Universidade de Brasília – UnB
Faculdade de Tecnologia - FT
Curso de Engenharia Elétrica

AVALIAÇÃO DE MÉTODOS CRIPTOGRÁFICOS QUÂNTICO-RESISTENTES NO *WIREGUARD*

Autor: Higor Gabriel Araújo A. de Souza
Orientador: William Ferreira Giozza
Co-orientador: Robson de Oliveira Albuquerque

Brasília, Distrito Federal
2021

HIGOR GABRIEL ARAÚJO ALVES DE SOUZA

UNIVERSIDADE DE BRASÍLIA

Faculdade de Tecnologia
Departamento de Engenharia Elétrica

AVALIAÇÃO DE MÉTODOS CRIPTOGRÁFICOS QUÂNTICO-RESISTENTES NO *WIREGUARD*

Trabalho submetido ao curso de graduação Engenharia Elétrica da Universidade de Brasília como Trabalho de Conclusão de Curso para obtenção do Título de Bacharel em Engenharia Elétrica.

Banca Examinadora

Dr. William Ferreira Giozza, UnB/ENE _____
Orientador
Dr. Robson de Oliveira Albuquerque, UnB/ENE _____
Co-orientador
Dr. João José Costa Gondim, UnB/CIC _____
Membro Convidado

Brasília, 09 de novembro de 2021

AGRADECIMENTOS

Agradeço primeiramente a Deus, pelos milagres que fez em minha vida, que me permitiram chegar até aqui, e por ser a força que continua me movendo todos os dias de maneiras misteriosas que sequer sou capaz de compreender completamente.

Agradeço à minha família por todo o suporte me dado, sem o qual não poderia estar onde estou. Aos meus avós Isaura e João, por quem faço tudo o que faço, agradeço por todo o amor de uma vida. Agradeço à minha mãe, Graça, pela minha vida e por sempre ter me proporcionado o mundo ao seu alcance, sabendo que sou a razão pela qual ela faz tudo o que faz e fez mais do que pôde muitas vezes. Agradeço ao meu pai João, por sempre me dar todo o suporte e ser o meu exemplo nos estudos; e mais importante, agradeço a Deus pela sua vida e por poder estar presente neste momento tão importante para mim.

Agradeço à Universidade de Brasília, nome que poderei carregar com muito orgulho como estandarte do ensino de qualidade do qual pude me beneficiar, onde, muito mais do que aluno e profissional, me tornei cidadão, com pensamento crítico e uma ampla visão de mundo, que me permitem ver o mundo com novos olhos e reconhecer o meu potencial de transformá-lo. Agradeço a todos os meus professores que fizeram parte da minha trajetória, que acreditaram de alguma forma em mim e compartilharam comigo todo o seu conhecimento. Um agradecimento em especial aos meus orientadores William e Robson, por terem me aceitado e me guiado nesta etapa tão difícil e importante, e por terem a paciência de construir este trabalho comigo. Agradeço também aos meus, mais do que chefes, amigos André Serpa e Marcelo Buz, por todo o incentivo, por me abrirem as portas que precisei, por acreditarem em mim e por me fazerem acreditar em mim também.

Agradeço a todos os meus queridos colegas e amigos que me acompanharam por todos estes anos. Agradecimentos em especial aos meus tão amados amigos Daniel Clark (*in memoriam),* Tiago Menezes, Ana Machado e Ludmila Pimenta, por me acompanharem desde a jornada de ingresso na Universidade, pelas companhias naturais até demais, por estarem ao meu lado me inspirando diariamente e por compartilharem e construírem comigo muito do que sou hoje. Aos meus amigos Bryan e Nara por todo o amor, por cuidarem tão bem de mim, e por sempre estarem comigo nos momentos que mais precisei, mantendo minhas pernas firmes, meus pés no chão e meu sorriso no rosto. Ao meu amigo Vitor C., por todo o companheirismo, por ser um excelente ouvinte, por tornar suportáveis os dias mais difíceis e por ser o abraço e o sorriso que dissolveram todos os meus problemas, mesmo em meio a uma pandemia. Aos meus amigos, que amo há uma vida e por toda ela, Rayanne, Anna, Fernanda, Luís, Ludmila E. e Isadora por todo o suporte, carinho e os momentos inesquecíveis que vivemos. A todos os outros que só cabem em meu coração, mas que não haveria folhas suficientes para elencar, obrigado por fazerem parte destes anos incríveis na Universidade e na vida.

"*É preciso reconhecer com a criptografia que nenhuma quantidade de violência resolverá um problema de matemática".*

Jacob Applebaum

"Se, na verdade, não estou no mundo para simplesmente a ele me adaptar, mas para transformá-lo; se não é possível mudá-lo sem um certo sonho ou projeto de mundo, devo usar toda possibilidade que tenha para não apenas falar de minha utopia, mas participar de práticas com ela coerentes."

Paulo Freire

RESUMO

O emprego de técnicas criptográficas é a base de diversos processos envolvendo a Internet, dentre eles o estabelecimento de redes privadas virtuais (VPNs) sobre redes públicas, garantindo a tais processos a segurança necessária no que diz respeito principalmente à confidencialidade, privacidade e autenticidade. A criptografia moderna, entretanto, pode ver-se ameaçada pelo desenvolvimento da computação quântica, tendo assim a sua segurança comprometida. Neste contexto, se faz necessário a análise quântico-resistente dos processos empregados em soluções criptográficas frente a ameaças do desenvolvimento da computação quântica, traçando alternativas e soluções mais seguras quando possível. Tal análise, no escopo deste trabalho, será restrita ao *software* de VPN *WireGuard* e os métodos criptográficos empregados nos seus processos, e será avaliada sob a ótica dos parâmetros estabelecidos pelo NIST para segurança criptográfica quântico-resistente.

Palavras-chave: Criptografia. WireGuard. VPN. Computação Quântica. Análise quântico-resistente. NIST.

ABSTRACT

The use of cryptographic techniques is the basis of several Internet processes, among them the establishment of virtual private networks (VPNs) over public networks, assuring to such processes the necessary security with regard mainly to confidentiality, privacy, and authenticity. Modern cryptography, however, may be threatened by the development of quantum computing, thus having its security compromised. In this context, it is necessary to analyze the quantum-resistant processes used in cryptographic solutions against threats arising from the development of quantum computing, outlining safer alternatives and solutions when possible. Such analysis, in the scope of this work, will be restricted to the VPN WireGuard software and the cryptographic methods applied in its processes, and will be evaluated under the perspective of the parameters established by NIST for quantum-resistant cryptographic security.

Keywords: Cryptography. WireGuard. VPN. Quantum Computing. Quantum Resistant Analysis. NIST.

Sumário

1 INTRODUÇÃO 1

1.1 OBJETIVOS 2
1.2 METODOLOGIA 3

2 FUNDAMENTAÇÃO TEÓRICA 4

2.1 CRIPTOGRAFIA 4
2.2 CONCEITOS CRIPTOGRÁFICOS BÁSICOS 4
2.3 ALGORITMOS CRIPTOGRÁFICOS 6
2.3.1 ESQUEMAS CRIPTOGRÁFICOS 7
2.3.2 FUNÇÕES HASH 8
2.3.3 CRIPTOGRAFIA DE CHAVE SIMÉTRICA 10
2.3.4 CRIPTOGRAFIA DE CHAVE PÚBLICA 11
2.3.5 CIFRAS DE FLUXO 12
2.3.6 CURVE25519 14
2.3.7 POLY1305-AES 15
2.3.8 AEAD – AUTHENTICATED ENCRYPTION WITH ASSOCIATED DATA 16
2.3.9 *PERFECT FORWARD SECRECY* (SIGILO DE ENCAMINHAMENTO PERFEITO) 16
2.4 COMPUTAÇÃO QUÂNTICA 16
2.5 CIRCUITOS QUÂNTICOS 18
2.6 ALGORITMOS QUÂNTICOS 18
2.6.1 COMPLEXIDADE COMPUTACIONAL 19
2.6.2 ALGORITMO DE SHOR 19
2.6.3 ALGORITMO DE GROVER 20
2.7 NIST E A PADRONIZAÇÃO DA CRIPTOGRAFIA PÓS-QUÂNTICA 20
a) Sigilo de Encaminhamento Perfeito (*Perfect Forward Secrecy*): 22
b) Resistência a Ataques de Canais Laterais (*Side-Channel Atacks*): 22
c) Resistência a Ataques de Múltiplas Chaves (*Multi-Key Attacks*): 22
d) Resistência a Más-Práticas (*Misuse*): 22
2.8 REDES PRIVADAS VIRTUAIS (*VPN'S*) 23
2.9 *WIREGUARD* 23
2.9.1 KEY MANAGEMENT 24
2.9.2 CRIPTOGRAFIA NO *WIREGUARD* 24
2.9.3 TRANSPARÊNCIA PARA *PEERS* ILEGÍTIMOS 25
2.9.4 MODO DE PRÉ-COMPARTILHAMENTO DE CHAVE SIMÉTRICA 25
2.9.5 PREVENÇÃO A ATAQUES DoS 25

3 MÉTODOS EMPREGADOS NO *WIREGUARD* 26

3.1 AVALIAÇÃO DO PROCESSO DE *HANDSHAKE* 28
3.1.1 TROCA DE MENSAGENS: MENSAGENS DO INICIADOR E DO RESPONDEDOR. 29
3.2 DERIVAÇÃO DE CHAVES DE DADOS DE TRANSPORTE 32
3.3 MENSAGENS DE DADOS DE TRANSPORTE 32
3.4 MENSAGEM DE RESPOSTA DE *COOKIE* 33

4 AVALIAÇÃO DOS MÉTODOS E PROPOSTAS DE SEGURANÇA PÓS-QUÂNTICA 35

4.1 IMPLEMENTAÇÃO DO *WIREGUARD* NO LINUX 35
4.2 NÍVEIS DE SEGURANÇA INTRÍNSECOS À IMPLEMENTAÇÃO 36
4.3 PRINCIPAIS ALGORITMOS CRIPTOGRÁFICOS UTILIZADOS 37
4.3.1 BLAKE2S 38

4.3.2 CURVE25519 39
4.3.3 CHACHA20 E POLY1305 39
4.4 SIGILO DE ENCAMINHAMENTO PERFEITO NO WIREGUARD 40
5 CONCLUSÕES E TRABALHOS FUTUROS 41

LISTA DE FIGURAS

Figura 1: Máquina criptográfica alemã Enigma [2]. 1
Figura 2: Esquemático de um esquema de criptografia simples. Adaptado de [8]. 5
Figura 3: Esquemático de uma comunicação entre duas partes utilizando criptografia. Adaptado de [8]. 6
Figura 4: Taxonomia das primitivas criptográficas. Adaptado de [8]. 7
Figura 5: Representação genérica da atuação de funções hash. 8
Figura 6: Representação genérica de criptografia de chave simétrica. 11
Figura 7: Representação genérica de criptografia de chave pública. 11
Figura 8: Esquemático da Função de Cifragem ChaCha20. 13
Figura 9: Diagrama do algoritmo CURVE25519. Adaptado de [14]. 14
Figura 10: Esquema genérico de AEAD. Adaptado de [17]. 16
Figura 11: Esquemático do funcionamento de um Túnel VPN. 23
Figura 12: Logo do Projeto WireGuard [4]. 24
Figura 13: Esquemático do estabelecimento de chaves no Handshake. 28
Figura 14: Estrutura da mensagem do iniciador [4]. 29
Figura 15: Estrutura da mensagem do respondedor [4]. 30
Figura 16: Estrutura das Mensagens de Dados de Transporte [4]. 33
Figura 17: Estrutura da mensagem de resposta de cookies [4] 33
Figura 18: Par de chaves estáticas públicas e privadas geradas pelo WireGuard. 35
Figura 19: Exemplo de arquivo de configuração do WireGuard. 35
Figura 20: Diagrama Genérico de Implementação do WireGuard nas máquinas virtuais. 36
Figura 21: Anel de Segurança da Implementação do WireGuard. 37

LISTA DE TABELAS

Tabela 1: Parâmetros da Função de Hash BLAKE2s. Adaptado de [11]. 9
Tabela 2: Níveis de Segurança para Algoritmos Pós-Quânticos. Adaptado de [23]. 21
Tabela 3: Relação de Profundidade de Circuito para Prováveis Ataques Bem Sucedidos em Algoritmos Criptográficos. Adaptado de [23]. 21
Tabela 4: Tabela de variáveis envolvidas nos processos do WireGuard. Adaptado de [4]. 26
Tabela 5: Funções e Variáveis Específicas no WireGuard. Adaptado de [4]. 26
Tabela 6: Computações das Variáveis do Iniciador. Adaptado de [4]. 29
Tabela 7: Computações das Variáveis do Respondedor. Adaptado de [4]. 30
Tabela 8: Computação dos parâmetros do MAC. Adaptado de [4]. 31
Tabela 9: Computação das Mensagens de Dados de Transporte. Adaptado de [4]. 33
Tabela 10: Computação dos Parâmetros da Mensagem de Resposta de cookies 34
Tabela 11: Resumo da avaliação dos métodos. 38

LISTA DE SÍMBOLOS

$\mathcal{A}$ – Alfabeto de Definição
$\mathcal{C}$ – Espaço da Cifra
C – Mensagem cifrada
C_* –Valor de encadeamento no WireGuard
d – Chave de decifragem
D_d – Transformação de decifragem
e – Chave de cifragem
E – Curva Elíptica genérica
E_e – Transformação de cifragem
e_*^{priv} – Chave efêmera privada do iniciador/respondedor no WireGuard
e_*^{pub}– Chave efêmera pública do iniciador/respondedor no WireGuard
H_* – Valor *hash* no WireGuard
h – Função de *hash* genérica
I_i – Índice identificador do iniciador no WireGuard
I_r - Índice identificador do respondedor no WireGuard
s_*^{priv} – Chave estática privada do iniciador/respondedor no WireGuard
s_*^{pub} – Chave estática pública do iniciador/respondedor no WireGuard
$\mathcal{K}$ – Espaço de Chaves
k – Chave criptográfica genérica
κ – Chave criptográfica temporária no Wireguard
L – Identificador de *cookies* no WireGuard
$\mathcal{M}$ – Espaço de Mensagem
M – Mensagem base
m – mensagem genérica
mod – Módulo
N_*^{send} – Contador *nonce* para o transporte de dados para envio no WireGuard
N_*^{recv} – Contador *nonce* para o transporte de dados para recebimento no WireGuard
P – Pacotes no WireGuard
Q_* – Chave opcional pré compartilhada do iniciador/respondedor no WireGuard
R_m – Variável secreta na resposta de *cookies* no WireGuard
T_*^{send} – Chave simétrica transporte de dados para envio no WireGuard
T_*^{recv} – Chave simétrica transporte de dados para recebimento no WireGuard
τ – Variável temporária no WireGuard

LISTA DE ACRÔNIMOS

AEAD – *Authenticated Encryption with Associated Data*
AES – Padrão Criptográfico *Advanced Encryption Standard*
DH – Diffie-Hellman
ECDH – *Elliptic-curve Diffie–Hellman*
HKDF – *Hash-based Key Derivation Function*
HMAC - *Hash-based Message Authentication Code*
KDF – *Key Derivation Function*
KEM – *Key Encapsulation Mechanism*
ksg – *Key Stream Generator*
MAC – *Message Authentication Code*
PFS – *Perfect Forward Secrecy*
PKI – *Public Key Infrastructure*
RFC – *Request for Comments*

1 INTRODUÇÃO

Desde que foi desenvolvida a capacidade de comunicação pelo homem inserido em uma organização social, o ser humano viu como uma necessidade inerente ao seu convívio em comunidade a classificação e a proteção de certos tipos de conhecimentos organizados. Conhecimentos estes que chamamos hoje, a rigor, de Informação, e que poderiam caracterizar algum tipo de vantagem em potencial na sua sobrevivência e predominância em relação aos demais. Foi dentro desse contexto que, antes de ser uma ciência, um protótipo do que viria a ser chamado de Criptografia teve seu nascimento. A premissa básica de técnicas criptográficas é a possibilidade de comunicação de forma segura entre duas ou mais partes através de um canal não-seguro, de forma a garantir privacidade e autenticidade em suas trocas [1].

Para que a criptografia tenha êxito, de forma que mensagem e código possam ser diferenciados, é necessário o estabelecimento de esquemas. Um esquema criptográfico, de forma genérica, é definido por um par $(\mathcal{E}, D)$ de algoritmos de cifragem e decifragem, respectivamente. O primeiro, executado pelo remetente, toma uma *chave k* e a mensagem *M* e cria uma cifra *C*, que será transmitida para o receptor. Este aplica o algoritmo *D*, que toma uma *chave k* e a mensagem cifrada a fim de recuperar a mensagem original; importando desta forma como qualidade de segurança de que um terceiro não possa identificar nenhum dado importante constante na mensagem durante ou antes mesmo da transmissão.

A criptografia foi amplamente utilizada nos meios de comunicações, seja nas eras clássica, e moderna, como durante as grandes guerras do século XX; seja na contemporaneidade, nos sistemas de telecomunicações mais tecnológicos. Na Figura 1 pode ser observada a máquina criptográfica Enigma utilizada na segunda guerra mundial pelas forças militares alemãs para cifrar mensagens nas comunicações entre si.

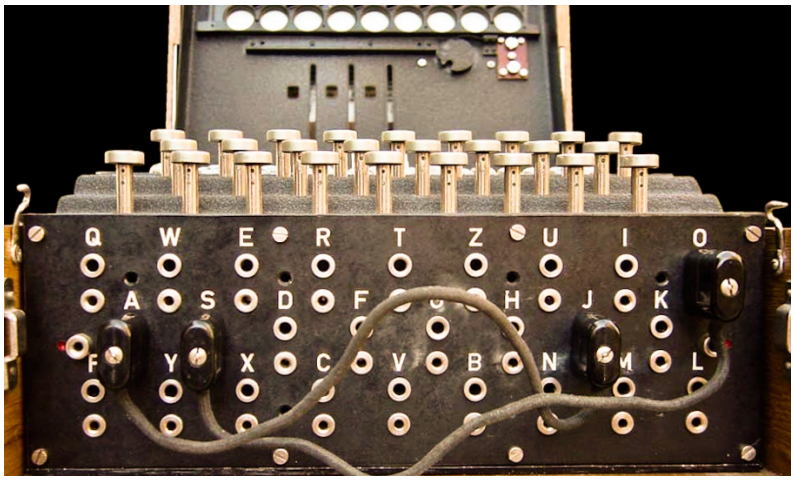

Figura 1: Máquina criptográfica alemã Enigma [2].

A título de exemplo, o meio de telecomunicação mais utilizado atualmente, a Internet, enquanto uma rede mundial de comunicações, tem como um de seus recursos mais importantes a

utilização da criptografia como forma de garantir a segurança e a operacionalidade dos seus recursos no que diz respeito à confidencialidade, integridade e a disponibilidade dos seus canais e dos dados trocados. O que garante essa sustentação é a impossibilidade prática de se quebrar a criptografia, se bem implementada, em sistemas e aplicações com os recursos computacionais hoje existentes.

Surge no horizonte, entretanto, um cenário diferente e incerto, com o desenvolvimento de novas tecnologias computacionais, que colocam em xeque os protocolos de segurança adotados nas redes de comunicação. As pesquisas e primeiras implementações na área da computação quântica são um prelúdio de uma mudança de paradigmas e servem como um presságio para a elaboração de novos modelos de segurança que resistam ao poder computacional de uma nova geração de computadores com novos mecanismos de funcionamento. Tal cenário ameaça quebrar a criptografia de chaves públicas e a maioria das cifras que protegem a arquitetura do mundo digital atual, incluindo as utilizadas em aplicações tais como o protocolo de comunicação HTTPS (*Hipertext Transfer Protocol Secure)*, o protocolo criptográfico de rede SSH (*Secure Shell*), Infraestruturas de Chaves Públicas (PKI's), os métodos utilizados em *smartcards*, HSMs (*Hardware Secure Modules*), criptomoedas, certificados digitais; algoritmos criptográficos como os baseados em RSA, Diffie-Hellman, Criptografia de Curvas Elípticas; protocolos de redes sem fio (Wi-Fi) e a maioria das soluções de redes privadas virtuais (VPNs), dentre outros [3].

Preparar-se para este cenário de computação quântica demanda criatividade, atenção, educação, escolhas críticas e planejamento dos interessados em contribuir para as soluções de segurança numa era pós-quântica. Existem muitas soluções implementáveis, algumas que fogem do escopo deste trabalho a nível de graduação. Este trabalho, entretanto, avaliará meios quântico-resistentes – não necessariamente quânticos – que podem contornar o advento da computação quântica como recurso computacional na quebra da criptografia no escopo das redes privadas virtuais (VPNs), e mais especificamente no software *WireGuard* [3].

1.1 OBJETIVOS

Visando um possível cenário iminente, ainda que incerto, no qual a criptografia clássica seria ameaçada pelo desenvolvimento de técnicas de computação quântica num futuro ainda incerto, ou ainda no qual dados cifrados atualmente já possam estar sendo capturados para futura quebra de criptografia, torna-se pertinente o estudo das alternativas de soluções criptográficas que estão sendo desenvolvidas atualmente a fim de avaliar a sua segurança frente às futuras ameaças.

Desta forma, este trabalho tem como objetivo a investigação dos métodos e das técnicas criptográficas implementados no software de VPN *WireGuard* [4] de forma a construir uma pesquisa qualitativa que explora a possível resistência aos avanços da computação quântica entregue pelo software *Wireguard* com base em suas referências bibliográficas, nos critérios estabelecidos pelo NIST [5] e nos estudos de avaliação de desempenho pertinentes. Desta forma, após a apresentação da teoria necessária para o completo entendimento de todos os tópicos abordados neste trabalho, será feita a exposição dos principais recursos do software *WireGuard,* serão detalhados os recursos elencados e destacados os seus papéis na construção de um cenário computacional quântico-resistente.

1.2 METODOLOGIA

Neste trabalho, será abordado o software de criação de VPN's *WireGuard* [4],que se propõe a ser uma nova alternativa, em sua implementação original, segura e escalável para lidar com um cenário pós-quântico; e no qual serão explorados os diferentes tipos de algoritmos criptográficos utilizados e avaliados com base na segurança fornecida por estes, levando em consideração os requisitos mais aceitos convencionalmente para esta avaliação. Desta forma, nas respectivas seções seguintes serão expostos, de forma estritamente pertinente, descritiva e explicativa os fundamentos para a compreensão:

- **Fundamentação Teórica:**
    - das técnicas criptográficas,
    - da computação quântica,
    - dos algoritmos de Shor [6]e de Groover [7],
    - da ameaça à criptografia clássica frente à computação quântica, e
    - das bases que fundamentam os critérios de segurança criptográfica pós-quântica.

- **Métodos Empregados no *WireGuard*:**
    - do funcionamento do software *Wireguard* [4] e das técnicas de segurança implementadas neste, e

- **Avaliação dos Métodos e Propostas de Segurança Pós-Quântica:**
    - das bases que fundamentam propostas de implementações a fim de aprimorar ou garantir a segurança criptográfica pós-quântica.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 CRIPTOGRAFIA

Criptografia é a ciência e o estudo de técnicas matemáticas aplicadas à segurança da informação. Tais técnicas têm o intuito de garantir aspectos importantes tais como confidencialidade, integridade e autenticação [8]. As técnicas utilizadas envolvem a cifragem de dados, testes de integridade e a utilização de algoritmos criptográficos que estabelecem protocolos para as comunicações entre as partes. A criptografia permite que a confidencialidade e a integridade dos dados, das comunicações e dos participantes seja mantida entre as partes autorizadas e/ou designadas, como softwares, dispositivos, e indivíduos [3]

A criptografia é o conceito primordial que rege a segurança da informação moderna. Ao garantir alguns fundamentos, chamados de pilares da Segurança da Informação, como Confidencialidade, Integridade, Autenticação e Não-Repúdio; ela se torna essencial para a segurança nas telecomunicações, e, em especial, para a segurança da rede mundial de computadores Internet.

Confidencialidade é um atributo que garante que uma informação está segura de não ser acessada por alguém não autorizado, e tem como sinônimo o termo sigilo. Integridade é a característica de algo que não foi alterado, seja por modificação, inclusão ou exclusão de informação. Autenticação é um serviço relacionado à identificação, e é relativo tanto à informação quanto às entidades comunicantes. Em uma comunicação, ambas as partes devem se identificar uma para a outra, e a troca de informações, ou a concessão do acesso, só deve se dar caso ambas as partes sejam capazes de comprovar que são quem alegam ser. Não-repúdio é a propriedade que previne que uma entidade possa negar o histórico de ações executados por ela, e é assegurada pelos fundamentos anteriores. Portanto, a criptografia tem como objetivo a aplicação adequada e conjunta desses quatro fundamentos na teoria e na prática, com o intuito de prevenir e detectar atividades maliciosas [8].

2.2 CONCEITOS CRIPTOGRÁFICOS BÁSICOS

A fim de tratar a criptografia de forma mais formal, é necessário estabelecer no escopo deste trabalho algumas terminologias e conceitos básicos [8] que serão retomados sempre que necessário ao longo da exposição dos assuntos nele tratados.

a) Alfabeto de definição ($\mathcal{A}$)

$\mathcal{A}$ é o conjunto finito que delimita o Alfabeto de definição, i.e., o conjunto de elementos empregados na linguagem utilizada.

b) Espaço da Mensagem ($\mathcal{M}$)

O conjunto $\mathcal{M}$ é definido como o Espaço da Mensagem, e consiste no conjunto de cadeias derivadas do alfabeto de definição.

c) Espaço da Cifra ($\mathcal{C}$)

O Espaço da Cifra $\mathcal{C}$ é o conjunto que consiste na cadeia de símbolos do alfabeto de definição, que pode ser diferente do alfabeto de definição de $\mathcal{M}$.

d) Espaço de Chaves ($\mathcal{K}$)

$\mathcal{K}$ denota o conjunto chamado Espaço de Chaves, e os elementos que compõem este espaço são chamados de "chaves".

Cada elemento $e \in \mathcal{K}$ determina de forma única uma bijeção E_e de $\mathcal{M}$ para $\mathcal{C}$. E_e é o que chamamos de uma função ou transformação criptográfica, i.e., a função de cifragem. Note-se que cada transformação criptográfica E_e deve ser uma bijeção para que se consiga reverter o processo, e ter uma mensagem base recuperada para cada texto-cifrado diferente. O processo E_e em uma mensagem $m \in \mathcal{M}$ é o que se conhece por cifra de m.

Todo d pertencente a $\mathcal{K}$ implica um D_d que denota uma bijeção de $\mathcal{C}$ para $\mathcal{M}$ ($D_d : \mathcal{C} \rightarrow \mathcal{M}$), que é chamado de função de decifragem, ou reversão de criptografia. O processo de D_d sobre um texto cifrado c é o que se conhece por decifragem de c.

Um *esquema criptográfico* consiste num conjunto de transformações criptográficas $\{Ee : e \in \mathcal{K}\}$ e um conjunto correspondente de transformações de reversão da criptografia $\{D_d : d \in \mathcal{K}\}$, tais que para todo $e \in \mathcal{K}$ existe uma chave única $d \in \mathcal{K}$ de forma que $D_d = {E_e}^{-1}$, i.e., $D_d(E_e(m)) = m$ para todo $m \in \mathcal{M}$. Tal esquema de criptografia é o que chamamos de *cifra*. As chaves e e d é o que chamamos de *par de chaves* e são denotados por (*e,d*). Todos os atributos mencionados anteriormente são necessários para se construir um *esquema criptográfico* conforme ilustrado na Figura 2.

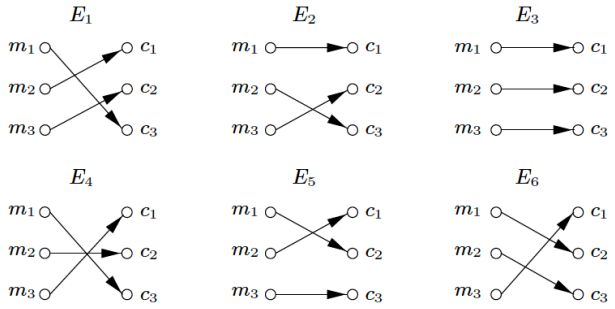

Figura 2: Esquemático de um esquema de criptografia simples. Adaptado de [8].

Um esquema criptográfico é o que possibilita a utilização da criptografia para assegurar a confidencialidade. Um exemplo clássico utilizado para representar o enlace de todos os conceitos apresentados é a comunicação entre duas partes, Alice e Bob, as quais estabelecem e secretamente trocam um par de chaves (*e,d*) conforme ilustrado na Figura 3. Em algum momento, se Alice pretende mandar uma mensagem $m \in \mathcal{M}$ para Bob, ela deve calcular $c = Ee(m)$ e transmitir o

resultado para Bob, que para recuperar a mensagem original deverá computar $Dd(c) = m$ e assim recuperará a mensagem enviada.

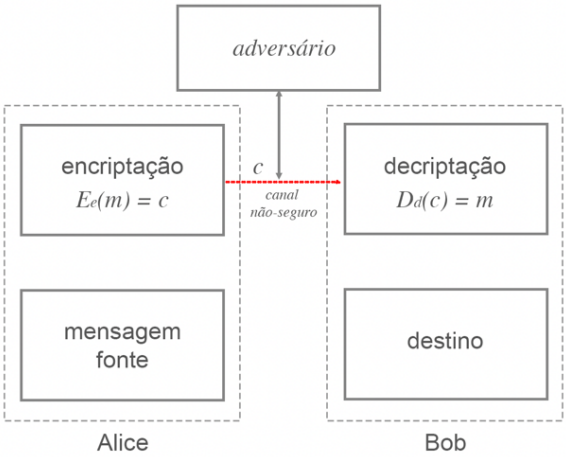

Figura 3: Esquemático de uma comunicação entre duas partes utilizando criptografia. Adaptado de [8].

Define-se como *quebrável* um esquema criptográfico cujo texto cifrado pode ser sistematicamente recuperado por um terceiro (adversário) sem o conhecimento prévio do *par de chaves (e,d)*, em um período de tempo apropriado, i.e., um período de tempo relevante em comparação com o ciclo de vida do dado que está sendo protegido. A quebra de um esquema criptográfico pode-se dar, por exemplo, por meio tentativa e erro (*pesquisa exaustiva*) de todas as chaves até descobrir a que está sendo utilizada por uma das partes, caso o algoritmo utilizado seja publicamente conhecido. Posto isso, deve-se ter em mente, ao considerar proteção de dados ou sistemas por criptografia, implementar protocolos com arquiteturas complexas e/ou tamanho do espaço de chaves suficientemente grande para tornar a quebra deste computacionalmente inviável [8].

2.3 ALGORITMOS CRIPTOGRÁFICOS

Como já abordado, o processo utilizado para transformar uma mensagem simples em uma mensagem criptografada é chamado de "*cifragem*" e o processo de reversão é chamado *"decifragem",* formas estas que serão utilizadas doravante. O processo documentado envolvendo todos os passos utilizados na cifragem e na decifragem dos dados é conhecido como cifra, ou algoritmo criptográfico [3]. Estes algoritmos são, em sua essência, implementações mais complexas de fundamentos criptográficos mais básicos, e servem como protocolos para garantir a segurança em sistemas computacionais. Os algoritmos criptográficos podem ser distribuídos em grupos mais específicos de acordo com os seus princípios de funcionamento, como *primitivas sem-chave*, *primitivas de chaves simétricas* e *primitivas de chaves públicas*; e podem ser avaliadas de acordo com o seu nível de segurança, a sua funcionalidade, seus métodos de operação, sua performance e sua aplicabilidade em cada situação [1]. O diagrama da Figura 4 representa a

taxonomia das primitivas criptográficas. Serão abordadas, entretanto, neste trabalho, apenas as que dizem respeito ao funcionamento dos sistemas abrangidos no seu escopo.

Figura 4: Taxonomia das primitivas criptográficas. Adaptado de [8].

2.3.1 ESQUEMAS CRIPTOGRÁFICOS

Esquemas criptográficos, sejam eles simétricos ou assimétricos, são compostos por algoritmos responsáveis por papéis diferentes nos processos criptográficos. Pode-se elencar ao menos três algoritmos básicos que podem compor um esquema criptográfico com chaves: um para a geração das chaves, um para a cifragem dos dados a serem trocados e um para a decifragem destes [9]. Em esquemas criptográficos as partes envolvidas são responsáveis ainda pelo gerenciamento das chaves com as quais interage com os pares, e consequentemente pelos dados sob criptografia publicamente disponíveis. Desta forma, podemos elencar ainda os seguintes processos específicos que tomam parte nas relações em soluções criptográficas:

a) Estabelecimento de Chaves (*Key Stablishment*):

Entende-se por Estabelecimento de Chaves qualquer processo que estabeleça as chaves a serem usadas no processo criptográfico e as divulgue para os pares envolvidos para a devida aplicação criptográfica [8].

b) Gerenciamento de Chaves (*Key Management*):

Processos de Gerenciamento de Chaves são mecanismos que viabilizam o Estabelecimento de Chaves em soluções criptográficos, responsáveis por guardar as chaves, mantê-las disponíveis, e até mesmo substituí-las por novas quando necessário. Fazem parte dos processos de gerenciamento de chaves os subprocessos de Acordo de Chaves (*Key Agreement*), que é o acordo mútuo entre as partes das chaves utilizadas; e de Transporte de Chaves (*Key Transport*), que é a comunicação das chaves em questão entre as partes [8].

c) Derivação de Chaves (*Key Derivation*):

Derivação de Chaves, como o próprio nome sugere, é o método que utiliza uma chave k_i para a computação de outra chave k_j por meio da utilização de um *token t* publicamente disponível entre os pares. Uma Função de Derivação de Chaves (*Key Derivation Function – KDF*) é uma função que realiza a derivação de um conjunto de chaves $\mathcal{K}$ utilizando uma cadeia de *tokens* $\mathcal{T}$, $\mathcal{T}: \mathcal{K} \mapsto 2^{\mathcal{K}}$ e é definida como $\mathcal{T}(k_i) = \{k_j \in \mathcal{K} : \exists\, t_{i,j} \in \mathcal{T}\}$ [10].

2.3.2 FUNÇÕES HASH

Funções criptográficas de *hash*, ou simplesmente *hashes*, são funções que tomam uma mensagem como entrada e produzem uma saída referenciada como um código de *hash*, valor de *hash*, ou simplesmente *hash*. Uma função *hash h* mapeia *strings* de um tamanho arbitrário para *strings* de tamanho *n* fixo numa transformação 1 para *N*, $D \rightarrow R$, tal que $|D| > |R|$, onde *D* é o domínio e *R* o alcance *(range)*, de modo que não são aceito pares de entrada com saída idênticas, i.e., não se admite colisões [8].

Funções *hash* têm como princípio uma representação compacta da imagem de uma mensagem, como uma impressão digital, e podem ser usadas para identificar de forma única um conjunto de dados [8]. Uma representação genérica da atuação de uma função de *hash* sobre mensagens base de tamanhos variados pode ser ilustrada pela Figura 5.

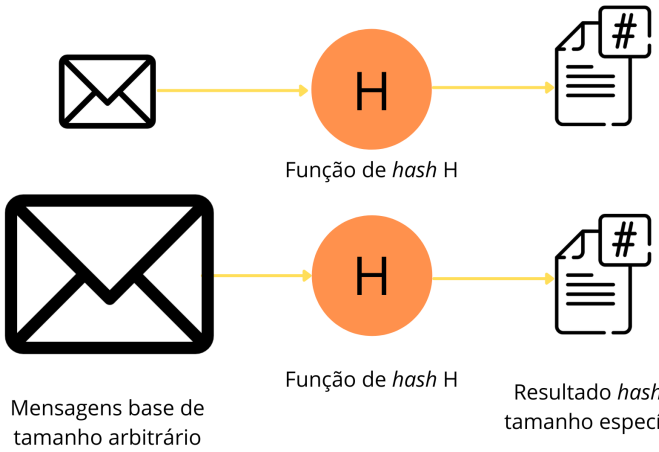

Figura 5: Representação genérica da atuação de funções *hash*.

2.3.2.1 BLAKE2s

BLAKE2s é uma função criptográfica de *hash* que faz parte da família de funções BLAKE2 [11]. Essa família é um aprimoramento da sua antecessora BLAKE, e possui mecanismos de *hash* rápidos e altamente seguros, fornecendo segurança similar à família SHA3 [12] e pode fornecer resultados compilados de tamanhos de 160 a 512 bits. Esta função pode ser utilizada por protocolos de assinatura digital, autenticação e mecanismos de proteção de integridade como para aplicações em Infraestrutura de Chaves Públicas (PKI), protocolos de comunicação seguros, armazenamento em nuvem, dentre outros; e é otimizada para plataformas de 8 a 32 bits, produzindo resultados compilados de tamanhos entre 1 e 32 bytes [11].

A função criptográfica de *hash* BLAKE2 na sua variação BLAKE2s é optimizada para plataformas de 8 a 32 bits e pode produzir resultados compilados de tamanho entre 1 e 32 bytes. Os parâmetros adotados nessa variação encontram-se relacionados na Tabela 1:

Tabela 1: Parâmetros da Função de *Hash* BLAKE2s. Adaptado de [11].

Parâmetros da função - BLAKE2s

Parâmetro	Representação	Valor
Bits por palavra	w	32
Rounds em F	r	10
Block bytes	bb	64
Hash bytes	nn	1-32
Key bytes	kk	0-32
Input bytes	ll	$0\text{-}2^{64}$
Constantes de Rotação de G	R1, R2, R3, R4	16, 12, 8, 7

Os seus fundamentos de processamento são baseados em duas funções básicas: uma função de mistura *G* e uma de compressão *F*, como definidas em sua RFC [11]. A função de mistura *G* toma duas *words* "x" e "y" e as embaralha entre quatro *words* de índices "a", "b", "c" e "d" em um vetor funcional de 16 posições, de forma que a função retorna o vetor modificado com suas devidas rotações, como pode ser visto no seguinte pseudocódigo [11]:

```
FUNCTION G( v[0..15], a, b, c, d, x, y)
|
| v[a] := (v[a] + v[b] + x) mod 2**w
| v[d] := (v[d] ^ v[a]) >>> R1
| v[c] := (v[c] + v[d])     mod 2**w
| v[b] := (v[b] ^ v[c]) >>> R2
| v[a] := (v[a] + v[b] + y) mod 2**w
| v[d] := (v[d] + v[a]) >>> R3
| v[c] := (v[c] + v[d])     mod 2**w
```

```
    | v[b] := (v[b] + v[c]) >>> R4
    |
    | RETURN v[0..15]
    |
END FUNCTION.
```

Em seguida, uma função de compressão F, que tem como parâmetros um vetor de estados "h", um vetor de bloco de mensagem "m", um contador de *offset* com tamanho de *2w* bits um indicador de final de bloco "f"; toma o vetor local "v" resultante da função de mistura e o utiliza no processamento em 10 rounds para o caso da BLAKE2s conforme mostrado pelo seguinte pseudo-código [11]:

```
FUNCTION F( h[0..7], m[0..15], t, f)
|
|     // Initialize local work vector v[0..15]
|     v[0…7] := h[0..7]  //First half from state
|     v[8..15] := IV[0..7] //Second half from IV
|
|     v[12] := v[12] ^ (t mod 2**w) // Low word of the offset
|     v[13] :=. V[13] ^ (t >> w) // High word
|
|     IF f = TRUE THEN // last block flag?
|     |     v[14] := v[14] ^ 0xFF..FF // Invert all bits.
|     END IF.
|
|     // Cryptographic mixing
|     FOR i=0 TO 7 DO         // XOR the two halves.
|     |     h[i] := h[i] ^ v[i] ^ v[i+8]
|     END FOR.
|
|     RETURN [0..7]     //New state
|
END FUNCTION.
```

O *padding* dos dados e a computação do compilado do BLAKE2 são descritos integralmente na RFC 7693 [11].

2.3.2.2 *MESSAGE AUTHENTICATION CODE (MAC)*

São algoritmos também conhecidos como *funções de hash chaveadas*, e são funções de *hash* utilizadas para autenticar mensagens entre as partes a partir do compartilhamento de um segredo.

2.3.3 CRIPTOGRAFIA DE CHAVE SIMÉTRICA

Diz-se um algoritmo criptográfico de chaves simétricas se para encriptar e desencriptar uma mensagem a mesma chave *k* é utilizada. Matematicamente, um algoritmo de chave simétrica é um algoritmo que utiliza as transformações $\{E_e : e \in \mathcal{K}\}$ e $\{D_d : d \in \mathrm{K}\}$ para encriptar e desencriptar,

respectivamente, uma mensagem tal que para o par (e,d) seja computacionalmente plausível determinar d conhecendo-se apenas e, ou vice-versa. Note-se que na maior parte das aplicações de criptografia de chave simétrica $e=d$ [8]. Estes algoritmos, quando aplicados em mecanismos menos simples do que o exemplificado na Figura 6, tendem a requerer chaves menores mantendo-se razoavelmente fortes, e tendem também a serem mais rápidos e mais fáceis de validar do que os baseados em chaves assimétricas, e são amplamente os mais utilizados atualmente [3].

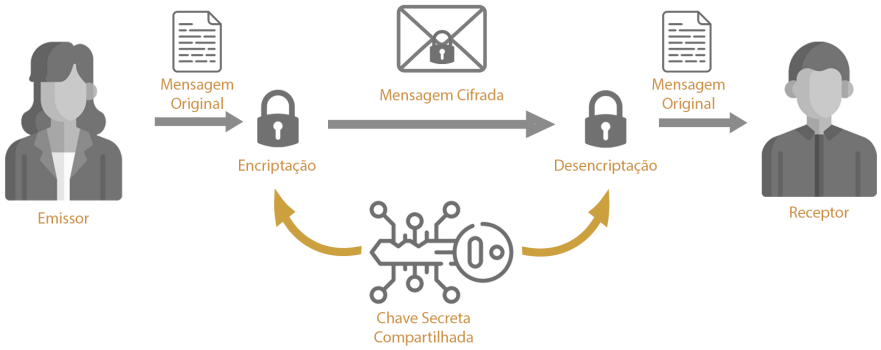

Figura 6: Representação genérica de criptografia de chave simétrica.

2.3.4 CRIPTOGRAFIA DE CHAVE PÚBLICA

A criptografia de chave pública, ou criptografia de chave assimétrica, envolve em seu conceito duas chaves utilizadas no processo de cifragem e decifragem, chaves estas cunhadas como chave pública e chave privada conforme ilustrado na Figura 7. Neste tipo de algoritmo, a chave privada de cada par envolvido na transformação é utilizada para desencriptar a mensagem e permanece secreta, sendo a chave pública, relacionada à chave privada dos pares, conhecida por ambos e utilizada para encriptar. Um esquema criptográfico é dito de chave pública se para cada par de chaves de cifragem/decriptação (e, d), a chave pública e é disponibilizada para a comunicação enquanto a chave privada d é mantida em segredo.

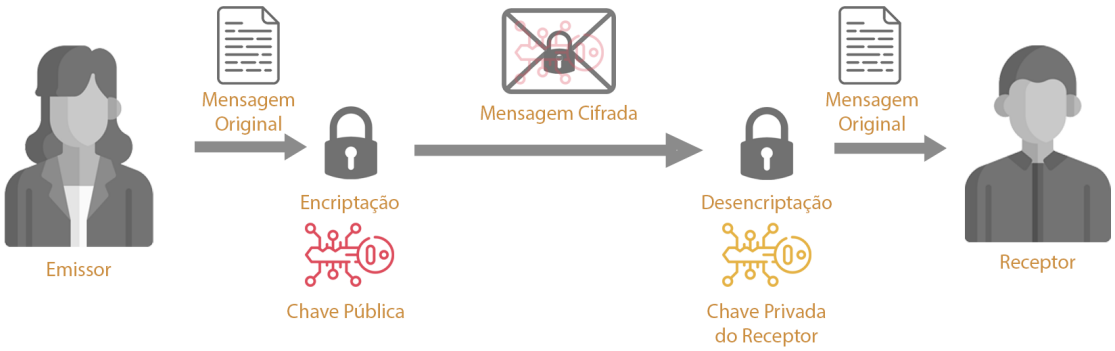

Figura 7: Representação genérica de criptografia de chave pública.

Em termos matemáticos, sejam os conjuntos $\{E_e : \ e \in \mathcal{K}\}$ e $\{D_d: \ d \in \mathcal{K}\}$, que representam conjuntos de transformações de cifragem e decifragem, respectivamente, e o par $(E_e,$

D_d) de transformação de cifragem/decriptação associado. Um algoritmo de cifragem de chave simétrica garante que, para que uma dada mensagem $m \in \mathcal{M}$ cuja mensagem cifrada $c \in \mathcal{C}$ sob E_e, tal que $E_e(m) = c$; a chave e de correspondente d seja computacionalmente inviável de calcular [8].

2.3.5 CIFRAS DE FLUXO

Uma cifra de fluxo (Fig. 8) toma uma mensagem base *m,* que pode ser referenciada como uma *string* do tipo $m_1m_2...m_i$, e a transforma por meio da utilização de um fluxo de chaves (*keystream*) $e_1e_2...e_i \in \mathcal{K}$ em uma *string* de mensagem cifrada do tipo $c_1c_2...c_i$ tal que $c_i = E_{e_i}(m_i)$. A sua transformação inversa pode ser representada por $D_{d_i}(c_i) = m_i$ [8].

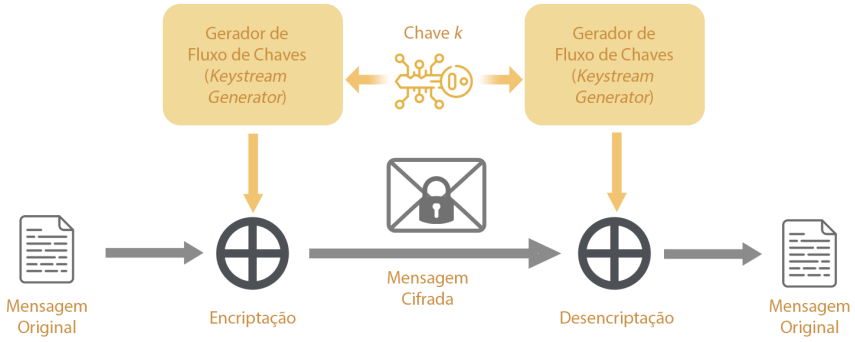

Figura 8: Representação genérica de cifragem e decifragem de cifras de fluxo.

Cifras de fluxo podem ser tanto de chaves simétricas, quanto de chaves assimétricas. Restringir-se-á, entretanto, à definição de cifras de fluxo de chave simétrica, como a do algoritmo ChaCha20.

2.3.5.1 CHACHA20

Os algoritmos da família ChaCha são algoritmos de cifra de fluxo de chave simétrica, que possuem grande potencial de substituição em caso de comprometimento de algoritmos amplamente utilizados como o AES simples. Essa cifra se apresenta como insuscetível a ataques de temporização, i.e., ataques nos quais o atacante tenta realizar o comprometimento do algoritmo por meio da análise do tempo de sua execução. O algoritmo original é chamado ChaCha, sendo ChaCha20 uma variação que executa 20 rounds – ou 80 quartos-de-round – do algoritmo. O algoritmo ChaCha toma uma chave de 256 bits e a expande em um grupo de 2^{64} fluxos de acesso randômico, cada um com blocos de 64 bytes igualmente de acesso randômico. Este algoritmo é basicamente uma função de *hash* utilizada no modo de contagem como uma cifra de fluxo, de 64 bytes de entrada e 64 bytes de saída, na qual encripta-se a mensagem aplicando sob um princípio de *ksg (key stream generator)* uma função de *hash* que tem como parâmetros uma chave, um *nonce* e um número de bloco; fazendo um XOR do resultado com a mensagem a ser criptografada [13].

Uma *word* neste algoritmo é um elemento de {0, 1, ..., 2^{32}-1} e é expressa usualmente como números em hexadecimal indicados por 0x, e.g., *0xc0a8787e* = 3232266366_{10}. A soma de duas

palavras u e v é denotada por $u+v = u+v \ mod \ 2^{32}$. Um XOR de duas palavras u e v é denotado por $u \oplus v$ e tem como resultado a soma $u+v$ com carries suprimidos [13].
Para cada elemento $c \in \{0, 1, 2, ...\}$, descreve-se uma rotação esquerda de c-bits de uma palavra u como $u <<< c$, e é a única palavra diferente de zero congruente com $2^c u \ modulo \ 2^{32}-1$, a não ser que $0 <<< c = 0$.

Seja $y = (y_0, y_1, y_2, y_3)$, um quarto de *round* (*quarteround*) pode ser descrito como uma sequência de 4 palavras de modo que $quarteround(y) = (z_0, z_1, z_2, z_3)$ onde:

$$z_1 = y_1 \oplus ((y_1 + y_2) <<< 7), \quad (1)$$

$$z_2 = y_2 \oplus ((z_1 + y_0) <<< 9), \quad (2)$$

$$z_3 = y_3 \oplus ((z_2 + z_1) <<< 13), \quad (3)$$

$$z_0 = y_0 \oplus ((z_3 + z_2) <<< 18). \quad (4)$$

Um *quarteround* pode ser explicado então como uma função de modificação de lugar do y, na qual y_1 muda para z_1, y_2 muda para z_2, y_3 muda para z_3 e y_0 muda para z_0; de modo que cada mudança é irreversível e, consequentemente, a função inteira também será irreversível [14].

2.3.5.2 FUNÇÃO DE CIFRAGEM DO CHACHA20

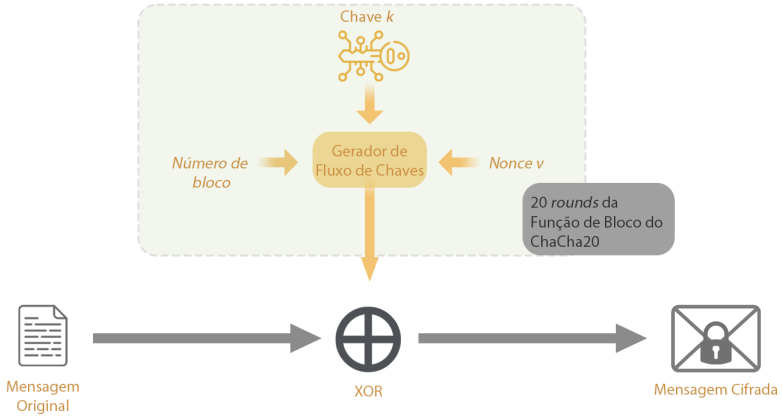

Figura 8: Esquemático da Função de Cifragem ChaCha20.

Tome-se k como uma sequência de 32 ou 16 bytes, v como uma sequência de 8 bytes, m como uma sequência de ℓ bytes para $\ell \in \{0, 1, ..., 2^{70}\}$, conforme esquemático mostrado na Figura 8, cifragem feita pelo Chacha20 de m com um *nonce* v sob uma chave k, denotada como $Salsa20_k(v) \oplus m$, é uma sequência de ℓ bytes. Usualmente k é uma chave privada preferencialmente de 32 bytes, v é um número único da mensagem e m é a mensagem original [13].

2.3.6 CURVE25519

O algoritmo Curve25519 é baseado em uma função de Curva-Eliptica-Diffie-Hellman em estado da arte, e é diferenciado por ter um ótimo desempenho em velocidade de processamento e uma segurança forte [14].

A função de curva elíptica da Curve25519 tem como base matemática a consideração de um número primo $p \geq 5$ *n,* e *A* números inteiros*,* de forma que $A^2 - 4$ não seja um módulo quadrático de *p.* A curva elíptica *E* é definida sobre o campo $\mathbf{F}_p$ como:

$$y^2 = x^3 + Ax^2 + x \tag{5}$$

Definam-se X_0*: E* $(F_{p^2}) \rightarrow F_{p^2}$ como $X_0(\infty)=0$; $X_0(x,y) = x$. Para um elemento *q* pertencente a F_p, então existe um único $s \in \mathbf{F}_p$ tal que $X_0(nQ) = s$ para todo $Q \in E$ $(\mathbf{F}_{p^2})$ tal que $X_0(Q) = q$.

Posto isso, assume-se então *p* como um número primo $2^{255} - 19$, $\mathbf{F}_p$ como o campo primo $Z/p = Z/(2^{255} - 19)$, F_{p^2} como o campo $(Z/(2^{255} - 19))\sqrt{2}$.

Assume-se $A = 486662$. Levando-se em conta ainda todas as definições anteriores, define-se então *X: E* $(F_{p^2}) \rightarrow \{\infty\} \cup \mathbf{F}_{p^2}$, de modo que $X_0(\infty) = 0$; e $X_0(x,y) = x$. Portanto, dado um $n \in 2^{254} + 8$ $\{0,1,2,3,..., 2^{251}-1\}$ e $q \in \mathbf{F}_p$ o algoritmo é capaz de gerar *s* [14].

Na geração de chaves sob função Curve25519, para uma explicação em alto nível, tem-se o seguinte mecanismo: cada parte em um processo de autenticação e cifragem entre duas partes tem uma chave privada e uma pública, ambas de 32 bytes e ambas as partes trocam um segredo também de 32 bytes [14]. Em médio nível, o processo é descrito pelo diagrama da Figura 9.

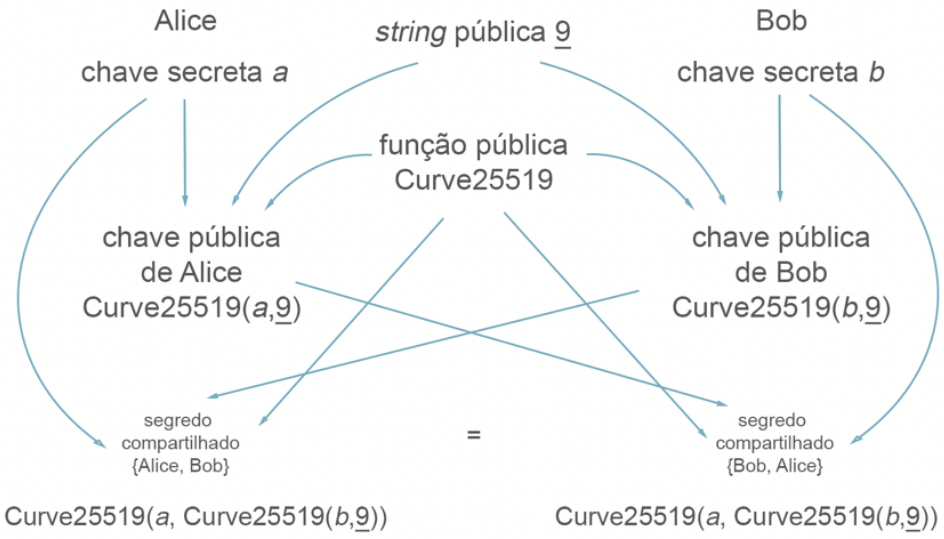

Figura 9: Diagrama do algoritmo CURVE25519. Adaptado de [14].

Um *hash* do segredo Curve25519(*a, Curve25519*(*b,9*)) é utilizado como chave para autenticação de mensagens em um sistema baseado em chaves privadas, ou para sistemas baseados em autenticação e cifragem simultaneamente [14].

2.3.7 POLY1305-AES

O Poly1305-AES é um algoritmo de autenticação de mensagens (*MAC – message authentication code*), utilizado para garantir a autenticidade e a integridade de mensagens. A sua função é uma avaliação polinomial direta (*straightforward polynomial evaluation*) mod 2^{130}-5. O Poly1305 é um MAC que toma uma chave secreta única de 32 bytes e uma mensagem e retorna uma *tag* de 16 bytes, *tag* esta utilizada para realizar a autenticação. A sua função utiliza como atributos uma chave AES de 128 bits – utilizada para encriptar o *nonce*, uma "chave adicional" de 128 bits, e um *nonce* não-secreto [15].

Mais especificamente a chave de 32 bytes compartilhada é composta de duas partes, a primeira é uma AES de 16 bytes representada como *k,* a segunda uma *string* de 16 bytes representada como *r*[0..15]. Esta por sua vez representa em *little-endian* um inteiro do tipo *unsigned* de 128 bits, de modo que $r = r[0]+2^8r[1]+,...,2^{120}r[15]$.

Cada mensagem é acompanhada, obrigatoriamente, por um *nonce* de 16 bytes, e cada *nonce n* é obtido através de uma $AES_k(n)$. Segundo o próprio artigo original [15], não há nenhum motivo especial para a utilização do padrão AES aqui, podendo o seu uso ser substituído por qualquer outra função que utiliza uma chave pertencente a um grupo arbitrário de *nonces* para *strings* de 16 bytes.

Neste algoritmo, é definido o processo de conversão e de *padding* (preenchimento) de forma que a cada fragmento de 16 bytes é anexado o número 1 a fim de compor então um fragmento de 17 bytes. No caso de fragmentos de tamanho entre 1 e 15 bytes, é adicionado um "1" ao final, e os espaços vazios para atingir o tamanho de 17 bytes são completados com zeros. Desta forma, o resultado desse processo será sempre um fragmento – ou "*chunk*", como definido na sua documentação – inteiro *unsigned* de 17 bytes *little-endian*. De forma matemática, seja *m*[0], *m*[1],...,m[l-*1*] uma mensagem *m,* $q = [\frac{l}{16}]$e $c_1, c_2,..., c_q \in \{1, 2, 3,..., 2^{129}\}$ são inteiros definidos pelas Equações de 6 a 8:

$$c_i = m[16i - 16] + 2^8 m[16i - 15] + 2^{16} m[16i - 14] + \cdots + 2^{120} m[16i - 1] + 2^{128} \quad (6)$$

se $1 \leq i \leq [\frac{l}{16}]$, e no caso de *l* não ser múltiplo de 16:

$$c_q = m[16q - 16] + 2^8 m[16q - 15] + \cdots + 2^{8(l \bmod 16)-8} m[l - 1] + 2^{8(l \bmod 16)-16} \quad (7)$$

Por fim, a transformação $Poly1305_r(m, AES_k(n))$ é definida como o autenticador Poly1305 de uma mensagem *m,* com um *nonce n*, sob as chaves (*k, r),* de representação:

$$\left(\left(\left(c_1 r^q + c_2 r^q + \cdots + c_q r^1\right) mod\ 2^{130} - 5\right) + AES_k(n)\right) mod\ 2^{128} \quad (8)$$

onde $AES_k(n)$ é uma *string* de 16 bytes tratada como um inteiro *unsigned little-endian*.

2.3.8 AEAD – AUTHENTICATED ENCRYPTION WITH ASSOCIATED DATA

AEs (*Authenticated Encryption*), ou em português Cifragem Autenticada, são mecanismos que envolvem chaves simétricas através dos quais uma mensagem simples M sofre uma transformação em uma mensagem cifrada C de modo que a mensagem cifrada C garante simultaneamente a privacidade e a autenticidade. De modo complementar, os esquemas de *Authenticated Encryption with Associated Data,* i.e., Cifragem Autenticada com Dados Associados, são esquemas AE que não demandam uma mensagem cifrada integral em aplicações que envolvem dados secretos e não-secretos; de modo que as partes secretas são cifradas e as não secretas podem continuar em mensagem base, fornecendo privacidade para a parte secreta e autenticidade a ambas as partes [16]. Uma generelização de como funcionam algoritmos de AEAD pode ser observado na Figura 10.

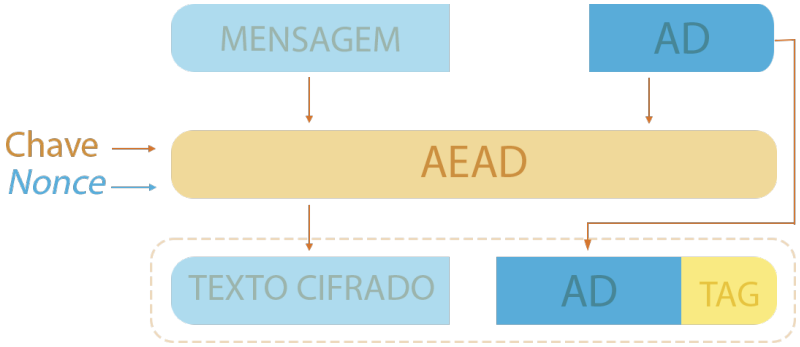

Figura 10: Esquema genérico de AEAD. Adaptado de [17].

2.3.9 *PERFECT FORWARD SECRECY* (SIGILO DE ENCAMINHAMENTO PERFEITO)

Alguns aspectos relevantes para os protocolos de estabelecimento de chaves podem impactar diretamente na sua segurança, como o potencial de comprometimento das chaves de uso prolongado ou o comprometimento das chaves de sessões anteriores. Para um protocolo ser considerado como tendo o atributo de Sigilo de Encaminhamento Perfeito (*Perfect Forward Secrecy),* também conhecido como "proteção de quebra-reversa" (*break-backward protection*), o comprometimento das chaves em utilização e/ou de uso prolongado não pode comprometer as demais chaves utilizadas anterior ou posteriormente.

2.4 COMPUTAÇÃO QUÂNTICA

Os computadores atuais têm como base sistemas digitais cujo funcionamento é fundamentado em uma lógica binária. O avanço destes sistemas na capacidade de processamento se dá principalmente pelo aumento da capacidade de inserção de micro ou nano circuitos em unidades de área cada vez menores, aumentando-se assim a complexidade do sistema e sua capacidade de processamento simultâneo de bits. Ainda assim, estas máquinas ainda operam em uma lógica binária com sinais de tensão. Na computação clássica, de forma geral, existem dois

fatores que limitam sua capacidade de resolução de problemas extensos, tempo (passos computacionais) e memória [6]. Toda esta lógica impõe restrições ao poder de processamento dos computadores clássicos, fazendo com que estes não sejam capazes de realizar determinadas operações em um tempo plausível. A título de exemplo, computadores binários são capazes de fatorar números inteiros em números primos, uma atividade não muito prática de ser realizada manualmente; entretanto, caso essa fatoração envolva números inteiros grandes o suficiente, como os comumente utilizados para criptografia, essa atividade se torna impraticável até mesmo para computadores comuns [3]. Outro exemplo que não envolve diretamente poder de processamento, mas artifícios de lógica computacional, é a geração de números ou *strings* de caracteres verdadeiramente aleatórios por parte destes computadores. A sua construção lógica só permite que simulem aleatoriedade para estes tipos de valores, o que causa prejuízo ao funcionamento de aplicações que se baseiam em números verdadeiramente aleatórios.

Sistemas quânticos, por outro lado, têm sua arquitetura baseada em *Qubits, "*bits quânticos", que possuem *spin*, polarização e a propriedade de superposição. O funcionamento dos *qubits* é possível porque os objetos base da arquitetura de computadores quânticos são os estados quânticos de partículas quânticas. Um *qubit* é um sistema ainda assim baseado em 1's e 0's, mas devido à superposição, um *qubit* pode encontrar-se em todos os estados possíveis ao mesmo tempo antes que seja possível medi-lo [3]. Um sistema computacional clássico requer em sua modelagem de estado *n* bits, enquanto numa arquitetura quântica um estado requer *2n – 1* bits para a sua representação [18]. Deixando de lado os estados mistos, Peter Shor elenca em seu artigo *Polynomial-Time Algorithms Prime Factorization* [6] que "(...) o estado de um sistema quântico é um ponto em um vetor de espaço de dimensão *2n*. Para cada uma das posições clássicas possíveis dos componentes, existe uma base de estado desse espaço de vetor o qual é representado, por exemplo, com $|0111\ldots0\rangle$, significando que o primeiro bit é 0, o segundo bit é 1, e assim por diante. Aqui, a notação *ket* $|x\rangle$ denota que x é um estado quântico (puro)" [18].

Na definição de Shor, e de forma geral a sustentar aplicações quânticas, esse espaço associado ao sistema quântico é um espaço de Hilbert, i.e., um espaço vetorial dotado de produto interno que não é limitado por um número de dimensões fixo. Neste caso específico, o espaço vetorial complexo tem como bases esses 2^n estados, e um estado deste sistema em um determinado momento pode ser representado por um vetor unitário. A multiplicação deste estado por uma fase complexa unitária não altera o seu comportamento, portanto são necessários 2^n-1 números complexos para descrever o estado de forma integral nesta superposição de estados. Tal superposição pode ser representada por um somatório do tipo representado na Equação 9:

$$\sum_{i=0}^{2^n-1} a_i|S_i\rangle \tag{9}$$

De modo que a_i são amplitudes que representam números complexos tais que $\sum_i |a_i|^2 = 1$, e $|S_i\rangle$ é uma base vetorial do espaço de Hilbert. Isto implica que, adotando como referencial a base $|S_i\rangle$, caso a máquina venha a ser medida durante algum passo específico, a probabilidade de observação do estado base é $|a_i|^2$. Por outro lado, a tentativa de realizar a medição de um estado da

máquina implica na projeção deste estado para o vetor base $|S_i\rangle$ observado. Tal generalização é suficiente para permitir a expansão dos estados quânticos em combinações probabilísticas que aproximam o comportamento de circuitos quânticos em função de seus estados com base em seu número de *qubits.*

Computadores quânticos também precisam computar dados seguindo algoritmos devidamente programados para chegar a qualquer tipo de processamento, mas é seguro dizer que, ao atingir o seu funcionamento ideal, estes computadores terão um processamento muito mais rápido do que os computadores digitais. A empresa multinacional americana IBM, que tem grande relevância na área da computação, detém uma métrica utilizada para a avaliação da performance de computadores quânticos chamada *quantum volume*, que avalia a quantidade de trabalho quântico que um computador quântico pode realizar em um determinado período. Essa métrica envolve diversos fatores, incluindo número de *qubits* e a sua interação com outros componentes, tempo de coerência, medidas de erros, conectividade com outros dispositivos e eficiência de compilação [3].

Até o presente momento, ainda não há um consenso sobre a arquitetura a ser adotada em computadores quânticos, nem os mecanismos a serem utilizados para sua implementação funcional. Os resultados obtidos resumem-se a demonstrar as características quânticas em pequena escala que viriam a possibilitar a resolução de problemas que computadores digitais ainda não conseguem. Porém os riscos à segurança da informação baseada na computação digital em decorrência da iminência do cenário quântico são reais e devem ser considerados no desenvolvimento de novos métodos de segurança. Nas palavras de Roger A. Grimes, ex-principal arquiteto de segurança da *Microsoft* [3]:

> Em algum ponto, computadores quânticos serão capazes de solucionar problemas que computadores tradicionais não conseguem solucionar de maneira alguma, bem como solucionar problemas que computadores tradicionais conseguem resolver, mas resolvê-los de maneira significativamente mais rápida. Este momento é conhecido como Supremacia Quântica (ou Vantagem Quântica, como cunhado pela IBM). Nós parecemos estar bem próximos.

2.5 CIRCUITOS QUÂNTICOS

O modelo de circuitos quânticos é o modelo mais utilizado para se tomar como ponto de partida a fim de generalizar o estudo de computação e algoritmos quânticos, atendo-se ao estudo de um grupo particular de circuitos chamados *uniformes*, grupo de circuitos em que o circuito que representa uma dada entrada (*input*) de tamanho *n* possa ser gerado em função de *n* [19]. No escopo deste trabalho, para discutir questões acerca das aplicações da computação quântica com base em seu funcionamento, é suficiente considerar a atuação de circuitos quânticos como circuitos uniformes envolvendo medições em *qubits* sob a ação de operadores quânticos.

2.6 ALGORITMOS QUÂNTICOS

Um algoritmo computacional clássico é uma sequência finita e ordenada de passos a ser executada por uma máquina computacional clássica a fim de executar uma ação ou resolver um

problema. Analogamente, um algoritmo quântico segue esta mesma definição, entretanto, aplicado ao campo da computação quântica. Algoritmos quânticos são algoritmos teóricos para execução em um modelo computacional quântico realístico. A maioria destes algoritmos foi desenvolvida sob modelos de computação discretos, i.e., que se baseiam em um espaço de estados e passos de tempo (*time steps)* discretos [19].

A maioria dos algoritmos quânticos foi desenvolvida com base na ideia geral da computação reversiva. Uma operação computacional reversiva é uma operação para a qual todas as distribuições de probabilidade para uma regra de transmissão probabilística sobre os estados inicial e final possuem valores diferentes de zero que não se sobrepõem; i.e., para a qual cada valor possível num conjunto de estados finais existe no máximo um valor correspondente num conjunto de estados iniciais em que a distribuição de probabilidade é maior que zero, e igual a zero para valores inexistentes no conjunto [20].

A tese de Church-Turing [21] diz que toda função que poderia ser naturalmente computável pode ser realizada por uma máquina de Turing determinística. Para a consideração do uso de algoritmos quânticos é possível realizar uma aproximação para a tese por meio da baseada em uma máquina de Turing probabilística, poderia implementar eficientemente, em termos específicos, qualquer modelo computacional [19]. Posto isso, a consideração de algoritmos quânticos se faz plausível quando consideramos a sua execução em qualquer modelo realista de um computador quântico.

2.6.1 COMPLEXIDADE COMPUTACIONAL

A Complexidade Computacional, do ponto de vista do algoritmo, pode ser depreendida como o custo de recursos para a execução sucedida de um algoritmo no pior dos cenários, i.e., esgotando-se todas as possíveis tentativas até o seu êxito. Comumente é formulada em função do tamanho da entrada do algoritmo. Em relação à solução de um problema, a Complexidade Computacional é o custo mínimo de recursos utilizados por um algoritmo para a solução do problema em questão [19].

2.6.2 ALGORITMO DE SHOR

Um dos principais motivos para a preocupação com um cenário pós-quântico da computação se deve ao fato de que em 1996 o matemático Peter Shor desenvolveu um algoritmo computacional quântico que, em tese, é capaz de solucionar o problema da fatoração por números primos e do logaritmo discreto em tempo praticável; de forma que dado um número inteiro N, o algoritmo é capaz de encontrar os seus fatores primos em tempo polinomial *log* N [6]. Em seu artigo, Shor propõe um método de construção de uma porta lógica reversível que toma um espaço O(l) e um tempo O(l^3) para computar uma solução do tipo (a, x^a (mod n)) a partir de a, considerando a, x e n como números de comprimento de l-bits onde x e n são primos entre si (*relatively prime numbers*).

No caso da implementação concreta deste algoritmo por computadores quânticos, seria possível realizar ataques contra esquemas criptográficos que envolvem primitivas redutíveis a fatorização por números primos ou logaritmo discreto, como RSA, Diffie-Hellman, Diffie-Hellman-Curva-Elíptica (ECDH), dentre outros [6].

2.6.3 ALGORITMO DE GROVER

Outro algoritmo que traz à tona a discussão sobre as possíveis implicações da implementação funcional de um computador quântico é o proposto por Lov Grover no ano de 1996 em seu artigo "*A fast quantum mechanical algorithm for database search"* [7]*,* que traz uma proposta para o aprimoramento na resolução do problema da busca não-estruturada.

Uma busca não estruturada, tratada originalmente no inglês como *unstructured search*, é assim chamada devido à ação de realizar uma pesquisa em um banco de dados, *lato sensu,* sob o fato do desconhecimento de como este encontra-se ordenado, i.e., quando se desconhece as regras de estruturação da base de dados. Para bases de dados classificadas, é possível, por exemplo, executar uma busca a fim de encontrar um elemento em tempo logarítmico; o que não ocorre em bases não-estruturadas, nas quais uma pesquisa utilizando circuitos computacionais clássicos demandaria uma quantidade linear de consultas para encontrar um elemento específico.

Dados N elementos em uma base de dados genérica não-estruturada, uma pesquisa necessitaria varrer linearmente uma média de $N/2$ elementos para encontrar o elemento desejado, e, no pior dos cenários, necessitaria varrer os N elementos. Uma definição apropriada para este problema é: dado um problema de busca não-estruturada em um conjunto de N elementos os quais formam um conjunto do tipo $X = \{x_1, x_2, \ldots x_n\}$ e dada uma função booleana $f : X \{0,1\}$; o problema tem como meta encontrar o elemento x^* em X tal que $f(x^*) = 1$ [22].

O algoritmo de Grover provoca a discussão que envolve o uso da computação quântica para realizar ataques quânticos de pesquisa de chave, por exemplo.

2.7 NIST E A PADRONIZAÇÃO DA CRIPTOGRAFIA PÓS-QUÂNTICA

O NIST (*National Institute of Standards and Technology*) é um instituto equiparável a uma agência não governamental estadunidense que possui relevância internacional em atividades de cunho econômico e tecnológico. No que diz respeito ao desenvolvimento de novas tecnologias no campo da criptografia pós quântica, o NIST em 2016 [5] assumiu a liderança ao iniciar a avaliação de submissões de implementações de algoritmos criptográficos candidatos a serem considerados quântico-resistentes, e estabeleceu como objetivo do processo de avaliação "desenvolver sistemas criptográficos que sejam seguros contra computadores quânticos e clássicos e possam interoperar com protocolos e redes de comunicação existentes" [5]. O processo de submissão e avaliação de candidatos a algoritmos criptográficos quântico-resistentes foi organizado em rodadas, onde os candidatos foram avaliados em 3 rodadas.

Neste contexto, foram estabelecidas premissas criptográficas pós-quânticas a serem seguidas pelos candidatos, a fim de garantir a segurança almejada em soluções quântico-resistentes para aplicações baseadas em criptografia de chaves públicas, bem como definições para segurança em criptografia/estabelecimento de chaves (*key-estabilishment*) simples e com uso chaves efêmeras e em assinaturas digitais.

A definição mais importante feita pelo NIST para o escopo deste trabalho é a das categorias de força de segurança a serem almejadas pelos algoritmos criptográficos candidatos de acordo com os propósitos específicos de cada algoritmo. Tais categorias foram definidas em intervalos de força de segurança baseados nos critérios conhecidos e estabelecidos pelo NIST para criptografia simétrica, considerada significativamente resistente à criptoanálise quântica [23]; e encontram-se

discriminadas na Tabela 2, por ordem de recursos computacionais necessários para se realizar um ataque bem-sucedido.

Tabela 2: Níveis de Segurança para Algoritmos Pós-Quânticos. Adaptado de [23].

NÍVEL DE SEGURANÇA	TIPO DE ATAQUE	REQUISITO DE SEGURANÇA (COMPARÁVEL OU SUPERIOR)
I	Pesquisa de Chave	Cifra de Bloco de Chave de 128 bits (AES128)
II	Pesquisa de Colisão	Função de Hash de 256 bits (SHA256/SHA3-256)
III	Pesquisa de Chave	Cifra de Bloco de Chave de 192 bits (AES192)
IV	Pesquisa de Colisão	Função de Hash de 384 bits (SHA384/SHA3-384)
V	Pesquisa de Chave	Cifra de Bloco de Chave de 256 bits (AES256)

O NIST considera que ataques quânticos podem ser medidos em termos de profundidade de circuito ou em tempo de execução, definindo um parâmetro chamado MAXDEPTH, que é considerado relevante num cenário de execução de séries computacionais grandes o suficiente. O MAXDEPTH pode ser definido em um intervalo de 2^{40} portas lógicas a aproximadamente 2^{96} portas lógicas, que são, respectivamente, os números de portas lógicas com previsão de implementação bem-sucedida no período de 1 ano e o estimado de portas que *qubits* de escala atômica com a velocidade de propagação da luz poderia contemplar em um milênio [23]. A complexidade de ataques quânticos pode ser medida em termos de tamanho do circuito, parâmetro que pode ser usado como comparação para recursos necessários para a quebra de algoritmos SHA3 e AES. O NIST usa como estimativa para a contagem de portas lógicas clássicas e quânticas necessárias para ataques bem-sucedidos de recuperação de chave e ataques de colisão em AES e SHA3 os parâmetros definidos na Tabela 3, com base no MAXDEPTH:

Tabela 3: Relação de Profundidade de Circuito para Prováveis Ataques Bem Sucedidos em Algoritmos Criptográficos. Adaptado de [23].

ALGORITMO EM ATAQUE	CONTAGEM DE PORTAS
AES128	2^{170}/MAXDEPTH portas quânticas ou 2^{143} portas clássicas
SHA3-256	2^{146} portas clássicas
AES192	2^{233}/MAXDEPTH portas quânticas ou 2^{207} portas clássicas
SHA3-384	2^{210} portas clássicas
AES256	2^{298}/MAXDEPTH portas quânticas ou 2^{272} portas clássicas
SHA3-512	2^{274} portas clássicas

O NIST assegura que o uso das primitivas AES e SHA3 como referências para avaliar a segurança pós-quântica no uso de novos algoritmos é suficientemente relevante. Elenca-se como exemplo que a categoria dos algoritmos AES128, AES192 e AES256 “são definidas em termos de cifras de bloco, que podem ser quebradas usando o algoritmo de Grover, com uma aceleração quântica quadrática. Mas o algoritmo de Grover requer uma computação serial de longa execução, que é difícil de implementar na prática. Em um ataque realista, é preciso executar muitas instâncias menores do algoritmo paralelamente, o que torna a aceleração quântica menos dramática” [23].

Não obstante, o NIST estabelece como norteadores outros princípios qualitativos que devem ser levados em conta no desenvolvimento de soluções quântico-resistentes tais como:

a) Sigilo de Encaminhamento Perfeito (*Perfect Forward Secrecy*):

Deve-se fazer uma avaliação precisa da utilização de algoritmos que garantem PFS, como por meio de criptografia simétrica e de protocolos de assinatura digital, dado que os efeitos da sua utilização podem agregar outros custos, a exemplo da utilização de algoritmos de geração de chaves muito lentos, como no RSA, que devem ser evitados. Portanto, é necessário ponderar a interação significativa entre custos e segurança prática de um algoritmo.

b) Resistência a Ataques de Canais Laterais (*Side-Channel Atacks*):

Ataques de canais laterais são ataques que se aproveitam dos efeitos colaterais dos processamentos em máquinas, como vazamentos de informações como tempo de execução e recursos computacionais utilizados na execução de tarefas; por meio de irradiação de campos eletromagnéticos, calor ou som dissipados [24]. O NIST orienta que os algoritmos candidatos almejem a resistência a estes a ataques através do menor custo possível, de forma que a performance do algoritmo não seja prejudicada pela tentativa de resistência ao ataque.

c) Resistência a Ataques de Múltiplas Chaves (*Multi-Key Attacks*):

Os algoritmos candidatos devem procurar meios de evitar que o atacante consiga realizar ataques com a tentativa de diversas chaves de uma vez, com o objetivo de comprometer muitas chaves ou apenas uma delas.

d) Resistência a Más-Práticas (*Misuse*):

Os esquemas criptográficos devem ser suficientemente sólidos a fim de evitar falhas nas suas execuções, sejam elas devido a erros na codificação dos algoritmos, mau funcionamento dos geradores de números aleatórios, reutilização de *nonces*, reutilização de pares de chaves na criptografia com chaves efêmeras, dentre outros.

2.8 REDES PRIVADAS VIRTUAIS (*VPN'S)*

Redes privadas virtuais, ou VPNs (*Virtual Private Networks)*, são redes que funcionam sob arquitetura de túneis criptográficos entre pontos de acesso autorizados, implementados sobre uma rede pública, como a Internet, ou privada, para garantir o tráfego seguro de informações entre pontos distintos conforme ilustrado na Figura 10. As VPNs podem servir como mecanismos de acesso remoto via Internet, como no acesso remoto a redes corporativas por meio de um ponto local e um provedor de acesso (*Internet Service Provider*); podem também realizar uma conexão entre redes locais distintas, dispensando-se assim circuitos de transmissão de longa distância; ou ainda como conexão direta entre computadores em uma rede interna [25].

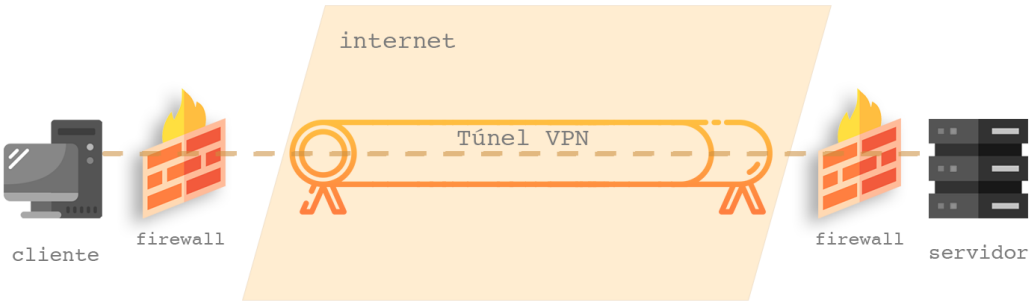

Figura 11: Esquemático do funcionamento de um Túnel VPN.

A implementação de VPNs possui alguns requisitos básicos a serem cumpridos para sua operação segura, como controle de acessos com autenticação, o gerenciamento adequado de endereços, arquitetura criptográfica com gerenciamento de chaves e a definição dos protocolos de redes. As VPNs têm como princípio de funcionamento o tunelamento, i.e., encapsulamento, transmissão ao longo de uma rede intermediária e desencapsulamento dos pacotes. O encapsulamento diz respeito ao encapsulamento de protocolos, aplicando camadas de criptografia. O protocolo de tunelamento encapsula o pacote com um cabeçalho contendo as informações de roteamento que permitem o tráfego das informações entre os pontos do túnel, o caminho lógico pelo qual serão trafegados os dados; e pode ocorrer na camada de enlace ou na de rede [25].

2.9 *WIREGUARD*

Com base no descrito no *whitepaper "WireGuard: Next Generation Kernel Network Tunnel"* por Jason Donenfeld [4], o *WireGuard* [Fig. 11], é um software de VPN de uso geral de código aberto baseado na utilização de criptografia estado-da-arte de forma mais simples, ágil e segura do que os seus equivalentes-concorrentes, e se encontra em constante desenvolvimento e aprimoramento.

Figura 12: Logo do Projeto WireGuard [4].

A configuração do *WireGuard* e sua implementação se tornam mais simples em comparação aos demais softwares ao utilizar uma interface visual que pode ser configurada usando os utilitários *ip(8)* e *ifconfig(8)* padrão, e de forma resumida, configurando-se a chave privada e as chaves públicas dos *peers* com os quais se deseja estabelecer uma conexão segura. Desta forma, as trocas de chaves e os procedimentos de conexão acontecem de forma transparente para o administrador, que pode se abster dos detalhes se assim o quiser. Toda essa aparente simplicidade tem o intuito de tornar o software mais "enxuto" em termos de código, tornando-o também menos propenso a erros críticos e facilitando sua auditabilidade.

2.9.1 KEY MANAGEMENT

Para a distribuição de chaves, o *WireGuard* tem como inspiração o conjunto de utilitários de rede *OpenSSH* [4] para realizar a gestão de chaves através de um conjunto diverso de mecanismos não restritamente especificados, desde a simples utilização do software de criptografia *PGP* (*Pretty. Good Privacy*) a mecanismos mais complexos como a distribuição de chaves utilizando o protocolo *Lightweight Directory Access Protocol* (*LDAP*) e autoridades certificadoras. A não especificação destes mecanismos é devido à inclinação da implementação do *WireGuard* a não se ater a esta camada para resolução do problema de segurança na distribuição de chaves. Portanto, a interface é simples ao ponto de qualquer solução de distribuição de chaves poder ser utilizada. As chaves públicas utilizadas são especificamente do tamanho de 32 bytes e podem ser facilmente representadas em codificação Base64 em 44 caracteres [4].

2.9.2 CRIPTOGRAFIA NO *WIREGUARD*

O *WireGuard* é intencionalmente carente no que diz respeito à agilidade das cifras e dos protocolos, dado que a agilidade das cifras aumenta a complexidade do sistema de forma relevante [4]. O seu protocolo utiliza diversos recursos diferentes para garantir a segurança, por exemplo, se lacunas são encontradas nas primitivas subjacentes, todos os *endpoints* serão requisitados a atualizar. No *WireGuard* são utilizadas uma variação do protocolo *Noise* [26], que é um protocolo criptográfico Diffie-Hellman desenvolvido por Trevor Perrin; para uma troca de chaves em 1-RTT (*round trip time*), com *Curve25519* para *ECDH*; uma função de derivação de chave *hash HKDF* para expansão dos resultados da *ECDH*, um algoritmo ChaCha20Poly1305 para cifragem autenticada e um algoritmo BLAKE2s para um *hash.* Todos estes procedimentos serão descritos de forma mais detalhada na próxima seção.

2.9.3 TRANSPARÊNCIA PARA *PEERS* ILEGÍTIMOS

O *WireGuard* foi projetado para mitigar a alocação de qualquer estado prévio à autenticação de um *peer* e também para evitar estabelecer qualquer tipo de comunicação com pacotes sem autenticação, de forma que tais estados e pacotes não são, respectivamente, armazenados e respondidos, visando assim tornar-se invisível para *peers* ilegítimos e *scanners* de rede [4].

2.9.4 MODO DE PRÉ-COMPARTILHAMENTO DE CHAVE SIMÉTRICA

Existe um modo opcional no *WireGuard* que permite, de forma adicional, a troca prévia de chaves simétricas de 256 bits entre dois pares, como forma de prover uma camada a mais de segurança; considerando que o funcionamento do software já é baseado na troca de chaves públicas estáticas sustentadas numa função de Curva Elíptica Diffie-Hellmann Curve25519 como forma de atribuir identificação aos pares. Este modo é uma alternativa fornecida a mais para mitigar os efeitos da computação quântica [4].

2.9.5 PREVENÇÃO A ATAQUES DoS

Para atender aos critérios de autenticação nas mensagens de *handshake*, uma operação de Curve25519 deve ser computada, e este fato *per se* oferece a possibilidade de mais uma ferramenta para ser utilizada como forma de prevenção a ataques de negação de serviço (DoS) e outros ataques que partilham de características semelhantes. Com base nisto, é possível prevenir um ataque de exaustão no receptor da solicitação do *handshake*, de modo que este pode se recusar a processar a solicitação caso esteja sendo estressado e/ou reconheça como uma tentativa de ataque, respondendo à solicitação com um *cookie*, o qual será necessário para o iniciador da mensagem enviar uma nova solicitação. Este mecanismo também será melhor explorado na próxima seção.

3 MÉTODOS EMPREGADOS NO *WIREGUARD*

O protocolo de troca de pacotes criptografados encapsulados é iniciado com a interação entre iniciador (*iniciator)* e respondedor (*responder)* através de um *handshake* de troca de chaves de 1-RTT. Antes de iniciar a troca de chaves*,* assume-se que ambas as partes, iniciador e respondedor, possuem pares de chaves gerados e que a chave pública do respondedor é conhecida pelo iniciador. O processo de envio de mensagens encriptadas inicia-se por meio do uso destes pares de chaves simétricas compartilhados, uma para o envio e outra para recebimento, e com o envio da primeira mensagem pelo iniciador, e após o envio, o respondedor encontra-se apto também a mandar mensagens para o iniciador. Tais mensagens utilizam o padrão "IK" do protocolo *Noise* [27] combinado com um mecanismo de resposta de *cookie* para evitar ataques DoS.

Doravante, serão utilizados para denotar os titulares das componentes os subscritos *i* e *r* para iniciador e respondedor, respectivamente. Quando não houver restrição a qualquer um dos titulares, será utilizado o subscrito *.

Na troca de mensagens, ambas as partes detêm algumas variáveis localmente, e neste trabalho, assim como no trabalho original [4], serão denotadas conforme mostrado na Tabela 4.

Tabela 4: Tabela de variáveis envolvidas nos processos do WireGuard. Adaptado de [4].

I_*	Um índice de 32 bits que representa localmente o outro dispositivo envolvido na comunicação.
S_*^{PRIV}, S_*^{PUB}	Os valores das chaves públicas e privadas estáticas.
E_*^{PRIV}, E_*^{PUB}	Os valores das chaves privada e pública efêmeras.
Q_*	Valor da chave simétrica pré-compartilhada opcional. Caso o modo não seja escolhido, o valor default é 0^{32}.
H_*, C_*	Um valor de resultado de hash e um valor de encadeamento.
T_*^{SEND}, $T_*^{RECV.}$	Valores de chave simétrica de dados de transporte de envio e recepção.
N_*^{SEND}, N_*^{RECV}	Contadores *Nonce* da mensagem de dados de transporte.

Definam-se ainda as funções envolvidas no processo e suas constantes como o sinalizado na Tabela 5.

Tabela 5: Funções e Variáveis Específicas no WireGuard. Adaptado de [4].

REPRESENTAÇÃO DAS FUNÇÕES E VARIÁVEIS	DESCRIÇÃO
DH(PRIVATE KEY, PUBLIC KEY)	Multiplicação de ponto **Curve25519** entre a chave privada e a chave pública, retornando uma saída de 32 bytes.

DH-GENERATE( )	Geração de uma chave privada Curve25519 aleatória e sua chave pública correspondente, retornando uma saída de 32 bytes (privada, pública).
AEAD(KEY, COUNTER, PLAIN TEXT, AUTH TEXT)	**AEAD** utilizando os algoritmos **ChaCHa20 e Poly1305 (RFC7539)** com um *nonce* de 32 bits de zeros seguidos de um contador *little-endian* de 64 bits.
XAEAD(KEY, NONCE, PLAIN TEXT, AUTH TEXT)	AEAD utilizando XChaCha20Poly1305, com um *nonce* aleatório de 24 bytes baseado em uma HChaCha20 e uma ChaCha20Poly1305.
HASH(INPUT):	Um *hash* do tipo BLAKE2s(INPUT, 32) com uma saída de 32 bytes.
MAC(KEY, INPUT):	Uma variação do BLAKE2s para MAC chaveado, do tipo KEYED-BLAKE2s (Key, Input, 16), com uma saída de 16 bytes.
HMAC(KEY, INPUT)	Um HMAC-BLAKE2s(KEY, INPUT, 32), um tipo de BLAKE2s para implementação de HMACs, com saída de 32 bytes.
TIMESTAMP( )	Retorna um *timestamp* do tipo TAI64N, ou o tempo real, no tamanho de 12 bytes.
CONSTRUCTION	Uma *string* literal UTF-8 “Noise_IKpsk2_25519_ChaChaPoly_BLAKE2s” com saída de 37 bytes.
IDENTIFIER	Uma *string* literal UTF-8 “WireGuard v1 zx2c4 Jason@zx2c4.com”, de 34 bytes na saída.
LABEL-MAC1	Uma *string* literal UTF-8 “mac1----”, de saída de 8 bytes.
LABEL-COOKIE	Uma *string* literal UTF-8 “cookie—“, de saída de 8 bytes.

Definidas estas funções, pode-se prosseguir para o detalhamento do processo de *handshake*.

3.1 AVALIAÇÃO DO PROCESSO DE *HANDSHAKE*

De forma geral, deixando de lado a abordagem de prevenção a ataques DoS por enquanto, o protocolo de *handshake* é realizado da forma como consta na Figura 12. O detalhamento de cada passo de forma isolada será realizado nas subseções posteriores. Note-se que, o primeiro dos passos para inicialização da comunicação entre duas partes quaisquer se dá com a geração de um par de chaves estáticas assimétricas, das quais serão derivadas as chaves efêmeras a serem utilizadas nas mensagens.

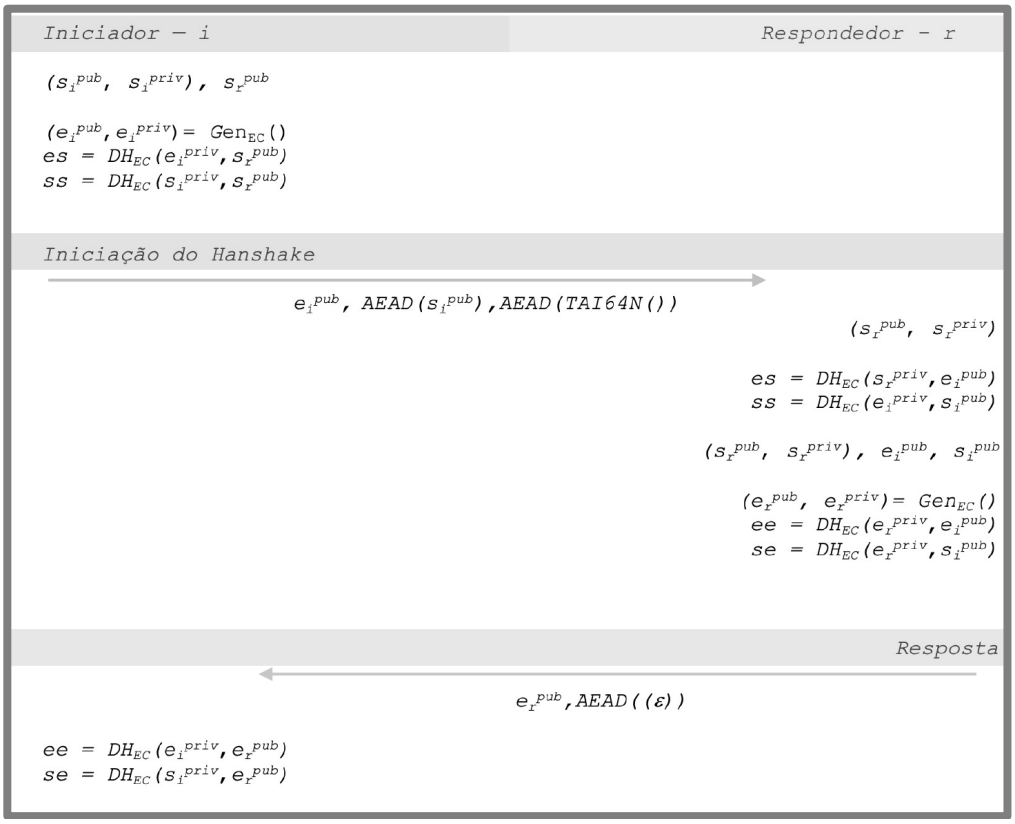

Figura 13: Esquemático do estabelecimento de chaves no Handshake.

Pode-se observar na Figura 12 que, com os respectivos pares de chaves estáticas e efêmeras gerados, na mensagem inicial de troca de chaves, o iniciador envia a sua chave efêmera pública, a sua chave estática pública sob *AEAD* e um carimbo de tempo TAI64N também sob AEAD. Note-se que para o cálculo das funções AEADs são utilizadas as chaves geradas sob criptografia de curva elíptica Diffie-Hellmann.

Com as mensagens devidamente trocadas e o *handshake* feito, o procedimento para a geração de chaves para o tráfego de dados é seguido, calculando-se as chaves pelo iniciador e pelo respondedor. O detalhamento dos processos envolvidos na computação dos parâmetros transmitidos será realizado mais a.

3.1.1 TROCA DE MENSAGENS: MENSAGENS DO INICIADOR E DO RESPONDEDOR.

Considerando-se que as chaves estáticas assimétricas foram geradas por ambas as partes, podemos dar início à descrição do processo de *handshake*. A mensagem inicial, que é enviada pelo iniciador da solicitação de *handshake*, é estruturada conforme a Figura 13.

type := 0x1 (1 byte)	reserved := 0^3 (3 bytes)
sender := I_i (4 bytes)	
ephemeral (32 bytes)	
static ($\widehat{32}$ bytes)	
timestamp ($\widehat{12}$ bytes)	
mac1 (16 bytes)	mac2 (16 bytes)

Figura 14: Estrutura da mensagem do iniciador [4].

A princípio, é gerado o índice I_i de 32 bits referente ao iniciador de forma aleatória (ρ^4) no envio da mensagem, a fim de atrelar as respostas decorrentes, após o início da solicitação, ao iniciador. Os campos a serem calculados após essa geração seguem os seguintes passos apresentados na Tabela 6:

Tabela 6: Computações das Variáveis do Iniciador. Adaptado de [4].

1	$C_i := HASH(CONSTRUCTION)$
2	$H_i :=$ HASH $(C_i \,\|\|$ IDENTIFIER)
3	$H_i :=$ HASH $(H_i \,\|\| \, S_r^{pub})$
4	$(E_t^{priv}, E_t^{pub}) :=$ DH-GENERATE()
5	$C_i :=$ KDH$_1(C_i, E_t^{pub})$
6	$msg.ephemeral := E_t^{pub}$
7	$H_i :=$ HASH $(H_i \,\|\| \, msg.ephemeral)$
8	$(C_i, \kappa) :=$ KDF$_2$ $(C_i,$ DH$(E_t^{priv}, S_r^{pub}))$
9	$msg.static :=$ AEAD$(\kappa, 0, S_t^{pub}, H_i)$
10	$H_i :=$ HASH $(H_i \,\|\| \, msg.static)$
11	$(C_i, \kappa) :=$ KDF$_2$ $(C_i,$ DH$(S_t^{priv}, S_r^{pub}))$
12	$msg.timestamp :=$ AEAD$(\kappa, 0, TIMESTAMP()\,, H_i)$
13	$H_i :=$ HASH$(H_i \,\|\| \, msg.timestamp)$

1. Primeiramente é gerado um valor de encadeamento atribuído a C_i como resultado de uma operação *hash* da *string* “Noise_IKpsk2_25519_ChaChaPoly_BLAKE2s”.
2. Em seguida, é realizado o *hash* da concatenação entre C_i e I_i e o resultado é atribuído a H_i, como um valor de *hash*.
3. A esse valor é então atribuído o resultado de uma função *hash* da concatenação entre H_i e a chave estática pública do respondedor, S_r^{pub}.

4. É então gerado um par de chaves DH efêmeras pública e privada do iniciador.
5. É atribuído à chave de encadeamento C_i o resultado de uma função de derivação de chave (*key derivation function – KDF)* entre C_i e a chave efêmera pública do iniciador, E_i^{pub}.
6. Um atributo de mensagem efêmera, *msg.ephemeral,* recebe o valor de E_i^{pub}.
7. É atribuído o resultado do *hash* entre H_i e *msg.ephemeral* a H_i.
8. C_i e κ são derivados por uma KDF a partir da chave C_i e o resultado de uma função DH entre a chave efêmera do iniciador E_i^{priv} e a chave estática pública do respondedor S_r^{pub}.
9. Ao atributo *msg.static* é atribuído o resultado da computação de uma AEAD (*Authenticated Encryption with Associated Data*) entre C_i e o resultado de uma DH entre as chaves estáticas privada e pública do iniciador e do respondedor, respectivamente.
10. H_i recebe a atribuição do valor do *hash* de si mesma concatenado com *msg.static.*
11. É atribuído então aos atributos C_i e κ o resultado da derivação de chaves utilizando C_i, e o resultado de uma DH envolvendo a chave estativa privada do iniciador e a chave estática pública do respondedor. Note-se que essa derivação é feita novamente após a utilização de κ anteriormente na AEAD do passo 9.
12. Um atributo *msg.timestamp* recebe o valor de uma computação AEAD entre κ, 0, o carimbo de tempo e H_i.
13. Por fim, H_i recebe o resultado do *hash* entre H_i e *msg.timestamp*.

De modo análogo à primeira mensagem enviada, uma mensagem de resposta à mensagem do iniciador é enviada após o processamento da requisição aplicando os mesmos mecanismos para chegar a um estado idêntico. A geração de I_r é feita ao enviar da mensagem de forma aleatória (ρ^4), e é igualmente utilizada para atar as respostas inicializadas pela mensagem. Os parâmetros da mensagem de resposta encontram-se resumidos na Figura 14.

type := 0x2 (1 byte)	reserved := 0^3 (3 bytes)
sender := I_r (4 bytes)	receiver := I_i (4 bytes)
ephemeral (32 bytes)	
empty ($\widehat{0}$ bytes)	
mac1 (16 bytes)	mac2 (16 bytes)

Figura 15: Estrutura da mensagem do respondedor [4].

Os parâmetros da mensagem do respondedor são calculados conforme disposto na Tabela 7 com os seguintes passos:

Tabela 7: Computações das Variáveis do Respondedor. Adaptado de [4].

1	(E_r^{priv}, E_r^{pub}) := DH-GENERATE()
2	$C_r := \text{KDF}_1(C_r, E_r^{pub})$
3	$msg.ephemeral_i := E_r^{pub}$
4	$H_r := \text{HASH}(H_r \,\|\|\, msg.ephemeral)$
5	$C_r := \text{KDH}_1(C_r, \text{DH}(E_r^{priv}, E_r^{pub}))$
6	$C_r := \text{KDH}_1(C_r, \text{DH}(E_r^{priv}, S_i^{pub}))$

7	$(C_r, \tau, \kappa) := \text{KDF}_3\ (C_r, Q)$
8	$H_r := \text{HASH}\ (H_r \| \tau)$
9	$msg.empty := \text{AEAD}(\kappa, 0, \epsilon, H_r)$
10	$H_r := \text{HASH}\ (H_r \| msg.empty)$

1. É gerado um par de chaves Diffie-Helmann e ele é atribuído às chaves efêmeras pública e privada do respondedor.
2. É atribuído a C_r o resultado de uma função de derivação de chave (*key derivation function* – KDF) entre C_r e a chave efêmera pública do respondedor, E_r^{pub}.
3. O objeto *msg.ephemeral* recebe E_r^{pub} .
4. H_r recebe a computação do *hash* da concatenação entre H_r e *msg.*ephemeral.
5. É atribuído a C_r o valor da função de derivação de chaves entre C_r e a computação Diffie-Hellman entre as chaves efêmeras privada do respondedor e pública do iniciador.
6. Em seguida é computada novamente uma KDF entre C_r e a computação Diffie-Hellman entre a chave efêmera privada do respondedor e a chave estática pública do iniciador e é atribuído o resultado a C_r.
7. É realizada então uma função de derivação de chaves envolvendo C_r e o valor da chave simétrica pré-compartilhada (ou 0^{32}), da qual resulta uma tripla de valores que cada um destes é atribuído a C_r, τ e κ.
8. É então atribuído a H_r o resultado de uma computação de uma função *hash* entre H_r e o τ obtido no passo anterior.
9. É atribuído à mensagem *msg.empty* o resultado da computação de uma função AEAD entre $\kappa, 0, \epsilon$ e H_r.
10. Por fim, H_r então recebe o resultado da computação do *hash* entre H_r e *msg.empty.*

A mensagem do respondedor é menor do que a mensagem de iniciação do *handshake*, a fim de mitigar as possibilidades de ataques de amplificação por sobrecarga no processamento das respostas. Após todos os passos executados, o iniciador recebe esta mensagem para executar as mesmas operações com as devidas substituições, de forma a produzir valores equivalentes por meio da utilização dos seus respectivos operandos das funções DH.

Em tempo, ambas as mensagens possuem os parâmetros *msg.mac1* e *msg.mac2.* Dentro do escopo do *handshake,* assume-se que o parâmetro msg_α representa todos os bytes referentes ao parâmetro *msg.mac1* e msg_β todos os bytes referentes ao parâmetro *msg.mac2.* Estes parâmetros são utilizados para a geração de *cookies* a fim de, por exemplo, mitigar as possibilidades de ataques como os de negação de serviço (DoS). Estes *cookies* são utilizados para possibilitar ao respondedor da mensagem validar se aceita a solicitação do iniciador ou não. O *cookie* recebido mais recentemente, identificado como $\widetilde{L_*}$, tem sua atualização representada como L_*. Posto isso, os parâmetros *msg.mac1* e *msg.mac2* têm a sua atribuição realizada conforme mostrado na Tabela 8.

Tabela 8: Computação dos parâmetros do MAC. Adaptado de [4].

$msg.mac1 := \text{MAC}(\text{HASH}(\text{LABEL-MAC1} \| S_m{}^{pub}), msg_\alpha)$
Se $L_m = \in$ ou $\widetilde{L_*} \geq 120$:
$msg.mac2 := 0^{16}$

caso contrário:

$$msg.mac2 := \text{MAC}(L_m, msg_\beta)$$

Textualmente, pode-se descrever a atribuição dos parâmetros MAC (Tab. 8) utilizados para os *cookies* da seguinte forma:

1. É atribuído a *msg.mac1* a computação do MAC entre o *hash* da concatenação entre o Label-Mac1 e a chave estática pública do titular da mensagem; e o os bits de msg_α.
2. Para o caso de o último *cookie* recebido ser uma *string* de bits vazia (ϵ) ou o *cookie* atual ter o seu tamanho maior que 120 é atribuída uma *string* de 0^{16} à *msg.mac2.* Para os demais casos, é atribuído ao parâmetro *msg.mac2* o cálculo do MAC entre o último *cookie* recebido e os bits de msg_β.

Em suma, o fundamento é que o iniciador, ao reenviar sua mensagem, envie um MAC da mensagem utilizando um *cookie* como uma chave MAC. No caso de sobrecarga de processamento do respondedor, este pode utilizar-se destes *cookies* para responder ou não à solicitação. A utilização de *cookies* neste mecanismo é feita através da computação de *hashes* e MACs para evitar a interceptação de um *cookie* por um terceiro não autorizado. Note-se que a utilização de *cookies* sob mecanismos suficientemente seguros, para além dos cenários de ataques de amplificação, podem também ser utilizados para contornar ataques de força bruta.

3.2 DERIVAÇÃO DE CHAVES DE DADOS DE TRANSPORTE

Com a primeira etapa do *handshake* concluída, o iniciador e o respondedor são capazes de realizar a computação de chaves para as mensagens de transporte de dados (Tab. 9).

Tabela 9: Computação das chaves de dados de transporte. Adaptado de [4].

$$(T_i^{send} = T_r^{send},\ T_i^{recv} = T_r^{send}) := \text{KDF}_2\,(C_i = C_{r,} \in)$$
$$N_i^{send} = N_r^{recv} = N_i^{recv} = N_r^{send} := 0$$
$$E_i^{priv} = E_i^{pub} = E_r^{priv} = E_r^{pub} = C_i = C_r := \in$$

Conforme mostrado na Tabela 9, os valores das chaves simétricas de dados de transporte de envio e recepção são derivados a partir de uma KDF utilizando os valores de C_* e ϵ. Os contadores de *nonce* são equiparados a 0 e às chaves efêmeras E_*^{priv} e E_*^{publ}, bem como às chaves de encadeamento C_* é atribuída a *string* vazia ϵ, de forma a apagar da memória o maior número possível dos estágios prévios do *handshake.*

3.3 MENSAGENS DE DADOS DE TRANSPORTE

A troca de mensagens subsequente é a de mensagens de dados de transporte entre iniciador e respondedor, e é utilizada para o envio de pacotes encapsulados criptografados. O conteúdo dos

pacotes, em mensagem base, é representado por P, e o seu comprimento é representado por $||P||$. A mensagem de dados de transporte tem a estrutura mostrada na Figura 15 cujos parâmetros são calculados de acordo com a Tabela 10.

type := 0x4 (1 byte)	reserved := 0^3 (3 bytes)
receiver := $I_{m'}$ (4 bytes)	
counter (8 bytes)	
packet ($\widehat{\|P\|}$ bytes)	

Figura 16: Estrutura das Mensagens de Dados de Transporte [4].

Tabela 9: Computação das Mensagens de Dados de Transporte. Adaptado de [4].

$$P := P \,||\, 0^{16[||P||/16]-||P||}$$
$$msg.counter := N_m{}^{send}$$
$$msg.packet := \text{AEAD}(T_m{}^{send}, N_m{}^{send}, P, \in)$$
$$N_m{}^{send} := N_m{}^{send} + 1$$

$T_m{}^{recv}$ é utilizada pelo recebedor da mensagem para lê-la. Note-se que o pacote P é preenchido com zeros antes da cifragem de modo a dificultar a análise criptográfica, sem prejuízo por incremento do tamanho do pacote. Os demais procedimentos podem ser compreendidos analogamente pela descrição feita nos passos para as trocas de mensagem entre iniciador e respondedor. O campo *msg.counter* funciona como um *nonce* para a AEAD e é monitorado pelo recebedor por meio de $N_m{}^{send}$, funcionando também para evitar ataques de repetição.

3.4 MENSAGEM DE RESPOSTA DE *COOKIE*

Como já mencionado na Subseção 3.8.5, o *WireGuard* tem um mecanismo para prevenção de ataques DoS, ou seja, para prevenir o sistema de eventual sobrecarga. No eventual caso que o sistema esteja sob uma carga de processamento grande, ao receber uma mensagem com uma *msg.mac1* válida, porém uma *msg.mac2* inválida ou que já tenha expirado; o *peer* retorna uma resposta com uma mensagem de *cookie*. O índice $I_{m'}$ é derivado do campo *msg.sender* da mensagem que desencadeou a resposta com *cookie*, e tem a estrutura conforme Figura 16 e a computação dos parâmetros conforme Tabela 11:

type := 0x3 (1 byte)	reserved := 0^3 (3 bytes)
receiver := $I_{m'}$ (4 bytes)	
nonce := ρ^{24} (24 bytes)	
cookie ($\widehat{16}$ bytes)	

Figura 17: Estrutura da mensagem de resposta de *cookies* [4].

Tabela 10: Computação dos Parâmetros da Mensagem de Resposta de *cookies*.

$$\tau := \text{MAC}(R_m, A_{m'})$$

$$msg.cookie := \text{XAEAD}\ (\text{HASH}(\text{LABEL-COOKIE} \parallel S_m^{pub}), msg.nonce, \tau, M)$$

Na Tabela 11, o parâmetro R_m é uma variável secreta que tem seu valor alterado aleatoriamente a cada segundo, $A_{m'}$ é a concatenação do endereço de IP e da porta UDP externos *e M* é o valor da *msg.mac1* a qual será respondida pelo *cookie.*

4 AVALIAÇÃO DOS MÉTODOS E PROPOSTAS DE SEGURANÇA PÓS-QUÂNTICA

4.1 IMPLEMENTAÇÃO DO *WIREGUARD* NO LINUX

Neste trabalho, com o intuito de elucidar algumas características práticas do *Wireguard*, implementou-se um túnel criptográfico utilizando duas máquinas virtuais em duas máquinas físicas distintas. A implementação do *WireGuard* foi considerada prioritariamente no ambiente Linux, devido à sua arquitetura direcionada de forma otimizada nativamente neste ambiente. Todo o projeto do *WireGuard* foi desenvolvido para ser uma alternativa mais segura ao já amplamente utilizado IPsec, e atualmente encontra-se presente no próprio *kernel* do Linux [4].

O sistema operacional utilizado nas máquinas virtuais foi o Ubuntu 20.04.2.0 LTS. Pormenores acerca da instalação e configuração das interfaces de rede não serão abordados pois fogem do escopo deste trabalho. Serão elencadas nesta seção apenas algumas etapas da implementação que podem ajudar no entendimento do cenário de segurança a ser abordado.

A configuração do *WireGuard* deve ser feita no maior nível de privilégio, i.e., *root*. Garantindo-se isso, se torna possível e necessária a geração, pela máquina destinada ao servidor, de um par de chaves pública-privada no repositório do *WireGuard*. É possível observar um exemplo destas chaves na Figura 17.

```
root@ubuntu:/etc/wireguard# cat privatekey publickey
cD4XiLdQmihdkDBKa2auK6z2JiVagNz0qOhbY/u5LV4=
FAHlzVRZNkDSHMsj57wYJhXPW2OS1FSnXfY5yAGNNwk=
```

Figura 18: Par de chaves estáticas públicas e privadas geradas pelo *WireGuard*.

Essas chaves são utilizadas para a configuração do servidor, e é a partir delas que são derivados diversos atributos utilizados no *handshake*, conforme exposto na Seção 4.1.1. Para a configuração, também é necessária a chave pública do cliente VPN, e a especificação dos IPs e portas envolvidos, conforme mostrado na Figura 18.

```
[Interface]

## Endereco IP do meu Servidor VPN

Address = [redacted]

## Porta do meu Servidor VPN

ListenPort = 41194

## Chave privada do servidor VPN

PrivateKey = cD4XiLdQmihdkDBKa2auK6z2JiVagNz0qOhbY/u5LV4=

[Peer]

#Public Key do Cliente VPN

PublicKey = SF1o+WPB4g5mRCCXwPR506S77SQXa+f5NVCeteZV3XE=

AllowedIPs = [redacted]
```

Figura 19: Exemplo de arquivo de configuração do *WireGuard*.

Outro passo necessário na implementação do *WireGuard* é a configuração do *firewall*. Para esta implementação foi utilizado *UncomplicatedFirewall* [28], que é uma interface de configuração

de *firewall* do *iptables* nativo do *kernel* do Linux. Esta solução permite a configuração descomplicada, porém precisa das regras de *firewall* a serem implementadas integradamente com o *WireGuard*. Tais regras, a título de exemplo (Fig. 17), devem habilitar a porta UDP 41194. Para o lado do cliente, a configuração se dá de forma similar.

De forma sucinta, para cada ponto de acesso, o *WireGuard* associa os endereços de IP às suas respectivas chaves públicas. No envio de pacotes para um dos pares conectados à rede, o *WireGuard* identifica o IP de referência, criptografa-o com a respectiva chave pública, identifica a porta UDP e envia os dados criptografados através desta. No recebimento das mensagens, primeiro acontece a identificação da porta por onde o pacote foi recebido, para então ocorrer a identificação do endereço IP referente à decifragem com a devida chave pública e assim obter o IP do par em questão. Com este IP, o *WireGuard* consegue observar se o par está habilitado para recebimento das mensagens.

Um diagrama que representa de forma genérica a implementação utilizada neste trabalho pode ser observada na Figura 19.

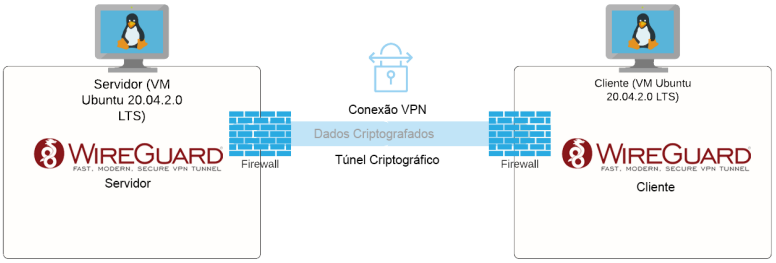

Figura 20: Diagrama Genérico de Implementação do *WireGuard* nas máquinas virtuais.

4.2 NÍVEIS DE SEGURANÇA INTRÍNSECOS À IMPLEMENTAÇÃO

Os processos de segurança intrínsecos ao *WireGuard* podem ser delimitados por meio de uma análise de Anel de Proteção. Considerando-se como Núcleo o acesso ao *kernel* da aplicação do *WireGuard* que detem privilégios de *root*, pode-se decrementar o nível de segurança a partir dela para mecanismos de menor privilégio, conforme ilustrado na Figura 20.

O Anel de Nível 3, mais externo, pode ser considerado o nível no qual os dados criptografados estão na rede, sejam as chaves públicas compartilhadas, ou os dados de transmissão, tendo como recurso de segurança “apenas” os mecanismos utilizados na sua criptografia. Aqui estão sendo considerados como nível de segurança mais externo porque seria o nível menor de esforço para ter acesso aos dados criptografados.

No nível 2, pode-se elencar o *Firewall*, que servirá como um filtro de restrição para a interação com o *WireGuard* por meio da rede, de forma que somente determinados pares, pelas portas e critérios especificados; sejam capazes de trocar mensagens com a aplicação, que só então será capaz de manusear essas mensagens.

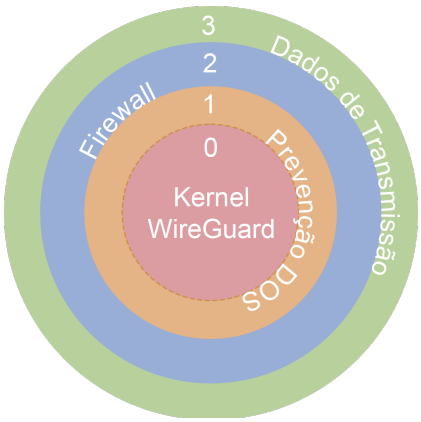

Figura 21: Anel de Segurança da Implementação do *WireGuard*.

O Nível 1 pode ser considerado como a camada onde o *WireGuard* implementa o seu mecanismo de Prevenção DoS por meio de mensagens de resposta com *cookies,* conforme explorado na Seção 4.1.1.4. Além da prevenção contra ataques de exaustão, essa camada pode ser utilizada como um recurso adicional para reconhecimento de ataques de força bruta.

No nível 0, correspondente ao Núcleo do *WireGuard*, no *kernel* do Linux, é onde ocorre a geração das chaves estáticas dos pares, das quais são derivados os demais atributos por meio de algoritmos criptográficos clássicos previamente conhecidos. Portanto, caso um atacante tenha acesso a este nível de privilégio, não restará qualquer alternativa de segurança do canal criptográfico. É neste nível de segurança que ocorrem os processos de rechaveamento, ou seja, onde novas chaves de Sessão e de Transporte são geradas, garantindo a rotação das chaves e um certo nível de *PFS.*

4.3 PRINCIPAIS ALGORITMOS CRIPTOGRÁFICOS UTILIZADOS

Os critérios estipulados pelo NIST para resistência criptográfica dos algoritmos utilizados frente a ataques possibilitam a comparação de algoritmos a serem utilizados nos processos de geração de chaves e de autenticação de forma isolada, sem prejuízo a outros mecanismos clássicos que podem elevar a segurança pós-quântica quando utilizados de forma integrada, almejando a contemplação dos demais princípios recomendados pelo NIST. Incrementos nos tamanhos das chaves utilizadas nos algoritmos, embora possam ser considerados, devem ser evitados pois tendem a diminuir a performance com a sua utilização. Com base nisso, é possível restringir a avaliação aos mecanismos clássicos, levando-se em conta tais critérios.

É possível notar, de forma suficientemente clara na Seção 3, que os processos de geração de atributos e chaves é consideravelmente complexo, nos quais a computação destes é feita por diversos algoritmos concatenados e cascateados.

A criptografia aplicada na geração das chaves e dos atributos é baseada na ECDH Curve25519 de 256 bits com segurança clássica de 128 bits e numa função de *hash* BLAKE2s no processo de derivação de chaves. No que diz respeito às cifras e chaves pré-compartilhadas de 256 bits, elas têm o seu comprimento definido por normalização de um tamanho de chave adequado "considerando as margens de segurança criptoanalítica, compensação de memória e tempo, ataques multi-chaves, recodificação e ataques quânticos" [27]. Um resumo das conclusões obtidas na

avaliação derivada da pesquisa pode ser observado na Tabela 11, e os mecanismos serão descritos com maiores detalhes nas subseções a seguir, correspondentes a cada um dos atributos avaliados de forma isolada.

Tabela 11: Resumo da avaliação dos métodos.

MECANISMO AVALIADO	RESULTADO DA AVALIAÇÃO
BLAKE2S	Nenhum nível de resistência do NIST
CURVE25519	Custo de ataque muito alto para chance de sucesso muito baixa (Expansão da abordagem clássica para a abordagem quântica)
CHACHA20	Resistência quântica Nível V (NIST)
POLY1305	Resistência quântica Nível I (NIST)
SIGILO DE ENCAMINHAMENTO PERFEITO	Insuficiente
RESISTÊNCIA A ATAQUES DE CANAL LATERAL	Presente em ao menos um dos processos (Utilização do algoritmo ChaCha20)
RESISTÊNCIA A ATAQUES DE MÚLTIPLAS CHAVES	Presente em ao menos um dos processos (Derivação de Chaves por Hash)
RESISTÊNCIA A MÁS-PRÁTICAS	Implementação simples e objetiva, alta auditabilidade.

4.3.1 BLAKE2S

As utilizações da função de *hash* BLAKE2s no *WireGuard* são feitas no modo de 256 bits na saída, conforme especificações em [4]. Esta função também é utilizada para derivação de chaves baseada em *hash* (HKDF), como um artifício de simulação de aleatoriedade, i.e., como uma função pseudorrandômica, que é o tipo de mecanismo mais próximo para simular aleatoriedade no escopo clássico.

Esta implementação na arquitetura de 32 bits e com saída igualmente de 32 bits oferece uma segurança de colisão de 2^{128} segundo [11]. Desta forma, não chega a atingir nenhum dos níveis de resistência quântica estipulados pelo NIST, onde o critério mínimo para resistência de uma função de *hash* (Nível II) é de 2^{256}, oferecido por exemplo pela cifra SHA256. A família BLAKE2 conta com o modo BLAKE2b512 que oferece uma segurança contra colisão de 2^{256}, o que seria uma solução viável, em uma análise desconsiderando implicações na performance do *WireGuard* com o uso deste algoritmo. Uma resistência a ataques de colisão de 2^{128} deixaria os mecanismos que envolvem a cifra BLAKE2s, em análise isolada, suscetível a ataques de colisão.

Entretanto, no que diz respeito à derivação de chaves por meio da BLAKE2s, é este mecanismo que entrega algum nível de segurança contra-ataques de múltiplas-chaves, de forma a mascarar qualquer relação direta entre as chaves utilizadas nas demais etapas.

4.3.2 CURVE25519

Em uma abordagem clássica, a ECDH Curve25519 mostra-se resistente contra ataques clássicos, conforme elencado em [14], por exemplo, a ataques pelos métodos *kangaroo* ou *rho* de Pollard, que computam algoritmos discretos contidos em um intervalo [*a,b*] em grupos de ciclos arbitrários [29]. Este método, por ser um método paralelizável, se associado à computação quântica, poderia em tese oferecer riscos ao comprometimento da cifra. Por outro lado, como Berstein [7] pontua, em uma Curve25519(*n,* 9), o número de operações necessárias seria da ordem de $\sqrt{n}$, com chances de sucesso de no máximo $\frac{a^2}{2^{251}}$, com *a* adições, i.e., operações, para esta utilização específica.

Estima-se que um atacante, utilizando os métodos *kangaroo* ou *rho* de Pollard em máquinas clássicas paralelizadas teria uma chance de sucesso na computação do logaritmo discreto de 251-bits de uma ECDH Curve25519 de aproximadamente 2^{-90}, que caracteriza uma chance muito baixa para um custo muito elevado [29]. Em seu artigo original [14], Bernstein exemplifica outros métodos que podem ser utilizados para a computação do logaritmo discreto da Curve25519, utilizando ainda o método *rho* de Pollard mencionado com mecanismos paralelizados, que fogem do escopo deste trabalho, mas consolida que o algoritmo permanece confortavelmente seguro, com a utilização de chaves de tamanho superior a 251-bits, em termos de custo e tempo para quebrá-lo.

4.3.3 CHACHA20 E POLY1305

O *WireGuard* utiliza o algoritmo ChaCha20_Poly1305, i.e., uma combinação dos algoritmos ChaCha20 e Poly1305 em um algoritmo de cifragem autenticada com dados associados. As entradas para este algoritmo são uma chave de 256 bits, um *nonce* de 96 bits (que é diferente para cada utilização de uma mesma chave), uma parte de tamanho arbitrário de texto-simples e uma AAD (*additional authenticated data*) de tamanho arbitrário. Primeiramente, é gerada uma chave de uso único nos padrões do algoritmo Poly1305 utilizando-se da chave de 256 bits e do *nonce*; e então é feita uma cifragem do texto-simples sob o algoritmo ChaCha20 por meio dos mesmos chave e *nonce* e de um contador. Em seguida, é chamada a função Poly1305 com a sua chave já calculada e uma mensagem é então construída de acordo com o estipulado para a combinação. Essa aplicação da AEAD tem como saída uma mensagem cifrada de mesmo comprimento que o texto-simples inicial e uma *tag* de 128 bits como saída do algoritmo Poly1305 [13].

O ChaCha20 é uma cifra de bloco que tem uma segurança em nível de uma cifra de bloco de 256-bits, atendendo ao critério de resistência quântica de Nível V pelos critérios estabelecidos pelo NIST [Tab. 2], o que em tese fornece uma alta segurança contra ataques de pesquisa de chave.

Para uma discussão relevante da quebra do algoritmo do ChaCha20 por um ataque de força bruta por pesquisa de chaves, utilizando-se por exemplo um computador quântico, é relevante elencar que, ao menos numa abordagem clássica, é impossível alcançar concomitantemente a atuação de máquinas paralelizadas com preço razoável, alto desempenho e chance de sucesso em distinguir $n \mapsto (\text{Salsa20}_k(n))$ maior que 2^{-64} (considerada por Bernstein uma máquina clássica suficientemente grande de aproximadamente 2^{64} unidades de pesquisa de chave independentes) [30].

Pela abordagem de resistência a ataques de canais laterais, Bernstein elenca que "Ataques de *timing* contra o ChaCha20 são, portanto, tão difíceis quanto à criptoanálise pura dos resultados do ChaCha20. As operações em ChaCha20 também estão entre as mais fáceis de proteger contra-ataques de energia e outros ataques de canal lateral" [30], enquadrando a cifra no princípio apontado pelo NIST de resistência a ataques de canais laterais.

O algoritmo de MAC Poly1305 utilizado na autenticação de dados assegura que a única maneira de se quebrar a criptografia é por meio da quebra do algoritmo AES envolvido na cifra, que é uma AES de 128 bits, conferindo individualmente à cifra uma resistência Nível I (Tab. 2). Em adendo, segundo a RFC do algoritmo ChaCha20_Poly1305, o Poly1305 rejeita mensagens forjadas com uma probabilidade de $1 - \left(\frac{n}{2^{102}}\right)$ para uma mensagem de tamanho $16n$ bytes, mesmo com o envio de 2^{64} mensagens genuínas, tornando o algoritmo altamente resistente à forja de mensagens em ataques de mensagem-escolhida (*strong unforgeability against chosen-message attacks* – SUF-CMA) [13].

4.4 SIGILO DE ENCAMINHAMENTO PERFEITO NO WIREGUARD

O *PFS* no *WireGuard*, a princípio, é almejado por meio do sistema de rotações de chave, que gera novas chaves de sessão a cada 120 segundos, ou seja, a cada 120 segundos é criada uma chave simétrica efêmera de sessão. Note-se, entretanto, que o *WireGuard* guarda os estados relativos à sessão atual, à sessão passada e à próxima sessão pretendida, com o intuito de prevenir perdas de sessão. No caso do estabelecimento de uma sessão, os estados são substituídos pelos novos em rotação. Embora assim elencado em [4], este artifício por si só não garante o *PFS*.

A fim de atingir o *PFS* em um nível quântico-resistente, pode-se propor um mecanismo de encapsulamento de chaves (*Key Encapsulation Mechanism – KEM)*. Para proteger as chaves públicas utilizadas em um nível quântico resistente, pode-se utilizar uma rodada adicional de cifragem com uma cifra que atenda aos padrões estipulados em (Tab. 2), como o uso de um algoritmos de curva elíptica, como o próprio Curve25519. Um *KEM* adequado pode, por exemplo, ser utilizado sobre a chave efêmera pública gerada pelo iniciador, derivando uma chave simétrica a ser trocada com o respondedor. Portanto, a derivação de novas chaves utilizando o encapsulamento da chave efêmera atrelada ao mecanismo de rotações de chave intrínseco ao *WireGuard* [4], que cria novas chaves de sessão a cada, aproximadamente, 120 segundos [4], se apresenta como uma boa alternativa para atingir o *PFS*. Entretanto, um ponto relevante a ser considerado, é a utilização da função de *hash* a um nível quântico-resistente sob os parâmetros do NIST, i.e., a derivação de chaves deve-se atentar para a utilização de um *hash* com uma segurança de colisão de no mínimo 2^{256}.

Em consideração paralela, o próprio artigo do *WireGuard* [4] traz como uma sugestão de implementação para escalar a segurança uma alteração no passo 9 da Seção 4.1.1.1 de geração da mensagem do iniciador, que seria a computação da *msg.static* como uma $AEAD(\kappa, 0, Hash(S_i^{pub}), H_i)$, ou seja, incrementar com uma operação de *hash*, por um outro prisma ao elencado anteriormente, na chave estática pública do iniciador S_i^{pub}. Essa alteração teria o efeito de garantir que a chave em curva elíptica não seja transmitida de forma direta, implicando em uma proximidade maior a uma desejável segurança pós-quântica por meio de um segredo *non-*

forward, contanto que seja garantido o sigilo das chaves [4]. Para este fim também, deve-se avaliar a utilização da função de *hash* com nível de segurança elencado nas definições do NIST.

5 CONCLUSÕES E TRABALHOS FUTUROS

Neste trabalho foram apresentados os conceitos básicos que envolvem a criptografia clássica moderna, e como estes conceitos se aplicam na implementação da solução de VPN *WireGuard*. A partir da exploração destes conceitos, foi possível dissecar as operações criptográficas inerentes ao funcionamento do *WireGuard* na geração de chaves de Sessão e de Transporte de dados, a fim de construir um cenário para uma análise de segurança quântico-resistente da solução. Em um segundo momento, foram avaliados os algoritmos criptográficos envolvidos nas operações criptográficas e suas avaliações de segurança quântico-resistente com base nos critérios estipulados pelo NIST [23].

Os algoritmos, quando utilizados de forma integrada nas diversas transformações que ocorrem no *WireGuard*, são operados de forma inteligente com o seu cascateamento de operações, para garantir que as chaves envolvidas não sejam comprometidas. Entretanto, o uso combinado destas técnicas de segurança, quando consideradas de forma unificada, se faz difícil de precisar quanto à sua segurança em termos quantitativos. Para reduzir este problema a uma análise em termos mensuráveis, foram feitas as considerações dos algoritmos e dos atributos quântico-resistentes de forma isolada.

Entre os algoritmos envolvidos nos processos do *WireGuard*, foram considerados como os mais relevantes a função de *hash* BLAKE2s, o algoritmo de ECDH Curve25519, a cifra de bloco *ChaCha20* e sua variação combinada *ChaCha20_Poly1305* para autenticação de mensagens. Três destes quatro algoritmos, sob análises considerando os critérios estipulados pelo NIST, atendem aos critérios de níveis de segurança para algoritmos pós-quânticos numa abordagem clássica; sendo a função BLAKE2s a única a não se encaixar de forma isolada em nenhum dos níveis. É possível considerar, entretanto, a sua substituição por uma função da mesma família BLAKE2 [11] de mecanismos similares e segurança mais elevada.

Para além dos critérios estipulados pelo NIST, as cifras utilizadas pelo *WireGuard* são primordialmente baseadas em *hash* e mecanismos que envolvem criptografia de chave secreta, que são considerados mecanismos de resistência quântica; dado que ainda não existem aplicações do algoritmo de Shor com perspectiva de quebra de cifras equivalentes à AES e a aplicação do algoritmo de Grover para a sua quebra seria muito custosa em termos de tempo. Portanto, pequenas compensações para a segurança poderiam ser feitas por meio do incremento do tamanho de chaves [31].

No que diz respeito aos princípios elencados pelo NIST, os mecanismos envolvidos no *WireGuard* buscam atender à resistência a ataques de canais-laterais, ao *PFS* e às boas práticas de implementação, sendo os dois primeiros alcançados em pelos menos uma das etapas de geração de chaves e o último almejado na implementação da solução como um todo.

Posto isso, é possível afirmar que o *WireGuard* é uma solução de VPN que atende suficientemente aos critérios de segurança no que diz respeito à resistência criptográfica frente a métodos quânticos, sob a ótica dos aspectos considerados pelo NIST e em aspectos gerais. Em complemento à análise dirigida à solução de VPN em si, é importante ressaltar que ainda não se encontra consolidado um cenário de ameaça quântica à criptografia clássica. A relevância do tipo de análise explorada neste trabalho, entretanto, se mostra como uma preparação para um possível salto para um cenário pós-quântico da computação, que pode acontecer a qualquer momento, ou

até mesmo nunca; possibilitando assim a estruturação de novos mecanismos prontos para serem aprimorados ou implementados num possível *quantum-break.*

Como trabalhos futuros, pretende-se realizar o aprofundamento nos princípios da computação quântica, no que diz respeito à sua complexidade computacional, os seus modelos de circuitos e o detalhamento do funcionamento dos computadores quânticos. Pretende-se também continuar com os estudos na área de criptografia e suas implicações na área de telecomunicações, bem como desenvolver estudos na área de métodos criptográficos pós-quânticos.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] M. Bellare e P. Rogaway, Introduction to Modern Cryptography, La Jolla, CA, USA: Department of Computer Science and Engineering, University of California at San Diego, 2005.

[2] B. Lord, Artist, *German Enigma Machine.* [Art]. Wikipedia Commons, 2005.

[3] R. A. Grimes, Cryptography Apocalypse: Preparing for the Day When Quantum Computing Breaks Today's Crypto, Hoboken, NJ, USA: John Wiley & Sons, Inc., 2020.

[4] J. A. Donenfeld, "WireGuard: Next Generation Kernel Network Tunnel," *Proceedings of the Network and Distributed System Symposium,* 2020.

[5] National Institute of Standards and Technology, "Post-Quantum Cryptography," Information Technology Laboratory - Computer Security Resource Center, [Online]. Available: https://csrc.nist.gov/projects/post-quantum-cryptography. [Acesso em 08 Setembro 2021].

[6] P. W. Shor, "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer," *Proceedings of the 35th Annual Symposium on Foundations of Computer Science,* Novembro 1997.

[7] L. K. Grover, "A fast quantum mechanical algorithm for database search," *Proceedings of the twenty-eighth annual ACM symposium on Theory of Computing,* vol. 96, 1996.

[8] A. J. Menezes, P. C. Oorschot e S. A. Vanstone, Handbook of Applied Cryptography, Boca Raton, FL, USA: CRC Press, Inc., 1996.

[9] D. Gordon, "ISA 562: Information Security, Theory and Practice - Lecture 1," 2017. [Online]. Available: https://cs.gmu.edu/~gordon/teaching/isa562/notes/lecture_1.pdf. [Acesso em 18 Outubro 2021].

[10] J. Camenisch, S. Fischer-Hübner e K. Rannenberg, Privacy and Identity Management for Life, Springer-Verlag Berlin Heidelberg, 2011.

[11] J.-P. Aumasson, "The BLAKE2 Cryptographic Hash and Message Authentication Code (MAC)," *RFC 7693,* Novembro 2015.

[12] G. Bertoni, J. Daemen, M. Peeters e G. V. Assche, "The Keccak SHA-3 submission," 14 Janeiro 2011.

[13] A. Langley, "RFC 7539: ChaCha20 and Poly1305 for IETF Protocols," *Internet Research Task Force (IRTF),* 2015.

[14] D. J. Bernstein, "Curve25519: new Diffie-Hellman speed records," em *Public Key Cryptography - PKC 2006. PKC 2006. Lecture Notes in Computer Science, vol 3958.* , Springer, Berlin, Heidelberg. , 2006.

[15] D. J. Bernstein, "The Poly1305-AES message-authentication code," em *Fast Software Encryption. FSE 2005. Lecture Notes in Computer Science, vol 3557.*, Chicago, IL, Springer, Berlin, Heidelberg. , 2005.

[16] M. Bellare, P. Rogaway e D. Wagner, "A Conventional Authenticated-Encryption Mode," *NIST,* 2013.

[17] L. C. Santos e J. López, “Pipeline Oriented Implementation of NORX for ARM Processors - Fast software implementation of AEAD,” em *XVII Simpósio Brasileiro em Segurança da Informação e de Sistemas* , Brasília, 2017.

[18] P. W. Shor, “Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer,” *SIAM J. COMPUT. Society for Industrial and Applied Mathematics,* October 1997.

[19] M. Mosca, “Quantum Algorithms,” *Institute for Quantum Computing and Dept. of Combinatorics & Optimization. University of Waterloo and St. Jerome’s University, and Perimeter Institute for Theoretical Physics,* 4 Agosto 2008.

[20] M. P. Frank, “Foundations of Generalized Reversible Computing,” *9th Conference on Reversible Computation,* p. 6, 25 Maio 2017.

[21] A. M. Turing, “On Computable Numbers, with an Application to the Entscheidungsproblem,” em *Proceedings of the London Mathematical Society* , London, 1936.

[22] T. T. John Wright, “Lecture 4: Grover’s Algorithm,” em *Quantum Computation - (CMU 15-859BB, Fall 2015)*, 2015.

[23] National Institute of Standards and Technology, “Post-Quantum Cryptography,” 03 Janeiro 2017. [Online]. Available: https://csrc.nist.gov/projects/post-quantum-cryptography. [Acesso em 18 Outubro 2021].

[24] F.-X. Standaert, “Introduction to Side-Channel Attacks,” em *Secure Integrated Circuits and Systems. Integrated Circuits and Systems.*, Boston, MA., Springer, 2010.

[25] L. K. Chin, Rede Nacional de Ensino e Pesquisa, 13 Novembro 1998. [Online]. Available: https://memoria.rnp.br/newsgen/9811/vpn.html. [Acesso em 21 Abril 2021].

[26] N. Kobeissi, G. Nicolas e K. Bhargavan, “Noise Explorer: Fully Automated Modeling and Verification for Arbitrary Noise Protocols,” *4th IEEE European Symposium on Security and Privacy,* 15 Agosto 2018.

[27] T. Perrin, “The Noise Protocol Framework,” nº Revision 34, 2018.

[28] cipherboy, “UncomplicatedFirewall,” Ubuntu.wiki, 27 04 2021. [Online]. Available: https://wiki.ubuntu.com/UncomplicatedFirewall. [Acesso em 06 10 2021].

[29] E. Teske, “Computing discrete logarithms with the parallelized kangaroo method,” *Discrete Applied Mathematics ,* vol. 130, p. 61 – 82, 2003.

[30] D. J. Bernstein, “Salsa20 security,” Department of Mathematics, Statistics, and Computer Science. The University of Illinois at Chicago, Chicago, IL, 2008.

[31] D. J. Bernstein, “Introduction to post-quantum cryptography,” em *Post-Quantum Cryptography.*, Springer, Berlin, Heidelberg, 2009, p. 2.

[32] A. L. Y. Nir, “RFC 7539 - ChaCha20 and Poly1305 for IETF Protocols,” em *Internet Research Task Force (IRTF), RFC: 7539*.