = IBMQ.load_account()
09: backend = provider.get_backend('ibmq_qasm_simulator')
10: shor = Shor(QuantumInstance(backend, shots=100))
11: result = shor.factor(39)
12: print(result)
```

Fonte – Elaborado pelo autor.

O tempo total de execução foi de 16 minutos e 43 segundos, como mostra a Figura 13, e o tempo em fila (*In queue*) foi de 1,4 segundos. No entanto, é difícil saber exatamente quanto tempo a simulação levou, pois alguns fatores aumentam o tempo total, como o tempo de autenticar o usuário. O qiskit não desconta tempo de execução *runtime* para o uso do simulador (*runtime usage: 0ms*). Então, dos 10 minutos disponíveis mensalmente, nada é descontado. A Figura 13 também traz mais informações, como: o *backend* usado (*Compute resource*), que no exemplo foi o "ibmq_qasm_simulator"; o número de vezes que o algoritmo foi executado (*# of shots*), que foi 100 vezes; e o *status*, que pode ser completado (*completed*) ou falhado (*failed*), tendo completado o exemplo; a data em que o processo entrou na fila (*Created*); a data em que o processo começou a executar no servidor (*Running*); e a data em que o processo encerrou (*Completed*).

Figura 13 – Resultado da execução do algoritmo de Shor na nuvem.



Fonte – Elaborado pelo autor.

4.4 EXECUTANDO NO *HARDWARE* REAL

No código da Figura 14, o número 39 é fatorado no computador ibm_brisbane de 127 qubits reais. O código é quase o mesmo de executar no simulador na nuvem, com a diferença de que o *backend* é outro (ibm_brisbane) e a autenticação da conta IBM já foi realizada, não sendo mais necessário informar o *token* do usuário e salvar as informações da conta, bastando apenas carregar as informações de autenticação de usuário salvas na máquina (linha 05).

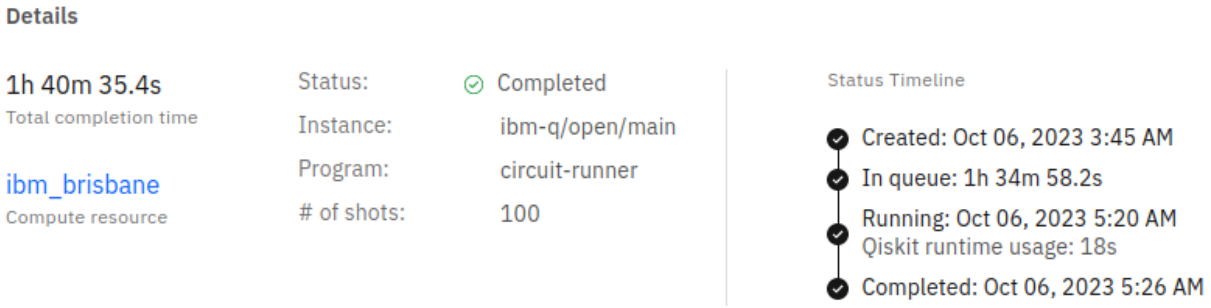
Figura 14 – Algoritmo de Shor num computador quântico real.

```
01: from qiskit.utils import QuantumInstance
02: from qiskit.algorithms import Shor
03: from qiskit import IBMQ,providers
04:
05: provider = IBMQ.load_account()
06: backend = provider.get_backend('ibm_brisbane')
07: shor = Shor(QuantumInstance(backend, shots=100))
08: result = shor.factor(39)
09: print(result)
```

Fonte – Elaborado pelo autor.

O tempo total do trabalho foi de 1 hora 40 minutos e 35 segundos, como pode ser visto na Figura 15, com tempo de fila de 1 hora 34 minutos e 58 segundos e tempo de execução *runtime* descontado de 18 segundos. Pode-se perceber que mais de 5 minutos estão faltando nesses dados, possivelmente tempo de autenticação do usuário e tempo de processamento na máquina do cliente.

Figura 15 – Resultado da execução do algoritmo de Shor no computador quântico real.



Fonte – Elaborado pelo autor.

Na Tabela 5 estão detalhados os tempos de execução do algoritmo de Shor no simulador local, no simulador em nuvem e após executado num computador quântico real. Os tempos levam em consideração a fatoração do número 39, pois esse foi executado em todas as plataformas.

Tabela 5 – Tempo de execução do algoritmo de Shor com a plataforma da IBM.

Plataforma	**Tempo Total**	**Tempo Total - Tempo Fila**
QasmSimulator local	9m 40s	9m 40s
QasmSimulator remoto	16m 43s	16m 42s
Brisbane	1h 40m 35s	5m 37s

Fonte – Elaborado pelo autor.

Pode-se notar que há um ganho real em desempenho de um algoritmo quântico quando executado em um computador quântico. No entanto, o tempo em fila (aguardando outros processos de outros usuários) pode ser enorme.

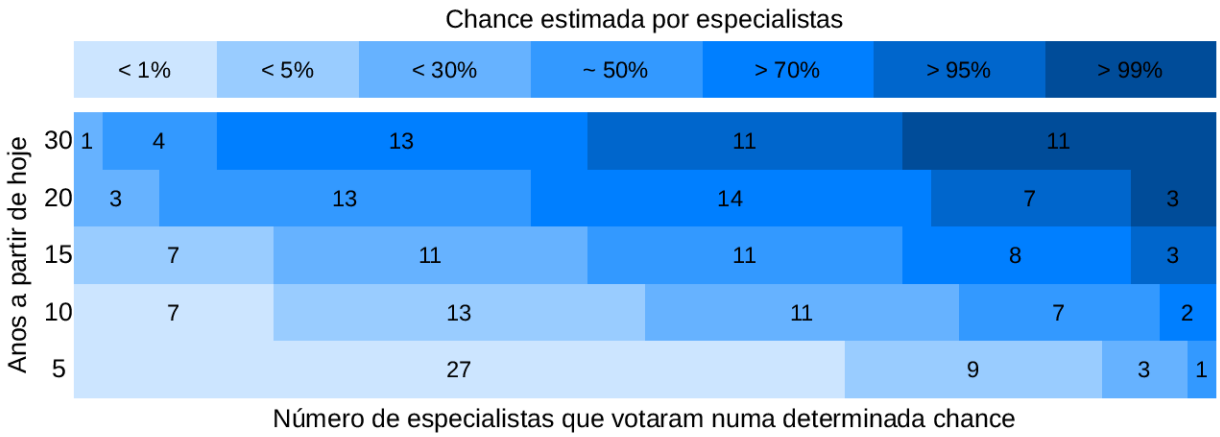
5 ATAQUES QUÂNTICOS AO TLS 1.3

5.1 MODELANDO A AMEAÇA

Seguindo a ideia de (RUNGE, 2023), é possível nomear alguns atores de ameaça e, para melhor respondermos o momento em que cada ator de ameaça se torna relevante, dividiremos os estágios da computação quântica em:

- **Era pré-quântica:** É o estágio em que nos encontramos. Os computadores quânticos ainda não são fortes o suficiente para ameaçarem a segurança da informação, encontrando-se na fase NISQ, com elevado grau de erro e de escala intermediária.
- **Era pós-quântica inicial:** A computação quântica já se torna uma ameaça real à segurança da informação, mas ainda está dando seus passos iniciais. Os computadores ainda são bastante caros e seu *hardware* ainda não apresenta um desempenho tão grande se comparado aos estágios seguintes.
- **Era pós-quântica intermediária:** Uma era intermediária entre os estágios inicial e avançado.
- **Era pós-quântica avançada:** Os computadores quânticos não só são uma ameaça real à segurança da informação, como também já são máquinas bastante acessíveis, em relação a estoque e preço. O *hardware* quântico também apresenta melhores *gate times* e melhores tempos de coerência.

Figura 16 – Chances de um computador quântico ser capaz de quebrar o RSA-2048 dentro de 24 horas nos próximos n anos.



Fonte – (MOSCA; PIANI, 2022b)

Para estimar o tempo t, quando acontece a virada do estágio pré-quântico para o estágio pós-quântico, pode-se usar os dados de (MOSCA; PIANI, 2022b), o qual coletou a opinião de especialistas na área sobre a probabilidade de um computador

quântico ser capaz de quebrar o RSA-2048 dentro de 24 horas em 5, 10, 15, 20, 25 ou 30 anos. A Figura 16 é extraída da pesquisa. Com ela, parece ser seguro estimar que esta virada acontecerá nos próximos 15 anos.

Tendo estabelecido os estágios da computação pós-quântica, dividiremos os atores de ameaça em três, considerando a quantidade de recursos financeiros e de recursos humanos disponíveis. Esses atores podem ser governos e grandes organizações, com um volume de recursos muito grande; grupos *hacker* e pequenas organizações, com recursos mais limitados; e o ataque também pode ser feito por apenas um indivíduo. Além disso, cada ator de ameaça terá um nível de habilidade em relação a operar um computador quântico e realizar um ataque com ele. Os níveis de ameaça podem ser 0, quando não há habilidade nenhuma e o ator também não se torna uma ameaça; 1, quando o nível de habilidade é básico; 2, quando intermediário; e 3, quando avançado. A Tabela 6 sintetiza esses conceitos e os correlaciona aos estágios em que cada ator de se torna uma ameaça. Em linhas gerais, o quanto maior a quantidade de recursos financeiros e humanos disponíveis, menos habilidade é necessária para realizar um ataque. No entanto, governos e grandes organizações que possuam altas habilidades podem ser tornar uma ameaça já nos estágios iniciais da computação pós-quântica. Ao contrário, um indivíduo só se tornará uma ameaça nos estágios avançados se possuir um nível alto de habilidade.

Tabela 6 – Correlacionando atores de ameaça e seus respectivos níveis de habilidade às eras pós-quânticas.

Recursos Disponíveis	**Níveis de Habilidade**	**Tornam-se Ameaça em Qual Era Pós-Quântica?**
Governos e Grandes Organizações	3	Inicial
	2	Intermediária
	1	Avançada
Grupos *Hacker* e Pequenas Organizações	3	Intermediária
	2	Avançada
	1	∞
Indivíduos	3	Avançada
	2	∞
	1	∞

Fonte – (MOURA; GIRON; CUSTODIO, 2023)

Os ataques ao TLS 1.3 facilitados por um computador quântico podem ter dois objetivos: impersonificação de uma das partes (agente malicioso finge ser uma das partes, atacando a autenticação) ou quebra de confidencialidade (agente malicioso escuta conteúdo não autorizado). E os ataques podem ser feitos de forma passiva (apenas escutando a rede) ou ativa (modificando a rede). Como a impersonificação exige que o *handshake* seja alterado, ela só pode ser um ataque ativo.

5.2 CENÁRIO DE QUEBRA DE CONFIDENCIALIDADE NO TLS 1.3

Se um agente malicioso quiser realizar um ataque de quebra de confidencialidade no TLS 1.3, para cada modo de *handshake* (Figura 1) terá que realizar os seguintes passos:

- ***Para os modos Certificate-based (server), Mutual Authentication e Post-Handshake Auth.***:
    1. coletar `Client` e `ServerHello` (CH e SH, respectivamente), extraindo as chaves públicas epk_{CH} e epk_{SH} presente nas mensagens `keyshare`;
    2. usar o algoritmo de Shor para ECDLP para quebrar a troca de chaves: isso computa a chave privada de epk_{CH} ou epk_{SH} para então recuperar a chave secreta efêmera usada para comunicação, a qual foi derivada com o algoritmo de derivação de chaves HKDF (KRAWCZYK; ERONEN, 2010); e
    3. usar a chave efêmera recuperada para obter as chaves simétricas, usando o *TLS Key Schedule* (RESCORLA, 2018), permitindo decriptar toda a comunicação.
- ***Para o modo PSK-based resumption***:
    1. usar os passos de 1 a 3 definidos anteriormente sobre o primeiro *handshake*;
    2. usar a chave efêmera recuperada para obter as chaves simétricas usadas durante a comunicação;
    3. decriptar a mensagem `NewSessionTicket`, recuperando sua informação (como *nonces* e *labels*);
    4. usar a informação recuperada para obter a PSK; e
    5. usar a PSK para obter a chave simétrica do segundo *handshake* (sessão retomada), permitindo violar a confidencialidade da conexão resumida.

5.3 CENÁRIO DE IMPERSONIFICAÇÃO NO TLS 1.3

Para o outro caso, em que o agente malicioso deseja realizar um ataque de impersonificação sobre um dos modos de *handshake* do TLS 1.3 (Figura 1), terá que seguir os seguintes passos:

- ***Certificate-Based (server)***:
    1. coletar `Client` e `ServerHello`, extraindo as chaves públicas epk_{CH} e epk_{SH} presentes nas mensagens `keyshare`;

2. usar o algoritmo de Shor para ECDLP para quebrar a troca de chaves: isso computa a chave privada de epk_{CH} ou epk_{SH} para então recuperar a chave secreta efêmera usada para autenticação, a qual foi derivada com o algoritmo de derivação de chaves HKDF (KRAWCZYK; ERONEN, 2010);
3. usar uma das chaves privadas recuperadas para obter as chaves simétricas, usando o *TLS Key Schedule* (RESCORLA, 2018), e então decriptar as mensagens de autenticação (que contém `Certificate`, `CertificateVerify`, and `Finished`); e
4. usar uma das alternativas para atacar a mensagem `Certificate` e conseguir a chave privada do certificado:
   - usar o algoritmo de Shor ou computação adiabática para resolver o problema da fatoração sobre a chave pública do RSA; ou
   - usar Shor para ECDLP sobre a chave pública baseada em curvas elípticas.

- ***Mutual Authentication***: mesmo do anterior para o caso de impersonificação do servidor. Mas o atacante pode escolher impersonificar o cliente ou o servidor. A principal diferença é a mensagem `Certificate` alvo.
- ***Post-Handshake Auth.***: impersonificar o servidor é idêntico aos modos anteriores, mas para impersonificar o cliente:
  1. conferir a presença da extensão `post_handshake_auth`;
  2. coletar `Client` e `ServerHello`, extraindo as chaves públicas epk_{CH} e epk_{SH} presentes nas mensagens `keyshare`;
  3. usar o algoritmo de Shor para ECDLP para quebrar a troca de chaves: isso computa a chave privada de epk_{CH} ou epk_{SH} para então recuperar a chave secreta efêmera usada para autenticação, a qual foi derivada com o algoritmo de derivação de chaves HKDF (KRAWCZYK; ERONEN, 2010);
  4. decriptar a comunicação usando as chaves simétricas recuperadas, procurando pela mensagem `CertificateRequest`; e
  5. usar uma das alternativas para atacar a mensagem `Certificate` do cliente e conseguir a chave privada:
     - solucionar o problema da fatoração com o algoritmo de Shor ou usar um computador adiabático; ou
     - usar Shor para ECDLP se necessário.
- ***PSK-based resumption***: similar à impersonificação do servidor exemplificada nos modos passados, mas os passos devem ser feitos no primeiro

handshake. Com a informação da PSK, o atacante pode impersonificar ambas as partes. Contudo, a duração da PSK pode ser limitada a até 7 dias (RESCORLA, 2018), então a janela de ataque é limitada.

5.4 O ALGORITMO DE SHOR NO TLS 1.3

O Capítulo 4 foi uma experimentação ou uma preparação com a ferramenta Qiskit, a qual executou o algoritmo de Shor em diferentes *backends*. O algoritmo de Shor tem diversas implementações diferentes para problemas diferentes. No Capítulo 4 o problema da fatoração foi explorado, mas apenas com chaves pequenas, dentro das capacidades dos computadores disponíveis. Com o lançamento do processador quântico Condor de 1121 qubits, da IBM, em 2023, e considerando que o algoritmo de Shor contido no pacote da IBM utiliza 4n + 2 qubits para fatorar um número de largura n bits, o processador Condor poderia fatorar números de até 279 bits. Usando a implementação de (GIDNEY, 2017), a qual utiliza apenas 2n + 1 qubits para fatoração, o processador Condor seria capaz de fatorar números de até 560 bits. Uma chave RSA comumente tem 2048 bits de largura e, como dito na Seção 2.3.1, 4097 qubits seriam necessários para a quebra. Esse valor é, contudo, teórico. Na realidade, qubits extra precisam ser adicionados para mitigar erros nas leituras dos qubits, ou rodar o mesmo experimento diversas vezes.

No TLS 1.3, uma implementação do RSA é usada apenas para a autenticação do usuário, não para a troca de chaves. Portanto o problema da fatoração teria efeito limitado no protocolo. A troca de chaves no TLS 1.3 ocorre por Diffie-Hellman com curvas elípticas e o uso de computação quântica para decriptação não é trivial. Além da não trivialidade, o Capítulo 4 mostrou a dificuldade de depender de serviços em nuvem — dificuldade presente independente do algoritmo utilizado. Como computadores quânticos são máquinas muito grandes e caras, o serviço em nuvem acaba sendo a única opção viável na maioria dos casos.

Por outro lado, o Capítulo 4 mostrou também uma grande facilidade de implementação de algoritmos quânticos e uma rica biblioteca de algoritmos prontos para execução. Essa visão é otimista para o futuro e pode se tornar pessimista, no sentido de facilitar ataques. Mas outra tecnologia compete fortemente com o algoritmo de Shor para a quebra da criptografia assimétrica, a computação quântica adiabática. Considerando a Tabela 3, esse trabalho conseguiu fatorar um número de 6 bits usando um simulador em um computador pessoal, o que mostra um grande avanço tecnológico desde 2012. Mas a computação adiabática sempre mostrou bons resultados na fatoração de números, o que pode indicar uma vantagem no uso dessa tecnologia em detrimento do uso do algoritmo de Shor em computadores quânticos universais na maior parte das situações.

5.5 RECURSOS PARA UM ATAQUE *STORE-NOW-DECRYPT-LATER*

Usando os métodos de impersonificação e quebra de confidencialidade, o agente malicioso pode conseguir acesso a informações valiosas (dados pessoais, dados bancários, dados médicos, senhas, informações familiares, etc). Mas o agente de ameaça pode estar querendo alguma informação mais imediata, sobre algo que está acontecendo agora. Para tal, deverá coletar os pacotes encriptados das comunicações que possam lhe interessar e esperar até que esteja em posse de um computador quântico poderoso o suficiente para decriptar os pacotes recolhidos. A Tabela 7 mostra o armazenamento necessário para guardar uma hora de pacotes capturados de alguns dos *sites* mais utilizados no mundo e também faz uma projeção desses números para períodos mais longos de captura.

Tabela 7 – Estimando recursos de armazenamento para um ataque SNDL.

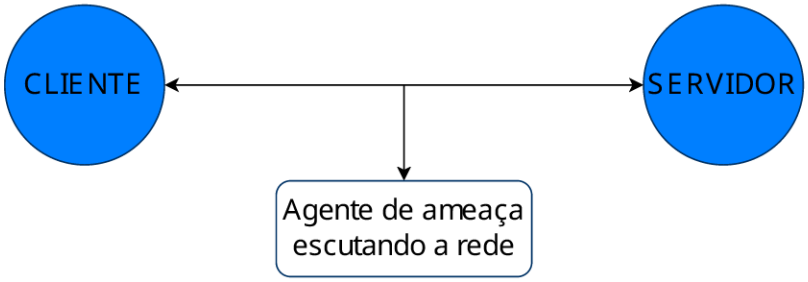
Site	**1 h de Pacotes Capturados (*MB*)**	**Armazenamento Esperado Para 24 h de Captura (*GB*)**	**Armazenamento Esperado Para 1 y de Captura (*TB*)**
instagram.com	835.4	19.6	7
youtube.com	723.7	17	6
amazon.com	272.6	6.4	2.3
gmail.com	124.8	2.9	1

Fonte – (MOURA; GIRON; CUSTODIO, 2023)

Todas as capturas foram feitas com o programa *Wireshark* filtrando-se pelo IP do servidor. A simulação de ataque imita um ataque do homem-do-meio (*man-in-the-middle*), capturando os pacotes enviados entre cliente e servidor. Esses pacotes coletados deveriam então ser armazenados em um disco rígido local ou em nuvem por uma certa quantidade de tempo. O tempo que esses pacotes devem permanecer armazenados depende de quando o agressor estará em posse de um computador quântico. A Figura 17 exemplifica o ataque de homem-do-meio. Na figura, a conexão se dá entre cliente (Client) e servidor (Server) por uma rede insegura (linhas desenhadas), e há também um terceiro indivíduo, o ator de ameaça, escutando e armazenando os pacotes (Threat Actor Listening).

Pode-se inferir que ataques SNDL custam bastante em recursos de armazenamento, a depender da natureza das informações capturadas, do tempo total de captura e do número de vítimas. A Tabela 7 considera apenas uma vítima, caso o ataque seja feito a cem usuários, o número seria cem vezes maior e assim por diante. Contudo, vale perceber que um número exato não é possível, visto que diferentes usuários têm diferentes perfis de navegação.

Figura 17 – Ataque homem-do-meio por meio do Wireshark.



Fonte – Elaborado pelo autor.

5.6 MÉTODOS PARA A MITIGAÇÃO DAS AMEAÇAS

Criptografia quântica

Criptografia quântica é o nome dado ao uso da física quântica para criar uma nova classe de criptografia. O exemplo mais simples dessa classe seria usar da superposição quântica para criar um número perfeitamente aleatório, mas o exemplo mais discutido dessa classe é a Distribuição Quântica de Chaves (QKD).

Protocolos de QKD comumente usam uma corrente de fótons polarizados através de um cabo de fibra ótica como qubits e são baseados no princípio da incerteza de Heisenberg ou no emaranhamento quântico. Baseados no princípio da incerteza, existem os protocolos: Bennett-Brassard 1984 (BB84) (BRASSARD; BENNETT, 1984), que é o protocolo principal dessa categoria, mas existem novas implementações, como o BB92 (BENNETT, 1992), "*Six-State Protocol*" (SSP) (BECHMANN-PASQUINUCCI; GISIN, 1999), "*Differential Phase Shift*" (DPS) (INOUE; WAKS; YAMAMOTO, 2002), Scarani-Acin-Ribordy-Gisim 2002 (SARG02) (GROSSHANS; GRANGIER, 2002) e SARG04 (SCARANI *et al.*, 2004). Baseados em emaranhamento, há os protocolos: Ekert 1991 (E91) (EKERT, A. K., 1991), Bennett-Brassard-Mermin 1992 (BBM92) (BENNETT; BRASSARD, Gilles; MERMIN, 1992), protocolo de Enzer (ENZER *et al.*, 2002) e "*Coherent One-Way*" (COW) (GISIN *et al.*, 2004).

- Prós de implementar QKD:
  - a matemática da mecânica quântica garante que a troca de chaves é perfeitamente segura;
  - a propriedade de não-cópia da mecânica quântica garante que não haverá ataques de homem-do-meio, pois uma medida do sistema modificaria o sistema.
- Contras de implementar QKD:
  - a propriedade de não-cópia impossibilita re-rotear ou realizar um

broadcast de um qubit, tornando necessário o desenvolvimento de *hardwares* e canais de rede especiais;
  - é afetado pela decoerência e distâncias muito longas podem ser impossíveis. A maioria dos sistemas de QKD atuais não permitem conexões com mais de 200 km de distância (XU *et al.*, 2023);
  - sua implementação tem um custo imenso para redes muito grandes, fazendo dessa uma solução viável apenas para um limitado número de casos.

Criptografia pós-quântica

Ao método de implementar uma matemática que não é facilmente resolvida por um computador quântico em um *hardware* clássico, dá-se o nome de Criptografia Pós-Quântica (PQC). O "*National Institute os Standardization and Technology*" (NIST) anunciou, em 2022, quatro algoritmos promissores de PQC: CRYSTALS-Kyber (BOS, J. *et al.*, 2018), um mecanismo de encapsulamento de chaves que pode ser usado para estabelecer chaves simétricas para o TLS ou outros protocolos; CRYSTALS-Dilithium (DUCAS *et al.*, 2018), um algoritmo de assinatura digital; Falcon (FOUQUE *et al.*, 2018), outro método de assinatura digital; e SPHINCS+ (BERNSTEIN, D. J. *et al.*, 2019), um método de assinatura digital baseado em hash.

- Prós de implementar PQC:
  - uma solução mais viável para troca de chaves do que a QKD;
  - há também implementações para assinatura digital.
- Contras de implementar PQC:
  - algoritmos de PQC tem sido testados por anos, mas ainda é impossível afirmar, com precisão, o nível de segurança deles para os próximos anos;
  - a maioria dos algoritmos de PQC são mais lentos ou requerem chaves maiores do que os algoritmos mais conhecidos (como ECDSA, RSA ou ECDHE) para troca de chaves e assinatura digital, o que causa um maior tempo de carregamento de páginas e risco de perda de pacotes.

Há também uma classe de implementações híbridas, que combinam um algoritmo pós-quântico a um algoritmo clássico para atenuar ou evitar possíveis falhas de segurança na aplicação de apenas um dos métodos. Esses algoritmos híbridos podem funcionar tanto para a troca de chaves, como para assinaturas digitais. Para exemplificar, o esquema de uma troca de chaves híbrida pode ser feita combinando-se as saídas de um algoritmo clássico e de um algoritmo pós-quântico com uma operação XOR. Um exemplo de uma aplicação mais prática envolvendo o TLS 1.3 pode

ser encontrado em (STEBILA; FLUHRER; GUERON, 2023). E, para o caso de uma assinatura digital híbrida, essa pode ser realizada com uma dupla assinatura, uma com um algoritmo clássico e outra com um algoritmo pós-quântico (BEULLENS *et al.*, 2021).

Algumas outras implementações, no que tange a segurança do TLS, são o Mecanismo de Encapsulamento de Chaves Pós-Quântico, do inglês "*Post-Quantum Key-Encapsulation Mechanism*" (PQKEM) (SCHWABE; STEBILA; WIGGERS, 2020), ou a Criptografia Baseada em Reticulados, do inglês "*Lattice-Based Cryptography*" (BOS, J. W. *et al.*, 2015).

Diversas ações que diminuem a vantagem do atacante

Além de todas as alternativas apresentadas, os métodos a seguir são opções que não são uma solução definitiva, mas que podem prejudicar o retorno sobre o investimento do atacante, talvez fazendo com que o atacante desista do ataque. Consultando a Tabela 2 — Requerimentos de hardware quântico para diferentes implementações do algoritmo de Shor — é possível perceber que o tamanho da chave a ser quebrada influencia diretamente o número de qubits e de portas lógicas necessárias para um ataque. Isso acaba por aumentar não só o preço do ataque, mas também o tempo de processamento, podendo acontecer do tempo de processamento total acabe se tornando mais longo do que o tempo de coerência dos qubits. Segundo (LI, K.; CAI, 2021), um ataque quântico ao RSA seria impraticável se a chave tivesse mais do que 8 KB.

O protocolo de concordância de chaves "*Extended Triple Diffie-Hellman*" (X3DH) (MARLINSPIKE; PERRIN, 2016), presente no protocolo Signal, promove múltiplas trocas de chaves em paralelo, o que pode aumentar o custo de um ataque, uma vez que o atacante quântico necessitaria quebrar cada uma das camadas. Algoritmos como este, que adicionam camadas de encriptação, são um tipo de defesa por dificultarem o ataque. Também pode ser uma ideia válida tunelar a comunicação TLS para uma "*Virtual Private Network*" (VPN) encriptada diferente (RUNGE, 2023).

Além de aumentar o custo do ataque, uma outra estratégia que pode ser adotada é diminuir a quantidade de informação recuperada em cada ataque — o que, indiretamente, faz com que o ataque tenha que ser feito mais vezes, aumentando seu custo também. Controles de segurança, como "*Perfect Forward Secrecy*" (PFS) e "*Post Compromise Security*" (PCS), são medidas que podem limitar o acesso a informação por parte dos atacantes (RUNGE, 2023). Outros exemplos são o "*Double Ratchet Key Management Algorithm*" (PERRIN; MARLINSPIKE, 2016), também parte do protocolo Signal, e os certificados de tempo curto, como presentes no protocolo ACME (SHEFFER *et al.*, 2020). Certificados de tempo curto são boas escolhas por diminuírem o intervalo de tempo que o atacante tem para quebrar o algoritmo do certificado e realizar

a impersonificação.

Ataques do tipo SNDL são mais urgentes, uma vez que indivíduos maliciosos podem estar armazenando dados agora para conseguirem ler no futuro. Um modo para as organizações decidirem o melhor momento para migrarem do uso de criptografia clássica para pós-quântica (ou *quantum-safe*) é dado por (MOSCA; PIANI, 2022b):

- **tempo de prateleira:** tempo em que as chaves criptográficas devem permanecer seguras;
- **tempo de migração:** tempo necessário para implementar um conjunto de ferramentas seguras contra ataques quânticos;
- **tempo de colapso:** tempo restante até que os computadores quânticos possam quebrar a segurança vigente.

A organização deve então pensar que o tempo de prateleira, somado ao tempo de migração, não deve ultrapassar o tempo de colapso. De outra maneira, quanto maior for esta diferença, maior a segurança ou o risco para a organização. E a organização também deve estar atenta, desde já, a ataques de engenharia social. Pois, como visto no quadro 7, o custo em armazenamento para o ataque SNDL é alto e o indivíduo malicioso deve querer filtrar melhor quais pacotes coletar.

6 CONCLUSÃO

Ao que se pode observar pelo modelo de ameaça, a capacidade de realizar um ataque por meio ou com o auxílio de um computador quântico não virá toda de uma vez. Ao contrário, certas classes atingirão esse patamar mais cedo do que outras. Para o TLS, um dos mais conhecidos e usados protocolos de segurança de redes, todos os seus modos apresentam vulnerabilidades em frente à ameaça quântica. Principalmente a respeito da criptografia assimétrica, atacando algoritmos de assinaturas digitais, trocas de chaves, impersonificando e ouvindo conversas privadas. Por essa razão é importante adotar os diversos métodos de mitigação da ameaça, estudando-os mais profundamente para reduzir prejuízos e estar atento a ataques SNDL. Um agente malicioso pode estar agindo desde já, esperando estar em posse de um computador quântico dentro dos próximos anos. Mas, como visto, os recursos de armazenamento necessários para um ataque SNDL são grandes, então as organizações devem estar atentas a ataques de engenharia social que ladrões podem usar para filtrar melhor quais pacotes valem a pena recolher. Um outro adendo nesse sentido é que os ataques SNDL não estão restritos à computação quântica, uma vez que a computação em geral avança (tanto em *hardware*, quanto em *software*) e uma tecnologia que hoje garante segurança pode não fornecer a mesma segurança daqui a alguns anos.

Quanto às ferramentas mapeadas e às dificuldades que um atacante enfrentaria ao realizar um ataque, os resultados foram promissores para o surgimento de novas ferramentas nos próximos anos. A programação de computadores quânticos está usando linguagens de alto nível, com bibliotecas que facilitam bastante o desenvolvimento. Dentro dos próximos anos, especialmente nas próximas décadas, podemos esperar uma quantidade de bibliotecas ainda maior, então a dificuldade de um ataque não deverá residir na dificuldade de escrever os códigos. A maior dificuldade encontrada por um atacante, levando em consideração os experimentos realizados, será, possivelmente, o tempo de espera para utilizar os serviços em nuvem. Para contornar isso, os agentes maliciosos podem decidir comprar ou alugar um computador quântico. No entanto, principalmente nos estágios iniciais da computação quântica, os valores de comprar ou alugar um *hardware* quântico são muito elevados, o que diminui o retorno sobre o investimento ou até levantam um alarme sobre transações bancárias de quantias elevadas para este fim.

Tendo em vista o acima concluído e sem deixar de pensar de forma crítica os próprios resultados, o estudo abre espaço para maiores discussões, em diversas frentes — a abrangência de conjecturas sobre o tema é imensa. Este trabalho pode ser ampliado ou atualizado, gerando trabalhos futuros, nos seguintes aspectos:

- Experimentar com mais ferramentas para entender melhor os desafios de um ataque quântico real. Existem dezenas de ferramentas (dentre *frameworks*,

simuladores, compiladores, e demais) disponíveis e experimentar com uma amplia o conhecimento, mas não abrange o conhecimento completamente.

- Atualizar o mapeamento, esperando encontrar novas ferramentas. Como dito no parágrafo anterior, o cenário é promissor para o surgimento de novas ferramentas. Isso faz com que o mapeamento tenha que ser refeito de tempos em tempos.
- Aprofundar a análise das ameaças à segurança da criptografia simétrica. O estudo focou na ameaça à criptografia assimétrica por ser a que apresenta maior risco. Mas algoritmos, como o de Grover, apresentam um ganho quadrático em ataques de força bruta e, apesar de não ser uma ameaça tão forte, ainda assim é uma ameaça.
- Expandir o estudo para outros algoritmos, como o de Grover e o de Simon. Além do algoritmo de Shor e implementações de computação adiabática para o mesmo fim, há ainda outros algoritmos que podem ser usados para um ataque. Por exemplo, o algoritmo de Grover (ameaça funções de hash e ataques de força bruta) e o algoritmo de Simon (ameaça cifras de bloco).
- Investigar as ameaças que áreas não diretamente ligadas à segurança podem causar à área da segurança da informação em geral. Como por exemplo, o uso de *machine learning* em ataques. Computadores quânticos apresentam um futuro promissor para a área de inteligência artificial e esse ganho pode ser usado tanto para o bem, como para o mal.

A APÊNDICE A — GRUPOS

Seja G um conjunto e $\cdot$ uma operação. O grupo $(G, \cdot)$ possui as propriedades:

- $\forall p, q \in G, p \cdot q \in G$.
- $\exists e \in G : \forall p \in G, e \cdot p = p \cdot e = p$.
- $\forall p \in G, \exists q \in G : p \cdot q = q \cdot p = e$.
- $\forall p, q, r \in G, (p \cdot q) \cdot r = p \cdot (q \cdot r)$.
- Adicionalmente, o grupo é abeliano se $\forall p, q \in G, p \cdot q = q \cdot p$.

Para facilitar a representação, deste ponto em diante a operação sobre o grupo seguirá a notação multiplicativa. Assim, $p \cdot q$ será representado apenas por pq e a inversa de p será p^{-1}. Ao aplicar a operação sob um grupo uma quantidade m de vezes: $ppp...pp = p^m$. Valem também algumas propriedades:

- $p^m p^{m'} = p^{m+m'}$
- $(p^m)^{m'} = p^{mm'}$
- $p^1 = p$
- $p^0 = e$
- Se for abeliano: $p^m q^m = pq^m$

Usando as propriedades de grupos abelianos, podemos concluir que, se $ac = bc$, $a = b$ $(a, b, c \in G)$. Como mostra a equação (8).

$$a = acc^{-1} = bcc^{-1} = b \quad \blacksquare \tag{8}$$

Considere G um grupo abeliano finito e $|G| = N$ a ordem deste grupo. Este grupo se diz cíclico quando seus elementos são $\{p^0, p^1, p^2, ..., p^{N-1}\} = \{e, p^1, p^2, ..., p^{N-1}\}$ e N é o menor positivo não nulo em que $p^N = e$. Em grupos cíclicos vale que $p^x = p^{(x \mod N)}$ e podemos demonstrar que $p^x = p^y \Leftrightarrow x = y \mod N$. Vide equação (9).

$$p^x = p^y \Rightarrow p^x p^{-y} = p^{x-y} = e \tag{9}$$

O que só pode ser válido se $x = y \mod N$, pois segue a validade da equação (10).

$$p^x = p^{(x \mod N)} = p^y = p^{(y \mod N)} \quad \blacksquare \tag{10}$$

B APÊNDICE B — CURVAS ELÍPTICAS

Seja um campo F, uma **equação de Weierstrass** possui a forma descrita pela equação (11).

$$y^2z = x^3 + axz^2 + bz^3 \tag{11}$$

Para $char(F) \neq 2, 3$ e o discriminante da equação é descrito pela equação (12)

$$\Delta = 4a^3 + 27b^2 \tag{12}$$

Chamaremos de **equação de Weierstrass simplificada** o caso em que $z = 1$. Resultando na equação (13)

$$y^2 = x^3 + ax + b \tag{13}$$

Consideremos o campo K e $F \supseteq K$. Por definição, representamos como na equação (14).

$$E(F) = \{e\} \cup \{(x,y) \in F \times F : y^2 = x^3 + ax + b\} \tag{14}$$

Onde e é a identidade, também conhecida por ponto no infinito.

A curva elíptica E definida no campo F e uma operação $\cdot$ definem um grupo abeliano $(E(F), \cdot)$ com todas as suas propriedades. Esta operação é comumente chamada de soma ou lei aditiva de curvas elípticas e está definida entre dois pontos $p, q \in E(F)$ de modo que, usando a notação multiplicativa, pq $(p \cdot q)$ é calculada traçando-se uma reta sobre p e q, esta reta atravessará também um ponto $r \in E(F)$, então, $pq = r^{-1}$ é o valor r espelhado ao redor da abscissa.

A lei aditiva contém algumas propriedades:

- Sendo e o ponto no infinito, $pe = p$.
- Se $p_x = q_x$, $pq = pp^{-1} = e$.
- Se $p_y = 0$, $p^2 = e$.
- Se $p, q \in E(F)$, a soma $pq \in E(F) \setminus \{e\}$.

Para o caso em que $pq \in E(F) \setminus \{e\}$, temos duas possibilidades: quando $p \neq q$ e quando $p = q$. Começando pela mais simples, quando $p \neq q$, temos uma reta cuja tangente é dada pela equação (15).

$$m = \frac{q_x - p_x}{q_y - p_y} \tag{15}$$

Como p é um ponto conhecido da reta, cuja equação é $y = mx + c$, podemos definir que $c = p_y - mp_x$. Substituindo, temos que a equação da reta é dada pela equação (16).

$$y = mx + p_y - mp_x = m(x - p_x) + p_y \tag{16}$$

E os pontos da curva elíptica que interceptam a reta são dados pela equação (17)

$$y^2 = (m(x - p_x) + p_y)^2 = x^3 + ax + b \tag{17}$$

Alternativamente, podemos abrir as contas. Resultando na equação (18)

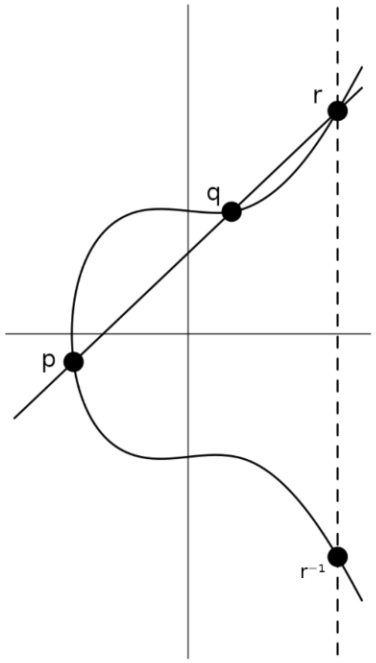
$$x^3 - m^2x^2 + (a + 2m^2p_x - 2mp_y)x + b + 2mp_xp_y - m^2p_x^2 - p_y^2 = 0 = (x - p_x)(x - q_x)(x - r_x) \quad (18)$$

E percebemos que $m^2 = p_x + q_x + r_x$. O que implica que, sendo p_x e q_x conhecidos, podemos concluir a equação (19)

$$r_x = m^2 - p_x - q_x \quad (19)$$

Por fim, $r^{-1} = pq$ tem coordenadas $(m^2 - p_x - q_x, m(p_x - r_x) - p_y)$.

Figura 18 – Lei aditiva de curvas elípticas.



Fonte – (WASHINGTON, 2008)

Para a segunda possibilidade, quando $p = q$, não temos um segundo ponto para traçarmos uma reta, mas podemos calcular uma distância infinitesimal seguindo as equações (20)(21)(22)

$$d(y^2)dy = d(x^3 + ax + b)dx \quad (20)$$

$$2ydy = (3x^2 + a)dx \quad (21)$$

$$m = \frac{dy}{dx} = \frac{3x^2 + a}{2y} \quad (22)$$

O procedimento restante é similar. A equação da reta tangente é a mesma, mas com o valor de m atualizado, o que significa que a equação (18), considerando $p = q$, deve ficar na forma da equação (23).

$$x^3 - m^2x^2 + (a + 2m^2p_x - 2mp_y)x + b + 2mp_xp_y - m^2p_x^2 - p_y^2 = 0 = (x - p_x)^2(x - r_x) \quad (23)$$

Que resulta num $m^2 = 2p_x + r_x$ e $r_x = m^2 - 2p_x$. Concluindo que agora $r^{-1} = pq$ tem as coordenadas $(m^2 - 2p_x,\ m(p_x - r_x) - p_y)$.

Explorando mais além, o campo K pode ser infinito ou finito, mas vamos imaginar um número primo N e $F_N \supseteq K$, tal que $F_N = \{0, ..., N - 1\}$. Por definição, temos a equação (24).

$$E(F_N) = \{e\} \cup \{(x,y) \in F_N \times F_N : y^2 = x^3 + ax + b \mod N\} \quad (24)$$

C APÊNDICE C — TROCA DE CHAVES DIFFIE-HELLMAN

Para resolver o problema de estabelecer uma chave simétrica entre Alice e Bob em uma rede insegura, a versão mais moderna do Diffie-Hellman define (WASHINGTON, 2008):

1. Alice e Bob concordam em uma curva elíptica E e num campo finito F_N, tal que o problema do logaritmo discreto[1] em $E(F_N)$ seja considerado difícil. Então também escolhem um $p \in E(F_N)$ de modo que o subgrupo gerado por p tenha um primo largo como ordem.
2. Alice escolhe um inteiro a e envia ap para Bob.
3. Bob faz o mesmo, escolhe um inteiro b e envia bp para Alice.
4. Alice calcula abp e Bob calcula bap.
5. Por fim, ambos utilizam algum método para extrair uma chave de abp.

As únicas informações que trafegam na rede são a curva E, o campo F_N, os pontos p, ap e bp.

Sabendo disso, um indivíduo malicioso, em posse de $p, ap, bp \in E(F_N)$, deve calcular abp para extrair a chave simétrica. Tal desafio é conhecido como o **Problema Diffie-Hellman**.

Uma das maneiras de encontrar abp é usando p e ap para encontrar a através de logaritmo discreto. Sendo p um gerador de $E(F_N) = \{p^0, p^1, p^2, ..., p^{N-1}\}$, há um único $x \in \mathbb{Z}_N : \forall h \in E(F_N), p^x = h$. Então chamamos $x = \log_g h$ de **logaritmo discreto de h em respeito a p**[2].

[1] O problema do logaritmo discreto é um problema computacional relacionado à dificuldade de encontrar a solução para a equação $a^x \equiv b \mod p$, onde a, b e p são conhecidos.

[2] Note que $x \in \mathbb{Z}_N$, portanto, se $x \geq N$, o que vale é $x \mod N$.

D APÊNDICE D — PORTA QUÂNTICA DE HADAMARD

A porta de Hadamard é uma porta de 1-qubit que atua criando superposição igual dos dois estados base. A equação (25) mostra a porta de Hadamard sendo aplicada ao qubit $|0\rangle$ e a equação (26) mostra a porta de Hadamard sendo aplicada ao qubit $|1\rangle$.

$$H|0\rangle = \frac{1}{2^{1/2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}\begin{pmatrix}1\\ 0\end{pmatrix} = \frac{1}{2^{1/2}}\begin{pmatrix}1\\ 1\end{pmatrix} = \frac{1}{2^{1/2}}(|0\rangle + |1\rangle) = |+\rangle \tag{25}$$

$$H|1\rangle = \frac{1}{2^{1/2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}\begin{pmatrix}0\\ 1\end{pmatrix} = \frac{1}{2^{1/2}}\begin{pmatrix}1\\ -1\end{pmatrix} = \frac{1}{2^{1/2}}(|0\rangle - |1\rangle) = |-\rangle \tag{26}$$

Ademais, é fácil verificar que:

- $H|+\rangle = |0\rangle$
- $H|-\rangle = |1\rangle$
- Portanto: $H^{2k} = I, \forall k \in \mathbb{N}$

E APÊNDICE E — PORTA QUÂNTICA R_N CONTROLADA

Essa é uma porta de múltiplos qubits, aplicada sobre o qubit alvo se o qubit controlado estiver no estado $|1\rangle$. Sua forma matricial é dada pela equação (27).

$$R_n = \begin{pmatrix} 1 & 0 \\ 0 & e^{2\pi i/2^n} \end{pmatrix} \tag{27}$$

F APÊNDICE F — PORTA QUÂNTICA *SWAP*

A porta *Swap*, como o nome indica, troca dois qubits entre si. Ela age da seguinte forma, descrita pela equação (28).

$$Swap(|q_0\rangle \otimes |q_1\rangle) = |q_1\rangle \otimes |q_0\rangle \tag{28}$$

E sua forma matricial é dada pela equação (29)

$$Swap = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{29}$$

G APÊNDICE G — TRANSFORMADA DE FOURIER QUÂNTICA

Conhecida em inglês por "*Quantum Fourier Transform*" (QFT), é a implementação da transformada discreta de Fourier, do inglês "*Discrete Fourier Transform*" (DFT), sobre as amplitudes das funções quânticas de onda. A transformada clássica mapeia um vetor $(x_0, x_1, ..., x_{N-1}) \in \mathbb{C}^N$ a um vetor $(y_0, y_1, ..., y_{N-1}) \in \mathbb{C}^N$ de acordo com a equação (30).

$$y_k = \frac{1}{N^{1/2}} \sum_{j=0}^{N-1} x_j e^{2\pi ijk/N} \tag{30}$$

Similarmente, a QFT mapeia um estado quântico $|X\rangle = \sum_{j=0}^{N-1} x_j |j\rangle$ a $|Y\rangle = \sum_{k=0}^{N-1} y_k |k\rangle$ de acordo com a equação (31).

$$y_k = \frac{1}{N^{1/2}} \sum_{j=0}^{N-1} x_j e^{2\pi ijk/N} \tag{31}$$

E, uma vez que $e^{2\pi i/N}$ é uma rotação, a transformada inversa é dada pela equação (32).

$$x_k = \frac{1}{N^{1/2}} \sum_{j=0}^{N-1} y_j e^{-2\pi ijk/N} \tag{32}$$

No caso em que $|x\rangle$ é uma base, temos a equação (33).

$$|x\rangle \mapsto \frac{1}{N^{1/2}} \sum_{k=0}^{N-1} e^{2\pi ixk/N} |k\rangle \tag{33}$$

Por exemplo, para 1 qubit, $N = 2^n = 2$. Então temos as equações (34)(35).

$$|0\rangle \mapsto \frac{1}{2^{1/2}} \sum_{k=0}^{1} e^{\pi i0k} |k\rangle = \frac{1}{2^{1/2}} e^0 |0\rangle + \frac{1}{2^{1/2}} e^0 |1\rangle = |+\rangle \tag{34}$$

$$|1\rangle \mapsto \frac{1}{2^{1/2}} \sum_{k=0}^{1} e^{\pi i1k} |k\rangle = \frac{1}{2^{1/2}} e^0 |0\rangle + \frac{1}{2^{1/2}} e^{\pi i} |1\rangle = |-\rangle \tag{35}$$

Para mais qubits, podemos tratar k como um número binário $k = \sum_{i=1}^{n} k_i 2^{n-1} = \sum_{i=1}^{n} k_i 2^n/2^i = N \sum_{i=1}^{n} k_i/2^i$ e mapear de acordo com a equação (36).

$$|x\rangle \mapsto \frac{1}{N^{1/2}} \sum_{k=0}^{N-1} e^{2\pi ix \sum_{i=1}^{n} k_i/2^i} |k_1...k_n\rangle = \frac{1}{N^{1/2}} \sum_{k=0}^{N-1} \prod_{i=1}^{n} e^{2\pi ixk_i/2^i} |k_1...k_n\rangle \tag{36}$$

Agora, expandindo o somatório $\sum_{k=0}^{N-1} = \sum_{k_1=0}^{1} ... \sum_{k_n=0}^{1}$, temos o produto tensorial definido na equação (37).

$$|x\rangle \mapsto \frac{1}{N^{1/2}} \bigotimes_{i=1}^{n} \left(|0\rangle + e^{2\pi ix/2^i} |1\rangle \right) \tag{37}$$

Para exemplificar, tomemos o número 5, representado 101 em binário. Neste caso, $n = 3$ e $N = 8$. Temos a equação (38).

$$|101\rangle \mapsto \frac{1}{8^{1/2}} \left(|0\rangle + e^{2\pi i 5/2}|1\rangle\right) \otimes \left(|0\rangle + e^{2\pi i 5/4}|1\rangle\right) \otimes \left(|0\rangle + e^{2\pi i 5/8}|1\rangle\right) \tag{38}$$

A transformada inversa é trivial, como explicado anteriormente, costuma-se chamá-la $QFT^{\dagger}$ (QFT dagger).

Construindo um circuito para a transformada:

Sabemos que $y_k = \frac{1}{N^{1/2}} \sum_{j=0}^{N-1} x_j e^{2\pi i j k/N}$. Vamos tentar para $N = 2$:

$$y_0 = \frac{1}{2^{1/2}} \sum_{j=0}^{1} x_j = \frac{1}{2^{1/2}}(x_0 + x_1)$$

$$y_1 = \frac{1}{2^{1/2}} \sum_{j=0}^{1} x_j e^{\pi i j} = \frac{1}{2^{1/2}}(x_0 + e^{\pi i} x_1)$$

Para $N = 3$:

$$y_0 = \frac{1}{3^{1/2}} \sum_{j=0}^{2} x_j = \frac{1}{3^{1/2}}(x_0 + x_1 + x_2)$$

$$y_1 = \frac{1}{3^{1/2}} \sum_{j=0}^{2} x_j e^{2\pi i j/3} = \frac{1}{3^{1/2}}(x_0 + e^{2\pi i/3} x_1 + e^{4\pi i/3} x_2)$$

$$y_2 = \frac{1}{3^{1/2}} \sum_{j=0}^{2} x_j e^{2\pi i j 2/3} = \frac{1}{3^{1/2}}(x_0 + e^{4\pi i/3} x_1 + e^{8\pi i/3} x_2)$$

Para $N = 4$:

$$y_0 = \frac{1}{2} \sum_{j=0}^{3} x_j = \frac{1}{2}(x_0 + x_1 + x_2 + x_3)$$

$$y_1 = \frac{1}{2} \sum_{j=0}^{3} x_j e^{\pi i j/2} = \frac{1}{2}(x_0 + e^{\pi i/2} x_1 + e^{\pi i} x_2 + e^{\pi i 3/2} x_3)$$

$$y_2 = \frac{1}{2} \sum_{j=0}^{3} x_j e^{\pi i j} = \frac{1}{2}(x_0 + e^{\pi i} x_1 + e^{\pi i 2} x_2 + e^{\pi i 3} x_3)$$

$$y_3 = \frac{1}{2} \sum_{j=0}^{3} x_j e^{\pi i j 3/2} = \frac{1}{2}(x_0 + e^{\pi i 3/2} x_1 + e^{\pi i 3} x_2 + e^{\pi i 9/2} x_3)$$

Podemos montar uma matriz para cada um desses casos. Para $N = 4$ a matriz ficaria:

$$\frac{1}{2} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & e^{\pi i/2} & e^{\pi i} & e^{\pi i 3/2} \\ 1 & e^{\pi i} & e^{\pi i 2} & e^{\pi i 3} \\ 1 & e^{\pi i 3/2} & e^{\pi i 3} & e^{\pi i 9/2} \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{pmatrix} = \frac{1}{2} \begin{pmatrix} x_0 + x_1 + x_2 + x_3 \\ x_0 + e^{\pi i/2} x_1 + e^{\pi i} x_2 + e^{\pi i 3/2} x_3 \\ x_0 + e^{\pi i} x_1 + e^{\pi i 2} x_2 + e^{\pi i 3} x_3 \\ x_0 + e^{\pi i 3/2} x_1 + e^{\pi i 3} x_2 + e^{\pi i 9/2} x_3 \end{pmatrix}$$

De modo geral, para um N qualquer, temos a equação (39).

$$\frac{1}{N^{1/2}}\begin{pmatrix} 1 & 1 & 1 & 1 & \cdots & 1 \\ 1 & e^{2\pi i/N} & e^{2\pi ij/N} & e^{2\pi ij/N} & \dots & e^{2\pi ij/N} \\ 1 & e^{2\pi ik/N} & e^{2\pi ijk/N} & e^{2\pi ijk/N} & \dots & e^{2\pi ijk/N} \\ 1 & e^{2\pi ik/N} & e^{2\pi ijk/N} & e^{2\pi ijk/N} & \dots & e^{2\pi ijk/N} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & e^{2\pi ik/N} & e^{2\pi ijk/N} & e^{2\pi ijk/N} & \dots & e^{2\pi ijk/N} \end{pmatrix}\begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_{N-1} \end{pmatrix} = \frac{1}{N^{1/2}}\begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_{N-1} \end{pmatrix} \tag{39}$$

onde $j = 2, 3, ..., N-1$ é a coluna da matriz e $k = 2, 3, ..., N-1$ é a linha.

Para que que possa ser implementada num computador quântico, as portas devem ser matrizes unitárias. Assim, devemos provar que suas colunas são ortonormais. Tomando duas colunas j e j', temos a equação (40).

$$\langle QFT_{j'}|QFT_j\rangle = \frac{1}{N}\sum_{k=0}^{N-1} e^{2\pi i(j-j')k/N} \tag{40}$$

Para o caso em que $j - j' = 0$, temos a equação (41).

$$\langle QFT_{j'}|QFT_j\rangle = \frac{1}{N}\sum_{k=0}^{N-1} 1 = 1 \tag{41}$$

Para o caso em que $j - j' \neq 0$, temos uma progressão geométrica cujo primeiro termo é $\frac{1}{N}$ e a razão é $e^{2\pi i(j-j')/N}$. Com isso podemos somar os termos da progressão geométrica finita, obtendo a equação (42).

$$\frac{1}{N}\sum_{k=0}^{N-1} e^{2\pi i(j-j')k/N} = \frac{\frac{1}{N}\left(1 - e^{2\pi i(j-j')}\right)}{1 - e^{2\pi i(j-j')/N}} \tag{42}$$

Mas podemos avaliar que a expressão $e^{2\pi i(j-j')} = \cos 2\pi(j-j') + i \sin 2\pi(j-j') = 1$

Assim, determinar que vale a equação (43).

$$\langle QFT_{j'}|QFT_j\rangle = \frac{1}{N}\sum_{k=0}^{N-1} e^{2\pi i(j-j')k/N} = 0 \quad \blacksquare \tag{43}$$

Para construirmos um circuito, analisamos que os estados têm superposição e fase relativa. Isso indica que devemos usar portas Hadamard e a expressão $e^{2\pi ijk/N}$ contida na matriz se assemelha a $e^{2\pi i/N}$ contida na porta R_n. Se começarmos com um registrador de n qubits $|q_1 q_2 ... q_n\rangle$ e aplicarmos a porta H (Hadamard) ao primeiro qubit, teremos a equação (44).

$$|q_1\rangle \mapsto \frac{1}{2^{1/2}}(|0\rangle + e^{\pi i q_1}) \tag{44}$$

A porta R_k irá multiplicar o qubit $|1\rangle$ por $e^{2\pi i/2^k}$ caso $q_k = 1$. Se aplicarmos R_k $n-1$ vezes, controlando de q_2 a q_n, teremos a equação (45).

$$|q_1\rangle \mapsto \frac{1}{2^{1/2}}(|0\rangle + e^{\pi i q_1} e^{\pi i q_2/2} e^{\pi i q_3/4} ... e^{\pi i q_n/2^{n-1}} |1\rangle)$$

$$|q_1\rangle \mapsto \frac{1}{2^{1/2}}(|0\rangle + e^{\pi i (q_1 + q_2/2 + q_3/4 + ... + q_n/2^{n-1})} |1\rangle)$$

$$|q_1\rangle \mapsto \frac{1}{2^{1/2}}(|0\rangle + e^{\pi i (q_1 2^{n-1} + q_2 2^{n-2} + q_3 2^{n-3} + ... + q_n)/2^{n-1}} |1\rangle)$$

$$|q_1\rangle \mapsto \frac{1}{2^{1/2}}(|0\rangle + e^{2\pi i q/2^n} |1\rangle) \tag{45}$$

onde $q = q_1 2^{n-1} + q_2 2^{n-2} + q_3 2^{n-3} + ... + q_n$.

Aplicando os mesmos passos para o segundo qubit, com a diferença de que a porta R_k será aplicada $n-2$ vezes, temos a equação (46)

$$|q_2\rangle \mapsto \frac{1}{2^{1/2}}(|0\rangle + e^{2\pi i q/2^{n-1}} |1\rangle) \tag{46}$$

Aplicando para o terceiro qubit, a porta R_k será aplicada $n-3$ vezes e assim por diante, de modo que obtemos a equação (47).

$$|q\rangle \mapsto \frac{1}{2^{1/2}}(|0\rangle + e^{\frac{2\pi i q}{2^n}} |1\rangle) \otimes (|0\rangle + e^{\frac{2\pi i q}{2^{n-1}}} |1\rangle) \otimes (|0\rangle + e^{\frac{2\pi i q}{2^{n-2}}} |1\rangle) \otimes ... \otimes (|0\rangle + e^{\frac{2\pi i q}{2^1}} |1\rangle) \tag{47}$$

O resultado é muito semelhante ao que já mostramos anteriormente, com a diferença que a ordem dos qubits está trocada. Contudo, isso pode ser resolvido com portas *Swap*. ■

A transformada de Fourier quântica desempenha um papel muito importante no algoritmo de Shor. Por esse motivo vale a pena calcular a sua complexidade, que, em termos de computação quântica, é dada pelo número de portas que seu circuito possui. No primeiro qubit, n portas são aplicadas; no segundo, $n-1$; no terceiro, $n-2$. Repetindo esse padrão até que no último qubit apenas uma porta é aplicada, resultando na progressão definida na equação (48).

$$n + (n-1) + (n-2) + ... + 1 = \frac{n(n+1)}{2} = \frac{n^2 + n}{2} \tag{48}$$

Também há as c portas *Swap* ao final do circuito. Então temos a equação (49).

$$complexidade = \frac{n^2 + n}{2} + c \tag{49}$$

No entanto, a quantidade de portas *Swap* não ultrapassa $n/2$ e o termo dominante define a complexidade do pior caso como $O(n^2)$.

No melhor caso, com apenas um qubit, a transformada é apenas uma porta Hadamard e tem complexidade $\Omega(1)$.

H APÊNDICE H — ALGORITMO DE SHOR

Consideremos a função $f(a) = x^a \mod N$, com N composto pelos primos pq, x entre 1 e N coprimo de N, $a \geq 0$. Vale lembrar que estamos lidando com teoria dos números, então, com inteiros. Para $N = 21$, os valores possíveis de x são $S = \{2, 4, 5, 8, 10, 11, 13, 16, 17, 19, 20\}$. Escolhendo x = 2 podemos montar a tabela:

Tabela 8 – Evolução de f(a).

a	0	1	2	3	4	5	6
$\mathbf{x^a}$	1	2	4	8	16	32	64
f(a)	1	2	4	8	16	11	1

Aqui podemos notar claramente que o período de $f(a) = 2^a \mod 21$ é 6. Mas podemos pensar num modo mais elegante de encontrar o período r da função considerando que $x^0 = x^r = 1 \mod N$. Daí podemos desenvolver a equação (50).

$$x^r = (x^{r/2})^2 = 1 \mod N \tag{50}$$

Que nos leva à equação (51)

$$(x^{r/2})^2 - 1 = (x^{r/2} + 1)(x^{r/2} - 1) = 0 \mod N \tag{51}$$

Com isso, concluímos que $(x^{r/2} + 1)(x^{r/2} - 1)|N$, o que significa que $(x^{r/2} + 1)$ ou $(x^{r/2} - 1)$ tem um fator em comum com N. Então $gcd((x^{r/2} + 1), N)$ revelará um fator de N ou $gcd((x^{r/2} - 1), N)$ o fará, onde a função $gcd(n1, n2)$ é encontrar o máximo divisor em comum entre $n1$ e $n2$.

O algoritmo de Shor pode ser dividido em três partes elementares: transformar o problema da fatoração em um problema de busca de período, que pode ser feito em um computador clássico; encontrar o período aplicando a transformada de Fourier quântica; e usar o período para descobrir os fatores. Sendo que a segunda parte é a crucial e a que promete uma melhora no desempenho.

Passos para o algoritmo:

1. Usando um computador clássico.
    a) Para colocarmos o algoritmo em funcionamento, precisamos de um inteiro x em $]1, N[$ que seja coprimo de N. Ou seja, precisamos que $gcd(x, N) = 1$ e, em caso falso, rejeitar.
2. Usar o computador quântico para definir o período da função.
    a) Inicializar dois registradores quânticos com $n = lgT$ qubits cada, onde lg é o logaritmo na base 2 e T é uma potência de 2 maior do que N. Equação (52).

$$\psi_0 = |0\rangle^{\otimes n} |0\rangle^{\otimes n} \tag{52}$$

b) Aplicar portas H no primeiro registrador. Equação (53).

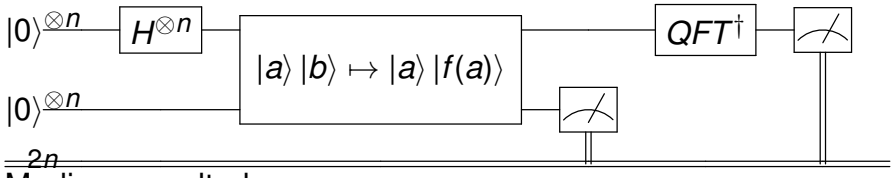
$$\psi_1 = \frac{1}{T^{1/2}} \sum_{a=0}^{T-1} |a\rangle |0\rangle^{\otimes n} \tag{53}$$

c) Aplicar um funcional que mapeie $|a\rangle |b\rangle \mapsto |a\rangle |f(a)\rangle$ no segundo registrador. Equação (54).

$$\psi_2 = \frac{1}{T^{1/2}} \sum_{a=0}^{T-1} |a\rangle |f(a)\rangle \tag{54}$$

d) Medir o segundo registrador. O resultado será um dos valores para $f(a)$. Então, projetar o estado quântico ao estado que estava emaranhado ao resultado.

e) Aplicar $QFT^\dagger$ ao circuito.



f) Medir os resultados.

3. De volta a um computador clássico.

a) O resultado será uma lista de valores em binário que devemos transformar em decimal.

b) Então, cada resultado R do circuito, dividido por T, será uma fase.

c) Cada fase deverá ser reduzida à fração mais próxima em que o denominador seja menor que N.

d) Uma lista com todos os denominadores r das fases é criada, excluindo os valores ímpares de r. Equação (55).

$$r \approx k\frac{T}{R}, k \in \mathbb{Z} \tag{55}$$

e) Por fim, $gcd((x^{r/2}+1), N)$ ou $gcd((x^{r/2}-1), N)$ devem revelar um fator de N, para algum valor possível de r.

I APÊNDICE I — QUBIT EM COORDENADAS ESFÉRICAS

Um qubit é matematicamente representado pela equação (1) $|\psi\rangle = a|0\rangle + b|1\rangle$ e queremos demonstrar que a equação (3) $|\psi\rangle = \cos\frac{\varphi}{2}|0\rangle + e^{i\theta}\sin\frac{\varphi}{2}|1\rangle$ também é uma representação válida.

Sendo $a, b \in \mathbb{C}$, é possível usar a identidade de Euler e representar um qubit pela equação (56). Então multiplicar um fator de fase global ao qubit, como visto na equação (57).

$$|\psi\rangle = |a|e^{i\varphi_a}|0\rangle + |b|e^{i\varphi_b}|1\rangle \tag{56}$$

$$|\psi\rangle = |a|\,|0\rangle + |b|e^{i\varphi_b - i\varphi_a}|1\rangle = |a|\,|0\rangle + |b|e^{i\delta}|1\rangle \tag{57}$$

Com $\delta = \varphi_b - \varphi_a$ (um grau de restrição; apenas a fase relativa importa).

Expandindo com a fórmula de Euler obtemos a equação (58).

$$|\psi\rangle = |a|\,|0\rangle + |b|\cos\delta\,|1\rangle + i|b|\sin\delta\,|1\rangle \tag{58}$$

Então podemos fazer:

- $X = |b|\cos\delta$
- $Y = |b|\sin\delta$
- $Z = |a|$

Ao que se obtém a equação (59).

$$|\psi\rangle = Z|0\rangle + (X + iY)|1\rangle \tag{59}$$

Para manter a normalização (um grau de restrição; normalização), a equação (60) deve ser verdadeira.

$$1 = |z|^2 + |X + iY|^2 = |X|^2 + |Y|^2 + |Z|^2 \tag{60}$$

Então percebe-se que $|X|^2 + |Y|^2 + |Z|^2 = 1$ é a equação de uma esfera unitária e as amplitudes do estado quântico de um qubit estão contidas nela (R = 1). Para transformar a equação (59) em coordenadas esféricas, usa-se:

- $X = R\sin\theta\cos\varphi$
- $Y = R\sin\theta\sin\varphi$
- $Z = R\cos\theta$

E obtém-se a equação (61).

$$|\psi\rangle = \cos\theta|0\rangle + \sin\theta(\cos\varphi + i\sin\varphi)|1\rangle = \cos\theta|0\rangle + \sin\theta e^{i\varphi}|1\rangle \tag{61}$$

A equação (61) é muito parecida com a equação (3), mas é necessário substituir θ por $\theta/2$. Isso é feito para que $\theta = 0 \Rightarrow |\psi\rangle = |0\rangle$ e $\theta = \pi \Rightarrow |\psi\rangle = |1\rangle$ (ou $|\psi\rangle = e^{i\varphi}|1\rangle$).

■

REFERÊNCIAS

AARONSON, Scott. The limits of quantum. **Scientific American**, JSTOR, v. 298, n. 3, p. 62–69, 2008.

ALGHADEER, Mohammed; ALDAWSARI, Eid; SELVARAJAN, Raja; ALUTAIBI, Khaled; KAIS, Sabre; ALHARBI, Fahhad H. Psitrum: An Open Source Simulator for Universal Quantum Computers, 2022. Publisher: arXiv Version Number: 2.

ALI, Shaukat; YUE, Tao. Modeling Quantum programs: challenges, initial results, and research directions. en. *In*: PROCEEDINGS of the 1st ACM SIGSOFT International Workshop on Architectures and Paradigms for Engineering Quantum Software. Virtual USA: ACM, nov. 2020. p. 14–21.

ALYAMI, Hashem; NADEEM, Mohd; ALOSAIMI, Wael; ALHARBI, Abdullah; KUMAR, Rajeev; GUPTA, Bineet Kumar; AGRAWAL, Alka; KHAN, Raees Ahmad. Analyzing the Data of Software Security Life-Span: Quantum Computing Era. **INTELLIGENT AUTOMATION AND SOFT COMPUTING**, v. 31, n. 2, p. 707–716, 2022. Publisher: TECH SCIENCE PRESS 871 CORONADO CENTER DR, SUTE 200, HENDERSON, NV 89052 USA. ISSN 1079-8587.

ANGARA, Prashanti Priya; STEGE, Ulrike; MACLEAN, Andrew; MÜLLER, Hausi A.; MARKHAM, Tom. Teaching Quantum Computing to High-School-Aged Youth: A Hands-On Approach. **IEEE Transactions on Quantum Engineering**, v. 3, p. 1–15, 2022. Conference Name: IEEE Transactions on Quantum Engineering. ISSN 2689-1808.

ANGARA, Prashanti Priya; STEGE, Ulrike; MÜLLER, Hausi A.; BOZZO-REY, Mehdi. Hybrid quantum-classical problem solving in the NISQ era. *In*: PROCEEDINGS of the 30th Annual International Conference on Computer Science and Software Engineering. USA: IBM Corp., nov. 2020. (CASCON '20), p. 247–252.

ANTIPOV, A. V.; KIKTENKO, E. O.; FEDOROV, A. K. Efficient realization of quantum primitives for Shor's algorithm using PennyLane library. en. Edição: Pietro Massignan. **PLOS ONE**, v. 17, n. 7, e0271462, jul. 2022. ISSN 1932-6203.

AOUN, Mohamed Raed El; LI, Heng; KHOMH, Foutse; TIDJON, Lionel. Bug Characteristics in Quantum Software Ecosystem, 2022. Publisher: arXiv Version Number: 1.

BEAUREGARD, Stephane. Circuit for Shor's algorithm using 2n+ 3 qubits. **arXiv preprint quant-ph/0205095**, 2002.

BECHMANN-PASQUINUCCI, Helle; GISIN, Nicolas. Incoherent and coherent eavesdropping in the six-state protocol of quantum cryptography. **Physical Review A**, APS, v. 59, n. 6, p. 4238, 1999.

BECKMAN, David; CHARI, Amalavoyal N; DEVABHAKTUNI, Srikrishna; PRESKILL, John. Efficient networks for quantum factoring. **Physical Review A**, APS, v. 54, n. 2, p. 1034, 1996.

BENIOFF, Paul. Quantum mechanical Hamiltonian models of Turing machines. **Journal of Statistical Physics**, Springer, v. 29, n. 3, p. 515–546, 1982.

BENNETT, Charles H. Quantum cryptography using any two nonorthogonal states. **Physical review letters**, APS, v. 68, n. 21, p. 3121, 1992.

BENNETT, Charles H; BRASSARD, Gilles; MERMIN, N David. Quantum cryptography without Bell's theorem. **Physical review letters**, APS, v. 68, n. 5, p. 557, 1992.

BERNSTEIN, Daniel J. Introduction to post-quantum cryptography. *In*: POST-QUANTUM cryptography. [*S.l.*]: Springer, 2009. p. 1–14.

BERNSTEIN, Daniel J; HÜLSING, Andreas; KÖLBL, Stefan; NIEDERHAGEN, Ruben; RIJNEVELD, Joost; SCHWABE, Peter. The SPHINCS+ signature framework. *In*: PROCEEDINGS of the 2019 ACM SIGSAC conference on computer and communications security. [*S.l.*: *s.n.*], 2019. p. 2129–2146.

BERNSTEIN, Ethan; VAZIRANI, Umesh. Quantum Complexity Theory. **SIAM Journal on Computing**, v. 26, n. 5, p. 1411–1473, 1997.

BERTELS, Koen; SARKAR, Aritra; ASHRAF, Imran. Quantum Computing—From NISQ to PISQ. **IEEE Micro**, v. 41, n. 5, p. 24–32, 2021.

BEULLENS, Ward; D'ANVERS, Jan-Pieter; HÜLSING, Andreas T; LANGE, Tanja; PANNY, Lorenz; SAINT GUILHEM, Cyprien de; SMART, Nigel P. **Post-Quantum Cryptography: Current state and quantum mitigation**. [*S.l.*], 2021.

BONE, Simon; CASTRO, Matias. A brief history of quantum computing. **Imperial College in London, http://www. doc. ic. ac. uk/˜ nd/surprise_97/journa l/vol4/spb3**, 1997.

BOS, Joppe W.; COSTELLO, Craig; NAEHRIG, Michael; STEBILA, Douglas. Post-Quantum Key Exchange for the TLS Protocol from the Ring Learning with Errors Problem. *In*: 2015 IEEE Symposium on Security and Privacy. [*S.l.*: *s.n.*], 2015. p. 553–570.

BOS, Joppe; DUCAS, Léo; KILTZ, Eike; LEPOINT, Tancrède; LYUBASHEVSKY, Vadim; SCHANCK, John M; SCHWABE, Peter; SEILER, Gregor; STEHLÉ, Damien. CRYSTALS-Kyber: a CCA-secure module-lattice-based KEM. *In*: IEEE. 2018 IEEE European Symposium on Security and Privacy (EuroS&P). [*S.l.*: *s.n.*], 2018. p. 353–367.

BRANDHOFER, Sebastian; DEVITT, Simon; POLIAN, Ilia. ArsoNISQ: Analyzing Quantum Algorithms on Near-Term Architectures. *In*: 2021 IEEE European Test Symposium (ETS). [*S.l.*: *s.n.*], mai. 2021. p. 1–6. ISSN: 1558-1780.

BRASSARD, G; BENNETT, Charles H. Quantum cryptography: Public key distribution and coin tossing. *In*: INTERNATIONAL conference on computers, systems and signal processing. [*S.l.*: *s.n.*], 1984. p. 175–179.

BUREK, Elżbieta; WROŃSKI, Michał; MAŃK, Krzysztof; MISZTAL, Michał. Algebraic attacks on block ciphers using quantum annealing. **IEEE Transactions on Emerging Topics in Computing**, IEEE, v. 10, n. 2, p. 678–689, 2022.

CHAN, Chia-ling; FONTUGNE, Romain; CHO, Kenjiro; GOTO, Shigeki. Monitoring TLS adoption using backbone and edge traffic. *In*: IEEE. IEEE INFOCOM 2018-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS). [*S.l.*: *s.n.*], 2018. p. 208–213.

CHARETON, Christophe; BARDIN, Sébastien; LEE, Dongho; VALIRON, Benoît; VILMART, Renaud; XU, Zhaowei. **Formal Methods for Quantum Programs: A**

Survey. [*S.l.*]: arXiv, abr. 2022. arXiv:2109.06493 [cs]. Disponível em: http://arxiv.org/abs/2109.06493. Acesso em: 22 out. 2022.

CHEN, Lily; JORDAN, Stephen; LIU, Yi-Kai; MOODY, Dustin; PERALTA, Rene; PERLNER, Ray; SMITH-TONE, Daniel. NIST: report on post-quantum cryptography. **NIST, Tech. Rep**, 2016.

COBB, Adrian; SCHNEIDER, Jean-Guy; LEE, Kevin. Towards Higher-Level Abstractions for Quantum Computing. en. *In*: AUSTRALASIAN Computer Science Week 2022. Brisbane Australia: ACM, fev. 2022. p. 115–124.

CÓRCOLES, Antonio D.; KANDALA, Abhinav; JAVADI-ABHARI, Ali; MCCLURE, Douglas T.; CROSS, Andrew W.; TEMME, Kristan; NATION, Paul D.; STEFFEN, Matthias; GAMBETTA, Jay M. Challenges and Opportunities of Near-Term Quantum Computing Systems. **Proceedings of the IEEE**, v. 108, n. 8, p. 1338–1352, ago. 2020. Conference Name: Proceedings of the IEEE. ISSN 1558-2256.

DA ROSA, Evandro Chagas Ribeiro; DE SANTIAGO, Rafael. Ket Quantum Programming. en. **ACM Journal on Emerging Technologies in Computing Systems**, v. 18, n. 1, p. 1–25, jan. 2022. ISSN 1550-4832, 1550-4840.

DE STEFANO, Manuel; PECORELLI, Fabiano; DI NUCCI, Dario; PALOMBA, Fabio; DE LUCIA, Andrea. Software Engineering for Quantum Programming: How Far Are We?, 2022. Publisher: arXiv Version Number: 2.

DEUTSCH, David. Quantum theory, the Church–Turing principle and the universal quantum computer. **Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences**, The Royal Society London, v. 400, n. 1818, p. 97–117, 1985.

DEUTSCH, David Elieser. Quantum computational networks. **Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences**, The Royal Society London, v. 425, n. 1868, p. 73–90, 1989.

DEY, Nivedita; GHOSH, Mrityunjay; KUNDU, Subhra Samir; CHAKRABARTI, Amlan. **QDLC – The Quantum Development Life Cycle**. [*S.l.*]: arXiv, out. 2020. arXiv:2010.08053 [cs]. Disponível em: http://arxiv.org/abs/2010.08053. Acesso em: 22 out. 2022.

DONG, Xiaoyang; WANG, Xiaoyun. Quantum key-recovery attack on Feistel structures. **Science China Information Sciences**, Springer, v. 61, p. 1–7, 2018.

DONKERS, Huub; MESMAN, Koen; AL-ARS, Zaid; MÖLLER, Matthias. QPack Scores: Quantitative performance metrics for application-oriented quantum computer benchmarking, 2022. Publisher: arXiv Version Number: 1.

DUCAS, Léo; KILTZ, Eike; LEPOINT, Tancrede; LYUBASHEVSKY, Vadim; SCHWABE, Peter; SEILER, Gregor; STEHLÉ, Damien. Crystals-dilithium: A lattice-based digital signature scheme. **IACR Transactions on Cryptographic Hardware and Embedded Systems**, p. 238–268, 2018.

EKERT, Artur K. Quantum cryptography based on Bell's theorem. **Physical review letters**, APS, v. 67, n. 6, p. 661, 1991.

ENZER, Daphna G; HADLEY, Phillip G; HUGHES, Richard J; PETERSON, Charles G; KWIAT, Paul G. Entangled-photon six-state quantum cryptography. **New Journal of Physics**, IOP Publishing, v. 4, n. 1, p. 45, 2002.

ESCANEZ-EXPOSITO, Daniel; CABALLERO-GIL, Pino; MARTIN-FERNANDEZ, Francisco. QuantumSolver: A quantum tool-set for developers, 2022. Publisher: arXiv Version Number: 1.

FEYNMAN, Richard P. Simulating physics with computers. **International journal of theoretical physics**, World Scientific, v. 21, n. 6/7, p. 467–488, 1982.

FOUQUE, Pierre-Alain *et al.* Falcon: Fast-Fourier lattice-based compact signatures over NTRU. **Submission to the NIST's post-quantum cryptography standardization process**, v. 36, n. 5, 2018.

FREITAS, Alessandro; FERREIRA, Marcello; SILVA FILHO, Olavo Leopoldino da. Uma Proposta de Ensino Investigativo Sobre a Física Moderna e Contemporânea: O Efeito Fotoelétrico. **Revista do Professor de Física**, v. 3, Especial, p. 37–38, 2019.

GIDNEY, Craig. Factoring with n+ 2 clean qubits and n-1 dirty qubits. **arXiv preprint arXiv:1706.07884**, 2017.

GIDNEY, Craig; EKERÅ, Martin. How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits. **Quantum**, Verein zur Förderung des Open Access Publizierens in den Quantenwissenschaften, v. 5, p. 433, 2021.

GILL, Sukhpal Singh; KUMAR, Adarsh; SINGH, Harvinder; SINGH, Manmeet; KAUR, Kamalpreet; USMAN, Muhammad; BUYYA, Rajkumar. Quantum computing: A taxonomy, systematic review and future directions. **Software: Practice and Experience**, Wiley Online Library, v. 52, n. 1, p. 66–114, 2022.

GISIN, Nicolas; RIBORDY, Grégoire; ZBINDEN, Hugo; STUCKI, Damien; BRUNNER, Nicolas; SCARANI, Valerio. Towards practical and fast quantum cryptography. **arXiv preprint quant-ph/0411022**, 2004.

GROSSHANS, Frédéric; GRANGIER, Philippe. Continuous variable quantum cryptography using coherent states. **Physical review letters**, APS, v. 88, n. 5, p. 057902, 2002.

GROVER, Lov K. A fast quantum mechanical algorithm for database search. *In*: PROCEEDINGS of the twenty-eighth annual ACM symposium on Theory of computing. [*S.l.*: *s.n.*], 1996. p. 212–219.

GUO, Jingzhe; LOU, Huazhe; LI, Riling; FANG, Wang; LIU, Junyi; LONG, Peixun; YING, Shenggang; YING, Mingsheng. isQ: Towards a Practical Software Stack for Quantum Programming. **arXiv preprint arXiv:2205.03866**, 2022.

GUSTAFSON, Erik *et al.* Large scale multi-node simulations of $\mathbb{Z}_2$ gauge theory quantum circuits using Google Cloud Platform. *In*: 2021 IEEE/ACM Second International Workshop on Quantum Computing Software (QCS). [*S.l.*: *s.n.*], nov. 2021. p. 72–79.

HAGOUEL, Paul Isaac; KARAFYLLIDIS, Ioannis G. Quantum computers: Registers, gates and algorithms. *In*: IEEE. 2012 28th International Conference on Microelectronics Proceedings. [*S.l.*: *s.n.*], 2012. p. 15–21.

HÄNER, Thomas; ROETTELER, Martin; SVORE, Krysta M. Factoring using 2n+ 2 qubits with Toffoli based modular multiplication. **arXiv preprint arXiv:1611.07995**, 2016.

HEESE, Raoul; BICKERT, Patricia; NIEDERLE, Astrid Elisa. Representation of binary classification trees with binary features by quantum circuits. en. **Quantum**, v. 6, p. 676, mar. 2022. ISSN 2521-327X.

HEN, Itay. Realizable quantum adiabatic search. **Europhysics Letters**, IOP Publishing, v. 118, n. 3, p. 30003, 2017.

HEVIA, Jose Luis; PETERSSEN, Guido; PIATTINI, Mario. QuantumPath: A quantum software development platform. **Software: Practice and Experience**, v. 52, n. 6, p. 1517–1530, 2022. Publisher: Wiley Online Library.

HIETALA, Kesha. A Verified Software Toolchain for Quantum Programming, 2022.

INOUE, Kyo; WAKS, Edo; YAMAMOTO, Yoshihisa. Differential phase shift quantum key distribution. **Physical review letters**, APS, v. 89, n. 3, p. 037902, 2002.

J., Abhijith *et al.* Quantum Algorithm Implementations for Beginners. en. **ACM Transactions on Quantum Computing**, v. 3, n. 4, p. 1–92, dez. 2022. ISSN 2643-6809, 2643-6817.

JIANG, Shuxian; BRITT, Keith A; MCCASKEY, Alexander J; HUMBLE, Travis S; KAIS, Sabre. Quantum annealing for prime factorization. **Scientific reports**, Nature Publishing Group UK London, v. 8, n. 1, p. 17667, 2018.

JUNIOR, Paulo Eduardo Zanni; CAMARGO, Valter Vieira de. A systematic mapping on quantum software development in the context of software engineering. **arXiv preprint arXiv:2106.00926**, 2021.

KAPLAN, Marc; LEURENT, Gaëtan; LEVERRIER, Anthony; NAYA-PLASENCIA, María. Breaking symmetric cryptosystems using quantum period finding. *In*: SPRINGER. ADVANCES in Cryptology–CRYPTO 2016: 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part II 36. [*S.l.*: *s.n.*], 2016. p. 207–237.

KHAN, Arif Ali; AKBAR, Muhammad Azeem; AHMAD, Aakash; FAHMIDEH, Mahdi; SHAMEEM, Mohammad; LAHTINEN, Valtteri; WASEEM, Muhammad; MIKKONEN, Tommi. Agile Practices for Quantum Software Development: Practitioners Perspectives. **arXiv preprint arXiv:2210.09825**, 2022.

KHAN, Tariq M; ROBLES-KELLY, Antonio. Machine learning: Quantum vs classical. **IEEE Access**, IEEE, v. 8, p. 219275–219294, 2020.

KIEFL, Niklas; HAGEL, Georg. Software Engineering Education of Classical Computing vs. Quantum Computing: A Competency-Centric Approach. en. *In*: PROCEEDINGS of the 4th European Conference on Software Engineering Education. Seeon/Bavaria Germany: ACM, jun. 2020. p. 27–31.

KIM, Jane; CHO, Seong-Min; SEO, Seung-Hyun. Comparison and Analysis of Quantum Software Simulators. *In*: p. 188–191.

KITCHENHAM, Barbara; CHARTERS, Stuart. **Guidelines for performing systematic literature reviews in software engineering**. [*S.l.*], 2007.

KRAWCZYK, Hugo; ERONEN, Pasi. **RFC 5869: HMAC-based extract-and-expand key derivation function (HKDF)**. [*S.l.*]: RFC Editor, 2010.

KUWAKADO, Hidenori; MORII, Masakatu. Quantum distinguisher between the 3-round Feistel cipher and the random permutation. *In*: IEEE. 2010 IEEE International Symposium on Information Theory. [*S.l.*: *s.n.*], 2010. p. 2682–2685.

LAROSE, Ryan *et al.* Mitiq: A software package for error mitigation on noisy quantum computers. en. **Quantum**, v. 6, p. 774, ago. 2022. ISSN 2521-327X.

LAUMANN, Christopher R; MOESSNER, Roderich; SCARDICCHIO, Antonello; SONDHI, Shivaji Lal. Quantum annealing: The fastest route to quantum computation? **The European Physical Journal Special Topics**, Springer, v. 224, n. 1, p. 75–88, 2015.

LI, Kai; CAI, Qing-yu. Practical security of RSA against NTC-architecture quantum computing attacks. **International Journal of Theoretical Physics**, Springer, v. 60, n. 8, p. 2733–2744, 2021.

LI, Zhaokai; DATTANI, Nikesh S; CHEN, Xi; LIU, Xiaomei; WANG, Hengyan; TANBURN, Richard; CHEN, Hongwei; PENG, Xinhua; DU, Jiangfeng. High-fidelity adiabatic quantum computation using the intrinsic Hamiltonian of a spin system: Application to the experimental factorization of 291311. **arXiv preprint arXiv:1706.08061**, 2017.

MANDRA, Salvatore; MARSHALL, Jeffrey; RIEFFEL, Eleanor G.; BISWAS, Rupak. HybridQ: A Hybrid Simulator for Quantum Circuits. *In*: 2021 IEEE/ACM Second International Workshop on Quantum Computing Software (QCS). St. Louis, MO, USA: IEEE, nov. 2021. p. 99–109.

MARLINSPIKE, Moxie; PERRIN, Trevor. The X3DH key agreement protocol. **Open Whisper Systems**, v. 283, p. 10, 2016.

MARTINA, Stefano; GHERARDINI, Stefano; BUFFONI, Lorenzo; CARUSO, Filippo. Noise fingerprints in quantum computers: Machine learning software tools. **Software Impacts**, v. 12, p. 100260, 2022. Publisher: Elsevier. ISSN 2665-9638.

MAVROEIDIS, Vasileios; VISHI, Kamer; ZYCH, Mateusz D; JØSANG, Audun. The impact of quantum computing on present cryptography. **arXiv preprint arXiv:1804.00200**, 2018.

MAXWELL, Joseph. Understanding and validity in qualitative research. **Harvard educational review**, Harvard Education Publishing Group, v. 62, n. 3, p. 279–301, 1992.

MCCASKEY, Alexander J; LYAKH, Dmitry I; DUMITRESCU, Eugene F; POWERS, Sarah S; HUMBLE, Travis S. XACC: a system-level software infrastructure for heterogeneous quantum–classical computing. **Quantum Science and Technology**, v. 5, n. 2, p. 024002, 2020. Publisher: IOP Publishing. ISSN 2058-9565.

METWALLI, Sara Ayman; VAN METER, Rodney. A Tool For Debugging Quantum Circuits, 2022. Publisher: arXiv Version Number: 1.

MINA-ZICU, Mina; SIMION, Emil. Threats to Modern Cryptography: Grover's Algorithm. Preprints, 2020.

MIRANSKYY, Andriy; KHAN, Mushahid; FAYE, Jean Paul Latyr; MENDES, Udson C. Quantum Computing for Software Engineering: Prospects. **arXiv preprint arXiv:2203.03575**, 2022.

MIRANSKYY, Andriy; ZHANG, Lei; DOLISKANI, Javad. On testing and debugging quantum software. **arXiv preprint arXiv:2103.09172**, 2021.

MOHAMMADBAGHERPOOR, Hamed; DREHER, Patrick; IBRAHIM, Mohannad; OH, Young-Hyun; HALL, James; STONE, Richard E.; STOJKOVIC, Mirela. **Exploring Airline Gate-Scheduling Optimization Using Quantum Computers**. [*S.l.*]: arXiv, nov. 2021. arXiv:2111.09472 [quant-ph]. Disponível em: `http://arxiv.org/abs/2111.09472`. Acesso em: 22 out. 2022.

MOSCA, Michele; PIANI, Marco. 2021 Quantum Threat Timeline Report, 2022.

MOSCA, Michele; PIANI, Marco. Quantum threat timeline report 2022. **Global Risk Institute, Toronto, ON**, 2022.

MOUEDDENE, Ahmed Abid; KHAMMASSI, Nader; BERTELS, Koen; ALMUDEVER, Carmen G. Realistic simulation of quantum computation using unitary and measurement channels. en. **Physical Review A**, v. 102, n. 5, p. 052608, nov. 2020. ISSN 2469-9926, 2469-9934.

MOURA, Luiz F A S; GIRON, Alexandre A; CUSTODIO, Ricardo F. Quantum Threats to the TLS 1.3 Protocol. *In*: IARIA. SECURWARE 2023, The Seventeenth International Conference on Emerging Security Information, Systems and Technologies. [*S.l.*: *s.n.*], 2023. p. 67–73.

MYKHAILOVA, Mariia; SVORE, Krysta M. Teaching Quantum Computing through a Practical Software-driven Approach: Experience Report. en. *In*: PROCEEDINGS of the 51st ACM Technical Symposium on Computer Science Education. Portland OR USA: ACM, fev. 2020. p. 1019–1025.

NAGARAJAN, Harsha; LOCKWOOD, Owen; COFFRIN, Carleton. QuantumCircuitOpt: An Open-source Framework for Provably Optimal Quantum Circuit Design. *In*: 2021 IEEE/ACM Second International Workshop on Quantum Computing Software (QCS). [*S.l.*: *s.n.*], nov. 2021. p. 55–63.

NGUYEN, Hoa T.; USMAN, Muhammad; BUYYA, Rajkumar. QFaaS: A Serverless Function-as-a-Service Framework for Quantum Computing, 2022. Publisher: arXiv Version Number: 1.

NGUYEN, Thien; MCCASKEY, Alexander J. Extending Python for Quantum-classical Computing via Quantum Just-in-time Compilation. en. **ACM Transactions on Quantum Computing**, v. 3, n. 4, p. 1–25, dez. 2022. ISSN 2643-6809, 2643-6817.

OLIVER, Jose Luis Hevia. Requirements for Quantum Software Platforms. *In*: p. 20–26.

OLIVIERI, Pierriccardo; ASKARPOUR, Mehrnoosh; NITTO, Elisabetta di. Experimental Implementation of Discrete Time Quantum Walk with the IBM Qiskit Library. *In*: 2021 IEEE/ACM 2nd International Workshop on Quantum Software Engineering (Q-SE). [*S.l.*: *s.n.*], jun. 2021. p. 33–38.

PAE, Chi-Un. Why systematic review rather than narrative review? **Psychiatry investigation**, Korean Neuropsychiatric Association, v. 12, n. 3, p. 417, 2015.

PALER, Alexandru; BASMADJIAN, Robert. Energy Cost of Quantum Circuit Optimisation: Predicting That Optimising Shor's Algorithm Circuit Uses 1 GWh. en. **ACM Transactions on Quantum Computing**, v. 3, n. 1, p. 1–14, mar. 2022. ISSN 2643-6809, 2643-6817.

PALTENGHI, Matteo; PRADEL, Michael. Bugs in Quantum computing platforms: an empirical study. en. **Proceedings of the ACM on Programming Languages**, v. 6, OOPSLA1, p. 1–27, abr. 2022. ISSN 2475-1421.

PEDURI, Anurudh; BHAT, Siddharth; GROSSER, Tobias. QSSA: an SSA-based IR for Quantum computing. en. *In*: PROCEEDINGS of the 31st ACM SIGPLAN International Conference on Compiler Construction. Seoul South Korea: ACM, mar. 2022. p. 2–14.

PENG, WangChun; WANG, BaoNan; HU, Feng; WANG, YunJiang; FANG, XianJin; CHEN, XingYuan; WANG, Chao. Factoring larger integers with fewer qubits via quantum annealing with optimized parameters. **SCIENCE CHINA Physics, Mechanics & Astronomy**, Springer, v. 62, p. 1–8, 2019.

PERRIN, Trevor; MARLINSPIKE, Moxie. The double ratchet algorithm. **GitHub wiki**, p. 10, 2016.

PETRENKO, Alexei. **Applied Quantum Cryptanalysis**. [*S.l.*]: CRC Press, 2023.

PRESKILL, John. Quantum computing and the entanglement frontier. **arXiv preprint arXiv:1203.5813**, 2012.

RESCORLA, E. **The Transport Layer Security (TLS) Protocol Version 1.3**. [*S.l.*], 2018.

ROY, Pradosh K. Quantum Logic Gates, ago. 2020.

RUNGE, Tilman. **Dismantling the Quantum Threat**. 2023. Tese (Doutorado) – Technische Hochschule Brandenburg.

SANCHEZ-RAMIREZ, Sergio; CONEJERO, Javier; LORDAN, Francesc; QUERALT, Anna; CORTES, Toni; BADIA, Rosa M; GARCIA-SAEZ, Artur. RosneT: A Block Tensor Algebra Library for Out-of-Core Quantum Computing Simulation. *In*: 2021 IEEE/ACM Second International Workshop on Quantum Computing Software (QCS). St. Louis, MO, USA: IEEE, nov. 2021. p. 1–8.

SANTOLI, Thomas; SCHAFFNER, Christian. Using Simon's algorithm to attack symmetric-key cryptographic primitives. **arXiv preprint arXiv:1603.07856**, 2016.

SCARANI, Valerio; ACIN, Antonio; RIBORDY, Grégoire; GISIN, Nicolas. Quantum cryptography protocols robust against photon number splitting attacks for weak laser pulse implementations. **Physical review letters**, APS, v. 92, n. 5, p. 057901, 2004.

SCEKIC, Marija; YAKARYILMAZ, Abuzer. Comparing Quantum Software Development Kits for Introductory Level Education. **Baltic Journal of Modern Computing**, v. 10, n. 1, p. 87–104, 2022. Publisher: University of Latvia. ISSN 2255-8942.

SCHMITT, Bruno; DE MICHELI, Giovanni. tweedledum: a compiler companion for quantum computing. *In*: PROCEEDINGS of the 2022 Conference & Exhibition on Design, Automation & Test in Europe. Leuven, BEL: European Design e Automation Association, mar. 2022. (DATE '22), p. 7–12.

SCHWABE, Peter; STEBILA, Douglas; WIGGERS, Thom. Post-Quantum TLS Without Handshake Signatures. *In*: PROCEEDINGS of the 2020 ACM SIGSAC Conference on Computer and Communications Security. Virtual Event, USA: Association for Computing Machinery, 2020. (CCS '20), p. 1461–1480.

SHEFFER, Yaron; LOPEZ, Diego; DIOS, Oscar Gonzalez de; PASTOR, Antonio; FOSSATI, Thomas. Support for short-term, automatically renewed (STAR) certificates in the automated certificate management environment (ACME). **RFC 8739**, 2020.

SHI, Tai-Rong; JIN, Chen-Hui; HU, Bin; GUAN, Jie; CUI, Jing-Yi; WANG, Sen-Peng. Complete analysis of Simon's quantum algorithm with additional collisions. **Quantum Information Processing**, Springer, v. 18, n. 11, p. 334, 2019.

SHOR, Peter W. Algorithms for quantum computation: discrete logarithms and factoring. *In*: IEEE. PROCEEDINGS 35th annual symposium on foundations of computer science. [*S.l.*: *s.n.*], 1994. p. 124–134.

SIKERIDIS, Dimitrios; KAMPANAKIS, Panos; DEVETSIKIOTIS, Michael. Assessing the overhead of post-quantum cryptography in TLS 1.3 and SSH. *In*: PROCEEDINGS of the 16th International Conference on emerging Networking EXperiments and Technologies. [*S.l.*: *s.n.*], 2020. p. 149–156.

SIKERIDIS, Dimitrios; KAMPANAKIS, Panos; DEVETSIKIOTIS, Michael. Post-quantum authentication in TLS 1.3: a performance study. **Cryptology ePrint Archive**, 2020.

SMITH, Robert S.; PETERSON, Eric C.; SKILBECK, Mark G.; DAVIS, Erik J. **An Open-Source, Industrial-Strength Optimizing Compiler for Quantum Programs**. [*S.l.*]: arXiv, mar. 2020. arXiv:2003.13961 [quant-ph]. Disponível em: `http://arxiv.org/abs/2003.13961`. Acesso em: 22 out. 2022.

SODHI, Balwinder; KAPUR, Ritu. Quantum Computing Platforms: Assessing the Impact on Quality Attributes and SDLC Activities. *In*: 2021 IEEE 18th International Conference on Software Architecture (ICSA). [*S.l.*: *s.n.*], mar. 2021. p. 80–91.

SOEKEN, Mathias; HAENER, Thomas; ROETTELER, Martin. Programming quantum computers using design automation. *In*: IEEE. 2018 Design, Automation & Test in Europe Conference & Exhibition (DATE). [*S.l.*: *s.n.*], 2018. p. 137–146.

SOUZA, Paulo J. P.; MENDONÇA, Taysa M.; OLIVEIRA, Estêvão V. B. de; VILLAS-BOAS, Celso J. Computação Quântica Adiabática: Do Teorema Adiabático ao Computador da D-Wave. **Revista Brasileira de Ensino de Física**, Sociedade Brasileira de Física, v. 43, e20210049, 2021. ISSN 1806-1117.

STEBILA, Douglas; FLUHRER, Scott; GUERON, Shay. **Hybrid key exchange in TLS 1.3**. [*S.l.*], fev. 2023. Work in Progress. Disponível em: `https://datatracker.ietf.org/doc/draft-ietf-tls-hybrid-design/06/`.

SUCHARA, Martin; FARUQUE, Arvin; LAI, Ching-Yi; PAZ, Gerardo; CHONG, Frederic T; KUBIATOWICZ, John. Comparing the overhead of topological and concatenated quantum error correction. **arXiv preprint arXiv:1312.2316**, 2013.

SUO, Jingwen; WANG, Licheng; YANG, Sijia; ZHENG, Wenjie; ZHANG, Jiankang. Quantum algorithms for typical hard problems: a perspective of cryptanalysis. **Quantum Information Processing**, Springer, v. 19, p. 1–26, 2020.

TAKAHASHI, Yasuhiro; KUNIHIRO, Noboru. A quantum circuit for Shor's factoring algorithm using 2n+ 2 qubits. **Quantum Information & Computation**, Rinton Press, Incorporated Paramus, NJ, v. 6, n. 2, p. 184–192, 2006.

VALENCIA, David; GARCIA-ALONSO, Jose; ROJO, Javier; MOGUEL, Enrique; BERROCAL, Javier; MURILLO, Juan Manuel. Hybrid classical-quantum software services systems: Exploration of the rough edges. *In*: SPRINGER. INTERNATIONAL Conference on the Quality of Information and Communications Technology. [*S.l.*: *s.n.*], 2021. p. 225–238.

VEDRAL, Vlatko; BARENCO, Adriano; EKERT, Artur. Quantum networks for elementary arithmetic operations. **Physical Review A**, APS, v. 54, n. 1, p. 147, 1996.

VEMULA, Dinesh Reddy; KONAR, Debanjan; SATHEESAN, Sudeep; KALIDASU, Sri Mounica; CANGI, Attila. A Scalable 5,6-Qubit Grover's Quantum Search Algorithm, 2022. Publisher: arXiv Version Number: 1.

VOGT, Sebastian; FUNKE, Holger. How Quantum Computers threat security of PKIs and thus eIDs. **Open Identity Summit 2021**, Gesellschaft für Informatik eV, 2021.

WANG, Sherry; ADAMS, Carlisle; BROADBENT, Anne. Password authentication schemes on a quantum computer. *In*: 2021 IEEE International Conference on Quantum Computing and Engineering (QCE). [*S.l.*: *s.n.*], out. 2021. p. 346–350.

WANG, Shuangbao Paul; SAKK, Eric. Quantum Algorithms: Overviews, Foundations, and Speedups. *In*: 2021 IEEE 5th International Conference on Cryptography, Security and Privacy (CSP). [*S.l.*: *s.n.*], jan. 2021. p. 17–21.

WASHINGTON, Lawrence C. **Elliptic curves: number theory and cryptography**. [*S.l.*]: Chapman e Hall/CRC, 2008.

WROŃSKI, Michał. Practical solving of discrete logarithm problem over prime fields using quantum annealing. *In*: SPRINGER. COMPUTATIONAL Science–ICCS 2022: 22nd International Conference, London, UK, June 21–23, 2022, Proceedings, Part IV. [*S.l.*: *s.n.*], 2022. p. 93–106.

WU, Sau Lan *et al.* Application of quantum machine learning using the quantum variational classifier method to high energy physics analysis at the LHC on IBM quantum computer simulator and hardware with 10 qubits. **Journal of Physics G: Nuclear and Particle Physics**, v. 48, n. 12, p. 125003, dez. 2021. ISSN 0954-3899, 1361-6471.

XU, Guobin; MAO, Jianzhou; SAKK, Eric; WANG, Shuangbao Paul. An Overview of Quantum-Safe Approaches: Quantum Key Distribution and Post-Quantum Cryptography. *In*: IEEE. 2023 57th Annual Conference on Information Sciences and Systems (CISS). [*S.l.*: *s.n.*], 2023. p. 1–6.

YARKONI, Sheir; RAPONI, Elena; BÄCK, Thomas; SCHMITT, Sebastian. Quantum annealing for industry applications: Introduction and review. **Reports on Progress in Physics**, IOP Publishing, 2022.

YU, Chao-Hua. Experimental Implementation of Quantum Algorithm for Association Rules Mining. **IEEE Journal on Emerging and Selected Topics in Circuits and Systems**, v. 12, n. 3, p. 676–684, set. 2022. Conference Name: IEEE Journal on Emerging and Selected Topics in Circuits and Systems. ISSN 2156-3365.

YUNAKOVSKY, Sergey E *et al.* Towards security recommendations for public-key infrastructures for production environments in the post-quantum era. **EPJ Quantum Technology**, Springer Berlin Heidelberg, v. 8, n. 1, p. 14, 2021.

ZAMBONI, Maurizio; GRAZIANO, Mariagrazia; TURVANI, Ph D Giovanna; RAGGI, Lorenzo. Arithmetic circuits for quantum computing: a software library, 2020.

ZHAO, Pengzhan; ZHAO, Jianjun; MA, Lei. Identifying Bug Patterns in Quantum Programs. *In*: 2021 IEEE/ACM 2nd International Workshop on Quantum Software Engineering (Q-SE). [*S.l.*: *s.n.*], jun. 2021. p. 16–21.