18 a 21 de setembro de 2023.
Juiz de Fora - MG

Minicursos
SBSeg 2023
XXIII SIMPÓSIO BRASILEIRO EM SEGURANÇA DA INFORMAÇÃO E DE SISTEMAS COMPUTACIONAIS

Por Alex Borges Vieira, Edelberto Franco Silva, Dianne Scherly Varela de Medeiros, Roberto Samarone Dos Santos Araujo.

XXIII SBSeg 2023
Juiz de Fora - MG

Minicursos do XXIII Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais

Editora
Sociedade Brasileira de Computação (SBC)

Organização
Alex Borges Vieira, UFJF
Edelberto Franco Silva, UFJF
Dianne Scherly Varela de Medeiros, UFF
Roberto Samarone Dos Santos Araujo, UFPA

Realização
Sociedade Brasileira de Computação (SBC)
Universidade Federal de Juiz de Fora (UFJF)

Promoção
Sociedade Brasileira de Computação (SBC)

18 a 21 de setembro de 2023.
Juiz de Fora - Minas Gerais

Capa (Edição): Alex Borges Vieira (UFJF)
Produção Editorial: Edelberto Franco Silva (UFJF), Alex Borges Vieira (UFJF), Dianne Scherly Varela de Medeiros (UFF), Roberto Samarone dos Santos Araújo (UFPA)

Cópias Adicionais:
Sociedade Brasileira de Computação (SBC)
Av. Bento Gonçalves, 9500- Setor 4 - Prédio 43.412 - Sala 219
Bairro Agronomia - CEP 91.509-900 - Porto Alegre - RS
Fone: (51) 3308-6835
E-mail: sbc@sbc.org.br

Dados Internacionais de Catalogação na Publicação (CIP)

S612 Simpósio Brasileiro em Segurança da Informação e Sistemas Ocupacionais (23. : 18 – 21 setembro 2023 : Juiz de Fora)
Minicursos do SBSeg 2023 [recurso eletrônico] / organização: Alex Borges Vieira [et al.]. – Dados eletrônicos. – Porto Alegre: Sociedade Brasileira de Computação, 2023.
314 p. : il. : PDF ; 9.20 MB

Modo de acesso: World Wide Web.
Inclui bibliografia
ISBN 978-85-7669-567-7 (e-book)

1. Computação – Brasil – Evento. 2. Ciência e Tecnologia. 3. Cibersegurança. 4. Redes de computadores. I. Vieira, Alex Borges. II. Silva, Edelberto Franco. III. Medeiros, Dianne Scherly Varela de. IV. Araujo, Roberto Samarone dos Santos. V. Sociedade Brasileira de Computação. VI. Título.

CDU 004(063)

Ficha catalográfica elaborada por Annie Casali – CRB-10/2339
Biblioteca Digital da SBC – SBC OpenLib

Índices para catálogo sistemático:

1. Ciência e tecnologia dos computadores : Informática – Publicação de conferências, congressos e simpósios etc. ... 004(063)

Sociedade Brasileira de Computação

Presidência
Presidente: Thais Vasconcelos Batista (UFRN)
Vice-presidente: Cristiano Maciel (UFMT)

Diretorias
Administrativa: Renata de Matos Galante (UFRGS)
Finanças: Lisandro Zambenedetti Granville (UFRGS)
Eventos e Comissões Especiais: Denis Lima do Rosário (UFPA)
Educação: Claudia Lage Rebello da Motta (UFRJ)
Publicações: José Viterbo Filho (UFF)
Planejamento e Programas Especiais: André Luís de Medeiros Santos (UFPE)
Secretarias Regionais: Eunice Pereira dos Santos Nunes (UFMT)
Divulgação e Marketing: Alirio Santos de Sá (UFBA)
Relações Profissionais: Tanara Lauschner (UFAM)
Competições Científicas: Carlos Eduardo Ferreira (USP)
Cooperação com Sociedades Científicas: Ronaldo Alves Ferreira (UFMS)
Articulação com Empresas: Michelle Silva Wangham (UNIVALI)

Diretoria Extraordinária
Ensino de Computação na Educação Básica: Leila Ribeiro (UFRGS)

Conselho

Mandato 2021-2025	**Mandato 2023-2027**
Tayana Uchoa Conte (UFAM)	Altigran Soares da Silva (UFAM)
Isabela Gasparini (UDESC)	José Carlos Maldonado (USP-São Carlos)
Avelino Francisco Zorzo (PUCRS)	Jussara Marques de Almeida (UFMG)
Alba Cristina M. A. de Melo (UnB)	Débora Christina Muchaluat Saade (UFF)
Alfredo Goldman (IME-USP)	Carla Maria dal Sasso Freitas (UFRGS)

Contato
Av. Bento Gonçalves, 9500
Setor 4 - Prédio 43.412 - Sala 219
Bairro Agronomia
91.509-900 – Porto Alegre RS
CNPJ: 29.532.264/0001-78
http://www.sbc.org.br

Comissão Especial em Segurança da Informação e de Sistemas Computacionais – CESeg

Coordenação
Igor Monteiro Moraes (UFF), Coordenador
Marcos Antonio Simplicio Junior (USP), Vice-Coordenador

Comitê Gestor
Aldri Luiz dos Santos (UFMG)
Carlos Raniery Paula dos Santos (UFSM)
Daniel Macêdo Batista (USP)
Edelberto Franco Silva (UFJF)
Emerson Ribeiro de Mello (IFSC)
Lourenço Alves Pereira Júnior (ITA)
Marco Aurélio Amaral Henriques (UNICAMP)
Michele Nogueira (UFMG)
Roberto Samarone dos Santos Araujo (UFPA)

Mensagem dos Coordenadores Gerais

É com imensa alegria e satisfação que recebemos todos no XXIII Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais (SBSeg 2023) que será realizado presencialmente, entre os dias 18 a 21 de Setembro de 2023. Esta edição do SBSeg é especial, pois após anos afastados, estamos nos encontrando presencialmente e contribuindo com a consolidação do SBSeg como o mais importante evento nacional na área de segurança.

Gostaríamos de agradecer a todos os envolvidos, pelo apoio, dedicação e comprometimento durante a organização do evento. Primeiro, aos patrocinadores que confiaram no SBSeg 2023, pelo incentivo à pesquisa, desenvolvimento e inovação, tão importantes nos dias de hoje: o Comitê Gestor da Internet no Brasil (CGI.br) e o Núcleo de Informação e Coordenação do Ponto BR (NIC.br), a Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), o Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), Fundação de Amparo à Pesquisa do Estado de Minas Gerais (FAPEMIG), à Rede Nacional de Ensino e Pesquisa (RNP), a Tempest, a CESAR School, a Escola Superior de Redes (ESR/RNP), a AltasNet, e a Faculdade IBPTech.

Nosso especial agradecimento à Universidade Federal Juiz de Fora (UFJF), que nos apoiou e deu indispensável suporte à realização do evento. Nosso agradecimento também para os coordenadores do comitê de programa, Aldri Luiz dos Santos (UFMG) e Weverton Luis da Costa Cordeiro (UFRGS). Aos coordenadores dos eventos satélites do evento: André Grégio (UFPR) e Giuliano Giova (IBPTech), coordenadores do Workshop de Forense Computacional (WFC); Diogo Menezes Ferrazani Mattos(UFF) e Eduardo Pagani Julio (UFJF), coordenadores do Salão de Ferramentas; Carlos Ferraz (UFPE), coordenador do Workshop de Gestão de Identidades Digitais (WGID); Lourenço Alves Pereira Júnior (ITA) e Diego Kreutz (UNIPAMPA), coordenadora do Workshop de Trabalhos de Iniciação Científica e de Graduação (WTICG); Marcos Simplício (USP) e Paulo Matias (UFScar), Coordenadores do Workshop de Tecnologia Eleitoral (WTE); e por fim, Dianne Scherly Varela de Medeiros (UFF) e Roberto Samarone Dos Santos Araujo (UFPA), Coordenadores dos minicursos do SBSeg 2023. Gostaríamos de agradecer também ao Diretor do Centro Ciências da UFJF, Marco Antônio Escher, que nos dispensou incalculável apoio. Finalmente, o apoio da Sociedade Brasileira de Computação (SBC) e da Coordenação e do Comitê Gestor da Comissão Especial de Segurança da Informação (CESeg) da SBC foram determinantes para o sucesso do evento.

Todos os envolvidos contribuíram incansavelmente para fornecer uma programação rica e diversificada, discutindo temas relevantes no cenário nacional e internacional. A contribuição da comunidade científica brasileira foi de fundamental importância para manter a qualidade técnica dos trabalhos e fortalecer a ciência, a tecnologia e a inovação no Brasil.

Desejamos a todos um produtivo SBSeg.

Alex Borges Vieira (UFJF) e Edelberto Franco Silva (UFJF)
Coordenadores Gerais do SBSeg 2023

Mensagem dos Coordenadores de Minicursos

É com grande alegria e satisfação que apresentamos a seleção de Minicursos para o XXIII Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais (SBSeg) sediado em Juiz de Fora - MG de 18 a 21 de Setembro de 2023. Recebemos 12 propostas de minicursos e 6 foram selecionadas para publicação e apresentação, representando uma taxa de aceitação de 50%. O Comitê de Avaliação foi composto por 16 pesquisadores para a elaboração dos pareceres e cada proposta recebeu 4 pareceres, gerando ao todo 48 revisões.

Os minicursos do SBSeg têm se adaptado para atender os anseios do público do evento. Isto significa estar alinhado não somente às expectativas de quem deseja conteúdos mais práticos, mas também de quem espera algo mais próximo da fronteira do conhecimento na área de cibersegurança. Assim, esta edição do SBSeg oferece minicursos mais teóricos e mais aplicados.

Aproveitamos para agradecer aos membros do Comitê de Programa por terem contribuído de forma voluntária no processo de avaliação e de seleção dos minicursos.

Cada minicurso selecionado corresponde a um capítulo de 47 a 56 páginas deste livro com conteúdo ministrado, por especialistas, de forma presencial durante o evento.

No Cap. 1 “Explorando esquemas criptográficos pós-quânticos considerados pelo NIST com implementação em Sage”, os autores mostram as bases teóricas com experimentação prática de uma seleção de esquemas criptográficos pós-quânticos que o NIST considera promissores. Com apoio da implementação em Sage, os autores explicam conceitos matemáticos e esquemas criptográficos de forma mais acessível aos leitores.

No Cap. 2 “Sistemas de votação fim-a-fim: teoria e prática”, os autores apresentam os benefícios e desafios de sistemas de votação fim-a-fim (E2E), considerando o cenário brasileiro como seu potencial alvo de aplicação. A discussão é pautada pelo estudo de dois exemplos: um sistema baseado em redes de misturadores, similar à primeira versão do sistema de votação online Helios, implementado com o auxílio da biblioteca Verificatum; e o sistema ElectionGuard.

No Cap. 3 “Introdução à engenharia social: da psicologia cognitiva aos ataques automatizados”, os autores fornecem uma visão abrangente sobre a interseção entre a engenharia social, a psicologia e a automatização computacional, abordando as técnicas de manipulação empregadas. São abordados os aspectos psicológicos relacionados com os principais ataques de engenharia social e a crescente preocupação com a engenharia social automatizada. São apresentados os principais vieses cognitivos explorados nos ataques de engenharia social e os fundamentos e técnicas aplicados no contexto de engenharia social automatizada.

No Cap. 4 “Ameaças e vulnerabilidades em Open RAN: desafios e soluções” apresentam-se as principais ameaças, vulnerabilidades e vetores de ataque às redes de acesso via rádio abertas, também conhecidas como Open RAN, discutindo-se também os desafios de pesquisa para prover segurança a essas redes.

No Cap. 5 “Desenvolvimento ágil de software seguro e a cultura DevSecOps”, o foco está em mostrar como a aplicação combinada de técnicas, práticas e ferramentas pode

aumentar a segurança do software já durante os primeiros ciclos de desenvolvimento ágil, incorporando a cultura de prevenção à cultura DevSecOps, na qual impera a prática de testar automaticamente, detectar os problemas cedo e corrigí-los rapidamente.

No Cap. 6 “Proteção de templates biométricos”, os autores apresentam mecanismos para proteção de templates biométricos armazenados em sistemas modernos, mostrando as principais características desses mecanismos e os principais critérios que podem ser utilizados para avaliação do desempenho, da segurança e da eficácia dos mecanismos. Os autores mostram uma visão geral sobre o estado atual das tecnologias de proteção de templates biométricos, bem como de problemas em aberto e potenciais necessidades futuras.

Gostaríamos de agradecer a todos os autores que submeteram suas propostas de minicursos ao SBSeg 2023, nos motivando a realizar anualmente este evento de interesse, visibilidade e sucesso crescentes. Em especial agradecemos aos autores dos minicursos aceitos que prepararam os capítulos deste livro.

Expressamos também nossos agradecimentos aos coordenadores gerais do SBSeg 2023, professores Alex Borges Vieira (UFJF) e Edelberto Franco Silva (UFJF), pela disponibilidade, ajustes operacionais, orientações providas e por confiarem em nós para coordenar os trabalhos de minicursos desta edição do SBSeg. Desejamos que todos aproveitem ao máximo os conteúdos deste livro!

Dianne Scherly Varela de Medeiros (UFF) e
Roberto Samarone Dos Santos Araujo (UFPA)
Coordenadores dos Minicursos do SBSeg 2023

Comitê de Organização

Coordenadores Gerais
Alex Borges Vieira (UFJF)
Edelberto Franco Silva (UFJF)

Coordenação de Minicursos
Dianne Scherly Varela de Medeiros (UFF)
Roberto Samarone Dos Santos Araujo (UFPA)

Comitê de Programa dos Minicursos
Altair Santin (PUCPR)
Daniel Batista (USP)
Diego Kreutz (Unipampa e Monash University)
Diogo Mattos (UFF)
Eduardo Feitosa (UFAM)
Eduardo Souto (UFAM)
Eduardo Viegas (PUCPR)
Igor Moraes (UFF)
Luiz Fernando Rust da Costa Carmo (Inmetro)
Marcia Henke (UFSM)
Marco Aurelio Amaral Henrique (Unicamp)
Marco Simplício (USP)
Natalia Castro Fernandes (UFF)
Raul Ceretta Nunes (UFSM)
Ricardo Custódio (UFSC)
Ricardo Dahab (Unicamp)

Sumário

Capítulo 1

Explorando esquemas criptográficos pós-quânticos considerados pelo NIST com implementação em Sage

Thales Paiva (CASNAV e Fundep), Vitor Ponciano (CASNAV, Fundep e UFRJ), Everaldo Moreira (CASNAV e UNICAMP), Rafael Oliveira (CASNAV e IME), Vilc Rufino (CASNAV), Cabral Melo (UFRJ), Julio López (UNICAMP), Eduardo Ueda (SENAC e IPT) e Routo Terada (USP)

Abstract

This chapter presents four post-quantum cryptography schemes that NIST considers to be promising: NTRU, Crystals-Kyber, Crystals-Dilithium, and HQC. To introduce the mathematical foundation of the schemes, we take a concrete approach by providing SageMath code that allows newcomers to the field to further explore with the mathematical objects. We cover a broad range of topics ranging from Algebra and efficient implementation of polynomial operations with the Number Theory Transform to Reed-Solomon codes and advanced cryptographic constructions.

Resumo

Este capítulo apresenta quatro algoritmos criptográficos pós-quânticos que o NIST considera promissores: NTRU, Crystals-Kyber, Crystals-Dilithium, e HQC. Para apresentar os fundamentos dos esquemas considerados, propomos uma abordagem prática em que descrevemos o código em SageMath no meio da discussão matemática, o que permite que pesquisadores interessados na área possam explorar os conceitos enquanto aprendem. Os esquemas escolhidos permitem uma discussão abrangente de tópicos, indo de Álgebra e a implementação eficiente de operações polinomiais com a Transformada da Teoria dos Números, até códigos de Reed-Solomon e construções criptográficas avançadas.

O presente trabalho foi realizado com o apoio da Financiadora de Estudos e Projetos (FINEP).

1.1. Introdução

O trabalho seminal de Shor apresentou novos algoritmos quânticos aplicáveis aos problemas da fatoração de inteiros e do logaritmo discreto [Shor 1994]. Embora ainda não sejam conhecidos computadores quânticos grandes e confiáveis o suficiente para aplicar esses algoritmos contra problemas reais, é possível que futuros avanços na construção de tais computadores tornem vulneráveis alguns dos esquemas criptográficos mais usados hoje, como os de curvas elípticas e o RSA.

Esquemas criptográficos para os quais não se conhecem ataques quânticos eficientes são chamados pós-quânticos. Com o objetivo de acelerar a implantação de esquemas pós-quânticos, o National Institute of Standards and Technology (NIST) deu início, em 2016, um processo de padronização de tais esquemas [Chen et al. 2016]. O processo foca em primitivas essenciais para a comunicação segura na Internet, como assinatura digital e troca de chaves. Este processo é dividido em etapas, em que os esquemas melhores avaliados avançam rumo à padronização.

Hoje o processo do NIST está em sua 4ª. rodada [Alagic et al. 2022], e já foram selecionados esquemas para serem padronizados, tanto para troca de chaves, como o Crystals-Kyber [Avanzi et al. 2019], quanto para assinatura digital, como o Cyrstals-Dilithium [Bai et al. 2021], o Falcon [Prest et al. 2020], e o SPHINCS+ [Bernstein et al. 2019b]. O NIST considera padronizar outros esquemas para troca de chaves escolhidos dentre o BIKE [Aragon et al. 2022], o HQC [Melchor et al. 2021], e o Classic McEliece [Bernstein et al. 2019a], chamados alternativos. Note que há esquemas de troca de chaves fora da lista do NIST considerados tão válidos quanto o Crystals-Kyber [Avanzi et al. 2019], como o NTRU [Hoffstein et al. 1998] e o Saber [D'Anvers et al. 2018], porém que foram preteridos por critérios de desempenho.

Objetivo: Apresentar uma seleção de esquemas criptográficos pós-quânticos, considerados promissores pelo NIST, de forma que os participantes os conheçam e possam experimentá-los. Os esquemas selecionados foram: (i) NTRU [Chen et al. 2020], (ii) Crystals-Kyber [Avanzi et al. 2019], (iii) Crystals-Dilithium [Bai et al. 2021], e (iv) HQC [Melchor et al. 2021].

Notas sobre o código: O capítulo apresenta os códigos em SageMath dos esquemas escolhidos, contudo essas implementações possuem caráter didático e não devem ser aplicadas diretamente em sistemas em produção. Todo o código mostrado neste capítulo está disponível em `https://github.com/thalespaiva/pqsage`.

1.2. Conceitos fundamentais

1.2.1. Anéis comutativos e corpos finitos

Esquemas de criptografia de chave pública, em geral, fundamentam sua segurança na dificuldade de resolver problemas matemáticos reconhecidamente difíceis. Porém, para que um problema matemático difícil seja útil, é requerido que sua representação computacional seja factível e eficiente.

Para definir muitos dos problemas matemáticos em que os esquemas atuais são baseados, precisamos trabalhar com conjuntos que admitem operações como soma e multiplicação. Conjuntos em que a multiplicação é distributiva sobre a soma, e onde ambas as operações são comutativas e associativas são chamados anéis comutativos.

Neste texto, há dois tipos de anéis comutativos em que nos interessamos: anéis de inteiros e anéis polinomiais. Dado um inteiro positivo q, denotamos por $\mathbb{Z}_q$ o conjunto $\mathbb{Z}_q = \{0, 1, \ldots, q-1\}$. Se $a, b \in \mathbb{Z}_q$, então as operações $a+b$, $a-b$ ou ab são feitas módulo q. Considere o exemplo em Sage abaixo em que definimos $\mathbb{Z}_{10}$. Aproveitamos para mostrar como ver a tabela de multiplicação, que ilustra como todas as operações são feitas módulo 10.

```
sage: Zq = Integers(10)
sage: Zq.multiplication_table('digits')
```

Note que, em $\mathbb{Z}_{10}$, alguns elementos têm inversa e outros não. Por exemplo, $3^{-1} = 7$, pois $3 \cdot 7 \equiv 1 \mod 10$. Porém 2 não é inversível pois $2a \not\equiv 1 \mod 10$ para todo $a \in \mathbb{Z}_{10}$. Em geral, se $a \in \mathbb{Z}_q$, então a é inversível se somente se $\gcd(a, q) = 1$.

Dado qualquer anel R, podemos definir um anel polinomial $R[x]$, que consiste em polinômios na variável x de coeficientes em R. Apesar de este conjunto ser infinito, podemos usá-lo para construir um anel finito de forma análoga ao que usamos para os anéis finitos de inteiros. Se tomarmos um polinômio $m(x) \in R[x]$, podemos construir o anel quociente $R[x]/m(x)$ da seguinte forma: $R[x]/m(x) = \{p \bmod m(x) : p(x) \in R[x]\}$. Neste caso, se $a(x), b(x) \in R[x]/m(x)$, então as operações $a(x)+b(x)$ e $a(x)b(x)$ são feitas módulo $m(x)$. Note que, neste texto, muitas vezes omitiremos a variável x, denotando o polinômio $a(x)$ simplesmente por a.

Neste texto, em geral estaremos interessados em anéis da forma $\mathbb{Z}_q[x]/(x^n+1)$ ou $\mathbb{Z}_q[x]/(x^n-1)$. Uma vantagem destes anéis é que a redução módulo (x^n+1) ou (x^n-1) é fácil pois $x^n \equiv 1 \mod (x^n-1)$ e $x^n \equiv -1 \mod (x^n+1)$.

Em Sage, podemos construir o anel $\mathbb{Z}_{11}[x]/(x^4-1)$ e realizar experimentos utilizando o código fornecido a seguir.

```
sage: PolyRing = PolynomialRing(Integers(11), 'x')
....: x = PolyRing.gen()
....: Rq = PolyRing.quotient(x^4 - 1, 'x')
....: a = Rq.random_element()
....: b = Rq.random_element()
....: print(f'{a = }\n{b = }\n{a * b = }')
a = x^3 + 7*x^2 + 7*x + 5
b = 3*x^3 + 7*x^2 + 2*x + 7
a * b = 8*x^3 + 2*x^2 + 10*x + 8
```

Em $\mathbb{Z}_{11}$, o polinômio (x^4-1) pode ser fatorado em $(x+1)(x+10)(x^2+1)$, então existem polinômios em $\mathbb{Z}_{11}[x]/\left(x^4-1\right)$ que não têm inversa. Em particular, múltiplos dos fatores de (x^4-1) não são inversíveis. Polinômios que não podem ser fatorados em polinômios não constantes com coeficientes em $\mathbb{Z}_q$ são ditos irredutíveis em $\mathbb{Z}_q$.

Anéis em que todos os elementos não-nulos são inversíveis são chamados corpos. Se q é um número primo, então $\mathbb{F}_q = \mathbb{Z}_q$ é um corpo finito de ordem q. O Kyber e o Dilithium usam anéis polinomiais sobre corpos finitos da forma $\mathbb{F}_q[x]/(x^{256}+1)$ para diferentes valores de q, enquanto o HQC é usa o corpo binário $\mathbb{F}_2$ e usa o anel polinomial

$\mathbb{F}_q[x]/(x^r-1)$.

O exemplo abaixo mostra como construir o corpo finito $\mathbb{F}_q$ para o primo $q = 13$. Note como de fato $4 \cdot 10 = 40 \equiv 1 \mod 13$.

```
sage: F = GF(13)
....: a = F.random_element()
....: print(f'{a = }, {a.inverse() = }')
a = 4, a.inverse() = 10
```

Há corpos finitos de ordem não prima $q = p^m$, onde p é primo. Em particular para nosso texto, um dos códigos corretores de erro usados no HQC é definido sobre $\mathbb{F}_{2^8} = \mathbb{F}_2[x]/\left(x^8+x^4+x^3+x^2+1\right)$. Este anel polinomial forma um corpo finito pois o polinômio $x^8+x^4+x^3+x^2+1$ é irredutível sobre $\mathbb{F}_2$. Isso faz com que, para todo polinômio $a \in \mathbb{F}_2[x]$ de grau menor do que 8, o máximo divisor comum entre a e $x^8+x^4+x^3+x^2+1$ seja 1. Como há $2^8 = 256$ polinômios de grau menor que 8 em $\mathbb{F}_2[x]$, este corpo é chamado de $\mathbb{F}_{2^8}$. O corpo finito $\mathbb{F}_{2^8}$ é particularmente importante para computação, pois cada um de seus polinômios pode ser associado a um byte. Intuitivamente, podemos pensar que $\mathbb{F}_{2^8}$ define operações de soma e multiplicação, ambas inversíveis, sobre bytes.

Todos os esquemas de que trataremos neste texto usam alguma noção de tamanho de polinômios ou vetores. Dado um polinômio $a = a_0 + \ldots + a_{n-1}x^{n-1} \in \mathbb{Z}_q/m(x)$, definimos a sua norma ifinito como $\|a\|_\infty = \max\{|\bar{a}_i| : i = 0,\ldots,n-1\}$, onde $\bar{a}_i = a_i \mod q$ e $-\lfloor q/2 \rfloor \leq \bar{a}_i < \lceil q/2 \rceil$. Isto é, $\bar{a}_i$ é o coeficiente a_i centralizado.

Para facilitar a leitura, quando um polinômio c ou vetor $\mathbf{u}$ tiver coeficientes pequenos, usaremos a cor azul para representá-lo. Note que vetores pequenos tipicamente são secretos, então isso também ajudará a identificar variáveis que representam chaves secretas ou segredos temporários usados durante a encriptação.

1.2.2. Criptografia

O processo de padronização do NIST considera duas primitivas criptográficas importantes, KEMs e assinaturas digitais. Para serem considerados, os esquemas devem propor parâmetros que, dados os melhores ataques conhecidos, instanciem esquemas seguros de sob algum dos 5 critérios definidos pelo NIST.

Os níveis de segurança definidos pelo NIST são apresentados na Tabela 1.1. Para um exemplo concreto, se um KEM apresenta parâmetros de nível de segurança igual a 3, então quebrar[1] o KEM deve ter custo computacional igual ou superior a quebrar uma chave de 192 bits do AES.

Esquemas de encriptação de chave pública: Esquemas de encriptação de chave pública, abreviados por PKE (*public-key encryption*), são definidos por 3 algoritmos: geração de chaves, encriptação e decriptação. A geração de chaves devolve um par de chaves pública e privada, denotadas por **pk** e **sk**, respectivamente. A encriptação recebe uma chave pública **pk** e uma mensagem **m**, e devolve um texto cifrado **c**. O algoritmo de decriptação usa a chave secreta **sk** para obter, a partir do encriptado **c**, a mensagem

[1]Por enquanto, podemos pensar que quebrar significa computar a chave secreta a partir da pública, ou descobrir a chave de sessão que foi encapsulada a partir do texto encriptado e a chave pública.

Tabela 1.1. Níveis de segurança pós-quântica de acordo com o NIST.

Nível de segurança	Dificuldade correspondente
Nível 1	Busca por chave do AES128
Nível 2	Busca por colisão no hash SHA3-256
Nível 3	Busca por chave do AES192
Nível 4	Busca por colisão no hash SHA3-384
Nível 5	Busca por chave do AES256

correspondente.

São considerados dois tipos de atacantes: passivos e ativos. Atacantes passivos não interagem com o portador da chave secreta. Ao atacante passivo, somente é dado acesso à chave pública do alvo, e o atacante pode então usá-la para fazer ataques chamados de CPA (*chosen-plaintext attack*), em que textos legíveis arbitrários são encriptados para tentar detectar padrões que facilitem o ataque. Atacantes ativos são poderosos e podem pedir para o portador da chave secreta decriptar textos encriptados escolhidos, caracterizando um ataque do tipo CCA (*chosen-ciphertext attack*).

Em geral, são considerados 3 tipos de objetivos ao atacar um PKE. Naturalmente, o objetivo mais natural, e devastador, é recuperar a chave secreta **sk** a partir da pública **pk**. Outro objetivo é recuperar o texto legível **m** associado a um encriptado **c** e uma chave pública **pk**. Quando tal recuperação é inviável para um PKE, dizemos que este é unidirecional, abreviado por OW (*one-way*). Há também um objetivo mais fraco, que é, dado um encriptado **c**, a chave pública **pk**, e duas mensagens $\mathbf{m}_0$ e $\mathbf{m}_1$, determinar qual destas duas mensagens foi encriptada para produzir **c**. Um PKE que resiste a este tipo de ataque é dito indistinguível, abreviado por IND.

Um ataque é caracterizado pelo seu objetivo e pelo poder do atacante. Assim, a segurança de um PKE é caracterizada pelo ataque mais poderoso a que ele resiste. Por exemplo, dizemos que um PKE é OW-CPA, quando ele é unidirecional contra ataques passivos. A forte noção de segurança tipicamente objetivada na construção de esquemas é IND-CCA, ou indistinguibilidade de encriptados contra atacantes ativos.

Esquemas de encapsulamento de chaves: Um KEM consiste em 3 algoritmos: geração de chaves, encapsulamento e desencapsulamento. A geração de chaves gera um par de chaves pública e privada, denotadas por **pk** e **sk**, respectivamente. O encapsulamento toma como único parâmetro uma chave pública **pk** e devolve uma chave de sessão **k** e um texto encriptado **c** destinado ao portador da chave secreta associada a **pk**. O desencapsulamento consiste em obter a chave de sessão **k** a partir do texto encriptado **c** usando a chave secreta **sk**. Então, se o desencapsulamento for bem sucedido, o remetente e o destinatário poderão usar a chave **k** num esquema simétrico, como o AES-GCM, para a comunicação segura. As mesmas noções de segurança se aplicam tanto a PKEs quando aos KEMs. Um KEM pode ser construído a partir de um PKE, por exemplo, ao se considerar uma chave de sessão gerada aleatoriamente como a mensagem. Tipicamente, KEMs são construídos sobre PKEs usando o que chamamos de conversões de segurança ou transformações-CCA. Estas transformações pegam um PKE seguro contra atacantes passivos, por exemplo IND-

Geração de chaves

$\mathbf{pk}_{\text{PKE}}, \mathbf{sk}_{\text{PKE}} \leftarrow$ Gere chaves do PKE
$\sigma \leftarrow$ 256 bits aleatórios
$\mathbf{pk} \leftarrow \mathbf{pk}_{\text{PKE}}$
$\mathbf{sk} \leftarrow (\mathbf{sk}_{\text{PKE}}, \sigma)$
Devolva $(\mathbf{pk}, \mathbf{sk})$

Encapsulamento $(\mathbf{pk})$

$\mathbf{m} \leftarrow$ Mensagem aleatória do PKE
$\mathbf{r} \leftarrow G(\mathbf{m}, \mathbf{pk})$
$\mathbf{c} \leftarrow$ Encripte $(\mathbf{pk}, \mathbf{m}; \mathbf{r})$ com PKE
$\mathbf{k} \leftarrow H(\mathbf{m}, \mathbf{c})$
Devolva $(\mathbf{c}, \mathbf{k})$

Desencapsulamento $(\mathbf{sk} = (\mathbf{sk}_{\text{PKE}}, \sigma), \mathbf{c})$

$\mathbf{k}_{\text{rejeição}} \leftarrow H(\sigma, \mathbf{c})$
$\mathbf{m} \leftarrow$ Decripte$(\mathbf{sk}_{\text{PKE}}, \mathbf{c})$ com PKE
Se $\mathbf{m} = \perp$, **devolva** $\mathbf{k}_{\text{rejeição}}$
$\mathbf{r} \leftarrow G(\mathbf{m}, \mathbf{pk})$
$\hat{\mathbf{c}} \leftarrow$ Encripte $(\mathbf{pk}, \mathbf{m}; \mathbf{r})$ com PKE
Se $\hat{\mathbf{c}} \neq \mathbf{c}$, **devolva** $\mathbf{k}_{\text{rejeição}}$
Devolva $\mathbf{k} \leftarrow H(\mathbf{m}, \mathbf{c})$

Figura 1.1. Transformação de Fujisaki-Okamoto com rejeição implícita para construir um KEM IND-CCA a partir de um PKE (OW ou IND)-CPA.

CPA ou OW-CPA, e produzem um KEM IND-CCA. Ter transformadas desse tipo é muito conveniente, pois é muito mais fácil explicar PKEs na forma OW-CPA ou IND-CPA do que na forma IND-CCA.

A Figura 1.1 mostra uma conversão de segurança muito usada chamada Fujisaki-Okamoto com rejeição implícita [Hofheinz et al. 2017]. A ideia geral tornar inviável para um atacante obter qualquer informação sobre a chave secreta através de pedidos de decriptação. Para isso, a transformação garante propriedades como as seguintes: (i) ser inviável gerar um texto encriptado válido sem conhecer a mensagem em claro associada, pois a aleatoriedade usada na encriptação depende da mensagem; (ii) ser viável a detecção de textos encriptados inválidos durante o desencapsulamento, de modo que o valor $\mathbf{k}_{\text{rejeição}}$ devolvido não comprometa informações secretas.

Esquemas de assinatura: Um esquema de assinatura também consiste em 3 algoritmos: geração de chaves, assinatura e verificação. A geração de chaves é análoga à dos KEMs. O algoritmo de assinatura toma uma mensagem **m** e uma chave secreta **sk**, associada a uma chave pública **pk**, e produz uma assinatura **s**. O algoritmo de verificação toma a chave pública **pk** da entidade que supostamente produziu a assinatura, uma mensagem **m** e a assinatura **s**, e verifica se **s** é de fato uma assinatura válida de **m** sob **pk**. A noção de segurança exigida pelo NIST é chamada de EUF-CMA (*existentially unforgeable with respect to chosen message attacks*). Intuitivamente, um esquema é EUF-CMA se um atacante ativo, que pode pode pedir assinaturas para mensagens arbitrárias, não consegue forjar uma assinatura válida para uma mensagem cuja assinatura válida ele nunca viu.

Hashes, XOFs e geradores pseudoaleatórios: Em geral, as implementações de esquemas criptográficos isolam a aleatoriedade real usada pelas funções num parâmetro chamado `randomness`. Este parâmetro é usado para instanciar geradores pseudo-aleatórios criptograficamente seguros, que então serão usados para a amostragem de valores secretos. Funções de hash criptográficas são muito conhecidas e usadas não só em criptografia, como também em segurança da informação. Estas funções tomam uma sequência de bytes de qualquer tamanho e produzem uma saída com número fixo de bytes. Boas funções de hash para criptografia têm propriedades pseudoaleatórias que garantem, por exemplo, que é intratável encontrar duas cadeias de bytes de mesmo hash. Propriedades como essa fazem tais funções particularmente importantes em sistemas de verificação de integridade,

ou mais recentemente, como provas de trabalho.

Objetos muito convenientes para a geração pseudoaleatória com segurança criptográfica são os XOFs (*extendable-output functions*). Exemplos comumente usados de XOFs são as funções SHAKE128 e SHAKE256. Estas funções funcionam como hashes criptográficos, porém sua saída não tem tamanho fixo. Isso faz com que possamos usá-las em algoritmos genéricos para geração pseudoaleatória.

A biblioteca PyCryptodome[2] implementa XOFs e Hashes em Python. Abaixo mostramos como podemos importar o SHAKE128 para instanciar um XOF com a semente 1234 vista como uma sequência de 32 bytes. Note como o método `read`, que aceita um número de bytes como parâmetro, pode ser usado para obter um novo byte de cada vez.

```
sage: xof = SHAKE128.new(int(1234).to_bytes(32))
sage: int.from_bytes(xof.read(1))
130
sage: int.from_bytes(xof.read(1))
99
```

Podemos então definir uma coleção de geradores pseudo-aleatórios baseados nos XOFs da seguinte forma. Comece por um gerador de bits.

```
def xof_random_bit(xof):
    byte = int.from_bytes(xof.read(1))
    return (byte & 1)
```

Também definimos funções mais complexas para gerar inteiros módulo m, isto é entre 0 e $m-1$, ou até entre m_1 e m_2-1.

```
MAX_MOD = 2**256
MAX_MOD_NBYTES = ceil(log(MAX_MOD, 2**8))

def xof_random_int_mod(xof, mod):
    assert mod < MAX_MOD
    bias_region = MAX_MOD - (MAX_MOD % mod)
    v = int.from_bytes(xof.read(MAX_MOD_NBYTES))
    while v >= bias_region:
        v = int.from_bytes(xof.read(MAX_MOD_NBYTES))
    return v % mod

def xof_randrange(xof, min_include, max_exclude):
    diff = max_exclude - min_include
    return min_include + xof_random_int_mod(xof, diff)
```

Usando a função `xof_randrange`, podemos implementar o algoritmo de Fisher-Yates para selecionar k itens de uma lista de itens.

```
def xof_sample_k_indexes(xof, n_items, k):
    a = list(range(n_items))
    for i in range(n_items - 1):
        j = xof_randrange(xof, i, n_items)
        a[i], a[j] = a[j], a[i]
    return a[:k]
```

1.3. NTRU

O NTRU [Chen et al. 2020] é inspirado num esquema de criptografia de chave pública (PKE) introduzido por Hoffstein, Pipher e Silverman em 1998 [Hoffstein et al. 1998].

[2]https://github.com/Legrandin/pycryptodome

Este PKE é baseado em álgebra de polinômios e aritmética modular. Nesta seção, apresentamos a última revisão do NTRU com o conjunto de parâmetros chamado HPS.

Seleção de parâmetros e inicialização: Para cada nível de segurança, o NTRU usa parâmetros (n,q,d,t). Os parâmetros n e q definem o anel polinomial $R_q = \mathbb{Z}_q[x]/(x^n-1)$ usado como base para as operações. O parâmetro d define o número de coordenadas não nulas nos vetores secretos usados para gerar os polinômios das chaves secretas e textos legíveis, enquanto $t = n-2$ determina seu grau máximo.

Seja $\mathcal{T}$ o conjunto em R_q de grau máximo igual a $t = n-2$, cujos coeficientes estão no conjunto $\{-1,0,1\}$. Considere também o seu subconjunto $\mathcal{T}(d) \subset \mathcal{T}$ que contém polinômios com exatamente $d/2$ coeficientes iguais a -1 e outras $d/2$ coeficientes iguais a 1, onde $d = q/8-2$ é um parâmetro do NTRU.

Para que a decriptação seja possível, é necessário que os parâmetros satisfaçam à seguinte condição. Para quaisquer polinômios $r, f \in \mathcal{T}$ e $g, m \in \mathcal{T}(d)$, o polinômio $\hat{a} = 3rg + fm \in \mathbb{Z}[x]/(x^n-1)$, deve ter todos os seus coeficientes no intervalo $\{-q/2, \ldots, q/2-1\}$.

Os parâmetros de segurança do NTRU são definidos da seguinte forma:

```python
@dataclass
class NTRUHPS_Parameters:
    n: int
    q: int
    d: int
    max_degree_t: int

class NTRU_DPKE_OWCPA:
    SecurityParameters = {
        1: NTRUHPS_Parameters(n=509, q=2048, d=254, max_degree_t=509 - 2),
        3: NTRUHPS_Parameters(n=677, q=2048, d=254, max_degree_t=677 - 2),
        5: NTRUHPS_Parameters(n=821, q=2048, d=254, max_degree_t=821 - 2),
    }
```

Considere a fatoração de (x^n-1) em $\Phi_1\Phi_n$, onde $\Phi_1 = (x-1)$ e $\Phi_n = \sum_{i=0}^{n-1} x^i$. Além de R_q, o NTRU usa outros 3 anéis onde certas operações polinomiais são feitas. São eles os anéis $S_q = \mathbb{Z}_q[x]/\Phi_n$, $S_3 = \mathbb{Z}_3[x]/\Phi_n$, e $S_2 = \mathbb{Z}_2[x]/\Phi_n$. O código abaixo mostra a inicialização dos anéis usados pelo NTRU, seguido da inicialização do esquema.

```python
    def __init_rings(self):
        ring_mod_q = PolynomialRing(Integers(self.params.q), 'x')
        x = ring_mod_q.gen()
        phi   = x**self.params.n - 1
        phi_1 = x - 1
        phi_n = sum(x**i for i in range(self.params.n))
        self.Rq = ring_mod_q.quotient_ring(phi)
        self.Sq = ring_mod_q.quotient_ring(phi_n)
        self.S3 = ring_mod_q.change_ring(Integers(3)).quotient(phi_n)
        self.S2 = ring_mod_q.change_ring(Integers(2)).quotient(phi_n)

    def __init__(self, security_level):
        self.params = self.SecurityParameters[security_level]
        self.__init_rings()
```

Para o funcionamento do NTRU, é necessário converter polinômios entre os anéis. Suponha que queremos converter um polinômio f em $R_1 = \mathbb{Z}_{q_1}/m_1(x)$ para um polinômio noutro anel $R_2 = \mathbb{Z}_{q_2}/m_2(x)$. Então, primeiro escrevemos o polinômio com coeficientes

Geração de chaves

$f \leftarrow \textbf{Uniforme}(\mathcal{T})$
$g \leftarrow \textbf{Uniforme}(\mathcal{T}(d))$
$h \leftarrow 3R_q(g)R_q\left(S_q(f)^{-1}\right)$
$\textbf{pk} \leftarrow h$
$\textbf{sk} \leftarrow \left(f, S_3(f)^{-1}, S_q(h)^{-1}\right)$
Devolva $(\textbf{pk}, \textbf{sk})$

Encriptação $(\textbf{pk}, (r,m) \in \mathcal{T} \times \mathcal{T}(d))$

$h \leftarrow \textbf{pk}$
$c \leftarrow R_q(h)R_q(r) + R_q(m)$
Devolva c

Decriptação $(\textbf{sk}, c)$

Se $R_q(c) \neq 0 \bmod \Phi_1$, **devolva** $\perp$
$f, S_3(f)^{-1}, S_q(h)^{-1} \leftarrow \textbf{sk}$
$a \leftarrow S_q(c)S_q(f)$
$m \leftarrow S_3(a)S_3(f)^{-1}$
$r \leftarrow S_q\left(R_q(c) - R_q(m)\right)S_q(h)^{-1}$
Se $r, m \notin \mathcal{T} \times \mathcal{T}(d)$, **devolva** $\perp$
Devolva (r,m)

Figura 1.2. PKE determinístico do NTRU.

centralizados $f = f_0 + f_1 x + \ldots + f_n x^n$, onde cada f_i está no intervalo $-q_1/2 \leq f_i < q_1/2$. Depois computamos o seguinte polinômio convertido, denotado por $R_2(f)$, como $R_2(f) = (f_0 \bmod q_2) + \ldots + (f_n \bmod q_2)x^n \mod m_2(x)$.

Note que, como o módulo $\Phi = \Phi_1\Phi_n$ de R_q é um múltiplo dos módulos Φ_n usado em S_q, então: $S_q\left(R_q(a) + R_q(b)R_q(c)\right) = S_q(a) + S_q(b)S_q(c)$. No entanto, esta propriedade não vale no outro sentido, isto é, em geral $R_q(S_q(a) + S_q(b)) \neq R_q(a) + R_q(b)$.

A operação de centralização dos coeficientes pode ser escrita da seguinte forma:

```
    def centered(self, polynomial):
        modulo = polynomial.base_ring().order()

        def mod_centered(v):
            v = int(v) % modulo
            if v <= modulo // 2:
                return v
            return - (modulo - v)

        return [mod_centered(c) for c in polynomial]
```

Assim, a conversão de um polinômio para outro anel é dada por:

```
    def to_Rq(self, polynomial):
        return self.Rq(self.centered(polynomial))
    def to_Sq(self, polynomial):
        return self.Sq(self.centered(polynomial))
    def to_S3(self, polynomial):
        return self.S3(self.centered(polynomial))
    def to_S2(self, polynomial):
        return self.S2(self.centered(polynomial))
```

A Figura 1.2 mostra os algoritmos usados no NTRU. Nas próximas seções, vamos descrever cada uma delas com a sua respectiva implementação.

Geração de chaves: A geração de chaves é implementada da seguinte forma:

```
    def keygen(self, randomness):
        xof = SHAKE256.new(randomness)
        f_in_s3 = self.S3(self.gen_ternary_coeffs(xof))
        g_in_s3 = self.S3(self.gen_ternary_coeffs_for_d(xof, self.params.d))
        f_inv_3 = f_in_s3.inverse()
        f = self.to_Rq(f_in_s3)
        g = self.to_Rq(g_in_s3)
        h = 3 * g * self.to_Rq(self.Sq_inverse(f))
        h_inv_q = self.to_Rq(self.Sq_inverse(h))
        return h, (f, f_inv_3, h_inv_q)
```

Primeiro, polinômios f e g são escolhidos de $\mathcal{T}$ e $\mathcal{T}(d)$, respectivamente, usando a XOF e as funções apropriadas. Os coeficientes de polinômios em $\mathcal{T}$ e $\mathcal{T}(d)$ são gerados usando o código abaixo:

```
    def gen_ternary_coeffs(self, xof):
        return [choice([-1, 0, 1]) for _ in range(self.params.max_degree_t + 1)]

    def gen_ternary_coeffs_for_d(self, xof, d):
        assert d % 2 == 0
        coeffs = [0] * (self.params.max_degree_t + 1)
        supp = xof_sample_k_indexes(xof, self.params.max_degree_t + 1, d)
        for i in range(d // 2):
            coeffs[supp[i]] = 1
            coeffs[supp[d//2 + i]] = -1
        return coeffs
```

Note que, como $f \in \mathcal{T}$, então 2 não divide f, e portanto f admite inversa em S_q. Além disso, lembre que S_2 e S_3 são corpos, então quaisquer de seus elementos não nulos admitem inversos, inclusive f.

Nos corpos S_2 e S_3, calcular a inversa é fácil através do algoritmo de Euclides estendido, e o Sage as calcula simplesmente através do método `inverse`. Porém, para S_q que não é um corpo, vamos implementar uma função eficiente para o cálculo da inversa customizada para este caso. A função `Sq_inverse` calcula o inverso de um polinômio $a \in S_q$, supondo que existe. Seu código é dado abaixo, e em seguida explicamos por que ela funciona.

```
    def Sq_inverse(self, a):
        assert is_power_of_two(self.params.q)
        a_in_Sq = self.to_Sq(a)
        v0 = self.to_Sq(self.to_S2(a).inverse())
        t = 1
        while t < log(self.params.q, 2):
            v0 = v0 * (2 - a_in_Sq * v0)
            t = 2*t
        assert (v0 * a_in_Sq) == 1
        return v0
```

A ideia da função é começar com v_0 sendo o inverso de a em S_2 e usar este inverso para construir inversos em $S_{2^2}, S_{2^4}, S_{2^8}, \ldots$, progressivamente, onde $S_i = \mathbb{Z}_i[x]/(\Phi_n)$. O critério de parada garante que sejam feitas pelo menos $\log_2(\log_2 q)$ iterações, que são suficientes para que o inverso em S_q seja encontrado.

Vamos analisar a primeira iteração. Para isso, considere $v_1 = v_0(2 - av_0)$. Pela definição de v_0, sabemos que $av_0 \equiv 1 \bmod 2$, então $av_0 = 2m+1$, para algum polinômio m. Mas isso implica que: $av_1 = av_0(2-av_0) = (2m+1)(2-2m-1) = (2m+1)(-2m+1)$. Portanto $av_0 = -4m^2+1 \equiv 1 \bmod 4$, isto é, v_1 é o inverso de a em S_4. Este argumento pode ser iterado para mostrar, por indução, que o resultado de `Sq_inverse` de fato é uma inversa em S_q.

Encriptação: O texto legível do NTRU é um par (r,m) em que $r \in \mathcal{T}$ e $m \in \mathcal{T}(d)$. Implementamos abaixo uma função que gera um texto legível (r,m) através de bytes usados como semente em um XOF.

```
    def sample_plaintext(self, randomness):
        xof = SHAKE256.new(randomness)
        r = self.gen_ternary_coeffs(xof)
```

```
        m = self.gen_ternary_coeffs_for_d(xof, self.params.d)
        return r, m
```

A encriptação é simples, basta computar $c = hr + m$ com as operações em R_q.

```
    def encrypt(self, pk, plaintext):
        h = pk
        r_coeffs, m_coeffs = plaintext
        r = self.Rq(r_coeffs)
        m = self.Rq(m_coeffs)
        c = h * r + m
        return c
```

Decriptação: Abaixo, listamos o código da decriptação:

```
    def decrypt(self, sk, ciphertext):
        (f, f_inv_3, h_inv_q) = sk
        c = ciphertext
        if sum(c) != 0:
            return None
        a = (self.to_Sq(c) * self.to_Sq(f))
        m = self.to_S3(a) * f_inv_3
        r = (c - self.to_Rq(m)) * h_inv_q
        m = self.centered(m)
        r = self.centered(self.to_Sq(r))
        if self.is_valid_plaintext(r, m):
            return r, m
        else:
            return None
```

A decriptação começa com um teste de validade do texto encriptado c. Se c foi calculado corretamente, então $c = hr + m$, e h é um polinômio múltiplo de g. Como tanto m quando g foram selecionados de $m, g \in \mathcal{T}(d)$, então ambos os polinômios têm mesmo número de coeficientes iguais a 1 e -1.

Dessa forma, $m(1) = g(1) = 0 \bmod q$, o que implica que $c(1) = h(1)r(1) + m(1) = 0 \bmod q$. Implementamos a computação de $c(1)$ como a soma de seus coeficientes. Note que, em Sage, não precisamos colocar o módulo q pois as operações sobre coeficientes de c já são feitas em $\mathbb{Z}_q$.

Vamos agora verificar que a decriptação funciona. O polinômio a calculado durante a decriptação é

$$\begin{aligned} a &= S_q(c)\, S_q(f) \\ &= S_q\left(R_q(h)\, R_q(r) + R_q(m)\right) S_q(f) \\ &= S_q(h)\, S_q(r)\, S_q(f) + S_q(m)\, S_q(f). \end{aligned}$$

Considere a primeira parcela

$$\begin{aligned} S_q(h) S_q(f) S_q(r) &= S_q\left(3 R_q(g)\, R_q\left(S_q(f)^{-1}\right)\right) S_q(f)\, S_q(r) \\ &= 3 S_q(g)\, S_q(f)^{-1}\, S_q(f)\, S_q(r) \\ &= 3 S_q(g)\, S_q(r). \end{aligned}$$

Então $a = 3S_q(g)S_q(r) + S_q(m)S_q(f)$. Mas lembre que os parâmetros do NTRU são selecionados de forma que $\hat{a} = 3gr + mf \in \mathbb{Z}[x]/(x^n - 1)$ tenha sempre coeficientes no intervalo $\{-q/2, \ldots, q/2 - 1\}$. Então $\hat{a} \bmod 3 = S_3(\hat{a}) = S_3(a) = S_3(m)S_3(f)$. Isso implica que $S_3(a)S_3(f)^{-1} = S_3(m)S_3(f)S_3(f)^{-1} = S_3(m)$. Basta então centralizar os coeficientes de $S_3(m)$ para obter o valor de m original com coeficientes em $\{-1, 0, 1\}$.

Agora que temos m, podemos recuperar a segunda parte r, da seguinte forma: $S_q\left(R_q(c) - R_q(m)\right)S_q(h)^{-1} = S_q(h)S_q(r)S_q(h)^{-1} = S_q(r)$.

Para garantir que a decriptação foi correta, verifica-se se de fato $r \in \mathcal{T}$ e $m \in \mathcal{T}(d)$. Esta verificação é implementada pela seguinte função.

```
    def is_valid_plaintext(self, r, m):
        if (m.count(-1), m.count(1)) != (self.params.d//2, self.params.d//2):
            return False
        r_non_ternary_coefficients = len([c for c in r if c not in {-1, 0, 1}])
        if r_non_ternary_coefficients:
            return False
        return True
```

1.4. Fundamentos do Kyber e Dilithium: NTT e módulos

Nesta seção, vamos discutir conceitos fundamentais para os esquemas da família Crystals, isto é, o Kyber e o Dilithium. Começamos discutindo a Transformada da Teoria dos Números (NTT), que é parte essencial de ambos os esquemas. Depois discutimos módulos, que são uma generalização de espaços vetoriais, e discutimos a sua implementação.

1.4.1. Multiplicação rápida de polinômios com a NTT

A multiplicação de polinômios é uma operação recorrente no contexto dos esquemas criptográficos apresentados neste minicurso. Então realizar essa operação de forma eficiente é fundamental para atingir desempenho adequado. Nesta seção, vamos apresentar a NTT, uma transformada da família da transformada de Fourier, que é usada tanto pelo Kyber quanto pelo Dilithium para fazer as multiplicações de polinômios.

A apresentação é dividida em três partes. Primeiro mostramos como o Teorema Chinês do Resto pode ajudar na multiplicação de polinômios. Depois mostramos o anel polinomial usado pelo Kyber e pelo Dilithium e como este pode ser quebrado em anéis menores onde as contas são mais eficientes. Finalmente, apresentamos a implementação rápida da NTT.

Teorema Chinês do Resto para polinômios: Fixe um primo q, e um polinômio $m(x)$ de grau n. Considere o anel polinomial $R_q = \mathbb{F}_q[x]/m(x)$ dos resíduos módulo m, isto é

$$\mathbb{F}_q[x]/m(x) = \left\{a \bmod m : a \in \mathbb{F}_q[x]\right\}.$$

Dados dois polinômios a e b de R_q, queremos computar o produto $c = ab$ de forma eficiente. Lembre que, como $a, b \in R_q$, ambos os polinômios têm grau menor que n. Além disso, como o produto $c = ab$ também tem grau menor que 256 pois, pois todas as operações em R_q são feitas $\bmod m(x)$.

Suponha que exista uma fatoração de $m(x)$ em s polinômios $f_1, \ldots, f_s$, isto é

$$m(x) = \prod_{i=1}^{s} f_i.$$

Além disso, suponha que os polinômios f_i são coprimos entre si, isto é $\gcd\left(f_i, f_j\right) = 1$ para $i \neq j$.

Então, o Teorema Chinês do resto garante que existe apenas um polinômio $a \bmod m(x)$ que satisfaz o seguinte sistema de equações modulares

$$\begin{cases} a &= a_1 \mod f_1, \\ \vdots & \vdots \quad \vdots \quad \vdots \\ a &= a_s \mod f_s. \end{cases}$$

Dessa forma, se soubermos os resíduos dos polinômios a e b em relação a cada um dos polinômios f_i, podemos multiplicar os resíduos correspondentes e usar o Teorema Chinês do Resto para obter o produto módulo $m(x)$.

Há, porém, duas observações importantes. A primeira é que se os fatores f_i tiverem grau muito alto, a multiplicação entre resíduos será ineficiente. A segunda é que precisamos ser capazes de computar eficientemente tanto os resíduos a_i e b_i quanto o polinômio c a partir dos resíduos $a_i b_i \mod f_i$.

Fatorando polinômios com raízes da unidade: Vimos que é desejável que o polinômio m seja fatorado em polinômios de grau pequeno para que a multiplicação de resíduos seja rápida. Vamos considerar um caso que é usado tanto pelo Kyber como pelo Dilithium: $m(x) = x^n + 1$, onde n é uma potência de 2.

Fixe um primo q para o qual $\mathbb{F}_q$ admite uma raiz primitiva ω de ordem $2n$ da unidade, isto é $\omega^{2n} = 1$, e não existe nenhum outro $0 < i < 2n$ tal que $\omega^i = 1$. Podemos então fatorar $m(x) = x^n + 1$ notando que $\omega^n = -1$. Escreva $m(x) = x^n - \omega^n$, e note que m então é a diferença de dois quadrados, podendo ser fatorado em

$$m(x) = \left(x^{(n/2)} - \omega^{(n/2)}\right)\left(x^{(n/2)} + \omega^{(n/2)}\right).$$

Se $n/2 = 1$, então a fatoração está completa. Suponha então que n é uma potência de 2 maior que 1. Note que podemos fatorar novamente $\left(x^{(n/2)} - \omega^{(n/2)}\right)$ usando a mesma observação. Considere então como fatorar $\left(x^{(n/2)} + \omega^{(n/2)}\right)$. Mas como $\omega^n = -1$, podemos transformar este fator em outra diferença de dois quadrados escrevendo

$$x^{(n/2)} + \omega^{(n/2)} = x^{(n/2)} - (\omega^n)\omega^{(n/2)} = x^{(n/2)} - \omega^{n+(n/2)}.$$

Usando este procedimento recursivamente $\log_2(n)$ vezes, é possível chegar até a fatoração em polinômios de grau 1.

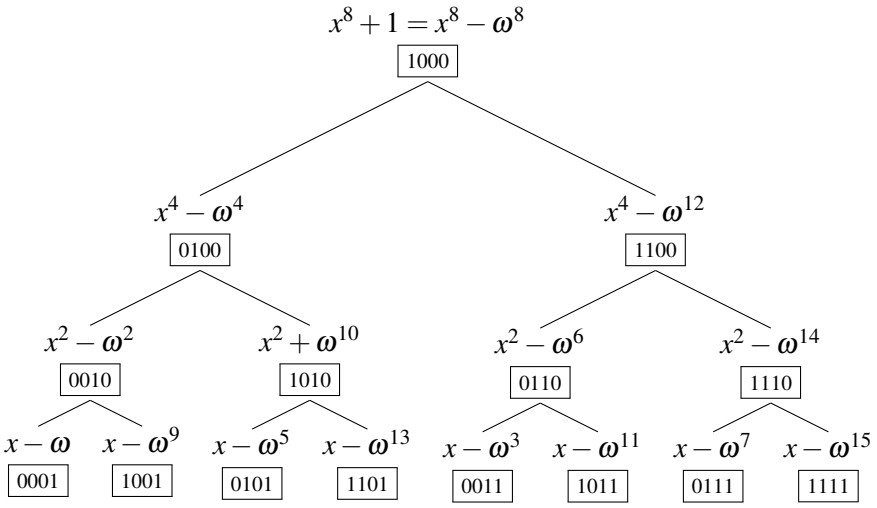


Figura 1.3. Decomposição de x^8+1 usando uma raiz da unidade ω de ordem 16.

A Figura 1.3 ilustra a árvore de decomposição para o polinômio x^8+1, supondo que existe uma raiz primitiva da unidade de ordem 16 em $\mathbb{F}_q$. As etiquetas abaixo dos polinômios indicam a representação em binário da potência de ω correspondente.

A ordem em que as potências de ω aparecem pode parecer misteriosa a princípio. Porém podemos notar um padrão interessante nas etiquetas usadas na Figura 1.3: o filho do nó de etiqueta a têm etiquetas $a/2$ e $a/2+8$. Isso faz com que a ordem em que as etiquetas são usadas na árvore, lendo linha por linha, seja igual à ordem das sequências em binário usando 4 bits, porém considerando a ordem invertida. Isto é 1000, 0100, 1100, e assim por diante. Essa ordem é chamada de bit-invertida, e pode ser implementada em Sage através da seguinte função.

```
def bit_rev(x, length):
    b = bin(x)[2:]
    b = '0' * (length - len(b)) + b
    return int(''.join(reversed(b)), 2)
```

Note que, apesar de termos mostrado a árvore de decomposição completa em polinômios de grau 1, existe também o caso incompleto. Quando n é uma potência de 2 e considerando o polinômio $m=1$, o caso incompleto ocorre quando ω é uma raiz primitiva de ordem 2^o em F_q, tal que $2 \leq o \leq \log_2 n$. Nesse caso, o mesmo procedimento é usado para decompor x^n+1, porém, a decomposição acaba com polinômios de grau $\log_2 n - o + 1$.

Sabemos então como construir polinômios x^n+1 que podem ser fatorados em polinômios de grau baixo, que são convenientes para a multiplicação baseada no Teorema Chinês do Resto. Porém, para que seja possível fazer a multiplicação eficiente entre dois polinômios a e b módulo x^n+1, é preciso definir um algoritmo rápido para fazer conversão a rápida dos polinômios a e b em resíduos em relação aos fatores, e vice-versa.

NTT: computação rápida dos resíduos na árvore de decomposição: Nesta seção, apresentamos a transformada da Teoria dos Números, do inglês Number Theory Transform (NTT), que consiste numa variação discreta da transformada Rápida de Fourier

(FFT). Como a FFT, a NTT pode ser calculada eficientemente. O que nos faz particularmente interessados na NTT em nossa discussão é que, com ela, é possível calcular precisamente os resíduos de um polinômio a numa árvore de decomposição de $x^n + 1$ em F_q, quando n é uma potência de 2.

Uma forma direta de implementar a NTT de um polinômio a num anel $R_q = \mathbb{F}_q[x]/(x^n + 1)$ seria a seguinte. Suponha que temos uma raiz da unidade ω de ordem $2n$. Comece na raiz da árvore de decomposição de $(x^n + 1)$. Então compute $a_0 = a \bmod x^n - \omega^n$ e $a_1 = a \bmod x^n + \omega^n$, e prossiga recursivamente nas subárvores de raízes $x^n - \omega^n$ e $x^n + \omega^n$, e polinômios a_0 e a_1, respectivamente.

Porém uma forma mais rápida de implementá-la é adaptando os algoritmos da FFT, como os de Cooley-Tukey e de Gentleman-Sande, para usar a nossa raiz da unidade num corpo finito.

Vamos mostrar a implementação da NTT rápida para o caso específico em que $n = 256$, e para o caso de $R_q = \mathbb{F}_q/(x^n + 1)$. Em inglês, é comum chamar a NTT no anel $R_q = \mathbb{F}_q/(x^n + 1)$ de *negacyclic* NTT pois, $ax^n = -a \bmod (x^n + 1)$.

```
def ntt_leveled_negacyclic_256(a, field, omega, n_levels):
    assert len(a) == 256
    omega = field(omega)
    assert omega.multiplicative_order() == 2**(n_levels + 1)

    a_hat = copy(a)
    for l in range(7, 7 - n_levels, -1):
        for i in range(0, 2**(7 - l)):
            phi = omega ** bit_rev(2**(7 - l) + i, n_levels)
            for j in range(i * 2**(l + 1), i * 2**(l + 1) + 2**l):
                t0 = a_hat[j]
                t1 = phi * a_hat[j + 2**l]
                a_hat[j] = t0 + t1
                a_hat[j + 2**l] = t0 - t1
    return vector(field, a_hat)
```

Similarmente, a inversa da NTT pode ser calculada da seguinte forma. Note que, de forma análoga à DFT, é necessário normalizar o resultado.

```
def inv_ntt_leveled_negacyclic_256(a_hat, field, omega, n_levels):
    assert len(a_hat) == 256
    omega = field(omega)
    assert omega.multiplicative_order() == 2**(n_levels + 1)

    a = copy(a_hat)
    for l in range(8 - n_levels, 8):
        for i in range(2**(7 - l)):
            phi = omega ** -bit_rev(2**(7 - l) + i, n_levels)
            for j in range(i * 2**(l + 1), i * 2**(l + 1) + 2**l):
                t0 = a[j] + a[j + 2**l]
                t1 = a[j] - a[j + 2**l]
                a[j] = t0
                a[j + 2**l] = phi * t1
    return vector(field, a) * field(2**n_levels)**(-1)
```

Vejamos como usar a NTT para multiplicar dois polinômios. Suponha que estamos em um anel que admite a decomposição completa. Para um exemplo concreto, tome o caso do Dilithium em que $q = 8380417$ e $\omega = 1753$. Como este caso a NTT rápida é completa, a multiplicação no domínio da NTT corresponde à multiplicação de coeficientes em $\mathbb{F}_q$.

Podemos usar o seguinte código para verificar que a multiplicação usando a NTT de fato dá o mesmo resultado que a multiplicação em Sage no anel R_q.

```
F = GF(8380417)
PolyRing = PolynomialRing(F, 'x')
x = PolyRing.gen()
Rq = PolyRing.quotient(x**256 + 1)
omega = F(1753)

def poly_prod_negacyclic_256_complete(a, b):
    a_ntt = ntt_leveled_negacyclic_256(a, F, omega, 8)
    b_ntt = ntt_leveled_negacyclic_256(b, F, omega, 8)
    c_ntt = [a_i * b_i for a_i, b_i in zip(a_ntt, b_ntt)]
    return inv_ntt_leveled_negacyclic_256(c_ntt, F, omega, 8)

def poly_prod_in_sage(a, b):
    return vector(Rq(list(a)) * Rq(list(b)))

a = random_vector(F, 256)
b = random_vector(F, 256)

result_ntt_mult = poly_prod_negacyclic_256_complete(a, b)
result_sage_mult = poly_prod_in_sage(a, b)
print(f'{(result_ntt_mult == result_sage_mult) = }')
```

1.4.2. Módulos

Módulos são generalizações de espaços vetoriais em que os escalares são elementos de um anel, e não necessariamente de um corpo finito. Essas estruturas são convenientes, pois permitem que trabalhemos com vetores e matrizes de polinômios, usando as regras e notações convencionais de álgebra linear, como a adição de vetores, multiplicação por escalares e produto escalar.

O conceito pode ser facilmente entendido com um exemplo. Para isso, consideremos o anel polinomial usado pelo Dilithium definido a seguir. Seja o primo $q = 8380417$, e considere o anel polinomial $R_q = \mathbb{F}_q[x]/\left(x^{256}+1\right)$. Para qualquer inteiro $k \geq 1$, o conjunto R_q^k

$$R_q^k = \left\{ \begin{bmatrix} p_1 \\ \vdots \\ p_k \end{bmatrix} : p_i \in R_q \right\},$$

é um módulo de dimensão k sobre R_q.

Os elementos dos módulos, que chamaremos de vetores, podem ser convenientemente representados pela classe `PolynomialVector` abaixo. Note como, para instanciar um módulo, devemos passar um anel onde a NTT pode ser computada eficientemente, além de uma lista de vetores de coeficientes dos polinômios e uma bandeira que indica se o vetor está no domínio da NTT ou não.

```
class PolynomialVector():

    def __init__(self, ntt_ring, poly_vec_coefficients, in_ntt_domain=False):
        assert all(len(p) == ntt_ring.n for p in poly_vec_coefficients)
        self.ntt_ring = ntt_ring
        self.poly_vec = [vector(ntt_ring.field, p) for p in poly_vec_coefficients]
        self.in_ntt_domain = in_ntt_domain
```

Podemos usar os métodos mágicos de Python para permitir que as operações como adição, subtração e negação sejam representadas de forma concisa no código através dos

respectivos operadores. Note que, para evitar problemas de domínio, temos um método que testa se dois vetores de polinômio são compatíveis.

```

    def test_compatibility_of_domains(self, poly_vec2):
        assert self.ntt_ring.field == poly_vec2.ntt_ring.field
        assert self.in_ntt_domain == poly_vec2.in_ntt_domain

    def __add__(self, poly_vec2):
        self.test_compatibility_of_domains(poly_vec2)
        sum_poly_vec = [v1 + v2 for v1, v2 in zip(self.poly_vec, poly_vec2.poly_vec)]
        return PolynomialVector(self.ntt_ring, sum_poly_vec, in_ntt_domain=self.
    in_ntt_domain)

    def __sub__(self, poly_vec2):
        self.test_compatibility_of_domains(poly_vec2)
        sum_poly_vec = [v1 - v2 for v1, v2 in zip(self.poly_vec, poly_vec2.poly_vec)]
        return PolynomialVector(self.ntt_ring, sum_poly_vec)

    def __neg__(self):
        return PolynomialVector(self.ntt_ring, [-p for p in self.poly_vec])
```

Note como o cálculo da NTT em cada polinômio é delegado ao anel usado para a inicialização.

```
    def ntt(self):
        poly_vec_ntt = [self.ntt_ring.ntt(p) for p in self]
        return PolynomialVector(self.ntt_ring, poly_vec_ntt, in_ntt_domain=True)

    def inv_ntt(self):
        poly_vec = [self.ntt_ring.inv_ntt(p) for p in self]
        return PolynomialVector(self.ntt_ring, poly_vec, in_ntt_domain=False)
```

Através de uma implementação polimórfica da multiplicação, podemos usar o mesmo operador $*$ para calcular: *(i)* o produtos de vetores de polinômios por inteiros, *(ii)* o produtos de vetores de polinômios por polinômios, e *(iii)* o produto escalar entre dois vetores de polinômios.

```
    def _multiply_poly_vecs(self, poly_vec2):
        self.test_compatibility_of_domains(poly_vec2)

        if (self.in_ntt_domain == True):
            prod = sum(self.ntt_ring.poly_product_ntt_domain(p1, p2)
                       for p1, p2 in zip(self, poly_vec2))
            return prod

        else:
            raise NotImplementedError

    def _multiply_poly_vec_and_poly(self, poly):
        assert self.in_ntt_domain

        prod = [self.ntt_ring.poly_product_ntt_domain(poly, p) for p in self]
        return PolynomialVector(self.ntt_ring, prod, in_ntt_domain=True)

    def _multiply_poly_vec_by_scalar(self, field_element):
        return PolynomialVector(self.ntt_ring, [p * field_element for p in self], self.
    in_ntt_domain)

    def __mul__(self, poly_or_poly_vec_or_int):
        if isinstance(poly_or_poly_vec_or_int, PolynomialVector):
            return self._multiply_poly_vecs(poly_or_poly_vec_or_int)
        elif poly_or_poly_vec_or_int in self.ntt_ring.field:
            return self._multiply_poly_vec_by_scalar(poly_or_poly_vec_or_int)
        else:
            return self._multiply_poly_vec_and_poly(poly_or_poly_vec_or_int)
```

1.5. Crystals-Kyber

O Kyber [Avanzi et al. 2019] é um esquema de encapsulamento de chaves baseado no problema (*Learning With Errors*) em módulos (MLWE). Começamos a seção definido o anel usado pelo Kyber para em seguida definir o MLWE. Depois apresentamos um esquema de chave pública geral baseado no MLWE, sobre o qual o Kyber é construído. Ao final, o Kyber e seus algoritmos são apresentados.

1.5.1. O anel polinomial usado pelo Kyber

Em todos os seus níveis de segurança, o Kyber usa o primo $q = 3329$ e o anel polinomial $R_q = \mathbb{F}_q/(x^n+1)$ para $n = 256$. Uma característica importante deste anel é que ele não admite raiz primitiva da unidade de ordem 512, que sabemos ser necessária para aplicar a NTT negacíclica rápida completa quando $n = 256$. Porém, $\omega = 17$ é uma raiz de ordem 256 da unidade em R_q, então podemos calcular a NTT rápida de 7 níveis, obtendo a decomposição de (x^n+1) como

$$x^{256}+1 = \prod_{i=0}^{127}\left(x^2-\omega^{2i+1}\right) = \prod_{i=0}^{127}\left(x^2-\omega^{2\texttt{bit\_rev}(i,7)+1}\right).$$

Representamos o anel R_q usado pelo Kyber pela seguinte classe.

```
class KyberNTTRing():

    def __init__(self):
        self.field = GF(3329)
        self.n = 256
        self.omega_n = self.field(17)
        self.polynomial_ring = PolynomialRing(self.field, 'x')
        self.x = self.polynomial_ring.gen()

    def ntt(self, a):
        return ntt_leveled_negacyclic_256(a, self.field, self.omega_n, 7)

    def inv_ntt(self, a_hat):
        return inv_ntt_leveled_negacyclic_256(a_hat, self.field, self.omega_n, 7)
```

Como a NTT e sua inversa são computadas de forma incompleta em 7 níveis, a multiplicação no domínio da NTT deve ser feita módulo cada um dos fatores $x^2 - \omega^{2\texttt{bit\_rev}(i,7)+1}$.

```
    def poly_product_ntt_domain(self, a_hat, b_hat):

        prod = [None] * 256
        for i in range(0, 256, 2):
            pa = a_hat[i + 1]*self.x + a_hat[i]
            pb = b_hat[i + 1]*self.x + b_hat[i]

            pol_prod = (pa * pb).mod(self.x**2 - self.omega_n**(2*bit_rev(i//2, 7) + 1))
            prod[i + 1] = pol_prod[1]
            prod[i] = pol_prod[0]

        return vector(self.field, prod)
```

1.5.2. O problema MLWE

Em alto nível, o MLWE consiste em encontrar o vetor de um módulo que é a solução de um sistema linear corrompido por erros pequenos. Então, antes de definir formalmente o MLWE, vamos apresentar a distribuição de erros usada.

Definição 1 (Distribuição binomial centrada)**.** Denotamos por $\mathcal{B}_\eta$ a binomial centrada no 0 com parâmetros $(n = 2\eta, p = 1/2)$. Para amostrar um valor b da distribuição $\mathcal{B}_\eta$, escolha 2η bits $b_1, \ldots, b_{2\eta}$ de forma uniformemente aleatória, e devolva

$$b = \sum_{i=1}^{\eta} b_i - \sum_{i=\eta}^{2\eta} b_i = \sum_{i=1}^{\eta} (b_i - b_{i+\eta}) .$$

□

Podemos implementar a amostragem de $\mathcal{B}_\eta$ usando um XOF já inicializado da seguinte forma.

```
def xof_centered_binomial_sample(xof, eta):
    a = sum(xof_random_bit(xof) for _ in range(eta))
    b = sum(xof_random_bit(xof) for _ in range(eta))
    return a - b
```

Denotamos por $\mathcal{B}_\eta\left(R_q\right)$ a distribuição sobre o anel polinomial R_q em que o coeficiente de cada polinômio é selecionado independentemente segundo a distribuição $\mathcal{B}_\eta$. Similarmente, usamos $\mathcal{B}_\eta\left(R_q^k\right)$ para denotar a distribuição sobre vetores de polinômios em que cada polinômio é escolhido segundo a distribuição $\mathcal{B}_\eta\left(R_q\right)$.

Definição 2 (Problema MLWE computacional)**.** Sorteie uma matriz $\mathbf{A}$ aleatoriamente de forma uniforme de $R_q^{m\times k}$. Escolha um vetor $\mathbf{s} \in R_q^k$ segundo a distribuição $\mathcal{B}_\eta\left(R_q^k\right)$ e outro vetor $\mathbf{e} \in R_q^m$ de acordo com $\mathcal{B}_\eta\left(R_q^m\right)$. Compute $\mathbf{b} = \mathbf{A}\mathbf{s} + \mathbf{e}$. O problema MLWE computacional de parâmetros $(m, k, \mathcal{B}_\eta)$ consiste em, dados $\mathbf{A}$ e $\mathbf{b}$, encontrar o vetor $\mathbf{s}$.

□

Definição 3 (Problema MLWE decisional)**.** $\mathbf{A} \in R_q^{m\times k}$ e um vetor $\mathbf{b} \in R_q^m$. O problema MLWE decisional de parâmetros $(m, k, \mathcal{B}_\eta)$ consiste em distinguir entre os dois casos possíveis.

1. $(\mathbf{A}, \mathbf{b})$ foram tirados de forma uniforme dos conjuntos $R_q^{m\times k}$ e R_q^m, respectivamente.
2. $\mathbf{A}$ foi escolhida de forma uniforme de $R_q^{m\times k}$ mas $\mathbf{b} = \mathbf{A}\mathbf{s} + \mathbf{e}$, onde $\mathbf{s}$ e $\mathbf{e}$ foram escolhidos segundo $\mathcal{B}\left(R_q^k\right)$ e $\mathcal{B}\left(R_q^m\right)$, respectivamente.

□

1.5.3. Um protótipo de esquema criptográfico baseado no MLWE

Suponha que o MLWE é intratável em sua versão de decisão, para o anel R_q. Pode-se argumentar que a ideia mais natural para usá-lo para definir um esquema criptográfico começa com usar vetores $(\mathbf{s}, \mathbf{e})$ como a chave secreta e o par $(\mathbf{A}, \mathbf{t})$ como a chave pública, e dessa forma seria intratável obter a chave secreta a partir da chave pública.

Geração de chaves: Escolha dois vetores $\mathbf{s}$ e $\mathbf{e}$ de acordo com as distribuições $\mathcal{B}_\eta\left(R_q^k\right)$, respectivamente. Sorteie uma matriz $\mathbf{A}$, de dimensões $k \times k$, formada por polinômios de R_q selecionados aleatoriamente. Compute $\mathbf{t} = \mathbf{A}\mathbf{s} + \mathbf{e}$. Então as chaves pública e privada são os pares $(\mathbf{A}, \mathbf{t})$ e $(\mathbf{s}, \mathbf{e})$, respectivamente.

Encriptação: Dada uma chave pública $(\mathbf{A}, \mathbf{t})$, a ideia da encriptação é criar instâncias do MLWE a partir de $\mathbf{A}$ e $\mathbf{t}$ que, quando combinadas com a chave secreta, permitem que a mensagem seja recuperada.

Seja $\mathbf{m}$ uma mensagem binária que queremos encriptar. Denote por $\text{Encode}(\mathbf{m})$ a codificação de $\mathbf{m}$ num polinômio de R_q da seguinte forma:

$$\text{Encode}(\mathbf{m})[i] = \begin{cases} 0, & \text{se } \mathbf{m}[i] = 0, \text{ e} \\ \lceil q/2 \rceil, & \text{se } \mathbf{m}[i] = 1. \end{cases}$$

A ideia da codificação é que seja possível recuperar $\mathbf{m}$ mesmo que o polinômio codificado sofra corrupções de tamanho pequeno em relação a q. A decodificação de um polinômio c é feita por $\text{Decode}(c)$ que verifica se cada coeficiente do polinômio c está mais próximo de 0 ou de $q/2 \mod q$, e decodifica para o bit correspondente. Na classe `Kyber` que definiremos na próxima seção, implementamos a codificação e decodificação de mensagens da seguinte forma.

```
    def encode_message(self, message):
        return vector(self.ntt_ring.field,
                      [ceil(bit * self.params.q/2) for bit in message])

    def decode_message(self, noisy_message):
        def decode_coeff(c):
            if abs(int(mod_centered(c, self.params.q))) <= self.params.q//4:
                return 0
            return 1
        return vector(self.ntt_ring.field, [decode_coeff(c) for c in noisy_message])
```

Pela hipótese da intratabilidade do MLWE decisional, $\mathbf{t}$ é indistinguível de um vetor aleatório de R_q^k mesmo para quem conhece $\mathbf{A}$. Então podemos usar $\mathbf{t}$ para gerar uma instância difícil do MLWE e esconder o polinômio $\text{Encode}(\mathbf{m})$ da seguinte forma. Escolha $\mathbf{r} \in R_q^k$ e $e_2 \in R_q$ segundo as distribuições $\mathcal{B}(R_q^k)$ e $\mathcal{B}(R_q)$, respectivamente. Como o polinômio $z = \langle \mathbf{t}, \mathbf{r} \rangle + e_2$ é uma instância do MLWE com segredo $\mathbf{r}$, então z é indistinguível de um polinômio aleatório de R_q. Portanto z pode ser usado para esconder a mensagem codificada obtendo $v = \text{Encode}(\mathbf{m}) + z$.

Note que, pela forma como $\mathbf{t}$ é definido, o polinômio v pode ser escrito como

$$\begin{aligned} v = \text{Encode}(\mathbf{m}) + z &= \text{Encode}(\mathbf{m}) + \langle \mathbf{t}, \mathbf{r} \rangle + e_2 \\ &= \text{Encode}(\mathbf{m}) + \langle \mathbf{A}\mathbf{s} + \mathbf{e}, \mathbf{r} \rangle + e_2 \\ &= \text{Encode}(\mathbf{m}) + \langle \mathbf{A}\mathbf{s}, \mathbf{r} \rangle + \langle \mathbf{e}, \mathbf{r} \rangle + e_2 \\ &= \text{Encode}(\mathbf{m}) + \left\langle \mathbf{s}, \mathbf{A}^\top \mathbf{r} \right\rangle + \langle \mathbf{e}, \mathbf{r} \rangle + e_2. \end{aligned}$$

O polinômio $\langle \mathbf{e}, \mathbf{r} \rangle + e_2$ tem coeficientes pequenos em relação a q, portanto a principal dificuldade de recuperar $\mathbf{m}$ a partir de v vem dos coeficientes grandes relativos à parcela $\left\langle \mathbf{s}, \mathbf{A}^\top \mathbf{r} \right\rangle$. Queremos então enviar uma dica, digamos $\mathbf{u}$, que permita ao destinatário remover a parcela densa da encriptação, e recuperar a mensagem $\mathbf{m}$. O principal desafio é que a dica $\mathbf{u}$ só deve ser útil para aquele que souber a chave secreta $\mathbf{s}$. Intuitivamente, uma boa solução seria um vetor $\mathbf{u}$ tal que $\langle \mathbf{s}, \mathbf{u} \rangle \approx \left\langle \mathbf{s}, \mathbf{A}^\top \mathbf{r} \right\rangle$.

Uma solução possível é tomar $\mathbf{u} = \mathbf{A}^\top \mathbf{r} + \mathbf{e}_1$, onde $\mathbf{e}_1$ é selecionado aleatoriamente segundo a distribuição $\mathcal{B}(R_q^k)$. Dessa forma, sob a intratabilidade do MLWE, o vetor $\mathbf{u}$

não revela informação sobre $\mathbf{r}$, e temos que

$$\langle \mathbf{s}, \mathbf{u} \rangle = \left\langle \mathbf{s}, \mathbf{A}^\top \mathbf{r} \right\rangle + \langle \mathbf{s}, \mathbf{e}_1 \rangle ,$$

onde $\langle \mathbf{s}, \mathbf{e}_1 \rangle$ é um vetor de polinômios de coeficientes pequenos.

Temos então que o par $(\mathbf{u}, z)$ contém $k+1$ amostras do MLWE relativo ao segredo $\mathbf{r}$. Supondo a intratabilidade do MLWE com parâmetros $(m, k+1, \mathcal{B}_\eta)$, esses valores são indistinguíveis de elementos aleatórios de $R_q^k \times R_q$. Portanto o par $(\mathbf{u}, v)$ encripta a mensagem $\mathbf{m}$.

Decriptação: Para a decriptação do texto encriptado $(\mathbf{u}, v)$, combinamos a chave secreta $\mathbf{s}$ com $\mathbf{u}$ e subtraímos o polinômio resultante de resultado de v, obtendo

$$\hat{c} = v - \langle \mathbf{s}, \mathbf{u} \rangle = \text{Encode}(\mathbf{m}) + \langle \mathbf{e}, \mathbf{r} \rangle - \langle \mathbf{s}, \mathbf{e}_1 \rangle + e_2.$$

Note como $\hat{c}$ é a mensagem codificada somada a um polinômio de erro dado por $\langle \mathbf{e}, \mathbf{r} \rangle - \langle \mathbf{s}, \mathbf{e}_1 \rangle + e_2$.

Para garantir que a decodificação seja possível sem erros, é preciso escolher os parâmetros q e η de forma que os coeficientes do polinômio de erro sejam pequenos em relação a q, porém tomando cuidado para manter o MLWE intratável.

1.5.4. Implementação do Kyber

O Kyber [Avanzi et al. 2019] consiste numa instanciação eficiente do esquema definido acima, com algumas otimizações. Em alto nível, o Kyber propõe 3 otimizações sobre o esquema protótipo que apresentamos

1. Definição de um anel polinomial R_q em que a NTT pode ser computada rapidamente, o que permite a multiplicação eficiente de polinômios de R_q.
2. A matriz pública $\mathbf{A} \in R_q^{k \times k}$ é representada por uma semente. Isso faz com que não sejam necessários $nk^2 \log_2(q)$ bits para representar a matriz. Além disso, a matriz $\mathbf{A}$ é calculada diretamente no domínio da NTT, já que ela só é usada para multiplicações.
3. O texto cifrado é comprimido $(\mathbf{u}, v)$ com perda, porém mantendo a probabilidade de falha de decriptação em patamares seguros.

Seleção de parâmetros e inicialização: Para cada nível de segurança, o Kyber usa um conjunto de parâmetros $(q, n, k, \eta_1, \eta_2, d_\mathbf{u}, d_v)$ diferentes. Já sabemos que o primo $q = 3229$ e $n = 256$ são fixos para todos os níveis. O parâmetro k representa a dimensão do módulo R_q^k usado. Os parâmetros η_1 e η_2 correspondem aos parâmetros de dispersão das binomiais centradas usadas para gerar instâncias do MLWE. Os parâmetros $d_\mathbf{u}$ e d_v podem ser vistos como índices de compressão dos elementos $\mathbf{u}$ e v que compõem o texto encriptado.

A tabela de parâmetros e a inicialização do Kyber são mostradas no código abaixo.

```
@dataclass
class KyberParameters:
    q: int; n: int; k: int; eta1: int; eta2: int; du: int; dv: int;

class KyberPKE_CPA():
    SecurityParameters = {
        1: KyberParameters(q=3329, n=256, k=2, eta1=3, eta2=2, du=10, dv=4),
        3: KyberParameters(q=3329, n=256, k=3, eta1=2, eta2=2, du=10, dv=4),
        5: KyberParameters(q=3329, n=256, k=4, eta1=2, eta2=2, du=11, dv=5),
    }

    def __init__(self, security_level):
        self.params = self.SecurityParameters[security_level]
        self.ntt_ring = KyberNTTRing()
```

Geração de chaves: A geração de chaves é essencialmente igual à do protótipo mostrado na seção anterior. Primeiro escolha aleatoriamente a matriz $\mathbf{A}$ de $R_q^{k\times k}$, e vetores secretos $\mathbf{s}$ e $\mathbf{r}$ de acordo com $\mathcal{B}_{\eta_1}\left(R_q^k\right)$. Compute $\mathbf{t} = \mathbf{A}\mathbf{s} + \mathbf{t}$. As chaves pública e secreta são $\mathbf{pk} = (\mathbf{A}, \mathbf{t})$ e $\mathbf{sk} = (\mathbf{s})$, respectivamente.

A primeira diferença é que toda a aleatoriedade da geração de chaves parte de uma semente d de 256 bits selecionada aleatoriamente. Com base nessa semente e com o uso de XOFs e hashes criptográficos, todas as amostragens são feitas de forma determinística. A vantagem é que, para recriar o par de chaves, basta armazenar a semente d.

Há porém, diferenças mais importantes, principalmente em relação à representação da matriz $\mathbf{A}$ no domínio da NTT. Note que a matriz $\mathbf{A}$, tanto na geração de chaves quanto na encriptação, só é usada na multiplicação. Podemos então gerá-la diretamente no domínio da NTT, e como a NTT é uma função bijetora, gerar uma matriz $\hat{\mathbf{A}}$ uniforme no domínio da NTT é equivalente a gerar uma matriz $\mathbf{A}$ uniformemente de $R_q^{k\times k}$. Outra decisão importante é representar a matriz $\mathbf{A}$ através de uma semente pública de 256 bits, o que diminui a representação da chave pública ao custo da recomputação de $\mathbf{A}$ durante a encriptação.

Antes de definir a função de geração de chaves, considere as funções que fazem a amostragem da binomial centrada em vetores e polinômios, que denotamos por $\mathcal{B}\left(R_q^k\right)$ e $\mathcal{B}\left(R_q\right)$, respectivamente.

```
    def sample_polynomial_from_cbd(self, eta, xof):
        coeffs = [xof_centered_binomial_sample(xof, eta) for i in range(self.params.n)]
        return vector(self.ntt_ring.field, coeffs)

    def sample_vector_from_cbd(self, eta, xof):
        v = [self.sample_polynomial_from_cbd(eta, xof) for i in range(self.params.k)]
        return self.to_poly_vec(v)
```

Então a geração de chaves do Kyber pode ser implementada da seguinte forma. Note como toda a aleatoriedade da função de fato vem de uma variável d que é inicializada com 256 bits realmente aleatórios. Um hash G de 512 bits é aplicado sobre d de forma a obter dois valores de 256 bits, chamados ρ e σ. A variável ρ pode ser vista como a semente pública, que será usada para gerar $\mathbf{A}$, enquanto σ é semente secreta, usada para gerar os vetores secretos $\mathbf{s}$ e $\mathbf{e}$.

```
    def keygen(self):
        d = get_random_bytes(32)
        hash_g_of_d = HASH_G.new(d).digest()
```

```
        rho, sigma = hash_g_of_d[:16], hash_g_of_d[16:]
        ntt_A = self.get_ntt_A_from_seed(rho)

        sigma_xof = PRF.new(sigma)
        s = self.sample_vector_from_cbd(self.params.eta1, sigma_xof)
        e = self.sample_vector_from_cbd(self.params.eta1, sigma_xof)
        ntt_s = s.ntt()
        ntt_e = e.ntt()

        ntt_t = self.matrix_poly_vec_product(ntt_A, ntt_s) + ntt_e
        return (ntt_t, rho), (ntt_s)
```

Note como é necessário computar a NTT de **s** e **e** para poder fazer o produto e obter o vetor **t** no domínio da NTT. Note também como a função devolve o vetor público **t** e a chave secreta **s** no domínio da NTT, pois, durante a encriptação e decriptação, eles são usados somente em multiplicações.

Encriptação: Considere a função de encriptação em Sage abaixo.

```
    def encrypt(self, pk, message, randomness):
        (ntt_t, rho) = pk

        ntt_A_transpose = self.get_ntt_A_from_seed(rho, transpose=True)

        rand_vectors_xof = PRF.new(randomness)
        r = self.sample_vector_from_cbd(self.params.eta1, rand_vectors_xof)
        e1 = self.sample_vector_from_cbd(self.params.eta2, rand_vectors_xof)
        e2  = self.sample_polynomial_from_cbd(self.params.eta2, rand_vectors_xof)
        ntt_r = r.ntt()
        u = self.matrix_poly_vec_product(ntt_A_transpose, ntt_r).inv_ntt() + e1

        encoded_message = self.encode_message(message)
        v = encoded_message + self.ntt_ring.inv_ntt(ntt_t * ntt_r) + e2

        u_compressed = [self.compress(u_i, self.params.du) for u_i in u]
        v_compressed = self.compress(v, self.params.dv)
        return (u_compressed, v_compressed)
```

Diferente do algoritmo de encriptação apresentado no protótipo, a encriptação do Kyber deve ser determinística, com a fonte da aleatoriedade usada na função sendo derivada da semente passada como argumento pela variável `randomness`.

A função começa destrinchando a chave pública em $\hat{\mathbf{t}}$ e ρ, e computando a matriz $\hat{\mathbf{A}}$ através de ρ. Depois, são gerados $\mathbf{r}, \mathbf{e}_1$ e e_2 segundo as distribuições $\mathcal{B}_{\eta_1}\left(R_q^k\right)$, $\mathcal{B}_{\eta_2}\left(R_q^k\right)$ e $\mathcal{B}_{\eta_2}\left(R_q\right)$, respectivamente. Em seguida $\mathbf{u}$ e v são computados da mesma forma, porém no domínio da NTT, isto é

$$\mathbf{u} = \mathbf{NTT}^{-1}\left(\mathbf{NTT}\left(\mathbf{A}\right) \cdot \mathbf{NTT}\left(\mathbf{r}\right)\right) + \mathbf{e}_1,$$
$$v = \text{Encode}\left(\mathbf{m}\right) + \mathbf{NTT}^{-1}\left(\left\langle \mathbf{NTT}\left(\mathbf{t}\right), \mathbf{NTT}\left(\mathbf{r}\right)\right\rangle\right) + e_2.$$

A diferença crítica está na compressão de $\mathbf{u}$ e v. Após a compressão com perda de informação, cada coeficiente de cada polinômio de $\mathbf{u}$ passa a ser representado por $d_{\mathbf{u}}$ bits, enquanto cada coeficiente do polinômio v passa a ser representado por d_v bits.

Formalmente, a compressão e decompressão de um coeficiente c são dadas por

$$\textbf{Compress}_d(c) = \left\lceil \frac{c2^d}{q} \right\rfloor \bmod 2^d,$$
$$\textbf{Decompress}_d(\hat{c}) = \left\lceil \frac{\hat{c}q}{2^d} \right\rfloor.$$

Implementamos as funções de compressão e decompressão de polinômios da seguinte forma.

```
    def compress(self, coefficients, d):
        def compress_coefficient(x, d):
            return mod(round(2**d / self.params.q * int(x)), 2**d)
        return [compress_coefficient(c, d) for c in coefficients]

    def decompress(self, coefficients, d):
        def decompress_coefficient(x, d):
            return round(self.params.q / 2**d * int(x))
        return vector(self.ntt_ring.field,
                      [decompress_coefficient(c, d) for c in coefficients])
```

Note que, como a compressão é com perda, após a decompressão pode haver um erro cujo tamanho depende do fator de compressão usado. Então, embora a compressão seja muito útil para gerar textos encriptados menores, é importante balancear os fatores de compressão para manter baixa a probabilidade de falha de decriptação.

Decriptação: A decriptação também é muito similar à decriptação mostrada no protótipo, exceto pela decompressão do texto encriptado feita logo no início da função.

```
    def decrypt(self, sk, ciphertext):
        ntt_s = sk

        u_compressed, v_compressed = ciphertext
        u = self.to_poly_vec([self.decompress(u_i, self.params.du)
                              for u_i in u_compressed])
        v = self.decompress(v_compressed, self.params.dv)

        ntt_u = u.ntt()
        noisy_message = v - self.ntt_ring.inv_ntt(ntt_s * ntt_u)

        return self.decode_message(noisy_message)
```

1.6. Crystals-Dilithium

O Dilithium [Bai et al. 2021] é um esquema de assinatura cuja segurança é baseada na dificuldade dos problemas *learning with errors* (LWE) e *short integer solution* (SIS) em módulos, denotados por MLWE e MSIS, respectivamente. O esquema é construído segundo a heurística Fiat-Shamir, em que uma prova de conhecimento interativa é transformada num esquema de assinatura.

Nesta seção, apresentamos o Dilithium de forma construtiva da seguinte forma. Primeiro definimos o módulo-SIS (MSIS), que, juntamente com o MLWE já apresentado na seção anterior, é um problema em que se apoia a segurança do esquema. Em seguida, mostramos um protótipo de assinatura baseada em provas de conhecimento, explicando passo a passo como a transformação de Fiat-Shamir é aplicada. Após essa construção,

que forma a base do Dilithium, discutimos quais modificações são necessárias para a implementação eficiente. Por fim, apresentamos a implementação do Dilithium.

1.6.1. O anel polinomial usado pelo Dilithium

O Dilithium usa um mesmo anel polinomial $R_q = \mathbb{F}_q/(x^n+1)$, com $q = 8380417$ e $n = 256$, para todos os seus níveis de segurança. Neste caso, a raiz primitiva da unidade $\omega = 1753$ tem ordem 512 em $\mathbb{F}_q$, o que significa que podemos quebrar o polinômio $x^{256}+1$ em 256 polinômios de grau 1 da forma

$$x^{256}+1 = \prod_{i=0}^{255}\left(x-\omega^{2i+1}\right) = \prod_{i=0}^{255}\left(x-\omega^{2\texttt{bit\_rev}(i,8)+1}\right).$$

De forma análoga ao Kyber, podemos implementar o anel R_q usado pelo Dilithium com o seguinte código.

```
class DilithiumNTTRing():

    def __init__(self):
        self.field = GF(8380417)
        self.n = 256
        self.omega_2n = self.field(1753)

    def ntt(self, a):
        return ntt_leveled_negacyclic_256(a, self.field, self.omega_2n, 8)

    def inv_ntt(self, a_hat):
        return inv_ntt_leveled_negacyclic_256(a_hat, self.field, self.omega_2n, 8)

    def poly_product_ntt_domain(self, a_hat, b_hat):
        return vector(self.field, [c_a * c_b for c_a, c_b in zip(a_hat, b_hat)])
```

Note como a NTT rápida pode ser calculada completamente em 8 níveis. Além disso, o produto de polinômios no domínio da NTT é mais simples do que aquele do Kyber, pois os resíduos são números, não polinômios de grau 1. Isso faz com que o produto seja simplesmente a multiplicação em $\mathbb{F}_q$.

1.6.2. Os problemas usados pelo Dilithium

A segurança do Dilithium é baseada em dois problemas: o MLWE e o MSIS. O MLWE é responsável por garantir a segurança da chave secreta, enquanto o MSIS garante a dificuldade em forjar assinaturas. Lembre que, na Seção 1.5.2, é introduzido o MLWE usado pelo Kyber. A variante do problema usada pelo Dilithium é muito similar, exceto pela distribuição de vetores pequenos, que no lugar da binomial centrada é a uniforme.

Definição 4 (Problema MLWE computacional para o Dilithium)**.** Considere o conjunto S_η de polinômios de R_q cuja norma infinito é menor ou igual a η. Formalmente

$$S_\eta = \left\{s \in R_q : \|s\|_\infty \leq \eta\right\}.$$

Sorteie uma matriz $\mathbf{A}$ aleatoriamente de forma uniforme de $R_q^{k\times\ell}$. Escolha dois vetores $\mathbf{s}_1$ e $\mathbf{s}_2$ aleatoriamente de forma uniforme dos módulos S_η^ℓ e S_η^k, respectivamente. Compute $\mathbf{t} = \mathbf{A}\mathbf{s}_1 + \mathbf{s}_2$. O problema MLWE computacional de parâmetros (m,k,η) consiste em, dados $\mathbf{A}$ e $\mathbf{t}$, encontrar o vetor $\mathbf{s}_1$.

□

O problema MSIS consiste em encontrar soluções pequenas de um sistema linear homogêneo em módulos. Este problema é definido formalmente abaixo.

Definição 5 (Problema MSIS computacional)**.** Seja $\mathbf{A}$ uma matriz aleatória tomada uniformemente de $R_q^{m\times k}$. O problema MSIS computacional na forma normal com parâmetros (m,k,γ) consiste em encontrar $\mathbf{y} \in R_q^{(m+k)}$ tal que $\|\mathbf{y}\|_\infty \leq \gamma$ e $[\,\mathbf{I} \mid \mathbf{A}\,]\,\mathbf{y} = \mathbf{0}$. □

1.6.3. Assinatura baseada em provas de conhecimento

Suponha que sabemos a solução de uma instância do MLWE. Isto é, seja $\mathbf{A} \in R_q^{k\times\ell}$ uma matriz de polinômios e $\mathbf{t} \in R_q^{\ell}$, nós temos em mãos vetores pequenos $\mathbf{s}_1 \in R_q^{\ell}$ e $\mathbf{s}_2 \in R_q^{k}$ tais que

$$\mathbf{A}\mathbf{s}_1 + \mathbf{s}_2 = \mathbf{t}.$$

Dado o par $(\mathbf{A},\mathbf{t})$, uma prova de conhecimento de $(\mathbf{s}_1,\mathbf{s}_2)$ é uma construção criptográfica que satisfaz as seguintes propriedades aparentemente contraditórias:

1. convence um verificador que de fato conhecemos uma solução $(\mathbf{s}_1,\mathbf{s}_2)$;
2. não permite que o verificador aprenda nenhuma informação sobre $\mathbf{s}_1$.

A solução tipicamente usada para esse problema é usar uma prova interativa entre o provador e o verificador. A interação consiste em o verificador fazer perguntas que devem ser respondidas pelo provador de forma satisfatória.

Nesta seção, vamos mostrar como construir uma prova de conhecimento de uma solução do MLWE. Depois, mostramos como esta construção pode ser transformada num esquema de assinatura digital por meio da transformada de Fiat-Shamir.

Conjuntos e funçõs auxiliares: Na prova interativa, vamos precisar de dois conjuntos de polinômios pequenos definidos a seguir.

Dado um inteiro $\gamma \geq 0$, definimos o conjunto S_γ como o conjunto de polinômios em R_q de norma infinito no máximo γ. Formalmente

$$S_\gamma = \{w \in R_q : \|w\|_\infty \leq \gamma\}.$$

Para um inteiro $\tau \geq 0$, o conjunto $B_\tau \subset R_q$ contém os polinômios com coeficientes em $\{-1,0,1\}$ mas com apenas τ coeficientes diferentes de 0. Isto é

$$B_\tau = \left\{ w = w_0 + \ldots + w_{n-1}x^{n-1} \in R_q : -1 \leq w_i \leq 1 \text{ e } \sum_{i=0}^{n-1} |w_i| = \tau \right\}.$$

Protótipo de prova de conhecimento: A Figura 1.4 mostra o protótipo da prova de conhecimento. Note que, embora útil para entender a construção do esquema, este protótipo é inseguro, como veremos adiante. Neste protocolo, o provador tenta convencer o

verificador de que ele sabe uma solução $(\mathbf{s}_1, \mathbf{s}_2)$ do MLWE tal que $\mathbf{t} = \mathbf{A}\mathbf{s}_1 + \mathbf{s}_2$, para uma matriz $\mathbf{A} \in R_q^{k \times \ell}$ e um vetor $\mathbf{t} \in R_q^{\ell}$.

O provador começa selecionando um vetor pequeno $\mathbf{y}$ chamado máscara, e computando $\mathbf{w} = \mathbf{A}\mathbf{y}$. Note que $\mathbf{w}$ é um vetor denso e é possível provar que, quando γ_1 é suficientemente grande, $\mathbf{w}$ tem distribuição indistinguível da uniforme sobre R_q^k, e portanto não revela nada sobre $\mathbf{y}$ [Peikert 2015, Seção 4.1.1]. O vetor $\mathbf{w}$ é então enviado ao verificador.

Ao receber $\mathbf{w}$, o verificador gera um polinômio uniformemente aleatório de B_τ, chamado desafio, e envia-o ao provador. O provador então computa $\mathbf{z} = \mathbf{y} + c\mathbf{s}_1$ e envia-o ao verificador. O protocolo termina com o verificador realizando duas verificações. A primeira é se o vetor $\mathbf{z}$ recebido de fato é pequeno. A segunda é se o vetor $\mathbf{w}$ recebido no protocolo próximo de $\mathbf{A}\mathbf{z} - c\mathbf{t}$. Se as duas verificações passarem, o verificador aceita a prova. Caso contrário, a prova é rejeitada.

Provador		**Verificador**
$\mathbf{y} \leftarrow \textbf{Uniforme}\left(S_{\gamma_1}^{\ell}\right)$		
$\mathbf{w} \leftarrow \mathbf{A}\mathbf{y}$	$\xrightarrow{\mathbf{w}}$	
	$\xleftarrow{c}$	$c \leftarrow \textbf{Uniforme}\,(B_\tau)$
$\mathbf{z} \leftarrow \mathbf{y} + c\mathbf{s}_1$	$\xrightarrow{\mathbf{z}}$	Aceite se $\|\mathbf{z}\|_\infty \leq (\gamma_1 + \tau\eta)$ e $\|\mathbf{w} - (\mathbf{A}\mathbf{z} - c\mathbf{t})\|_\infty \leq \tau\eta$
		Caso contrário, rejeite

Figura 1.4. Protótipo inseguro da prova de conhecimento interativa ainda pois a resposta final vaza informação sobre o segredo.

Primeiro, vejamos que alguém honesto e que sabe o segredo $\mathbf{s}_1$ consegue passar na prova. Neste caso, o provador computou honestamente $\mathbf{w} = \mathbf{A}\mathbf{y}$ e $\mathbf{z} = \mathbf{y} + c\mathbf{s}_1$. Note que, como a soma dos valores absolutos dos coeficientes de c é τ, vale que $\|c\mathbf{s}_1\|_\infty \leq \tau\eta$. Então, para a primeira verificação temos que

$$\|\mathbf{z}\|_\infty = \|\mathbf{y} + c\mathbf{s}_1\|_\infty \leq \|\mathbf{y}\|_\infty + \|c\mathbf{s}_1\|_\infty \leq \gamma_1 + \tau\eta.$$

Para a segunda verificação, observamos que

$$\begin{aligned}
\mathbf{A}\mathbf{z} - c\mathbf{t} &= \mathbf{A}(\mathbf{y} + c\mathbf{s}_1) - c\mathbf{t} \\
&= \mathbf{A}(\mathbf{y}) + \mathbf{A}(c\mathbf{s}_1) - c\mathbf{t} \\
&= \mathbf{w} + c(\mathbf{t} - \mathbf{s}_2) - c\mathbf{t} \\
&= \mathbf{w} - c\mathbf{s}_2.
\end{aligned}$$

Portanto, vale que $\mathbf{w} - (\mathbf{A}\mathbf{z} - c\mathbf{t}) = c\mathbf{s}_2$, que é um vetor de norma $\|c\mathbf{s}_2\|_\infty \leq \tau\eta$.

Considere o desempenho de um falso provador que não sabe $(\mathbf{s}_1, \mathbf{s}_2)$. Suponha que, após ele enviar a mensagem inicial $\mathbf{w}$, há somente um desafio c que ele sabe responder corretamente. Isso pode ocorre, por exemplo, quando o falso provador observou uma

interação passada do provador real. Para proteger o esquema contra esse tipo de ataque, basta que o conjunto B_τ seja grande o suficiente para que probabilidade de um mesmo desafio ser gerado duas vezes seja desprezável.

Suponha então que o falso provador, após enviar $\mathbf{w}$, é capaz de responder pelo menos 2 desafios diferentes c_1 e c_2, com respostas válidas $\mathbf{z}_1$ e $\mathbf{z}_2$. Neste caso, devido à validade das respostas, temos que

$$\mathbf{w} = \mathbf{A}\mathbf{z}_1 - c_1\mathbf{t} + \mathbf{e}_1 = \mathbf{A}\mathbf{z}_2 - c_2\mathbf{t} + \mathbf{e}_2,$$

onde $\mathbf{e}_1$ e $\mathbf{e}_2$ são vetores pequenos de norma infinito menor ou igual a $\tau\eta$. Portanto, vale que

$$\begin{gathered}\mathbf{A}(\mathbf{z}_1 - \mathbf{z}_2) + (c_2 - c_1)\mathbf{t} + (\mathbf{e}_1 - \mathbf{e}_2) = \mathbf{0} \\ \left[\, \mathbf{I} \mid \mathbf{A} \mid \mathbf{t} \,\right] \left[\begin{array}{c} \mathbf{e}_1 - \mathbf{e}_2 \\ \hline (\mathbf{z}_1 - \mathbf{z}_2)^\top \\ \hline c_1 - c_2 \end{array}\right] = \mathbf{0},\end{gathered}$$

o que implica que o falso provador consegue resolver o MSIS para a matriz $[\, \mathbf{I} \mid \mathbf{A} \mid \mathbf{t} \,]$.

Embora a dificuldade em dar uma resposta válida seja garantida pela dificuldade do MSIS, e a dificuldade em quebrar a chave secreta a partir da pública seja protegida pelo MLWE, há ainda dois problemas muito sérios no protocolo apresentado. O primeiro é que o verificador consegue computar $c\mathbf{s}_2 = \mathbf{w} - (\mathbf{A}\mathbf{z} - c\mathbf{t})$, e portanto pode obter o vetor secreto $\mathbf{s}_2$ com uma simples divisão de polinômio, e usá-lo para obter $\mathbf{s}_1$.

O segundo problema é que o vetor $\mathbf{z}$ é muito dependente do vetor secreto $\mathbf{s}_1$ pois a máscara $\mathbf{y}$ usada para ofuscá-lo é um polinômio pequeno. Então, observando um número suficientemente grande de interações, um atacante poderia ser capaz de obter alguma informação sobre $\mathbf{s}_1$.

Protegendo $\mathbf{s}_1$ com amostragem por rejeição: Queremos garantir que o verificador, que conhece c e $\mathbf{z}$, não consiga obter nenhuma informação sobre $\mathbf{s}_1$ e $\mathbf{s}_2$. Primeiramente, considere o caso de $\mathbf{s}_1$. Formalmente, queremos que $\Pr(\mathbf{s}_1)$ seja independente de $(c, \mathbf{z})$, isto é, que $\Pr(\mathbf{s}_1 | c, \mathbf{z}) = \Pr(\mathbf{s}_1)$.

Fixe um segredo $\mathbf{s}_1$, e considere a distribuição de probabilidade conjunta $\Pr(c, \mathbf{z})$

$$\begin{aligned}\Pr(c, \mathbf{z}) &= \Pr(\mathbf{z} \mid c)\Pr(c) \\ &= \Pr_{\mathbf{y}}(\mathbf{z} = \mathbf{y} + c\mathbf{s}_1 \mid c)\Pr(c) \\ &= \Pr_{\mathbf{y}}(\mathbf{y} = \mathbf{z} - c\mathbf{s}_1 \mid c)\Pr(c).\end{aligned}$$

Como $\mathbf{y}$ é um vetor uniformemente aleatório de $S^\ell_{\gamma_1}$, então se a condição $(\mathbf{z} - c\mathbf{s}_1) \in S^\ell_{\gamma_1}$ fosse satisfeita, valeria que

$$\Pr_{\mathbf{y}}(\mathbf{y} = \mathbf{z} - c\mathbf{s}_1 \mid c) = \frac{1}{|S^\ell_{\gamma_1}|},$$

que caracteriza uma distribuição uniforme sobre S_{γ_1}. Logo, se o provador abortar e recomeçar o protocolo sempre que $(\mathbf{z} - c\mathbf{s}_1) \notin S^{\ell}_{\gamma_1}$, então os valores de $(\mathbf{z}, c)$ serão de fato independentes de $\mathbf{s}_1$.

Podemos simplificar a condição de rejeição da seguinte forma. Como $\mathbf{s}_1 \in S^{\ell}_{\eta}$ e $c \in B_\tau$, então $\|c\mathbf{s}_1\|_\infty \leq \eta\tau$. Considerando então a desigualdade triangular sobre $(\mathbf{z} - c\mathbf{s}_1)$, obtemos

$$\|\mathbf{z} - c\mathbf{s}_1\|_\infty \leq \|\mathbf{z}\|_\infty + \|c\mathbf{s}_1\|_\infty \leq \|\mathbf{z}\|_\infty + \eta\tau.$$

Podemos então garantir que $\|\mathbf{z} - c\mathbf{s}_1\|_\infty \leq \gamma_1$ se forçarmos que o lado direito da inequação acima satisfaça

$$\|\mathbf{z}\|_\infty + \eta\tau \leq \gamma_1.$$

Ou seja, se aceitarmos $\|\mathbf{z}\|_\infty$ somente quando $\|\mathbf{z}\|_\infty \leq \gamma_1 - \eta\tau$, garantimos que $\|\mathbf{z} - c\mathbf{s}_1\|_\infty \leq \gamma_1$, e, portanto $\mathbf{z}$ pode ser enviado ao verificador sem vazamento de informação sobre $\mathbf{s}_1$.

Protegendo $\mathbf{s}_2$ com amostragem por rejeição: Considere agora o caso de $\mathbf{s}_2$, cujos coeficientes são vazados pois o verificador conhece todos os termos do lado direito da equação.

$$c\mathbf{s}_2 = \mathbf{w} - (\mathbf{A}\mathbf{z} - c\mathbf{t}).$$

Note que, como o vetor $c\mathbf{s}_2$ revelado tem coeficientes pequenos, a fonte de vazamento de informação está nos bits menos significativos dos coeficientes dos polinômios de $\mathbf{w}$. Intuitivamente, se o provador enviasse somente os bits mais significativos de $\mathbf{w}$, os bits menos significativos de $\mathbf{w}$ tratariam de esconder perfeitamente o vetor $c\mathbf{s}_2$.

Para formalizar esta ideia, vamos definir a função que, dado um parâmetro de α, decompõe um elemento r de $\mathbb{F}_q$ em duas partes r_1 e r_0 tais que

1. $r = r_0 + \alpha r_1$ e
2. $-(\alpha/2) \leq r_0 < (\alpha/2)$.

```
    def decompose(self, r, alpha):
        r = int(r) % self.params.q
        r0 = mod_centered(r, alpha)
        if r - r0 == self.params.q - 1:
            return (0, r0 - 1)
        return (r - r0) // alpha, r0
```

Usando a função `decompose`, podemos implementar funções específicas que devolvem a parte mais significativa r_1 ou a menos r_0.

```
    def high_bits_coefficient(self, r, alpha):
        return self.decompose(r, alpha)[0]

    def low_bits_coefficient(self, r, alpha):
        return self.decompose(r, alpha)[1]
```

Também usaremos a extensão das funções acima para vetores de polinômios, chamadas HighBits e LowBits, que simplesmente aplicam as respectivas funções a cada um dos coeficientes de cada polinômio no vetor.

```
    def high_bits(self, r, alpha):
        v = [[self.high_bits_coefficient(c, alpha) for c in poly] for poly in r]
        return self.to_poly_vec(v)

    def low_bits(self, r, alpha):
        v = [[self.low_bits_coefficient(c, alpha) for c in poly] for poly in r]
        return self.to_poly_vec(v)
```

Suponha então que provador deixe de enviar $\mathbf{w}$ e envie $\mathbf{w}_1 = \text{HighBits}(\mathbf{w}, 2\gamma_2)$ em seu lugar. Considere o valor computado para a verificação

$$\text{HighBits}(\mathbf{Az} - c\mathbf{t}, 2\gamma_2) = \text{HighBits}(\mathbf{w} - c\mathbf{s}_2, 2\gamma_2).$$

Podemos ver que ainda é possível que alguma informação sobre $c\mathbf{s}_2$ vaze para $\text{HighBits}(\mathbf{Az} - c\mathbf{t}, 2\gamma_2)$ por meio dos carregamentos que ocorrem durante a computação de

$$\text{LowBits}(\mathbf{w}, 2\gamma_2) - \text{LowBits}(c\mathbf{s}_2, 2\gamma_2) = \text{LowBits}(\mathbf{w}, 2\gamma_2) - c\mathbf{s}_2.$$

Considere $\mathbf{w}_0 = \text{LowBits}(\mathbf{w}, 2\gamma_2)$. Queremos então garantir que $\Pr(\mathbf{s}_2)$ seja independente de $\Pr(c, \mathbf{w}_0)$, isto é, que $\Pr(\mathbf{s}_2 | c, \mathbf{w}_0) = \Pr(\mathbf{s}_2)$.

Mas isso é exatamente análogo ao que computamos na seção anterior para $\mathbf{s}_1$ e $\mathbf{z}$, com os vetores $\mathbf{s}_2$ e $\mathbf{w}_0$ em seus lugares. Note que, pela definição de LowBits, qualquer vetor em $S_{\gamma_2 - 1}$ é um vetor $\|\mathbf{w}_0\|_\infty$ possível[3]. Portanto, usando o resultado mostrado na seção anterior, podemos garantir que

$$\Pr(\mathbf{s}_2 | c, \mathbf{w}_0) = \Pr(\mathbf{s}_2),$$

se somente aceitarmos valores de $\mathbf{w}_0$ tais que $\|\mathbf{w}_0\|_\infty < \gamma_2 - \eta\tau$. Esta condição pode ser reescrita como

$$\text{LowBits}(\mathbf{Ay} - c\mathbf{s}_2, 2\gamma_2) < \gamma_2 - \eta\tau.$$

Então, caso a condição não seja satisfeita, o provador aborda e recomeça o protocolo com um novo $\mathbf{y}$.

Construindo um esquema de assinatura: Uma assinatura digital pode ser vista como uma demonstração não interativa de que conhecemos ao mesmo tempo a chave secreta e a mensagem a ser assinada. Então, podemos transformar a prova de conhecimento acima num esquema de assinatura se fizermos o desafio $\mathbf{c}$ depender da mensagem e removermos a interação.

Suponha que há uma função de hash H que devolve polinômios de B_τ. A transformação de Fiat-Shamir consiste em eliminar o verificador e computar o desafio $\mathbf{c}$ como $\mathbf{c} = H(\mathbf{w}, \mathbf{m})$, onde $\mathbf{m}$ é a mensagem a ser assinada. Sendo H uma função de hash criptográfica segura, isso garante que seja computacionalmente inviável gerar $\mathbf{c}$ antes de $\mathbf{w}$ e reusar o mesmo par $(\mathbf{c}, \mathbf{w})$ para uma mensagem $\mathbf{m}$ diferente.

Com esta transformação, e adicionando a repetição do protocolo para aceitar vetores $\|\mathbf{z}\|_\infty$ somente quando $\|\mathbf{z}\|_\infty < \gamma_1$, obtemos o esquema de assinatura cuja segurança é baseada na dificuldade dos problemas MLWE e MSIS abaixo.

[3]Note que usamos $\gamma_2 - 1$ pois a $\text{LowBits}(w, 2\gamma_2)$ devolve um coeficiente w_0 no intervalo $-\gamma_2 \leq w_0 < \gamma_2$.

Geração de chaves

$\mathbf{A} \leftarrow \textbf{Uniforme}\left(R_q^{k\times\ell}\right)$

$\mathbf{s}_1 \leftarrow \textbf{Uniforme}\left(S_\eta^k\right) \in R_q^k$

$\mathbf{s}_2 \leftarrow \textbf{Uniforme}\left(S_\eta^\ell\right) \in R_q^\ell$

$\mathbf{t} \leftarrow \mathbf{A}\mathbf{s}_1 + \mathbf{s}_2$

$\mathbf{pk} \leftarrow (\mathbf{A}, \mathbf{t})$

$\mathbf{sk} \leftarrow (\mathbf{s}_1, \mathbf{s}_2)$

Devolva a assinatura$(c, \mathbf{z})$

Assinar $(\mathbf{sk} = (\mathbf{s}_1, \mathbf{s}_2), \mathbf{m})$

Faça :

$\mathbf{y} \leftarrow \textbf{Uniforme}\left(S_{\gamma_1}^\ell\right)$

$\mathbf{w} \leftarrow \mathbf{A}\mathbf{y}$

$\mathbf{w}_1 \leftarrow \text{HighBits}(\mathbf{w}, 2\gamma_2)$

$c \leftarrow \text{Hash}(\mathbf{w}_1, \mathbf{m})$

$\mathbf{z} \leftarrow \mathbf{y} + c\mathbf{s}_1$

Enquanto

1. $\|\mathbf{z}\|_\infty > \gamma_1 - \eta\tau$, ou
2. $\text{LowBits}(\mathbf{A}\mathbf{y} - c\mathbf{s}_2, 2\gamma_2) \geq \gamma_2 - \eta\tau$

Devolva a assinatura$(c, \mathbf{z})$

Verificar $(\mathbf{pk} = (\mathbf{A}, \mathbf{t}), (c, \mathbf{z}))$

$\mathbf{w}_1 \leftarrow \text{HighBits}(\mathbf{A}\mathbf{z} - c\mathbf{t}, 2\gamma_2)$

$c' = \text{Hash}(\mathbf{w}_1, \mathbf{m})$

Aceite se $c' = c$ e $\|\mathbf{z}\|_\infty \leq \gamma_1$

Figura 1.5. Esquema de assinatura usando a transformação de Fiat-Shamir com amostragem por rejeição.

Esse esquema de assinatura é a base do Dilithium. As diferenças principais estão nas várias otimizações para comprimir a chave pública e as assinaturas, além do uso de um anel R_q que admite a computação rápida da NTT completa para a multiplicação de polinômios.

1.6.4. Implementação do Dilithium

O Dilithium consiste na otimização do esquema mais simples apresentado na seção anterior. As otimizações são análogas às usadas no Kyber, como a representação de $\mathbf{A}$ pela semente, o uso da NTT para as operações de multiplicação entre polinômios e compressão das chaves públicas.

Seleção de parâmetros e inicialização: Por conta das várias otimizações que são aplicadas, o Dilithium é usa um grande número de parâmetros para definir cada nível de segurança. Os parâmetros $n = 256$ e $q = 8380417$ que definem o anel R_q são fixos para todos os níveis de segurança. Os parâmetros k e ℓ definem o espaço da matriz pública $\mathbf{A} \in R_q^{k\times\ell}$, enquanto η define o conjunto S_η^ℓ de onde é tirada a chave secreta $\mathbf{s}_1$.

O parâmetro τ define o raio do conjunto B_τ de onde os desafios $\mathbf{c}$ são tirados. O parâmetro d é um índice de compressão da chave pública. O inteiro γ_1 define o conjunto S_{γ_1} de máscaras $\mathbf{y}$, enquanto γ_2 é usada para definir as funções HighBits e LowBits. Finalmente, $\beta = \eta\tau$ e ω são números usados para definir os limites das normas de vetores que são esperados quando a assinatura é válida.

```
@dataclass
class DilithiumParameters:
    q: int; n: int; k: int; l: int; tau: int; eta: int; d: int;
    gamma1: int; gamma2: int; beta: int; omega: int

class Dilithium():
    SecurityParameters = {
        2: DilithiumParameters(
            q=8380417, d=13, tau=39, gamma1=2**17, gamma2=95232,
            n=256, k=4, l=4, eta=2, beta=78, omega=80),
        3: DilithiumParameters(
            q=8380417, d=13, tau=49, gamma1=2**19, gamma2=261888,
            n=256, k=6, l=5, eta=4, beta=196, omega=55),
```

```
        5: DilithiumParameters(
            q=8380417, d=13, tau=60, gamma1=2**19, gamma2=261888,
            n=256, k=8, l=7, eta=2, beta=120, omega=75),
    }

    def __init__(self, security_level):
        self.params = self.SecurityParameters[security_level]
        self.ntt_ring = DilithiumNTTRing()
```

Geração de chaves: A geração de chaves do Dilithium é muito similar à mostrada na Figura 1.5, exceto pela compressão da chave pública. Considere o código da geração de chaves abaixo.

```
    def keygen(self):
        zeta = get_random_bytes(32)
        xof_h = XOF_H.new(zeta)
        rho = xof_h.read(32)
        rho_prime = xof_h.read(64)

        ntt_A = self.get_ntt_A_from_seed(rho)
        s1, s2 = self.expand_S(rho_prime)
        ntt_s1 = s1.ntt()

        t = self.matrix_poly_vec_product(ntt_A, ntt_s1).inv_ntt() + s2
        t1, t0 = self.power2round_poly_vector(t, self.params.d)
        pk_hash = self.hash_H(rho + t1.as_bytes(), 32)

        assert (t1 * 2**self.params.d == t - t0)
        return (rho, t1), (rho, pk_hash, s1, s2, t0)
```

Podemos ver que, como no Kyber, a matriz $\mathbf{A}$ é representada compactamente pela semente e as multiplicações são todas feitas no domínio da NTT. Para a compressão da chave pública, o vetor $\mathbf{t}$ é dividido em dois vetores $\mathbf{t}_1$ e $\mathbf{t}_0$ tais que $\mathbf{t} = \mathbf{t}_1 2^d + \mathbf{t}_0$, sendo que apenas $\mathbf{t}_1$ faz parte da chave pública. A a parte menos significativa $\mathbf{t}_0$ é guardada na chave secreta, pois ela será importante para gerar assinaturas válidas. A função `power2round_poly_vector` que quebra $\mathbf{t}$ em $\mathbf{t}_1$ e $\mathbf{t}_0$ está implementada abaixo.

```
    def power2round(self, r, d):
        r = int(r) % self.params.q
        r0 = mod_centered(r, 2**d)
        return (r - r0)//2**d, r0

    def power2round_poly_vector(self, poly_vector, d):
        poly_vector0 = [[] for _ in poly_vector]
        poly_vector1 = [[] for _ in poly_vector]
        for i, poly in enumerate(poly_vector):
            for r in poly:
                r1, r0 = self.power2round(r, d)
                poly_vector0[i].append(r0)
                poly_vector1[i].append(r1)
        return self.to_poly_vec(poly_vector1), self.to_poly_vec(poly_vector0)
```

Assinatura: Devido à compressão de chave pública usada pelo Dilithium, a assinatura é significativamente mais complicada do que aquela mostrada na Figura 1.5. A assinatura pode ser implementada da seguinte forma.

```
    def sign(self, sk, message_bytes):
        (rho, pk_hash, s1, s2, t0) = sk
        ntt_A = self.get_ntt_A_from_seed(rho)
        mu = self.hash_H(pk_hash + message_bytes, 64)
```

```
5
6           sigma_prime = get_random_bytes(64)
7           s1_ntt, s2_ntt, t0_ntt = s1.ntt(), s2.ntt(), t0.ntt()
8
9           kappa, (z, h) = 0, (None, None)
10          while h == None:
11              y = self.expand_mask(sigma_prime, kappa.to_bytes(32))
12              kappa += self.params.l
13
14              w_c_z = self.get_w_c_z(y, ntt_A, s1_ntt, s2_ntt, mu)
15              if w_c_z:
16                  (w, w1), (c_tilde, c_ntt), z = w_c_z
17                  h = self.get_hints_for_w((w, w1), t0_ntt, s2_ntt, c_ntt)
18
19          return (c_tilde, z, h)
```

Em alto nível, o algoritmo de assinatura segue aquele do esquema apresentado na Figura 1.5. O núcleo da função de assinatura é o laço de rejeições, que garante que a assinatura não vaza informação sobre a chave secreta.

O laço começa da mesma forma que vimos na Figura 1.5, primeiro selecione $\mathbf{y}$ e tente gerar $(\mathbf{w}, c, \mathbf{z})$ válidos. Isto é, vetores fora dos dos intervalos de rejeição. A geração da máscara $\mathbf{y}$ é feita pela função `expand_mask`, que descrita abaixo. Note como é usado um XOF para gerar os inteiros no intervalo de $(-\gamma_1 + 1)$ a γ_1.

```
1       def expand_mask(self, rhoprime, kappa):
2           xof = SHAKE256.new(rhoprime + kappa)
3           y = []
4           min_inclusive, max_exclusive = -self.params.gamma1 + 1, self.params.gamma1 + 1
5           for i in range(self.params.l):
6               y.append([xof_randrange(xof, min_inclusive, max_exclusive)
7                         for j in range(self.params.n)])
8
9           return PolynomialVector(self.ntt_ring, y)
```

O procedimento que tenta gerar $(\mathbf{w}, c, \mathbf{z})$ válidos é implementado pela função `get_w_c_z` abaixo. Note como, ao final da função, as duas condições de rejeição são testadas. Caso qualquer das duas seja inválida, a função devolve `None` para indicar a falha.

```
1       def get_w_c_z(self, y, ntt_A, s1_ntt, s2_ntt, mu):
2           y_ntt = y.ntt()
3
4           w = self.matrix_poly_vec_product(ntt_A, y_ntt).inv_ntt()
5           w1 = self.high_bits(w, 2 * self.params.gamma2)
6
7           c_tilde = self.hash_H(mu + w1.as_bytes(), 32)
8           c = self.sample_in_ball(c_tilde)
9           c_ntt = self.ntt_ring.ntt(c)
10          z = y + (s1_ntt * c_ntt).inv_ntt()
11
12          r0 = self.low_bits(w - (s2_ntt * c_ntt).inv_ntt(), 2 * self.params.gamma2)
13          z_norm_cond = (z.norm_infinity() >= self.params.gamma1 - self.params.beta)
14          r0_norm_cond = (r0.norm_infinity() >= self.params.gamma2 - self.params.beta)
15
16          if z_norm_cond or r0_norm_cond:
17              return None
18
19          return ((w, w1), (c_tilde, c_ntt), z)
```

Exceto pelas multiplicações no domínio da NTT, a única diferença fundamental entre a função acima e o código dentro do laço na Figura 1.5 é em como o polinômio c é representado. Por motivos de eficiência, o Dilithium representa o polinômio $c \in B_\tau$

através da semente $\tilde{c}$ usada para gerá-lo. Dada uma semente $\tilde{c}$, a função `sample_in_ball` instancia um XOF com o SHAKE256. Este XOF é então usado para gerar os τ coeficientes em $\{-1,1\}$ usando uma variação do algoritmo de Fisher-Yates.

```
    def sample_in_ball(self, c_tilde):
        xof = SHAKE256.new(c_tilde)
        c = [0] * 256
        for i in range(256 - self.params.tau, 256):
            j = xof_randrange(xof, 0, i + 1)
            c[i] = c[j]
            c[j] = (-1)**xof_random_bit(xof)

        assert(c.count(1) + c.count(-1) == self.params.tau)
        return c
```

Após encontrados valores $(\mathbf{w}, c, \mathbf{z})$ válidos, entra em ação um procedimento particular do Dilithium que chamamos de `get_hints_for_w`. Este procedimento é responsável por produzir um vetor binário h chamado vetor de dicas, que é necessário para garantir a corretude da verificação usando a chave pública compactada $\mathbf{t}_1$.

Garantindo que a compressão de chaves não afeta a verificação: Primeiro, vejamos como a compressão de chaves dificulta o procedimento de verificação simplificado mostrado na Figura 1.5. Vimos que o primeiro passo da verificação consiste em computar $\mathbf{w}_1 = \text{HighBits}\left(\mathbf{Az} - c\mathbf{t}, 2\gamma_2\right)$.

Porém, a compressão de $\mathbf{t}$ implica que apenas a sua parte mais significativa $\mathbf{t}_1$ seja conhecida pelo verificador. Como ainda não é possível computar $\mathbf{w}_1$, o verificador pode computar o vetor

$$\hat{\mathbf{w}} = \mathbf{Az} - c2^d\mathbf{t}_1 = \mathbf{Az} - c\mathbf{t} + c\mathbf{t}_0 = \mathbf{w} - c\mathbf{s}_2 + c\mathbf{t}_0.$$

Formalmente, o problema enfrentado pelo verificador é que, em geral

$$\mathbf{w}_1 = \text{HighBits}\left(\mathbf{w} - c\mathbf{s}_2\right) \neq \text{HighBits}\left(\hat{\mathbf{w}}\right),$$

por conta do fator $c\mathbf{t}_0$. Uma solução possível seria incluir o vetor $\mathbf{t}_0$, que não precisa ser secreto, na assinatura. Assim, o verificador, poderia remover o fator $c\mathbf{t}_0$ e computar $\mathbf{w}_1$. Porém, isso aumentaria muito o tamanho das assinaturas, e não justificaria a vantagem obtida pela compressão da chave pública. A solução usada no Dilithium consiste em computar um vetor conciso de dicas que permitem que o verificador compute $\mathbf{w}_1$ corretamente.

Antes de mostrarmos como tal vetor de dicas é calculado, considere a seguinte propriedade das funções HighBits e LowBits. Fixe um inteiro r e sejam $r_1 = \text{HighBits}\left(r, 2\gamma_2\right)$ e $r_0 = \text{LowBits}\left(r, 2\gamma_2\right)$, de forma que $r = \gamma_2 r_1 + r_0$, e $\gamma_2 \leq r_0 < \gamma_2$. Suponha que $-\gamma_2 \leq a < \gamma_2$, então

$$\text{HighBits}\left(r+a\right) = \begin{cases} r_1, \text{ se } -\gamma_2 \leq r_0 + a < \gamma_2, \\ r_1 + 1, \text{ se } r_0 + a \geq \gamma_2, \\ r_1 - 1, \text{ se } r_0 + a < -\gamma_2. \end{cases}$$

Podemos interpretar a observação acima da seguinte forma. Se a é um inteiro desconhecido tal que $-\gamma_2 \leq a < \gamma_2$. Se soubermos que $\text{HighBits}\left(r+a, 2\gamma_2\right) \neq$

$\text{HighBits}(r, 2\gamma_2)$, então necessariamente

$$\text{HighBits}(r+a, 2\gamma_2) = \begin{cases} \text{HighBits}(r, 2\gamma_2) + 1, \text{ se } r_0 \geq 0, \\ \text{HighBits}(r, 2\gamma_2) - 1, \text{ se } r_0 < 0. \end{cases}$$

De forma que, se soubermos r, apenas 1 bit nos permite descobrir $\text{HighBits}(r+a, 2\gamma_2)$.

Voltemos então para o caso do Dilithium. Vimos que o verificador conhece $\hat{\mathbf{w}}$. Então, se valesse que $\|c\mathbf{t}_0\|_\infty < \gamma_2$, então poderíamos enviar um vetor $\mathbf{h}$ de bits, onde cada bit está associado a um coeficiente de cada polinômio, que permitiria

$$\text{HighBits}(\hat{\mathbf{w}} - c\mathbf{t}_0, 2\gamma_2) = \mathbf{w}_1.$$

Porém, caso $\|c\mathbf{t}_0\|_\infty \geq \gamma_2$, não seria possível garantir a validade de um tal vetor de dicas. Então, neste caso, recomeçamos o algoritmo de assinatura com um novo vetor $\mathbf{y}$.

O calculo do vetor binário $\mathbf{h}$ de dicas pode ser calculado usando a seguinte função. Quando $\|c\mathbf{t}_0\|_\infty \geq \gamma_2$, o valor `None` é devolvido para indicar que o algoritmo de assinatura deve ser reiniciado.

```
    def get_hints_for_w(self, w_and_w1, t0_ntt, s2_ntt, c_ntt):

        w, w1 = w_and_w1
        ct0 = (t0_ntt * c_ntt).inv_ntt()
        cs2 = (s2_ntt * c_ntt).inv_ntt()
        h = self.make_hint(-ct0, w - cs2 + ct0, 2 * self.params.gamma2)

        n_ones_in_h = sum(p.hamming_weight() for p in h)
        if ct0.norm_infinity() >= self.params.gamma2 or n_ones_in_h > self.params.omega:
            return None

        assert(self.use_hint(h, w - cs2 + ct0, 2 * self.params.gamma2) == w1)

        return h
```

Note que há também uma outra condição de rejeição, que é se o número de entradas não-nulas em $\mathbf{h}$ for maior do que ω. O valor ω é um parâmetro do esquema calculado de forma que `n_ones_in_h` $\leq \omega$ em 99% dos casos. A vantagem de ter um valor ω fixo é que $\mathbf{h}$ pode ser comprimido se forem enviadas apenas as posições das entradas não-nulas.

As funções auxiliares responsáveis por calcular cada um dos bits do vetor de dicas, dados os coeficientes, têm a seguinte implementação.

```
    def make_hint_coefficient(self, z, r, alpha):
        r1 = self.high_bits_coefficient(r, alpha)
        v1 = self.high_bits_coefficient(r + z, alpha)
        return int(r1 != v1)

    def make_hint(self, z, r, alpha):
        v = []
        for poly_z, poly_r in zip(z, r):
            v.append([self.make_hint_coefficient(c_z, c_r, alpha)
                      for c_z, c_r in zip(poly_z, poly_r)])
        return v
```

Verificação de assinatura: A verificação de assinaturas é análoga à verificação do esquema ilustrado na Figura 1.5. Sua implementação é dada a seguir.

```
    def verify(self, pk, message_bytes, signature):
        (rho, t1) = pk
        (c_tilde, z, h) = signature

        ntt_A = self.get_ntt_A_from_seed(rho)

        pk_hash = self.hash_H(rho + t1.as_bytes(), 32)
        mu = self.hash_H(pk_hash + message_bytes, 64)
        c = self.sample_in_ball(c_tilde)

        z_ntt = z.ntt()
        c_ntt = self.ntt_ring.ntt(c)
        t1_ntt = t1.ntt()

        r = self.matrix_poly_vec_product(ntt_A, z_ntt) - ((t1_ntt * c_ntt) * (2**self.
    params.d))
        w1 = self.use_hint(h, r.inv_ntt(), 2 * self.params.gamma2)

        if (z.norm_infinity() >= self.params.gamma1 - self.params.beta):
            return False

        if (c_tilde != self.hash_H(mu + w1.as_bytes(), 32)):
            return False

        if sum(p.hamming_weight() for p in h) > self.params.omega:
            return False

        return True
```

Note que, para recalcular $\mathbf{w}_1$, é chamada a função `use_hint` que usa as dicas da forma como mostramos na seção passada. Esta função é definida abaixo.

```
    def use_hint(self, h, r, alpha):
        v = []
        for poly_h, poly_r in zip(h, r):
            v.append([self.use_hint_coefficient(c_h, c_r, alpha)
                      for c_h, c_r in zip(poly_h, poly_r)])
        return self.to_poly_vec(v)

    def use_hint_coefficient(self, h, r, alpha):
        m = (self.params.q - 1) // alpha
        (r1, r0) = self.decompose(r, alpha)
        if h and r0 > 0:
            return (r1 + 1) % m
        if h and r0 <= 0:
            return (r1 - 1) % m
        return r1
```

1.7. Fundamentos do HQC: códigos lineares

Códigos corretores de erros são muito usados em sistemas de comunicação pois permitirem a recuperação de uma mensagem transmitida mesmo na presença de ruídos causados pelo canal de transmissão. Fixado um alfabeto de comunicação $\mathcal{A}$, que tipicamente será o corpo binário $\mathbb{F}_2$ ou um outro corpo finito $\mathbb{F}_q$. Suponha que queremos transmitir uma mensagem $\mathbf{m} \in \mathcal{A}^k$ de k elementos do nosso alfabeto $\mathcal{A}$. Um bom código corretor de erro deve explicitar dois procedimentos. Primeiro, um algoritmo de codificação, responsável por adicionar redundância a $\mathbf{m}$, obtendo uma palavra de código $\mathbf{c} \in \mathcal{A}^n$. Segundo, um procedimento de decodificação, responsável por reconstruir $\mathbf{m}$ a partir do vetor codificado mesmo que algumas entradas de $\mathbf{c}$ tenham sido corrompidas durante a transmissão.

A Figura 1.6 ilustra o uso de um código de repetição sobre o alfabeto $\mathcal{A} = \mathbb{F}_2$, com

mensagens de tamanho $k = 3$. Podemos ver que o processo de codificação consiste em repetir 3 vezes cada um dos 3 bits de $\mathbf{m}$. Para decodificar, basta verificar, em cada bloco, qual bit aparece mais vezes. É importante notar que, neste código, caso ocorressem dois erros num mesmo bloco, a mensagem seria incorretamente recuperada. Portanto, no pior caso, tal código só é capaz de corrigir um único bit errado.

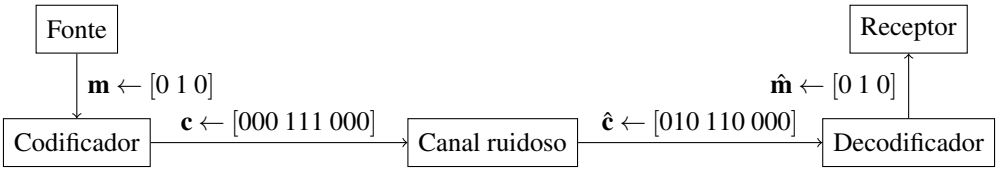


Figura 1.6. Transmissão de uma palavra através de um canal ruidoso.

Em geral, o processo de codificação pode ser qualquer mapeamento que leva as possíveis mensagens $\mathbf{m} \in \mathcal{A}^k$ em palavras de $\mathcal{A}^n$. Porém, mesmo para valores relativamente pequenos de k o tamanho $\left|\mathcal{A}^k\right|$ é muito grande, e fica inviável guardar uma tal função na memória eficientemente. Isso motiva a definição de códigos lineares, cujo alfabeto é um corpo finito e o procedimento de codificação é eficientemente descrito pela da multiplicação de uma matriz.

Seja p um primo e $q = p^m$, para algum inteiro m. Um código $[n,k]$-linear $\mathcal{C}$ é um subespaço vetorial de dimensão k do espaço $\mathbb{F}_q^n$. Note, portanto, o código $\mathcal{C}$ pode ser descrito tanto como a imagem de uma matriz $\mathbf{G} \in \mathbb{F}_q^{k\times n}$ quanto pelo núcleo de uma matriz $\mathbf{H} \in \mathbb{F}_q^{(n-k)\times n}$. Quaisquer matrizes $\mathbf{G}$ e $\mathbf{H}$ que definem um código $\mathcal{C}$ são ditas matrizes geradoras e de verificação de paridade, respectivamente. Formalmente, verifica-se para tais matrizes que

$$\mathcal{C} = \left\{\mathbf{mG} : \mathbf{m} \in \mathbb{F}_q^k\right\} = \left\{\mathbf{c} \in \mathbb{F}_q^n : \mathbf{cH}^\top = \mathbf{0}\right\}.$$

Dessa forma uma matriz de paridade $\mathbf{H}$ é capaz de identificar se uma palavra faz ou não parte do código $\mathcal{C}$. Isso motiva a definição da síndrome de um vetor $\hat{\mathbf{c}} \in \mathbb{F}_q^n$ como o produto $\mathbf{s} = \hat{\mathbf{c}}\mathbf{H}^\top$. Assim, se a síndrome $\mathbf{s}$ for $\mathbf{0}$, quer dizer que todas as equações lineares induzidas por $\mathbf{H}$ foram satisfeitas e $\hat{\mathbf{c}}$ faz parte do código. Caso contrário, então podemos ver $\hat{\mathbf{c}}$ como a soma de um vetor $\mathbf{c} \in \mathcal{C}$ e um vetor de erro $\mathbf{e} \in \mathbb{F}_q^n$, isto é, $\hat{\mathbf{c}} = \mathbf{c} + \mathbf{e}$. Portanto a síndrome

$$\mathbf{s} = \hat{\mathbf{c}}\mathbf{H}^\top = (\mathbf{c}+\mathbf{e})\,\mathbf{H}^\top = \mathbf{cH}^\top + \mathbf{eH}^\top = \mathbf{eH}^\top$$

pode ajudar na decodificação ao identificar as equações insatisfeitas por conta do erro $\mathbf{e}$.

Tome um vetor $\mathbf{c} \in \mathbb{F}_q^n$, e considere as seguintes definições. Definimos o suporte de $\mathbf{c}$, denotado por $\operatorname{supp}(\mathbf{c})$, como o conjunto de índices de suas entradas não nulas, isto é, $\operatorname{supp}(\mathbf{c}) = \{i : \mathbf{c}[i] \neq 0\}$. O peso de Hamming de $\mathbf{c}$ é definido como o número de elementos de seu suporte, e denotado por $\mathrm{w}(\mathbf{c}) = |\operatorname{supp}(\mathbf{c})|$. A distância de Hamming entre dois vetores $\mathbf{u}$ e $\mathbf{v}$ de $\mathbb{F}_q^n$, denotada por $\operatorname{dist}(\mathbf{u},\mathbf{v})$, é definida como o número de coordenadas em que $\mathbf{u}$ e $\mathbf{v}$ diferem, ou, alternativamente $\operatorname{dist}(\mathbf{u},\mathbf{v}) = \mathrm{w}(\mathbf{u}-\mathbf{v})$.

1.7.1. Códigos cíclicos e polinômios geradores

Há casos em que códigos é útil considerar códigos em que a dimensão k e o comprimento n são grandes. Nestes casos a representação por matrizes pode ser ineficiente, e, ainda pior, a multiplicação pela matriz geradora necessária para a codificação pode ser muito custosa. Uma subclasse de códigos lineares que cuja representação e codificação são mais eficientes é a de códigos cíclicos, que admitem uma matriz geradora $k \times n$ da forma

$$\mathbf{G} = \begin{bmatrix} 1 & g_1 & g_2 & \cdots & g_{n-k-1} & 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 1 & g_1 & \cdots & g_{n-k-2} & g_{n-k-1} & 1 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & g_1 & g_2 & g_3 & \cdots & g_{n-k-1} & 1 \end{bmatrix}.$$

Nota-se que uma tal matriz pode ser representada compactamente somente pelo vetor $\mathbf{g} = [1, g_1, \ldots, g_{n-k-1}, 1]$. Porém, mais importante ainda é o fato de que a codificação $\mathbf{c} = \mathbf{mG}$ de uma mensagem $\mathbf{m} \in \mathbb{F}_q^k$ pode ser computada da seguinte forma. Primeiro, associamos todo vetor $\mathbf{v} = [v_0, v_1, \ldots, v_{m-1}] \in \mathbb{F}_q^m$, ao polinômio $\mathbf{v}(x)$ de forma que

$$\mathbf{v}(x) = v_0 + v_1 x + \ldots + v_{m-1} x^{m-1} = \sum_{i=0}^{m-1} v_i x^i.$$

Seja $\mathbf{g} = [1, g_1, \ldots, g_{n-k-1}, 1] \in \mathbb{F}_q^{n-k+1}$ e $\mathbf{m} \in \mathbb{F}_q^k$ a mensagem que queremos codificar. Então, para o código cíclico gerado por $\mathbf{g}$, vale que

$$\mathbf{c} = \mathbf{mG} \text{ se, e somente se } \mathbf{c}(x) = \mathbf{m}(x)\mathbf{g}(x).$$

Como a multiplicação de polinômios pode ser computada eficientemente através de algoritmos como o de Karatsuba ou da trasformada rápida de Fourier, a codificação de códigos cíclicos é muito eficiente. Exemplos de códigos cíclicos muitos usados são as importantes famílias de códigos BCH e Reed-Solomon.

1.7.2. Decodificação

Há vários modelos de ruídos que podem ocorrer durante a transmissão. Podem ser erros uniformemente aleatórios, ou seguir uma gaussiana, por exemplo. Também há modelos em que coordenadas são apagadas, ou em que erros vêm em rajadas sobre posições contíguas do vetor codificado, como quando um CD é arranhado. Para cada tipo de erro, um diferente código corretor pode ser a melhor opção.

Seja q a potência de um primo e considere o corpo finito $\mathbb{F}_q$. Um modelo genérico e importante de ruído sobre códigos $[n,k]$-lineares sobre $\mathbb{F}_q$ é o canal simétrico q-ário. Este modelo usa um parâmetro $p \in [0,1]$ chamado probabilidade de transição, e gera erros $\mathbf{e} = [e_0, \ldots, e_{n-1}]$ de forma que cada e_i é gerado de forma independente como

$$e_i = \begin{cases} 0 \text{ com probabilidade } (1-p), \\ \text{elemento aleatório de } (\mathbb{F}_q - \{0\}) \text{ com probabilidade } p. \end{cases}$$

Dada uma palavra $\hat{\mathbf{c}} \in \mathbb{F}_q^n$, e um código $[n,k]$-linear $\mathcal{C} \in \mathbb{F}_q$, o problema da decodificação consiste em encontrar a palavra $\mathbf{c} \in \mathcal{C}$ mais próxima de $\hat{\mathbf{c}}$. Embora seja possível construir bons códigos cuja decodificação é eficiente, este problema é NP-difícil [Berlekamp et al. 1978].

O problema da decodificação pode ser formulado da seguinte maneira alternativa, resultando no chamado problema da decodificação por síndrome. Nesta formulação, é dada uma matriz de paridade $\mathbf{H} \in \mathbb{F}_q^{r \times n}$ de um código $\mathcal{C}$ junto com uma síndrome $\mathbf{s} \in \mathbb{F}_q^r$. O problema é encontrar o vetor $\mathbf{e} \in \mathbb{F}_q^n$ de menor peso de Hamming que satisfaz $\mathbf{eH}^\top = \mathbf{s}$.

1.7.3. Distância mínima de um código

Bons códigos possuem seus elementos a uma boa distância de Hamming uns dos outros, o que garante uma boa capacidade de correção de erros. Isso motiva a definição de distância mínima de um código, que corresponde à menor distância entre duas de suas palavras. Formalmente, a distância mínima de um código linear $\mathcal{C}$, denotada como $d(\mathcal{C})$, pode ser computada como

$$d(\mathcal{C}) = \min_{\mathbf{u},\mathbf{v} \in \mathcal{C}} \mathrm{dist}\,(\mathbf{u},\mathbf{v}) = \min_{\mathbf{u},\mathbf{v} \in \mathcal{C}} \mathrm{w}\,(\mathbf{u}-\mathbf{v})\,.$$

Porém, pela linearidade do código, a diferença dos vetores $\mathbf{a} = (\mathbf{u}-\mathbf{v})$ varre todos os elementos de $\mathcal{C}$. Portanto

$$d(\mathcal{C}) = \min_{\mathbf{u},\mathbf{v} \in \mathcal{C}} \mathrm{w}\,(\mathbf{u}-\mathbf{v}) = \min_{\mathbf{a} \in \mathcal{C}} \mathrm{w}\,(\mathbf{a})\,.$$

Dizemos que um código $\mathcal{C}$ é $[n,k,d]$-linear quando $\mathcal{C}$ é $[n,k]$-linear e $d(\mathcal{C}) = d$. Um tal código poderá decodificar erros de peso até $\lfloor (d-1)/2 \rfloor$.

1.8. HQC

O HQC [Melchor et al. 2021] é um KEM baseado no problema da decodificação de códigos quase cíclicos. Sua construção é muito similar àquela dos esquemas baseados no problema LWE vistos na Seção 1.6, porém com a significativa diferença de trabalhar no corpo binário. Isso faz com que a fase de reconciliação seja mais complicada pois é necessário o uso de códigos corretores de erro poderosos.

Começamos esta seção apresentando os códigos Reed-Muller e Reed-Solomon que, juntos, compõem o código corretor de erros usado para a reconciliação. Em seguida, definimos o problema difícil em que o HQC se baseia. Ao final, o HQC é apresentado em sua versão IND-CPA.

1.8.1. Códigos Reed-Muller

Códigos Reed-Muller são uma família de códigos binários lineares. Para quaisquer inteiros positivos r e m, tais que $0 \leq r \leq m$, é possível construir um código $[n,k,d]$-linear, denotado $\mathrm{RM}(r,m)$ com as seguintes propriedades. O comprimento do código é $n = 2^m$, sua dimensão é $k = 1 + \binom{m}{1} + \binom{m}{2} + \ldots + \binom{m}{r}$, e sua distância mínima é $d = 2^{m-r}$.

Há várias formas de definir tais códigos, porém, neste trabalho, vamos apresentar diretamente o código Reed-Muller usado no HQC, que usa como parâmetros os valores $(r,m) = (1,7)$. O leitor poderá conferir que tais parâmetros resultam num código

$[128,8,64]$-linear. Este código, denotado por $\mathrm{RM}(1,7)$, é gerado pela matriz representada na Figura 1.7.

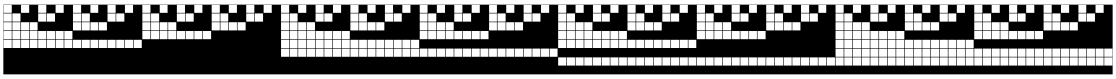


Figura 1.7. A matriz binária geradora do código RM$(1,7)$.

Para atingir a alta capacidade correção necessária pelo HQC, o código usado usa o código $\mathrm{RM}(1,7)$ repetido M vezes. Pode-se mostrar que, repetindo o código $[128,8,64]$-linear M vezes, obtém-se um novo código $[128M,8,64M]$-linear. Isto é, há uma expansão linear tanto no tamanho do código e em sua distância mínima, enquanto a dimensão não é alterada.

Como a matriz é do código é relativamente grande, podemos defini-la mais sucintamente em Sage usando uma função auxiliar que converte números inteiros em vetores binários de tamanho fixo.

```
def binary_vector_from_int(a, length):
    bits = bin(a)[2:]
    v = [0] * length
    for i, b in enumerate(reversed(bits)):
        v[i] = int(b)
    return v
```

Assim, podemos definir os códigos Reed-Muller repetidos usados pelo HQC da seguinte forma.

```
class RepeatedReedMullerForHQC():

    RMCodeLength = 128

    GeneratorMatrix = matrix([
        binary_vector_from_int(0xaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa, RMCodeLength),
        binary_vector_from_int(0xcccccccccccccccccccccccccccccccc, RMCodeLength),
        binary_vector_from_int(0xf0f0f0f0f0f0f0f0f0f0f0f0f0f0f0f0, RMCodeLength),
        binary_vector_from_int(0xff00ff00ff00ff00ff00ff00ff00ff00, RMCodeLength),
        binary_vector_from_int(0xffff0000ffff0000ffff0000ffff0000, RMCodeLength),
        binary_vector_from_int(0x00000000ffffffff00000000ffffffff, RMCodeLength),
        binary_vector_from_int(0x0000000000000000ffffffffffffffff, RMCodeLength),
        binary_vector_from_int(0xffffffffffffffffffffffffffffffff, RMCodeLength),
    ])

    def __init__(self, multiplicity):
        self.q = 2
        self.k = 8
        self.n = 128 * multiplicity
        self.multiplicity = multiplicity
        self.encoded_table = [self.encode(binary_vector_from_int(m, self.k))
                              for m in range(256)]
```

Codificação: A codificação de uma mensagem $\mathbf{m} \in \mathbb{F}_2^8$ pode ser feita simplesmente multiplicando-a pela matriz geradora do código $\mathrm{RM}(1,7)$ e devolvendo o vetor resultante repetido M vezes.

```
    def encode(self, message):  # Class RepeatedReedMullerForHQC
        repeated_encoded = (
            [vector(GF(2), message) * self.GeneratorMatrix] * self.multiplicity)
        return vector(chain(*repeated_encoded))
```

Decodificação: Em geral, a decodificação de códigos Reed-Muller pode ser feita eficientemente usando a transformada rápida de Hadamard [Lin and Costello 2004]. Porém, como o código $\mathrm{RM}(1,7)$ tem somente $2^k = 256$ elementos, podemos fazer a decodificação pela distância mínima usando um algoritmo exaustivo.

Para cada possível vetor de $\mathbb{F}_2^{256}$, codifique-o e calcule a distância entre o resultado e palavra ruidosa recebida. Devolva aquele vetor cuja codificação encontra-se à menor distância da palavra recebida.

```
    def decode(self, noisy_codeword):  # Class RepeatedReedMullerForHQC
        min_distance = self.n
        decoded_m = None
        for m in range(2**8):
            encoded_m = self.encoded_table[m]
            distance = (encoded_m - noisy_codeword).hamming_weight()
            if distance < min_distance:
                min_distance = distance
                decoded_m = m
        return binary_vector_from_int(decoded_m, self.k)
```

1.8.2. Códigos Reed-Solomon

Códigos Reed-Solomon são uma importante família de códigos cíclicos corretores de erros. Diferente dos códigos Reed-Muller, códigos Reed-Solomon trabalham com alfabetos q-ários, onde $q = p^m$ é a m-ésima potência de um primo p. Novamente, para deixar a nossa explicação o mais simples e concreta possível, vamos fixar alguns parâmetros para o código usado no HQC, onde $p = 2$, $m = 8$, portanto $q = 256$.

Lembre que, por serem cíclicos, trabalharemos com polinômios para a codificação eficiente. Uma propriedade importante desses códigos é poder garantir a sua capacidade de correção de erros durante sua construção. Seja δ o número de erros que o código deverá ser capaz de corrigir, e seja α um elemento primitivo de $\mathbb{F}_q$. Então o polinômio

$$\mathbf{g}(x) = \prod_{i=1}^{2\delta} (x - \alpha^i) = (x-\alpha)(x-\alpha^2)\dots(x-\alpha^{2\delta}).$$

gera um código $[n,k,d]$-linear de distância mínima $d = 2\delta + 1$, comprimento $n \leq q-1$ e dimensão $k \leq n-d$, chamado Reed-Solomon.[4]

```
class ReedSolomonForHQC():

    def __init__(self, n, k):
        self.n = n  # Block length
        self.k = k  # Dimension
        self.q = 256
        self.delta = (n - k) // 2
        self.base_field = GF(self.q)
        self.polynomial_ring = PolynomialRing(self.base_field, 'x')
        self.x = self.polynomial_ring.gen()
        self.generator_polynomial = self._compute_generator_polynomial()

    def _compute_generator_polynomial(self):
        alpha = self.base_field.primitive_element()
        return prod(self.x - alpha**i for i in range(1, 2*self.delta + 1))
```

[4]Em geral, adota-se $n = q-1$ na definição de códigos Reed-Solomon. Formalmente, os códigos usados pelo HQC são chamados Reed-Solomon encurtados pois $n < q-1$. Como a codificação e decodificação funcionam exatamente da mesma forma, preferimos trabalhar diretamente com os códigos encurtados.

Codificação: Suponha que queremos codificar uma mensagem $\mathbf{m} \in \mathbb{F}_{256}^k$. Lembre que identificamos vetores e polinômios, portanto $\mathbf{m}(x)$ é o polinômio associado à mensagem $\mathbf{m}$. Seja $\mathbf{g}$ o polinômio gerador do código Reed-Solomon $[n,k,d]$-linear. Então a codificação $\mathbf{c} \in \mathbb{F}_{256}^n$ da mensagem $\mathbf{m}$ na forma sistemática é dada por

$$\mathbf{c}(x) = \mathbf{m}(x)x^{n-k} + \left(\mathbf{m}(x)x^{n-k} \bmod \mathbf{g}(x)\right).$$

Considere o vetor codificado $\mathbf{c}$. Note que a mensagem $\mathbf{m}$ pode ser recuperada diretamente dos k coeficientes associados às potências de x^{n-k} até x^{n-1}. A redundância adicionada corresponde aos $n-k$ coeficientes de índice 0 a $(n-k-1)$. Em Sage, podemos implementar a codificação da seguinte forma.

```
    def encode(self, message):
        message_polynomial = self.polynomial_ring(message)
        a = self.x**(self.n - self.k) * message_polynomial
        b = a % self.generator_polynomial
        c = b + a
        return c
```

Note, porém, que os vetores usados no HQC são binários. Então, se quisermos codificar mensagens binárias usando com códigos Reed-Solomon sobre $\mathbb{F}_{256}$, devemos primeiro quebrar a mensagem binária em sequências de 8 bits, e interpretar cada bloco de 8 bits como um elemento de $\mathbb{F}_{256}$, para então codificar. Este procedimento é sintetizado no método abaixo.

```
    def encode_binary(self, binary_message):
        assert len(binary_message) == self.k * 8
        message = [self.base_field(binary_message[i * 8 : (i + 1) * 8])
                   for i in range(self.k)]
        encoded = self.encode(message)
        encoded_binary = chain(*[list(v) for v in encoded])
        return vector(GF(2), encoded_binary)
```

Decodificação: Quando descrevemos os códigos Reed-Muller na Seção 1.8.1, notamos que o código continha apenas 256 elementos, e, portanto, foi possível usar o algoritmo simples de distância mínima para a decodificação. Infelizmente, os códigos Reed-Solomon usados no HQC contêm $q^k = 256^k$ elementos, que tornam inviável usar a mesma forma geral de decodificação. Nesta seção, vamos descrever um algoritmo eficiente para a decodificação de códigos Reed-Solomon [Lin and Costello 2004].

Suponha que $\mathbf{c}$ é a codificação de uma mensagem $\mathbf{m}$, e seja $\mathbf{r} = \mathbf{c} + \mathbf{e}$ a palavra $\mathbf{c}$ corrompida por um erro $\mathbf{e}$. Considere o polinômio $\mathbf{r}(x) = r_0 + r_1 x + \ldots r_{n-1}x^{n-1}$ associado ao vetor corrompido $\mathbf{r}$. A síndrome do vetor recebido $\mathbf{r}$ em relação ao código Reed-Solomon é dado por $(s_1, \ldots, s_{2\delta})$ onde cada s_i é dado por

$$s_i = \mathbf{r}(\alpha^i) = \mathbf{c}(\alpha^i) + \mathbf{e}(\alpha^i) = \mathbf{e}(\alpha^i).$$

Suponha que sabemos que o o peso de $\mathbf{e}$ é t, isto é, há t entradas não nulas em $\mathbf{e} = [e_0, \ldots, e_{n-1}]$. Digamos que tais entradas são $\text{supp}(\mathbf{e}) = \{j_1, j_2, \ldots, j_t\}$. Então podemos escrever

$$s_i = \sum_{k=1}^{t} e_{j_k}\left(\alpha^{j_k}\right)^k = e_{j_1}\left(\alpha^{j_1}\right)^i + e_{j_2}\left(\alpha^{j_2}\right)^i + \ldots + e_{j_t}\left(\alpha^{j_t}\right)^i. \quad (1)$$

Na forma matricial, temos então o seguinte sistema

$$\begin{bmatrix} \alpha^{j_1} & \dots & \alpha^{j_t} \\ (\alpha^{j_1})^2 & \dots & (\alpha^{j_t})^2 \\ \vdots & \ddots & \vdots \\ (\alpha^{j_1})^{2\delta} & \dots & (\alpha^{j_t})^{2\delta} \end{bmatrix} \begin{bmatrix} e_{j_1} \\ e_{j_2} \\ \vdots \\ e_{j_t} \end{bmatrix} = \begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_t \end{bmatrix}. \tag{2}$$

Infelizmente, ainda não é possível resolver a equação pois não sabemos nem o número t de erros nem as suas posições $j_1, \dots, j_t$. Considere então o polinômio $\sigma(x)$ definido como

$$\sigma(x) = \left(1 - \alpha^{j_1} x\right)\left(1 - \alpha^{j_2} x\right) \dots \left(1 - \alpha^{j_t} x\right).$$

O polinômio $\sigma(x)$ é chamado localizador de erros pois, através de suas raízes $\alpha^{-j_1}, \dots, \alpha^{-j_t}$, é possível descobrir qual as posições j_i dos erros, já que α é um elemento primitivo de $\mathbb{F}_q$.

Os passos para resolver o sistema da Equação 2 são os seguintes.

1. Descobrir os coeficientes do polinômio $\sigma(x) = 1 + \sigma_1 x + \sigma_2 x^2 + \dots + \sigma_t x^t$.
2. Encontrar as t raízes α^{j_i} de $\sigma(x)$.
3. Obter as posições $j_1, \dots, j_t$, dos erros.
4. Montar a matriz de potências de α e resolver o sistema da Equação 2.

Pela definição de $\sigma(x)$, vale, para todo $k = 1$ até t, que

$$\sigma\left(\alpha^{-j_k}\right) = 1 + \sigma_1 \alpha^{-j_k} + \sigma_2 \left(\alpha^{-j_k}\right)^2 + \dots + \sigma_t \left(\alpha^{-j_k}\right)^t = 0.$$

A ideia é relacionar a equação acima com as entradas $(s_1, \dots, s_{2\delta})$ da síndrome. Compare a equação acima com a Equação 1 que define os valores s_i. Apesar de haver potências de α na equação, as potências são negativas. Além disso, faltam os fatores e_j.

Fixe então um ℓ qualquer tal que $1 \leq \ell \leq t$. Ao multiplicar a equação por $e_{j_k} \alpha^{j_k(\ell+t)}$, obtemos

$$\begin{aligned} 0 &= e_{j_k} \alpha^{j_k(\ell+t)} \left(1 + \sigma_1 \alpha^{-j_k} + \sigma_2 \left(\alpha^{-j_k}\right)^2 + \dots + \sigma_t \left(\alpha^{-j_k}\right)^t\right) \\ &= e_{j_k} \alpha^{j_k(\ell+t)} + \sigma_1 e_{j_k} \alpha^{j_k(\ell+t-1)} + \sigma_2 e_{j_k} \alpha^{j_k(\ell+t-2)} + \dots + \sigma_t e_{j_k} \alpha^{j_k(\ell)}. \end{aligned}$$

Como a equação acima vale para qualquer k, podemos somar as t equações para $1 \leq k \leq t$ e obtemos

$$\begin{aligned} 0 &= \sum_{k=1}^{t} \left(e_{j_k} \alpha^{j_k(\ell+t)} + \sigma_1 e_{j_k} \alpha^{j_k(\ell+t-1)} + \dots + \sigma_t e_{j_k} \alpha^{j_k(\ell)}\right) \\ &= \sum_{k=1}^{t} \left(e_{j_k} \alpha^{j_k(\ell+t)}\right) + \sum_{k=1}^{t} \left(\sigma_1 e_{j_k} \alpha^{j_k(\ell+t-1)}\right) + \dots + \sum_{k=1}^{t} \left(\sigma_t e_{j_k} \alpha^{j_k(\ell)}\right) \\ &= \sum_{k=1}^{t} \left(e_{j_k} \alpha^{j_k(\ell+t)}\right) + \sigma_1 \sum_{k=1}^{t} \left(e_{j_k} \alpha^{j_k(\ell+t-1)}\right) + \dots + \sigma_t \sum_{k=1}^{t} \left(\sigma_t e_{j_k} \alpha^{j_k(\ell)}\right). \end{aligned}$$

Lembre que, pela Equação 1, vale que $s_i = \sum_{k=1}^{t} \left(e_{j_k} \alpha^{j_k i}\right)$. Podemos então substituir cada somatória pelo respectivo valor de síndrome, reduzindo a equação acima a

$$s_{\ell+t} + \sigma_1 s_{\ell+t-1} + \ldots + \sigma_t s_\ell = 0.$$

Como tal equação vale para todo $1 \leq \ell \leq t$, obtemos o seguinte sistema

$$\begin{bmatrix} s_t & s_{t-1} & \cdots & s_1 \\ s_{t+1} & s_t & \cdots & s_2 \\ \vdots & \vdots & \ddots & \vdots \\ s_{2t-1} & s_{2t-2} & \cdots & s_t \end{bmatrix} \begin{bmatrix} \sigma_1 \\ \sigma_2 \\ \vdots \\ \sigma_t \end{bmatrix} + \begin{bmatrix} s_{t+1} \\ s_{t+2} \\ \vdots \\ s_{2t} \end{bmatrix} = \mathbf{0}. \tag{3}$$

Note que, apesar de sabermos cada s_i, para $1 \leq i \leq 2\delta$, nós não sabemos o valor de t. Neste momento, isso impede que possamos construir tal sistema e recuperar os coeficientes σ_j do polinômio localizador de erros $\sigma(x)$.

Há duas saídas. A primeira é simples: testar diferentes valores de t, começando em δ. Para cada valor de t, constrói-se o sistema correspondente e tenta-se resolvê-lo. Caso seja possível, o valor foi encontrado. Caso não seja, tentamos o valor $t \leftarrow (t-1)$, até chegar em $t = 0$. Apesar da busca exaustiva por t funcionar bem para casos pequenos, esse algoritmo pode ser muito ineficiente para códigos Reed-Solomon maires, devido ao grande número de sistemas lineares que devem ser resolvidos.

Uma alternativa mais eficiente é notar que este sistema linear tem uma forma especial, que consiste num *linear-feedback shift register*. Pode-se então usar o algoritmo de Berlekamp-Massey para encontrar ao mesmo tempo o menor t que satisfaz a equação e os coeficientes de $\sigma(x)$. Nosso código usa esta opção, porém, devido a limitações de espaço, não é possível explicar aqui o funcionamento desse algoritmo.

Lembre que o código Reed-Solomon usado pelo HQC usa $q = 256$. Dessa forma, podemos calcular as raízes de σ simplesmente testando para quais, dos 256 elementos $\beta \in \mathbb{F}_q$ possíveis, vale que $\sigma(\beta) = 0$. Com isso, conseguimos construir o sistema linear e, se possível, resolvê-lo. O código abaixo é responsável por este procedimento.

```
    def solve_error_linear_system(self, syndromes, error_location_poly):
        alpha = self.base_field.primitive_element()
        alpha_log_table = {alpha**i: i for i in range(1, self.n + 1)}
        roots = [a for a in self.base_field if error_location_poly(a) == 0]
        error_betas = [e.inverse() for e in roots]
        error_positions = [alpha_log_table[b] for b in error_betas]

        beta_matrix = matrix(GF(self.q), [
            [beta**i for beta in error_betas] for i in range(1, 2*self.delta + 1)])
        error_values = beta_matrix.solve_right(vector(syndromes))
        error = self.polynomial_ring(
                    sum(e * self.x**i for i, e in zip(error_positions, error_values)))
        return error
```

Podemos então descrever totalmente o algoritmo de decodificação da seguinte forma. Note que, caso haja problemas ao construir ou resolver o sistema de valores de erros, há uma falha de decodificação e o valor None é devolvido.

```
    def decode(self, noisy_codeword):
```

```
        alpha = self.base_field.primitive_element()
        syndromes = [noisy_codeword(alpha**i) for i in range(1, 2*self.delta + 1)]
        error_location_poly = self.berlekamp_massey(syndromes)

        try:
            error = self.solve_error_linear_system(syndromes, error_location_poly)
            codeword_polynomial = noisy_codeword - error
            return self.polynomial_ring(list(codeword_polynomial)[-self.k:])

        except (KeyError, ValueError):
            return None
```

Implementamos também, de forma análoga à codificação, a interface em binário para a decodificação.

```
    def decode_binary(self, noisy_binary_codeword):
        assert len(noisy_binary_codeword) == self.n * 8

        noisy_codeword = [self.base_field(noisy_binary_codeword[i * 8 : (i + 1) * 8])
                          for i in range(self.n)]

        decoded = self.decode(self.polynomial_ring(noisy_codeword))
        decoded_binary = chain(*[list(v) for v in decoded])
        return vector(GF(2), decoded_binary)
```

1.8.3. Concatenando os códigos Reed-Muller e Reed-Solomon

A Figura 1.8 ilustra como é feita a concatenação de códigos Reed-Muller e Reed-Solomon. Estes códigos concatenados podem ser implementados da seguinte forma. Note como a classe abaixo serve apenas como um orquestrador de chamadas aos códitos Reed-Muller e Reed-Solomon.

```
class RMRSCodeForHQC():

    def __init__(self, rs_n, rs_k, rm_multiplicity, n=None):
        self.rs_code = ReedSolomonForHQC(rs_n, rs_k)
        self.rm_code = RepeatedReedMullerForHQC(rm_multiplicity)

        if n == None:
            self.padding_length = 0
        else:
            self.padding_length = n - self.rm_code.n * self.rs_code.n

    def encode(self, binary_message):
        rs_codeword = self.rs_code.encode_binary(binary_message)
        rm_messages = [rs_codeword[i * 8 : (i + 1) * 8] for i in range(self.rs_code.n)]
        rm_codewords = [self.rm_code.encode(m) for m in rm_messages]
        return vector(GF(2), chain(*rm_codewords, [0] * self.padding_length))

    def decode(self, noisy_codeword):
        # By the way rm_noisy_codewords is defined, the padding is naturally removed
        rm_noisy_codewords = [noisy_codeword[i * self.rm_code.n : (i + 1) * self.rm_code
    .n]
                              for i in range(self.rs_code.n)]
        rm_codewords = [self.rm_code.decode(c) for c in rm_noisy_codewords]
        rs_noisy_codeword = vector(chain(*rm_codewords))
        return self.rs_code.decode_binary(rs_noisy_codeword)
```

1.8.4. O problema da decodificação por síndrome para códigos quase cíclicos

O principal desafio em usar problemas de Teoria de Códigos para construir esquemas criptográficos é o baixo desempenho inerente aos códigos necessários. Tome o problema da decodificação por síndrome, por exemplo. Como a decodificação deve ser

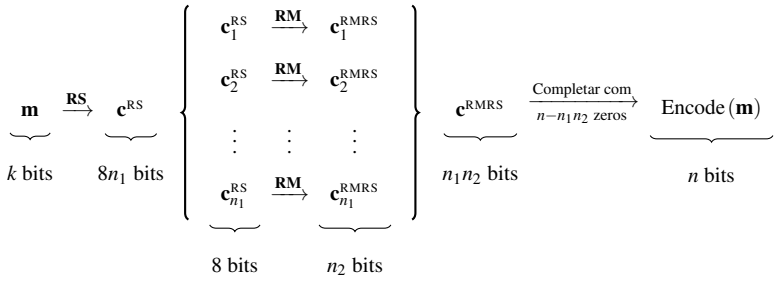


Figura 1.8. O processo de codificação usando códigos Reed-Muller e Reed-Solomon concatenados.

difícil para quem não tem a chave secreta, os códigos em questão devem ser aleatórios ou sua estrutura secreta deve estar muito bem escondida.

Dessa forma, para atingir níveis altos de segurança, os parâmetros $[n,k]$ devem ser relativamente grandes, fazendo com que as matrizes, tanto a geradora quanto a de paridade, ocupem muito espaço na memória e operações típicas como a multiplicação de matriz por vetor seja muito ineficiente. Esquemas modernos baseados em Teoria de Códigos, como o BIKE [Aragon et al. 2022] e o próprio HQC [Melchor et al. 2021], são muito eficientes por usarem códigos chamados quase-cíclicos, que permitem representação e operações mais eficientes.

Antes de vermos a definição de códigos quase cíclicos, primeiro lembre que uma matriz circulante definida por um vetor $\mathbf{v} = [v_0, \ldots, v_{n-1}]$ é a matriz

$$\operatorname{rot}(\mathbf{v}) = \begin{bmatrix} v_0 & v_{n-1} & \cdots & v_1 \\ v_1 & v_0 & \cdots & v_2 \\ \vdots & \vdots & \ddots & \vdots \\ v_{n-1} & v_{n-2} & \cdots & v_0 \end{bmatrix}.$$

O conjunto de matrizes $n \times n$ circulantes com entradas em $\mathbb{F}_q$ forma um anel. Em particular, soma e multiplicação de matrizes circulantes resulta em matrizes também circulantes.

Códigos quase-cíclicos são códigos que admitem matrizes geradoras formadas por blocos de matrizes circulantes. Assim, um código quase-cíclico de índice c admite uma matriz geradora sistemática da forma

$$\mathbf{G} = \begin{bmatrix} \mathbf{I} & \operatorname{rot}(\mathbf{g}_1) & \cdots & \operatorname{rot}(\mathbf{g}_{c-1}) \end{bmatrix} \in \mathbb{F}_2^{n \times cn}.$$

De forma equivalente, um código quase-cíclico admite uma matriz de paridade da forma

$$\mathbf{H} = \begin{bmatrix} \mathbf{I} & \mathbf{0} & \cdots & \mathbf{0} & \operatorname{rot}(\mathbf{h}_1) \\ \mathbf{0} & \mathbf{I} & \cdots & \mathbf{0} & \operatorname{rot}(\mathbf{h}_2) \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{I} & \operatorname{rot}(\mathbf{h}_{c-1}) \end{bmatrix} \in \mathbb{F}_2^{(cn-n) \times cn},$$

para vetores $\mathbf{h}_i \in \mathbb{F}_2^n$.

Definição 6 (Problema da decodificação por síndrome para códigos quase-cíclicos)**.** São dados inteiros n, w e c. Escolha aleatoriamente, de maneira uniforme, uma matriz de paridade $\mathbf{H} \in \mathbb{F}_2^{(cn-n)\times cn}$ de um código $[cn,n]$-linear quase-cíclico de índice c. Escolha um vetor esparso $\mathbf{x} = (\mathbf{x}_0, \ldots, \mathbf{x}_{c-1})$ aleatoriamente do conjunto $\mathbb{F}_2^{cn}$ de forma que o peso de cada $\mathbf{x}_i \in \mathbb{F}_2^n$ seja igual a w. Calcule a síndrome $\mathbf{s} = \mathbf{x}\mathbf{H}^\top$. Então, dados $(\mathbf{H}, \mathbf{s})$, o problema da decodificação por síndrome pede para encontrar um vetor $\mathbf{y} = (\mathbf{y}_0, \ldots, \mathbf{y}_{c-1}) \in \mathbb{F}_2^{cn}$, tal que $\mathbf{y}\mathbf{H}^\top = \mathbf{s}$ e o peso de cada $\mathbf{y}_i$ seja $\mathrm{w}(\mathbf{y}_i) = w$.

□

1.8.5. Implementação do HQC

Nesta seção, apresentamos o HQC e sua implementação. Veremos que, em alto nível, sua construção é similar à de esquemas baseados no LWE, como o Kyber. Porém, por trabalhar com vetores binários, a codificação e decoficação são significativamente mais complicadas, exigindo códigos corretores de erros poderosos.

Seleção de parâmetros e inicialização: Dado o nível de segurança λ desejado, são selecionados os parâmetros $n_1, n_2, M, n, k, w, w_{\mathbf{r}}, w_{\mathbf{e}}$. Estes parâmetros são responsáveis por definir o código concatenado formado pelos códigos Reed-Solomon e Reed-Muller repetido. O código Reed-Solomon usado será $[n_1, k/8]$-linear sobre $\mathbb{F}_{256}$, enquanto o código Reed-Muller repetido será $[n_2, 8]$-linear. Os parâmetros $w, w_{\mathbf{r}}$ e $w_{\mathbf{e}}$ correspondem ao peso dos vetores esparsos usados durante a geração de chaves e encriptação.

O código a seguir é responsável pela inicialização dos parâmetros usados no HQC.

```
class HQC_PKE_CPA():
    SecurityParameters = {
        128: HQCParameters(n1=46, n2=384, multiplicity=3, n=17669,
                           k=128, w=66, w_r=77, w_e=77),
        192: HQCParameters(n1=56, n2=640, multiplicity=5, n=35851,
                           k=192, w=100, w_r=114, w_e=114),
        256: HQCParameters(n1=90, n2=640, multiplicity=5, n=57637,
                           k=256, w=133, w_r=149, w_e=149),
    }
    def __init__(self, security_level):
        self.params = self.SecurityParameters[security_level]
        self.rmrs = RMRSCodeForHQC(rs_n=self.params.n1, rs_k=self.params.k // 8,
                                   rm_multiplicity=self.params.multiplicity,
                                   n=self.params.n)
```

Geração de chaves: Escolha um vetor $\mathbf{h}$ de n bits aleatoriamente de $\mathbb{F}_2^n$. Escolha vetores esparsos $\mathbf{x}$ e $\mathbf{y}$ aleatoriamente do conjunto $\left\{\mathbf{v} \in \mathbb{F}_2^n : \mathrm{w}(\mathbf{z}) = w\right\}$. Calcule o vetor $\mathbf{s} = \mathbf{x} + \mathbf{y} \cdot \mathbf{h}$. As chaves pública e secreta serão $\mathbf{pk} = (\mathbf{s}, \mathbf{h})$ e $\mathbf{sk} = (\mathbf{x}, \mathbf{y})$, respectivamente. Note que a chave secreta está protegida pelo problema da decodificação por síndrome.

```
    def generate_binary_vector_of_fixed_weight(self, xof, weight):
        support = xof_sample_k_indexes(xof, self.params.n, weight)
        v = zero_vector(GF(2), self.params.n)
        for s in support:
            v[s] = 1
        return v

    def vector_product(self, a, b):
        R = PolynomialRing(GF(2), 'x')
```

```
        x = R.gen()
        Q = R.quotient(x**self.params.n - 1)
        return vector(GF(2), Q(list(a)) * Q(list(b)))

    def keygen(self, randomness):
        xof = SHAKE256.new(randomness)
        h = random_vector(GF(2), self.params.n)
        x = self.generate_binary_vector_of_fixed_weight(xof, self.params.w)
        y = self.generate_binary_vector_of_fixed_weight(xof, self.params.w)
        s = x + self.vector_product(h, y)

        return (h, s), (x, y)
```

Encriptação: Seja $\mathbf{m} \in \mathbb{F}_2^k$ a mensagem a ser encriptada e $(\mathbf{s}, \mathbf{h})$ a chave pública do destinatário. Primeiro, escolha dois vetores esparsos $\mathbf{r}_1$ e $\mathbf{r}_2$ aleatoriamente do conjunto $\left\{\mathbf{z} \in \mathbb{F}_2^n : \mathrm{w}(\mathbf{z}) = w_{\mathbf{r}}\right\}$. Similarmente, escolha um outro vetor esparso $\mathbf{e}$ aleatoriamente de $\left\{\mathbf{z} \in \mathbb{F}_2^n : \mathrm{w}(\mathbf{z}) = w_{\mathbf{e}}\right\}$. Calcule os vetores $\mathbf{u} = \mathbf{r}_1 + \mathbf{r}_2 \cdot \mathbf{h}$ e $\mathbf{v} = \mathrm{Encode}(\mathbf{m}) + \mathbf{s} \cdot \mathbf{r}_2 + \mathbf{e}$. O texto encriptado é dado por $\mathbf{c} = (\mathbf{u}, \mathbf{v})$.

```
    def encrypt(self, pk, message, randomness):
        h, s = pk
        xof = SHAKE256.new(randomness)
        e = self.generate_binary_vector_of_fixed_weight(xof, self.params.w_e)
        r_1 = self.generate_binary_vector_of_fixed_weight(xof, self.params.w_r)
        r_2 = self.generate_binary_vector_of_fixed_weight(xof, self.params.w_r)
        u = r_1 + self.vector_product(h, r_2)
        v = self.rmrs.encode(message) + self.vector_product(s, r_2) + e
        return (u, v)
```

Decriptação: Dado um texto cifrado $\mathbf{c} = (\mathbf{u}, \mathbf{v})$ e a chave secreta $\mathbf{x}, \mathbf{y}$, primeiro calcule o vetor $\mathbf{c}' = \mathbf{v} + \mathbf{u} \cdot \mathbf{y}$. Note que este vetor é

$$\begin{aligned}\mathbf{c}' &= \mathbf{mG} + \mathbf{s} \cdot \mathbf{r}_2 + \mathbf{e} + (\mathbf{r}_1 + \mathbf{r}_2 \cdot \mathbf{h}) \cdot \mathbf{y} \\ &= \mathbf{mG} + (\mathbf{x} + \mathbf{y} \cdot \mathbf{h}) \cdot \mathbf{r}_2 + \mathbf{e} + (\mathbf{r}_1 + \mathbf{r}_2 \cdot \mathbf{h}) \cdot \mathbf{y} \\ &= \mathbf{mG} + \mathbf{x} \cdot \mathbf{r}_2 + \mathbf{r}_1 \cdot \mathbf{y} + \mathbf{e}.\end{aligned}$$

Como os vetores $\mathbf{x}, \mathbf{y}, \mathbf{r}_1, \mathbf{r}_2$, e $\mathbf{e}$ são todos esparsos, então é esperado que o vetor de erro $\mathbf{e}' = \mathbf{x} \cdot \mathbf{r}_2 + \mathbf{r}_1 \cdot \mathbf{y} + \mathbf{e}$ também seja relativamente esparso. Dessa forma, o código RMRS pode-se facilmente decodificar o erro $\mathbf{e}'$ e recuperar a mensagem $\mathbf{m}$.

```
    def decrypt(self, sk, ciphertext):
        u, v = ciphertext
        x, y = sk
        c_prime = v - self.vector_product(u, y)

        return self.rmrs.decode(c_prime)
```

1.9. Conclusão

Neste capítulo, mostramos detalhes da construção de esquemas criptográficos pós-quânticos. Embora os esquemas apresentem certas similaridades entre si, por abordarmos aspectos teóricos e práticos, pudemos tratar de uma coleção abrangente de temas atuais em criptografia. Com o NTRU, que é o esquemas mais antigo entre os apresentados, vimos como o Sage pode facilitar a experimentação com a prototipagem de esquemas

algébricos. Com o Kyber, apresentamos detalhes da Transformada da Teoria dos Números e vimos como ela é usada em esquemas modernos. Ao apresentar o Dilithium, discutimos a transformação de Fiat-Shamir, e a importância da amostragem por rejeição para proteger segredos. Por fim, o HQC nos permitiu discutir teoria de códigos corretores de erros, mostrando como implementar códigos Reed-Muller e Reed-Solomon em Sage e seu uso em criptografia.

É importante notar que este capítulo não substitui as especificações dos esquemas apresentados. Em particular, há questões que não puderam ser tratadas com maior profundidade por falta de espaço. Alguns pontos importantes de que não tratamos são os ataques contra os esquemas e os motivos por trás da escolha de parâmetros seguros. Há ainda questões mais profundas sobre os perigos das falhas de decriptação, e como podemos garantir que a probabilidade de falha seja desprezável. Além disso, este material não discute dificuldades de implementação segura de esquemas criptográficos nem ataques de canal lateral.

No entanto, esperamos que nosso material possa contribuir para que essa área de pesquisa seja mais acessível. Em particular, a implementação que disponibilizamos ao público poderá ser útil a novos pesquisadores interessados tanto em prototipar soluções mais eficientes quando em criptanálise.

Referências

[Alagic et al. 2022] Alagic, G., Apon, D., Cooper, D., Dang, Q., Dang, T., Kelsey, J., Lichtinger, J., Miller, C., Moody, D., Peralta, R., et al. (2022). Status report on the third round of the NIST post-quantum cryptography standardization process. Technical report, National Institute of Standards and Technology.

[Aragon et al. 2022] Aragon, N., Barreto, P. S. L. M., Bettaieb, S., Bidoux, L., Blazy, O., Deneuville, J.-C., Gaborit, P., Ghosh, S., Gueron, S., Güneysu, T., Aguilar-Melchor, C., Misoczki, R., Persichetti, E., Richter-Brockmann, J., Sendrier, N., Tillich, J.-P., Vasseur, V., and Zémor, G. (2022). BIKE: Bit flipping key encapsulation. https://bikesuite.org/files/v5.0/BIKE_Spec.2022.10.10.1.pdf.

[Avanzi et al. 2019] Avanzi, R., Bos, J., Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schanck, J. M., Schwabe, P., Seiler, G., and Stehlé, D. (2019). CRYSTALS-Kyber: Algorithm specifications and supporting documentation (2020). *NIST PQC Round*, 2(4):1–43.

[Bai et al. 2021] Bai, S., Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schwabe, P., Seiler, G., and Stehlé, D. (2021). CRYSTALS-Dilithium: Algorithm specifications and supporting documentation (version 3.1). *NIST Post-Quantum Cryptography Standardization Round*, 3.

[Berlekamp et al. 1978] Berlekamp, E. R., McEliece, R. J., and Van Tilborg, H. C. (1978). On the inherent intractability of certain coding problems. *IEEE Transactions on Information Theory*, 24(3):384–386.

[Bernstein et al. 2019a] Bernstein, D. J., Chou, T., Lange, T., Misoczki, R., Niederhagen, R., Persichetti, E., Schwabe, P., Szefer, J., and Wang, W. (2019a). Classic McEliece: conservative code-based cryptography. *NIST submissions*.

[Bernstein et al. 2019b] Bernstein, D. J., Hülsing, A., Kölbl, S., Niederhagen, R., Rijneveld, J., and Schwabe, P. (2019b). The SPHINCS+ signature framework. In *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security*, pages 2129–2146.

[Chen et al. 2020] Chen, C., Danba, O., Hoffstein, J., Hülsing, A., Rijneveld, J., Schanck, J. M., Saito, T., Schwabe, P., Whyte, W., Xagawa, K., Yamakawa, T., and Zhang, Z. (2020). NTRU: Algorithm specifications and supporting documentation. *Brown University and Onboard security company, Wilmington USA*.

[Chen et al. 2016] Chen, L., Jordan, S., Liu, Y.-K., Moody, D., Peralta, R., Perlner, R. A., and Smith-Tone, D. (2016). *Report on post-quantum cryptography*, volume 12. US Department of Commerce, National Institute of Standards and Technology.

[D'Anvers et al. 2018] D'Anvers, J.-P., Karmakar, A., Sinha Roy, S., and Vercauteren, F. (2018). Saber: Module-LWR based key exchange, CPA-secure encryption and CCA-secure KEM. In *Progress in Cryptology–AFRICACRYPT 2018: 10th International Conference on Cryptology in Africa, Marrakesh, Morocco, May 7–9, 2018, Proceedings 10*, pages 282–305. Springer.

[Hoffstein et al. 1998] Hoffstein, J., Pipher, J., and Silverman, J. H. (1998). NTRU: A ring-based public key cryptosystem. In *International Algorithmic Number Theory Symposium*, pages 267–288. Springer.

[Hofheinz et al. 2017] Hofheinz, D., Hövelmanns, K., and Kiltz, E. (2017). A modular analysis of the Fujisaki-Okamoto transformation. In *Theory of Cryptography Conference*, pages 341–371. Springer.

[Lin and Costello 2004] Lin, S. and Costello, D. J. (2004). *Error Control Coding*. Pearson Education.

[Melchor et al. 2021] Melchor, C. A., Aragon, N., Bettaieb, S., Bidoux, L., Blazy, O., Bos, J., Deneuville, J.-C., Dion, A., Gaborit, P., Lacan, J., Persichetti, E., Robert, J.-M., Véron, P., and Zémor, G. (2021). Hamming Quasi-Cyclic: HQC. https://pqc-hqc.org/doc/hqc-specification_2021-06-06.pdf.

[Peikert 2015] Peikert, C. (2015). A decade of lattice cryptography. Cryptology ePrint Archive, Paper 2015/939. https://eprint.iacr.org/2015/939.

[Prest et al. 2020] Prest, T., Fouque, P.-A., Hoffstein, J., Kirchner, P., Lyubashevsky, V., Pornin, T., Ricosset, T., Seiler, G., Whyte, W., and Zhang, Z. (2020). Falcon. *Post-Quantum Cryptography Project of NIST*.

[Shor 1994] Shor, P. W. (1994). Algorithms for quantum computation: Discrete logarithms and factoring. In *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*, pages 124–134. IEEE.

Capítulo 2

Sistemas de Votação Fim-a-Fim: Teoria e Prática

Eduardo Lopes Cominetti (USP), Marcos Antonio Simplicio Junior (USP), Paulo Matias (UFSCar), Roberto Samarone Araújo (UFPA)

Abstract

End-to-End voting systems provide greater security guarantees than the ones traditionally used. However, such systems are slightly used in large-scale elections, for positions such as president or governor. This short course aims to present the benefits and challenges of end-to-end voting systems, considering the Brazilian scenario as its potential target. In particular, the discussion is guided by the study of two examples: a system based on mixnets, similar to the first version of the Helios online voting system, implemented with the Verficatum library; and the ElectionGuard system.

Resumo

Sistemas de votação fim-a-fim fornecem maiores garantias de segurança que os tradicionalmente utilizados. No entanto, tais sistemas ainda são pouco utilizados em eleições grande porte, para cargos políticos concorridos como presidente ou governador. Este minicurso tem como objetivo apresentar os benefícios e desafios de sistemas de votação fim-a-fim, considerando o cenário brasileiro como seu potencial alvo de aplicação. Em particular, a discussão é pautada pelo estudo de dois exemplos: um sistema baseado em redes de misturadores, similar à primeira versão do sistema de votação online Helios, implementado com o auxílio da biblioteca Verificatum; e o sistema ElectionGuard.

2.1. Introdução

Atualmente, a democracia é o regime político legalmente mais adotado em escala global. Nesta forma de governo, o povo tem o direito de exercer o poder político, seja de maneira direta ou indireta. A democracia exercida de maneira indireta, ou democracia representativa, requer que o povo seja capaz de eleger seus representantes, que são legitimizados para atuar em seu lugar por um período determinado. Desta maneira, estes representantes,

durante seu mandato, adquirem grandes poderes para criar, promulgar, ou vetar leis, as quais influenciam milhares ou até mesmo milhões de indivíduos. Consequentemente, é de vital importância garantir que o processo de escolha destes representantes, denominado Eleição, seja realizada de maneira integra, refletindo o real desejo da população.

Dentro do processo de Eleição, o sistema de votação é o responsável pela coleta da escolha do eleitor, denominada Voto, bem como por sua guarda, tabulação e, por fim, a apuração dos resultados. Com o intuito de reduzir custos e agilizar o processo de tabulação de votos e apuração, sistemas eletrônicos de votação foram criados. Apesar da ideia de um sistema eletrônico de votação ser conceitualmente simples, ela envolve uma série de desafios, como descritos por Eddie Perez [Perez 2021]. Entre estes desafios, citam-se como exemplos: a quantidade de dados que o sistema trata; a enorme variedade de (falta de) conhecimento técnico que seus usuários possuem; a necessidade de garantir a segurança deste sistema contra ataques tanto de agentes externos como internos; e a necessidade do sistema ser extremamente preciso, mesmo dependendo de equipamentos sujeitos a falhas. Acima de todos estes desafios, é necessário destacar que a passagem pacífica de poder, algo vital em uma democracia, depende da confiança do público no correto funcionamento deste sistema. Nos últimos anos, este último ponto tem sido um desafio particularmente complexo, como observado em movimentos recentes nos Estados Unidos [Berlinski et al. 2021] e no Brasil [Nicas et al. 2022].

Duas das principais desconfianças do público em relação a um sistema eletrônico de votação são referentes à correta coleta do voto do eleitor e sua guarda, de maneira íntegra, e ao correto uso deste voto para a produção do resultado final. Dentre alguns métodos desenvolvidos para assegurar ao eleitor que estes dois processos estão sendo realizados de maneira correta, têm destaque os sistemas de votação fim-a-fim, ou E2E (*end-to-end*). Esses sistemas são capazes de fornecer ao eleitor evidências de que seu voto foi corretamente coletado e gravado pela equipamento eletrônico durante o momento da votação. Posteriormente, esses sistemas também são capazes de fornecer garantias matemáticas de que o voto do eleitor gravado no equipamento eletrônico foi utilizado na apuração da eleição. Portanto, como o voto foi corretamente gravado e, posteriormente, corretamente utilizado na totalização, um sistema E2E é capaz de promover uma maior confiança na integridade do resultado eleitoral.

Um exemplo de solução E2E bastante conhecido, adotado em muitos cenários de votação onde coerção não é considerada uma ameaça, é o sistema de votação online Helios [Adida 2008]. Outras propostas incluem Prêt à Voter [Ryan et al. 2005], Scantegrity II [Chaum et al. 2008], Punchscan [Popoveniuc and Hosp 2010], Remotegrity [Zagórski et al. 2013], VoteXX [Chaum et al. 2021], e ElectionGuard [Benaloh and Naehrig 2022], criando um ecossistema rico e bastante diverso de soluções. Alguns desses sistemas se baseiam em mecanismos criptográficos, comuns especialmente em propostas voltadas a votação online ou presencial, enquanto outros envolvem dispositivos físicos com características específicas, tendo como foco eleições presenciais.

Neste minicurso, são discutidas as principais características e diferenças entre os métodos de votação fim-a-fim baseados em redes de misturadores e criptografia homomórfica, bem como as ferramentas criptográficas utilizadas por cada um deles. Antes da apresentação destes dois sistemas, uma definição e uma breve discussão de um sistema

de votação fim-a-fim são apresentadas na Seção 2.3.1. Em seguida, um breve histórico de sistemas fim-a-fim é apresentado na Seção 2.3.2. A Seção 2.2 apresenta os blocos criptográficos fundamentais que serão utilizados no trabalho. Nas Seções 2.4 e 2.5, os sistemas Helios e ElectionGuard, que são considerados o estado da arte atualmente, são descritos. Na Seção 2.6, é feita uma comparação entre os dois sistemas, destacando-se as vantagens e desvantagens entre eles. Por fim, a Seção 2.7 apresenta a conclusão deste minicurso, juntamente com os desafios futuros em relação a sistemas de votação fim-a-fim.

2.2. Fundamentos de Criptografia

Nesta seção são apresentados os principais componentes criptográficos necessários para a operação dos sistemas fim-a-fim baseados em redes de misturadores e criptografia homomórfica.

2.2.1. Breve definição: chave simétrica e chave assimétrica

Um esquema de criptografia de chave simétrica utiliza uma única chave, ou chaves facilmente correlacionáveis, para tanto cifrar quanto decifrar um determinado texto.

Um esquema de criptografia de chave assimétrica utiliza duas chaves, uma chamada de chave pública, pois é amplamente divulgada, e uma chamada de chave secreta, pois é mantida em sigilo pelo usuário. Além disso, deve ser computacionalmente difícil de se obter a chave secreta a partir da chave pública. A operação no texto feita por uma das chaves apenas pode ser revertida pela outra. Desta maneira, por exemplo, qualquer usuário pode utilizar a chave pública para enviar uma informação que só poderá ser lida pelo dono da respectiva chave secreta. Por outro lado, o dono da chave secreta também pode utilizar a chave secreta para enviar uma informação que pode ter a origem comprovada através do uso da respectiva chave pública.

2.2.2. Breve definição: esquema determinístico e esquema probabilístico

Em um esquema determinístico, a operação de um determinado texto sempre obtém um mesmo resultado. Por exemplo, o simples uso da cifra de bloco AES para cifrar um texto específico sempre irá produzir o mesmo texto cifrado enquanto a chave permanecer inalterada.

Em um esquema probabilístico, a operação de um determinado texto produz resultados diversos, mesmo quando a chave, ou conjunto de chaves, permace inalterada. Isto em geral ocorre pela utilização de um número aleatório único, conhecido como *nonce*, a cada operação do algoritmo. Como exemplo, o esquema de assinatura ECDSA utiliza um ponto aleatório único a cada operação, de modo que ao assinar um mesmo texto duas vezes, duas assinaturas distintas sejam produzidas e ambas possam ser corretamente verificadas.

2.2.3. Criptosistema ElGamal

A cifra de ElGamal [ElGamal 1985] é um esquema de criptografia de chave assimétrica probabilístico proposto por ElGamal em 1985. O esquema tem sua segurança baseada no problema do logaritmo discreto e, portanto, opera através de exponenciações em um grupo cíclico. Mais precisamente, considere o grupo cíclico $\mathbb{Z}_p$, com p primo, onde

$\mathbb{Z}_p = \mathbb{Z} \mod p$. Um gerador g deste grupo cíclico é um número capaz de produzir todos os elementos deste grupo cíclico, com exclusão do elemento nulo, através de sua exponenciação por um número a:

$$\exists x, a \in \mathbb{Z}_p^* : g^a = x$$

A cifra de ElGamal é composta por três operações: geração de chaves, cifração e decifração.

Geração de chaves: Gera o par de chaves secreta **sk** e pública **pk** do sistema. A chave secreta **sk** neste algoritmo é um número aleatório de $\mathbb{Z}_p^*$. Por sua vez, a chave pública **pk** é definida como $\mathbf{pk} = g^{\mathbf{sk}}$.

Cifração ($\mathbf{Enc_{pk}}(m;y)$): Cifra a mensagem $m \in \mathbb{Z}_p$ através do uso da chave pública **pk** e de um número aleatório y, produzindo o texto cifrado $\mathbf{Enc_{pk}}(m;y) = c = (a,b)$, onde:

$$a = m \cdot \mathbf{pk}^y$$
$$b = g^y$$

Decifração ($\mathbf{Dec_{sk}}(c)$): Decifra o texto cifrado $c = (a,b)$ utilizando a chave secreta **sk**, produzindo a mensagem $\mathbf{Dec_{sk}}(c) = m' \in \mathbb{Z}_p$:

$$b' = b^{\mathbf{sk}} = g^{y \cdot \mathbf{sk}} = \mathbf{pk}^y$$
$$m' = a \cdot (b')^{-1} = m \cdot \mathbf{pk}^y \cdot (\mathbf{pk}^y)^{-1} = m$$

2.2.3.1. Propriedade de homomorfismo da cifra de ElGamal

Um leitor atento é capaz de percerber que dados $c_1 = (a_1, b_1)$ e $c_2 = (a_2, b_2)$, que cifram, respectivamente, as mensagens m_1 e m_2, utilizando y_1 e y_2, a multiplicação entre si destes dois textos cifrados produz $c_3 = (a_3, b_3)$, que cifra a mensagem $m_3 = m_1 \cdot m_2$, utilizando $y_3 = y_1 + y_2$. Logo, o esquema de ElGamal possuí um homomorfismo multiplicativo do texto às claras através da multiplicação de textos cifrados:

$$a_3 = a_1 \cdot a_2 = m_1 \cdot \mathbf{pk}^{y_1} \cdot m_2 \cdot \mathbf{pk}^{y_2} = m_1 \cdot m_2 \cdot \mathbf{pk}^{y_1+y_2} = m_3 \cdot \mathbf{pk}^{y_3}$$
$$b_3 = b_1 \cdot b_2 = g^{y_1} \cdot g^{y_2} = g^{y_1+y_2} = g^{y_3}$$

Observa-se também que a decifração também ocorre de maneira correta através do mesmo processo descrito anteriormente:

$$b'_3 = b_3^{\mathbf{sk}} = g^{y_3 \cdot \mathbf{sk}} = \mathbf{pk}^{y_3}$$
$$m'_3 = a_3 \cdot (b'_3)^{-1} = m_3 \cdot \mathbf{pk}^{y_3} \cdot (\mathbf{pk}^{y_3})^{-1} = m_3$$

2.2.3.2. Transformando a propriedade de homomorfismo de multiplicação para soma: ElGamal Exponencial

Apesar do homomorfismo multiplicativo ser útil em algumas situações, em um cenário de votação, o homomorfismo de soma, ou seja, a operação nos textos cifrados resulta em

uma soma dos textos às claras, é o ideal. Isto se deve ao fato de se desejar utilizar o esquema para somar os votos para produzir o resultado da eleição.

Felizmente, é possível transformar o homomorfismo multiplicativo da cifra de ElGamal para um homomorfismo de soma, contanto que as mensagens a serem cifradas sejam pequenas. Para tanto, ao invés de simplesmente cifrar as mensagens m_i, um usuário cifra as mensagens $m'_i = g^{m_i}$. Consequentemente, a multiplicação de c_1 por c_2 produz c_3, que cifra a mensagem $m'_3 = g^{m_1} \cdot g^{m_2} = g^{m_1+m_2} = g^{m_3}$. Esta variante da cifra de ElGamal recebe o nome de ElGamal Exponencial.

É importante observar que, após a decifração de c_3, é necessário calcular o logaritmo discreto da mensagem m'_3 resultante para a obtenção da soma das mensagens m_1 e m_2. Apesar do cálculo do logaritmo discreto ser um problema computacionalmente difícil, ele é possível de ser realizado contanto que as mensagens tenham um tamanho relativamente pequeno. Por exemplo, a eleição brasileira conta com cerca de 150 milhões de eleitores cadastrados, o que corresponde a uma mensagem de tamanho máximo inferior a 2^{28}. Este tamanho de mensagem ainda é considerado pequeno e apto para ser computacionalmente calculável ou mantido em uma tabela para consulta. Logo, a aplicação do ElGamal Exponencial é viável para o cenário eleitoral.

2.2.3.3. Recifração de texto cifrado por ElGamal

É possível utilizar a propriedade de homomorfismo em ambas as variantes da cifra de ElGamal para recifrar um determinado texto cifrado c_1, produzindo $c_3 \neq c_1$, mas com $m_3 = m_1$. Este fato é útil, pois é inviável para um observador externo relacionar c_1 e c_3. Em outras palavras, caso um observador veja três mensagens, c_1, c'_1 e c_3, ele não é capaz de dizer se c_3 foi produzido utilizando c_1, c'_1,ou até mesmo um terceiro c''_1 desconhecido. Este fato será utilizado nas provas de conhecimento zero relacionadas a redes de misturadores apresentadas futuramente neste texto.

Para realizar a recifração, basta escolher y_2 aleatório e calcular a cifração da mensagem $m_2 = 1$, ou $m'_2 = 1$ para o ElGamal Exponencial, obtendo c_2. Posteriormente, basta realizar a multiplicação de c_1 e c_2, obtendo-se c_3:

$$a_3 = a_1 \cdot a_2 = m_1 \cdot \mathbf{pk}^{y_1} \cdot 1 \cdot \mathbf{pk}^{y_2} = m_1 \cdot \mathbf{pk}^{y_1+y_2} = m_3 \cdot \mathbf{pk}^{y_3}$$
$$b_3 = b_1 \cdot b_2 = g^{y_1} \cdot g^{y_2} = g^{y_1+y_2} = g^{y_3}$$

2.2.3.4. ElGamal com decifração limiar (*Threshold ElGamal*)

Ao invés do par de chaves na cifra de ElGamal ser constituída por uma única chave secreta, é possível criar um conjunto de chaves secretas e uma única chave pública associada. Este é um fato interessante, pois enquanto a cifração de uma mensagem é executada de maneira análoga a construção com chave única, a decifração do texto cifrado requer um número mínimo pré-estabelecido destas chaves para ser realizada. A esta variante que possuí múltiplas chaves secretas com a necessidade de uma quantidade mínima delas para operar a decifração, é dado o nome de ElGamal com decifração limiar (*Threshold ElGamal*).

Primeiramente, é estabelecido quantas chaves existirão no sistema ao todo, n, e quantas são necessárias para realizar a decifração, k. Em geral, cada chave é criada por uma entidade independente. Para cada entidade i, temos que a chave secreta $\mathbf{sk}_i$ é um número aleatório de $\mathbb{Z}_p^*$, e a chave pública é definida como $\mathbf{pk}_i = g^{\mathbf{sk}_i}$. Por sua vez, a chave pública combinada é simplesmente o produtório de todas as chaves públicas individuais:

$$\mathbf{pk} = \prod_{i=1}^{n} \mathbf{pk}_i$$

Como segundo passo, a entidade i cria um polinômio P_i de grau $k-1$, sendo o coeficiente de grau 0 igual a chave secreta $\mathbf{sk}_i$ e os demais coeficientes $a_{i,j}$ escolhidos aleatoriamente em $\mathbb{Z}_p^*$:

$$P_i(x) = \prod_{j=1}^{k-1} a_{i,j} \cdot x^j$$

Para cada outra entidade ℓ, a entidade i calcula $P_i(\ell)$ e envia-o para ℓ.

Por fim, para decifrar um texto cifrado usando este esquema de chaves, temos dois casos: quando todas as entidades estão presentes e quando apenas o limiar delas encontra-se presente.

Caso todas elas estejam presentes, e lembrando que $\mathbf{Enc}_{\mathbf{pk}}(m;r) = (a,b) = (m \cdot \mathbf{pk}^r, g^r)$, cada entidade calcula $b^{\mathbf{sk}_i}$ e publica este resultado. Basta então combinar todos estes fatores através de um produtório, inverter o resultado e multiplicá-lo por a, obtendo a mensagem m:

$$a \cdot (\prod_{i=1}^{n} b^{\mathbf{sk}_i})^{-1} = m \cdot \mathbf{pk}^r \cdot (\prod_{i=1}^{n} g^{r \cdot \mathbf{sk}_i})^{-1} = m \cdot \mathbf{pk}^r \cdot ((g^{\sum_{i=1}^{n} \mathbf{sk}_i})^r)^{-1} =$$
$$= m \cdot \mathbf{pk}^r \cdot (\mathbf{pk}^r)^{-1} = m$$

Caso apenas um limiar destas entidades estejam presentes, estas entidades utilizam o resultado do polinomio enviado anteriormente pelas entidades faltantes para decifrar o texto cifrado. Isto é feito através do cálculo de parcelas do valor que deveria ser retornado pelas entidades ausentes. Caso a entidade i não esteja presente, uma entidade ℓ prossegue calculando e publicando o valor parcial $M_{i,\ell}$:

$$M_{i,\ell} = b^{P_i(\ell)}$$

Então, é calculado o valor do coeficiente de Lagrange para um conjunto de k entidades disponíveis $\{\ell \in U, |U| = k\}$:

$$\omega_\ell = \prod_{j \in (U - \{\ell\})} \frac{j}{j - \ell}$$

Por fim, o valor final referente a entidade ausente é computado como:

$$M_i = \prod_{\ell \in U} (M_{i,\ell})^{\omega_\ell}$$

Este valor M_i é utilizado no lugar de $b^{\mathbf{sk}_i}$ e a decifração é realizada como se todas as entidades estivessem presentes.

2.2.4. Esquema de Compromisso de Pedersen

O compromisso de Pedersen é um esquema baseado em um grupo cíclico. Um esquema de compromisso é um algoritmo que permite que um usuário se comprometa com uma ou mais mensagens perante outros usuários, mas sem revelar os seus valores. Posteriormente, o valor das mensagens pode ser revelado e pode-se averiguar que eles são condizentes com o compromisso previamente informado. O compromisso de Pedersen possuí a propriedade de esconder de forma perfeita ("*perfectly hiding*") e a propriedade de ligação de forma computacional ("*computationally binding*").

No compromisso de Pedersen, deseja-se criar um compromisso para n mensagens. Para tanto, é necessário escolher $n+1$ geradores indepentes do grupo cíclico, $g, h_1, h_2, \ldots, h_n$. Geradores independentes são geradores cuja relação entre eles é desconhecida, ou seja, não se conhece qual o valor do logaritmo discreto de um gerador em relação a outro. Estes geradores são tornados públicos.

Após a escolha dos geradores, um número aletório único r é escolhido o compromisso de Pedersen das n mensagens $\mathbf{m} = (m_1, m_2, \ldots, m_n)$, $\mathbf{Com}(\mathbf{m}; r)$, pode ser calculado por:

$$\mathbf{Com}(\mathbf{m}; r) = g^r \prod_{i=1}^{n} h_i^{m_i}$$

Observa-se que caso o grupo de mensagens $\mathbf{m}$ possua apenas uma mensagem x não nula, cujo valor seja $m_x = 1$, o compromisso pode ser simplesmente calculado por:

$$\mathbf{Com}(\mathbf{m}; r) = g^r \cdot h_x$$

Este fato será utilizado nas provas de conhecimento zero relacionadas a redes de misturadores apresentadas futuramente neste texto.

2.2.5. Provas de Conhecimento Não Interativas

Prova de conhecimento são em geral interativas, ou seja, elas requerem que o indivíduo que está realizando a prova interaja com o verificador durante a sua execução. Por meio da heuristica de Fiat-Shamir [Fiat and Shamir 1986], no entanto, provas de conhecimento zero interativas podem ser convertidas em provas não interativas. Dessa forma, o realizador da prova não precisa estar disponível no momento da verificação da prova. A seguir são apresentadas três provas de conhecimento não interativas utilizadas nesse minicurso.

2.2.5.1. Prova de Chaum-Pedersen

A prova de conhecimento zero de Chaum-Pedersen [Chaum and Pedersen 1992] permite provar o valor do texto às claras de um texto cifrado com ElGamal. Para isto, é necessário o conhecimento do número aleatório utilizado na cifração ou o conhecimento da chave privada.

Considere o criptosistema de ElGamal Exponencial, definido no grupo cíclico $\mathbb{Z}_p$,com p primo. Adicionalmente, considere o subgrupo cíclico de resíduos r deste grupo, $\mathbb{Z}_p^r = \{y \in \mathbb{Z}_p^*, \exists x \in \mathbb{Z}_p^* : y = x^r \mod p\}$, e seu gerador g. Por fim, seja o primo $q =$

$(p-1)/r$ e o máximo divisor comum entre q e r seja 1. Nesta configuração, a mensagem m a ser cifrada pertence ao subgrupo cíclico de resíduos r deste grupo, $\mathbb{Z}_p^r$ e o número aleatório a ser utilizado pertence a $\mathbb{Z}_q$.

Para a prova utilizando o conhecimento do número aleatório r, considerando o texto cifrado $\mathbf{Enc_{pk}}(m,r) = (g^m \cdot \mathbf{pk}^r, g^r) = (a,b)$, o provador escolhe um valor aleatório $u \in \mathbb{Z}_q$ e cria um compromisso deste valor através da tupla $(a',b') = (\mathbf{pk}^u, g^u)$. Em seguida, cria-se um desafio c, utilizando-se o método de Fiat-Shamir, com $c = H(a,b,a',b')$, sendo H uma função de hash criptográfico. Por fim, o provador calcula $v = u + c \cdot r$ e publica os valores (a',b',v)

O verificador, por sua vez, consegue confirmar a prova através da checagem das seguintes igualdades:

$$g^{m \cdot c} \cdot \mathbf{pk}^v \equiv a' \cdot a^c$$
$$g^v \equiv b' \cdot b^c$$

Para a prova utilizando o conhecimento da chave secreta **sk**, considerando o texto cifrado $\mathbf{Enc_{pk}}(m,r) = (g^m \cdot \mathbf{pk}^r, g^r) = (a,b)$, o provador escolhe um valor aleatório $u \in \mathbb{Z}_q$ e cria um compromisso deste valor através da tupla $(a',b') = (b^u, g^u)$. Em seguida, o provador calcula $d = b^{\mathbf{sk}}$ e cria um desafio c, com $c = H(a,b,a',b',d)$, de forma análoga a anterior. Por fim, o provador calcula $v = u + c \cdot \mathbf{sk}$ e publica os valores (a',b',d,v)

O verificador, por sua vez, consegue confirmar a prova através da checagem das seguintes igualdades:

$$g^v \equiv b' \cdot \mathbf{pk}^c$$
$$b^v \equiv a' \cdot d^c$$

2.2.5.2. Prova de Cramer-Damgård-Schoenmakers

A técnica de Cramer-Damgård-Schoenmakers [Cramer et al. 1994] permite provar a disjunção de dois predicados. Isto significa que dado duas provas de conhecimento zero referentes a um objeto, contraditórias entre si, uma delas é verdadeira, mas não se sabe qual.

Para isto, a técnica utiliza o fato de ser possível construir provas que verifiquem corretamente para afirmações falsas caso o desafio a ser escolhido seja previamente conhecido. Para garantir que em duas provas uma delas possua um valor pré-definido pelo provador e outra possua um valor aleatório, utiliza-se uma composição de um desafio global aleatório. Caso um desafio global c seja escolhido utilizando-se o método de Fiat-Shamir, é possível escolher um valor c_0 para um dos desafios, enquanto o outro desafio c_1 corresponde ao complemento deste valor em relação a c, ou seja, $c = c_0 + c_1$. Consequentemente, apesar de c_0 ser definido pelo provador, o seu complemento c_1 em relação a um aleatório c continua aleatório.

Neste minicurso, a técnica de Cramer-Damgård-Schoenmakers é utilizada para provar que um texto cifrado com ElGamal Exponencial corresponde a mensagem 0 ou a mensagem 1, tendo conhecimento do valor aleatório r. As construções para os dois casos, um com uma mensagem verdadeiramente 0 e um com uma mensagem verdadeiramente 1 são mostradas a seguir.

Para uma **mensagem de valor 0**, representada por $\mathbf{Enc_{pk}}(0,r) = (\mathbf{pk}^r, g^r) = (a,b)$, o provador escolhe $u,v,w \in \mathbb{Z}_q$ e cria dois compromissos:

$$(a_0,b_0) = (\mathbf{pk}^u, g^u)$$
$$(a_1,b_1) = (g^w \cdot \mathbf{pk}^v, g^v)$$

Em seguida, o valor do desafio global é definido através do cálculo de $c = H(a,b,a_0,b_0, a_1,b_1)$, seguindo a heurística de Fiat-Shamir. Por fim, o provador calcula os seguintes valores e publica (c_0,c_1,v_0,v_1):

$$c_0 = c - w$$
$$c_1 = w$$
$$v_0 = u + c_0 \cdot r$$
$$v_1 = v + c_1 \cdot r$$

O verificador, por sua vez, é capaz de confirmar o conjunto de provas validando as igualdades:

$$c = c_0 + c_1$$
$$(a_0 \cdot a^{c_0}, b_0 \cdot b^{c_0}) \equiv (\mathbf{pk}^u \cdot \mathbf{pk}^{c_0 \cdot r}, g^u \cdot g^{c_0 \cdot r}) \equiv (\mathbf{pk}^{v_0}, g^{v_0})$$
$$(a_1 \cdot a^{c_1}, b_1 \cdot b^{c_1}) \equiv (g^w \cdot \mathbf{pk}^v \cdot \mathbf{pk}^{c_1 \cdot r}, g^v \cdot g^{c_1 \cdot r}) \equiv (g^{c_1} \cdot \mathbf{pk}^{v_1}, g^{v_1})$$

Para uma **mensagem de valor 1**, representada por $\mathbf{Enc_{pk}}(1,r) = (g \cdot \mathbf{pk}^r, g^r) = (a,b)$, o provador escolhe $u,v,w \in \mathbb{Z}_q$ e cria dois compromissos:

$$(a_0,b_0) = (g^w \cdot \mathbf{pk}^v, g^v)$$
$$(a_1,b_1) = (\mathbf{pk}^u, g^u)$$

Em seguida, o valor do desafio global é definido através do cálculo de $c = H(a,b,a_0,b_0, a_1,b_1)$, seguindo a heurística de Fiat-Shamir. Por fim, o provador calcula os seguintes valores e publica (c_0,c_1,v_0,v_1):

$$c_0 = q - w$$
$$c_1 = c + w$$
$$v_0 = v + c_0 \cdot r$$
$$v_1 = u + c_1 \cdot r$$

O verificador, por sua vez, é capaz de confirmar o conjunto de provas validando as igualdades:

$$c = c_0 + c_1$$
$$(a_0 \cdot a^{c_0}, b_0 \cdot b^{c_0}) \equiv (g^w \cdot \mathbf{pk}^v \cdot g^{c_0} \cdot \mathbf{pk}^{c_0 \cdot r}, g^v \cdot g^{c_0 \cdot r}) \equiv (\mathbf{pk}^{v_0}, g^{v_0})$$
$$(a_1 \cdot a^{c_1}, b_1 \cdot b^{c_1}) \equiv (\mathbf{pk}^u \cdot g^{c_1} \cdot \mathbf{pk}^{c_1 \cdot r}, g^u \cdot g^{c_1 \cdot r}) \equiv (g^{c_1} \cdot \mathbf{pk}^{v_1}, g^{v_1})$$

2.2.5.3. Prova de embaralhamento de Wikström-Terelius

A prova de conhecimento zero de embaralhamento de Wikström-Terelius [Terelius and Wikström 2010] busca provar que duas listas de textos cifrados distintas, $\mathbf{e} = (e_1, e_2, \ldots, e_n)$

e $\mathbf{e}' = (e'_1, e'_2, \ldots, e'_n)$, contém os mesmos textos às claras $\mathbf{m} = (m_1, m_2, \ldots, m_n)$, mas em ordem permutada.

Para isto, considere a permutação $\psi : \{1, \ldots, n\} \to \{1, \ldots, n\}$ e a matriz de permutação de $\mathbf{B}_\psi = (b_{ij})_{n \times n}$, onde:

$$b_{ij} = \begin{cases} 1, & \text{se } \psi(i) = j, \\ 0, & \text{caso contrário.} \end{cases}$$

Ou seja, a coluna j da linha i é igual a 1 caso e'_j represente e_i e 0 caso contrário. A lista $\mathbf{e}$ é então multiplicada por esta matriz e os elementos resultantes são recifrados, produzindo a lista $\mathbf{e}'$. Desta maneira, é necessário provar que: 1) a matriz de permutação possui apenas um único elemento 1 em cada linha e em cada coluna; 2) a lista original $\mathbf{e}$ foi multiplicada por esta matriz; e 3) a lista $\mathbf{e}'$ é a recifração dos elementos do resultado de 2. Para construir estas provas será utilizado o compromisso de Pedersen previamente apresentado.

Sendo a coluna da matriz representada por $\mathbf{b_j} = (b_{1,j}, \ldots, b_{n,j})$, temos que o compromisso de cada coluna com aleatoriedade r_j é dado por:

$$\mathbf{Com}(\mathbf{b_j}; r_j) = g^{r_j} \prod_{i=1}^{n} h_i^{b_{ij}} = g^{r_j} \cdot h_i$$

Adicionalmente, considerando que $\mathbf{r} = (r_1, \ldots, r_n)$, adota-se a seguinte notação como o compromisso da matriz de permutação:

$$\mathbf{Com}(\psi; \mathbf{r}) = (\mathbf{Com}(\mathbf{b_1}; r_1), \ldots, \mathbf{Com}(\mathbf{b_n}; r_n)) = (c_1, \ldots, c_n) = \mathbf{c}$$

Para provar que a matriz de permutação $\mathbf{B}_\psi = (b_{ij})_{n \times n}$ possui apenas um único elemento 1 em cada linha e cada coluna, considere uma lista auxiliar de elementos aleatórios $\mathbf{x} = (x_1, \ldots, x_n)$. A afirmação é verdadeira se e somente se:

$$\sum_{j=1}^{n} b_{ij} = 1, \text{para } 1 \leq i \leq n$$
$$\prod_{i=1}^{n} \sum_{j=1}^{n} b_{ij} \cdot x_i = \prod_{i=1}^{n} x_i$$

Lembrando que $\mathbf{c}$ corresponde ao compromisso da matriz de permutação e considerando $\mathbf{r} = (r_1, \ldots, r_n)$ e $\bar{r} = \sum_{j=1}^{n} r_j$, é possível observar que podemos representar a primeira condição como (notar que a somatória de b_{ij} aparece como exponente de h_i):

$$\prod_{j=1}^{n} c_j = \prod_{i=1}^{n} g^{r_j} \prod_{i=1}^{n} h_i^{b_{ij}} = g^{\sum_{j=1}^{n} r_j} \prod_{i=1}^{n} h_i^{\sum_{j=1}^{n} b_{ij}} = g^{\bar{r}} \prod_{i=1}^{n} h_i = \mathbf{Com}(\mathbf{1}, \bar{r}) \tag{1}$$

Por sua vez, considerando uma lista de valores arbitrários $\mathbf{u} = (u_1, \ldots, u_n)$ e o resultado de sua multiplicação pela matriz de permutação $\mathbf{u}' = (u'_1, \ldots, u'_n)$, e sendo $\tilde{r} = \sum_{j=1}^{n} r_j \cdot u_j$, a segunda condição pode ser representada pelas equações (notar que o uso de c_j relaciona as duas próximas equações com a anterior):

$$\prod_{i=1}^{n} u_i = \prod_{i=1}^{n} u'_i \tag{2}$$

$$\prod_{j=1}^{n} c_j^{u_j} = \prod_{j=1}^{n} (g^{r_j} \prod_{i=1}^{n} h_i^{b_{ij}})^{u_j} = g^{\sum_{j=1}^{n} r_j \cdot u_j} \prod_{i=1}^{n} h_i^{\sum_{j=1}^{n} b_{ij} \cdot u_j} = g^{\tilde{r}} \prod_{i=1}^{n} h_i^{u'_i} = \\ = \mathbf{Com}(\mathbf{u}', \tilde{r}) \tag{3}$$

Por fim, recorda-se que $\mathbf{e}'$ é a lista permutada e recifrada da lista original $\mathbf{e}$, que estas listas tem como elementos textos cifrados utilizando ElGamal e que uma recifração em ElGamal é obtida através da multiplicação do texto cifrado por uma cifração do valor 1 utilizando um novo valor aleatório. Sendo o valor aleatório r'_j o valor empregado em cada e_j para produzir seu elemento equivalente na lista $\mathbf{e}'$, utilizando os valores $\mathbf{u}$ e $\mathbf{u}'$ definidos anteriormente, e sendo $r' = \sum_{j=1}^{n} = r'_j \cdot u_j$, para provar que $\mathbf{e}'$ é uma recifração de $\mathbf{e}$, temos:

$$\prod_{i=1}^{n} (e'_i)^{u'_i} = \prod_{j=1}^{n} (e_j \cdot \mathbf{Enc}_{pk}(1; r'_j))^{u_j} = \prod_{j=1}^{n} (e_j^{u_j} \cdot \mathbf{Enc}_{pk}(1; r'_j)^{u_j}) = \\ = \prod_{j=1}^{n} (e_j^{u_j} \cdot \mathbf{Enc}_{pk}(1; r'_j \cdot u_j)) = \mathbf{Enc}_{pk}(1; r') \cdot \prod_{j=1}^{n} e_j^{u_j} \tag{4}$$

Assim, a prova de conhecimento zero de embaralhamento de Wikström-Terelius consiste em demonstrar, utilizando uma prova de conhecimento não interativa de pré-imagem, estas quatro equações:

$$\text{NIZKP} \left[(\bar{r}, \tilde{r}, r', \mathbf{u}') : \begin{array}{l} \prod_{j=1}^{n} c_j = \mathbf{Com}(\mathbf{1}, \bar{r}) \\ \wedge \prod_{i=1}^{n} u_i = \prod_{i=1}^{n} u'_i \\ \wedge \prod_{j=1}^{n} c_j^{u_j} = \mathbf{Com}(\mathbf{u}', \tilde{r}) \\ \wedge \prod_{i=1}^{n} (e'_i)^{u'_i} = \mathbf{Enc}_{pk}(1; r') \cdot \prod_{j=1}^{n} e_j^{u_j} \end{array} \right]$$

Por questões de espaço, a transformação destas equações em provas de conhecimento zero não é mostrada neste minicurso. Entretanto, esta transformação, junto com um pseudo-código para sua implementação, encontra-se descrita no artigo [Haenni et al. 2017].

2.3. Sistemas de Votação Fim-a-Fim (E2E) e sua Evolução Histórica

Esta Seção apresenta as características de um sistema de votação fim-a-fim e como estas características buscam atender as duas principais propriedades desejadas de um sistema de votação: o **sigilo** do voto e a **integridade** do voto e do resultado.

Também são discutidas as diferenças entre o modelo de votação fim-a-fim e o mecanismo de Trilha em Papel para Auditoria Verificada pelo Eleitor (do inglês *Voter Verified Paper Audit Trail*, ou VVPAT).

Por fim, são apresentados alguns dos sistemas de votação que implementam a abordagem fim-a-fim, tanto de maneira mecânica (votação em papel) quanto de maneira eletrônica.

2.3.1. Sistemas de Votação Fim-a-Fim (E2E)

De acordo com Comissão de Assistência Eleitoral Amedicana (EAC) [Commission 2023], um sistema de votação fim-a-fim (E2E) é um sistema que permite com que eleitores individuais sejam capazes de verificar elementos cruciais de uma apuração eleitoral sem que sejam necessária a confiança no software de votação, no hardware de votação, em servidores do órgão eleitoral, ou até mesmo em observadores externos. Para atingir este objetivo, um sistema de votação E2E deve fornecer as seguintes propriedades:

- Voto é lançado como pretendido pelo eleitor – permite que o eleitor verifique que seu voto foi corretamente interpretado pelo sistema de votação enquanto estiver no local de votação;
- Voto é gravado como foi lançado - permite que o eleitor verifique que seu voto foi corretamente gravado pelo sistema de votação e incluído em um registro de votos público;
- Voto é apurado como foi gravado - o sistema eleitoral providencia um método público de apuração que utiliza o registro de votos público.

Com estas propriedades, cada eleitor é capaz de verificar, de maneira individual, que sua escolha de candidato foi corretamente salva pelo sistema e posteriormente corretamente utilizada pelo sistema para produzir a apuração da eleição. Portanto, um sistema E2E é capaz de prover garantias referentes à **integridade** do processo eleitoral.

Para satisfazer estas três propriedades E2E, os dois sistemas de enfoque deste trabalho adotam procedimentos similares, embora com o uso de técnicas diferentes. Consequentemente, nesta Seção a descrição destes procedimentos será feita de maneira única e as técnicas utilizadas para a suas implementações serão aprofundadas nas respectivas seções individuais para cada método.

2.3.1.1. Voto é lançado como pretendido pelo eleitor

Primeiramente, para prover garantias que o voto é lançado como pretendido pelo eleitor, o sistema E2E cifra este voto, utilizando um esquema assimétrico probabilístico de cifração. Na sequência, é computado um hash criptográfico deste valor cifrado, juntamente com alguns dados adicionais públicos. O resultado desta operação, chamado de **Código de Rastreio**, por razões que serão explicadas adiante, é então fornecido ao eleitor em conjunto com os dados adicionais (e.g., através da impressão de um código QR). Até o presente momento, o sistema ainda não realizou a gravação do voto do eleitor. Antes de realizar a gravação do voto, o sistema fornece ao eleitor a capacidade de "desafiar" o voto inserido, procedimento conhecido na literatura como "Desafio de Benaloh" [Benaloh 2006]. O eleitor então possui duas escolhas: 1) desafiar o voto associado ao Código

de Rastreio previamente fornecido; ou 2) permitir que o sistema salve (deposite) o voto cifrado associado ao Código de Rastreio previamente fornecido.

Caso o eleitor opte pelo desafio, o sistema fim-a-fim fornece ao eleitor uma informação extra (i.e., o número único utilizado para realizar a cifração do voto). Esta informação permite que o eleitor, em um dispositivo de sua posse, executando um programa de verificação fornecido por alguma entidade na qual ele confie (ou até mesmo um programa escrito por ele próprio), realize a verificação de correta construção do Código de Rastreio. Para isto, basta que o programa repita os passos do sistema de votação: com o voto do eleitor e o número aleatório utilizado originalmente na cifração, o verificador cifra o voto utilizando o esquema assimétrico e computa seu hash criptográfico. Realizado este procedimento, o programa então compara o resultado do hash que ele computou com o Código de Rastreio fornecido pelo sistema. Caso a comparação seja bem sucedida, o sistema originalmente comportou-se de maneira honesta e correta e utilizou o voto fornecido pelo eleitor. Esta afirmação é possível de ser feita pois para o sistema fornecer uma informação adicional que gere o mesmo Código de Rastreio para um voto diferente do inserido pelo eleitor, o sistema teria que ser capaz de resolver o problema matemático de segurança criptográfica associado ao algoritmo de cifração ou ser capaz de calcular a segunda pré-imagem do hash criptográfico utilizado.

Adicionalmente, uma vez que o eleitor opte pelo desafio, o sistema E2E apaga o voto do eleitor e todos os dados gerados e exige que o eleitor reinicie o processo de votação. Desta forma, toda vez que um desafio é realizado, o eleitor precisa reinserir seu voto no sistema e um novo Código de Rastreio, totalmente independente do Código de Rastreio anterior, é gerado. Isto é realizado para garantir o sigilo do voto do eleitor, afinal, caso o eleitor pudesse gravar um voto já desafiado, ele seria capaz de provar para qualquer pessoa em qual candidato ele votou. Consequentemente, o Desafio de Banaloh é um processo que fornece uma garantia estatística ao eleitor de que seu voto foi lançado como pretendido. Como o sistema não sabe se poderá gravar ou não o voto do eleitor antes de lhe fornecer o Código de Rastreio, caso o sistema se comporte de maneira errônea, haverá uma possibilidade do erro ser detectado. Uma análise simplificada desta garantia estatística é apresentada na Seção 2.3.1.5.

2.3.1.2. Voto é gravado como foi lançado

Para fornecer uma prova de que o voto é gravado como foi lançado, uma vez que o eleitor permita com que o sistema salve o voto associado ao Código de Rastreio, o sistema assina digitalmente o Código de Rastreio fornecido ao eleitor.

Uma assinatura digital é um algoritmo criptográfico que fornece o serviço de irretratabilidade a um sistema. Desta maneira, uma vez que o Código de Rastreio seja assinado pelo sistema fim-a-fim, o eleitor é capaz de provar, além de uma dúvida razoável, de que o voto cifrado associado a este Código de Rastreio deveria constar no registro de votos do sistema. Adicionalmente, está é a razão pelo nome do Código de Rastreio: através dele é possível rastrear que o voto cifrado do eleitor encontra-se no registro público da eleição. Como ele permite rastrear apenas o voto cifrado, o Código de Rastreio não compromete o sigilo do voto do eleitor.

Por fim, é importante notar que um eleitor malicioso não é capaz de produzir um Código de Rastreio falso para alegar que um possível voto não consta no registro público do sistema. Embora todas as etapas de criação do Código de Rastreio possam ser reproduzidas com sucesso fora do sistema, o eleitor malicioso não é capaz de gerar uma assinatura digital válida para este Código de Rastreio. Consequentemente, este tipo de ataque é mitigado em um sistema fim-a-fim.

2.3.1.3. Voto é apurado como foi gravado

Por fim, para fornecer uma prova de que o voto é apurado como foi gravado, um sistema E2E demonstra matematicamente que os votos cifrados no registro público foram utilizados na soma do resultado. Este processo envolve a decifração dos votos individuais ou de um agregado de votos cifrados somados homomorficamente, sem que esta decifração comprometa o sigilo do voto. A maneira que esta etapa é realizada é o principal diferencial entre os dois métodos estudados neste minicurso.

Como esperado, um sistema baseado em rede de misturadores embaralha os votos cifrados antes de decifrá-los, provando que o embaralhamento foi feito de maneira correta (i.e., nenhum voto foi removido ou inserido). Uma vez que os votos são decifrados, a apuração do resultado ocorre através do simples tabulamento e soma dos votos.

Por sua vez, um sistema baseado em criptografia homomórfica simplesmente soma todos os votos cifrados e realiza a decifração do agregado de votos. Desta maneira, uma vez que a decifração ocorra, o resultado final da eleição já encontra-se calculado.

Em ambos os casos, o sistema também prova que a decifração dos votos foi feita de maneira correta.

2.3.1.4. Sobre o Código de Rastreio

Como mencionado anteriormente, o Código de Rastreio é o resultado de um hash criptográfico que inclui o voto cifrado e alguns dados adicionais. Estes dados adicionais são o Código de Rastreio do último voto gravado no sistema, criando uma cadeia de hashes dos Códigos de Rastreio, e o instante de tempo no qual o eleitor inseriu o voto no sistema. Estes dados são incluídos no calculo do Código de Rastreio para prevenir dois ataques: o ataque de preplay do voto cifrado e um ataque de descarte do Código de Rastreio.

Um ataque de preplay permite com que o sistema desvie votos. Para isto, o sistema cria um voto cifrado para o candidato A, cria o Código de Rastreio para este voto cifrado, assina-o e lança este voto no sistema. Este Código de Rastreio, então, seria exibido para múltiplos eleitores. Sempre que um eleitor fizesse a escolha por candidato A, o sistema apresentaria esse Código de Rastreio para o eleitor. Caso o eleitor deposite este voto, a urna entrega a assinatura do Código de Rastreio para o eleitor, mas ela é capaz de criar um novo voto, com qualquer escolha de candidato, e depositá-lo no lugar do voto do eleitor. Caso o eleitor verifique o registro de votos público, ele será capaz de encontrar o Código de Rastreio provido pelo sistema e ver que o voto associado a este Código de Rastreio foi utilizado na apuração. Desta maneira, o sistema é capaz de criar um único voto que será

utilizado por todos os eleitores que fizerem a escolha do candidato A, enquanto desvia votos para outros candidatos.

Em um ataque de descarte do Código de Rastreio, votos no sistema podem ser substituídos. No momento anterior ao encerramento da votação, uma pessoa poderia procurar por todos os Códigos de Rastreio descartados pelos eleitores, como por exemplo, os Códigos de Rastreio que eleitores desinteressados em verificar o resultado jogaram no lixo. Pelo fato destes Códigos de Rastreio estarem descartados, sua validade nunca será checada perante o registro de votos público. Desta forma, estes Códigos de Rastreio podem ser apagados e substituídos no sistema. Uma vez realizada a substituição, a votação é encerrada normalmente, produzindo o registro público e as provas matemáticas de que os votos cifrados, agora alterados, foram utilizados para produzir o resultado.

O uso do Código de Rastreio em uma cadeia de hashes, que inclui o Código de Rastreio anterior, o instante de inserção do voto no sistema e o voto cifrado, previne ambos os ataques. Como o Código de Rastreio é criado usando o instante de inserção do voto no sistema, é possível verificar, ao receber o Código de Rastreio, que ele foi recém criado. Consequentemente, o sistema não conseguiria apresentar para múltiplos eleitores o mesmo Código de Rastreio, pois o eleitor conseguiria verificar que o instante de tempo presente no Código de Rastreio difere significativamente do momento atual.

Analogamente, o Código de Rastreio em uma cadeia de hashes previne o ataque de descarte do Código de Rastreio. Uma vez que o Código de Rastreio se encontra em uma cadeia de hashes, uma substituição de um Código de Rastreio descartado necessitaria que o sistema fosse capaz de substituir um elemento da cadeia sem com que os próximos elementos fossem afetados, o que é implausível. Esta é a mesma característica que garante que transações feitas em Blockchains não possam ser modificadas.

2.3.1.5. Análise do desafio de Benaloh

A seguir são apresentadas as análises estatísticas referente ao processo de desafio de Benaloh e a simulação de seu uso em casos reais.

Como mencionado anteriormente, o desafio de Benaloh é o processo estatístico que permite ao eleitor verificar que seu voto foi lançado como pretendido, ou seja, que o voto inserido no sistema representa o desejo do eleitor. O processo é considerado estatístico, pois o sistema não sabe se será ou não desafiado e, portanto, um eventual erro pode não ser detectado.

Entretanto, como será apresentado na análise estatística a seguir, o número de desafios que devem ocorrer no sistema para que eventuais erros não impactem no resultado da eleição é pequeno. Essa análise é aplicada, adicionalmente, a alguns cenários reais para confirmar a utilidade do mecanismo.

Para a análise estatística, nós consideramos uma eleição simples entre dois candidatos, A e B, onde o candidato A é o vencedor. Neste cenário, existem V_T eleitores e A recebe a dos votos, com $0,5 < a \leq 1$. Consequentemente, o candidato B recebe $(1-a) \cdot V_T$ dos votos. Adicionalmente, s dos votos são desafiados pelos eleitores, sendo o desafio incondicional a escolha do eleitor (i.e., os eleitores de A e B desafiam o sistema

com a mesma probabilidade).

Para o resultado desta eleição ser alterado, é necessário que o sistema desvie votos do candidato A para o candidato B. A análise considera que o sistema é poderoso o suficiente para desviar a quantidade mínima de votos para que o resultado da eleição mude. Em outras palavras, o sistema muda votos de A para B de tal modo que, ao final do processo, B ganhe a eleição por um único voto.

Para que isto ocorra, a urna deve desviar d votos de A para B, onde:

$$d = \frac{V_T \cdot (2 \cdot a - 1) + 1}{2 \cdot a \cdot V_T}$$

Uma vez que apenas votos de A são desviados, a probabilidade l de um voto ser desviado é $l = a \cdot d$, e a probabilidade deste desvio ser detectado c é a probabilidade de um voto desviado ser desafiado, ou seja, $c = l \cdot s$.

Finalmente, como um voto desafiado é destruído e o eleitor deve realizar novamente o fluxo de votação, onde o novo voto pode ser novamente desafiado com a mesma probabilidade s, o sistema deve criar, no total, $V_C = V_T/(1-s)$ votos.

Para que um sistema desonesto tenha sucesso em sua tentativa de alterar o resultado eleitoral, nenhum dos votos desviados pode ser detectado. A probabilidade deste fato ocorrer é dada por $P = (1-c)^{V_C}$.

Com as fórmulas acima, é possível aferir o risco de um erro não ser detectado. Também é possível calcular a quantidade de desafios necessários para que uma eleição com uma determinada quantidade de eleitores e uma determinada margem de vitória apresente um risco limite de um erro não ser detectado. A seguir, são apresentados alguns casos para ilustrar os resultados obtidos.

Na Figura 2.1, o número total de eleitores V_T é fixado em 1.000.000 (um milhão) e o candidato A é o vencedor com 50,1% dos votos. Com estes números, é possível averiguar a queda do risco de não detectar um eventual erro com o aumento da quantidade de desafios. Quando a quantidade de desafios s é próxima a 1,4%, o risco de um erro não ser detectado é inferior a 10^{-6} (1 em 1 milhão).

Na Figura 2.2, o número de eleitores é novamente fixado em 1.000.000, mas, neste caso, o número de desafios é fixado em 1 a cada 1000 votos ($s = 0,1\%$). O gráfico mostra que quando o candidato A ganha com aproximadamente 51,5% dos votos, o risco de um erro não ser detectado é novamente inferior a 10^{-6}.

Por fim, na Figura 2.3, o número de eleitores é fixado em 1.000.000 e o risco de não detectar um erro na apuração é fixado em 10^{-6}. O gráfico ilustra a quantidade de desafios necessário para atingir esse risco com a variação dos votos que o candidato A recebe. É possível observar que quando o candidato A recebe 50,3% dos votos, a quantidade de desafios necessária é de 0,5%.

Para confirmar a utilidade do desafio de Benaloh em cenários reais, o mecanismo foi aplicado em três cenários no qual a diferença entre o candidato vencedor e o candidato perdedor foi pequena. Estes cenários, em ordem cronológica, são a disputa presidencial de 2º turno brasileira em 2014, a disputa presidencial norte americana no estado da Georgia

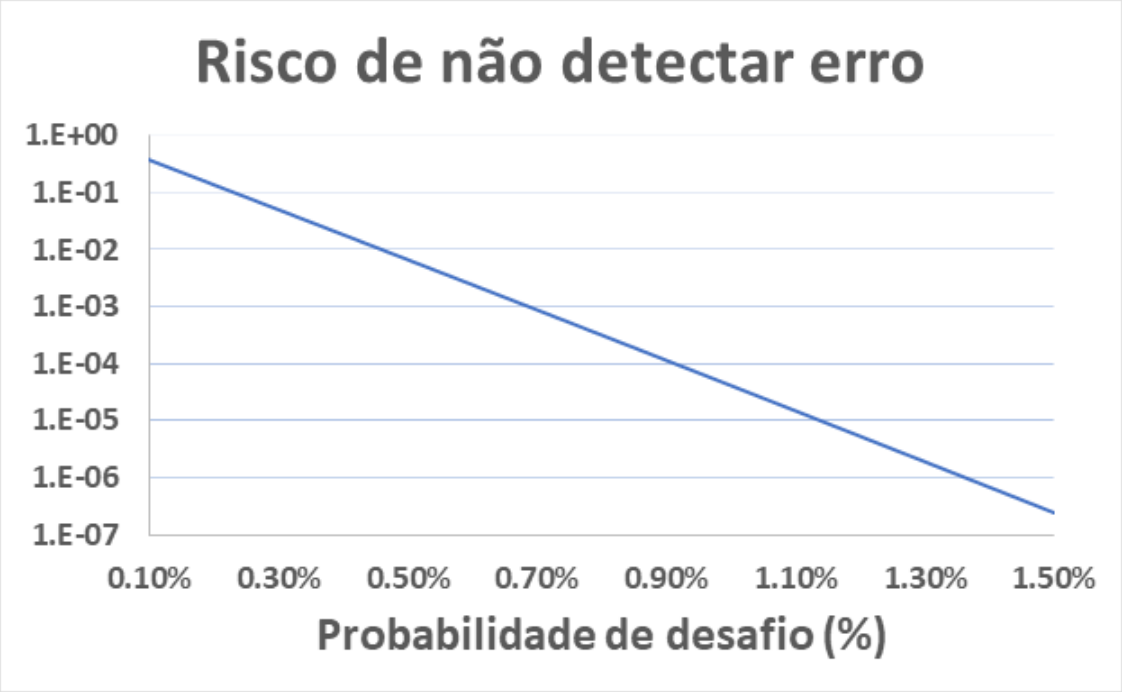


Figura 2.1: Risco de não detectar um erro no resultado eleitoral com 1.000.000 eleitores e vitória de A com 50,1%.

em 2020, e a disputa presidencial de 2º turno brasileira em 2022.

No caso das eleições brasileiras de 2014, o candidato vencedor obteve 54.501.118 votos do total de 105.542.273 votos, ou 51,64% dos votos. Para que o risco de não detectar um erro nessa eleição seja inferior a 10^{-6}, a quantidade de desafios necessária é de $s = 0,000008$, ou 1 desafio a cada 125.000 votos.

No caso das eleições americanas no estado da Georgia de 2020, o candidato vencedor obteve 2.473.633 votos do total de 4.935.487 votos, ou 50,12% dos votos. Para que o risco de não detectar um erro nessa eleição seja inferior a 10^{-6}, a quantidade de desafios necessária é de $s = 0,00235$, ou 1 desafio a cada 400 votos.

No último cenário, nas eleições brasileiras de 2022, o candidato vencedor obteve 60.345.999 votos do total de 118.552.353 votos, ou 50,90% dos votos. Para que o risco de não detectar um erro nessa eleição seja inferior a 10^{-6}, a quantidade de desafios necessária é de $s = 0,000013$, ou 1 desafio a cada 77.435 votos.

A Tabela 2.1 apresenta o resumo agregado dos resultados obtidos.

2.3.1.6. Sigilo do voto

As três propriedades fornecidas por um sistema fim-a-fim tem como objetivo garantir a **integridade** do processo de votação. Entretanto, outra propriedade de interesse em um sistema eleitoral é o **sigilo** do voto. De fato, o sigilo do voto protege o eleitor de tentativas de coerção e intimidação, garantindo sua liberdade de escolha. Infelizmente, um sistema

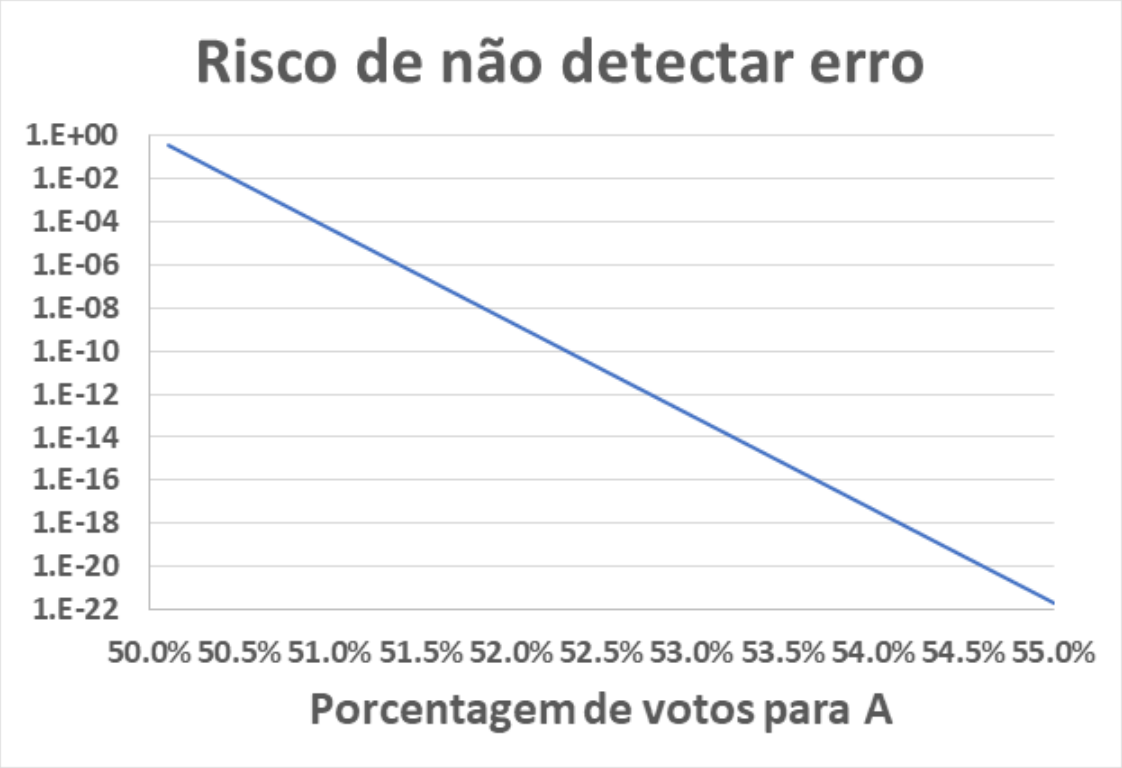


Figura 2.2: Risco de não detectar um erro no resultado eleitoral com 1.000.000 eleitores e probabilidade de desafio de 0,1%.

fim-a-fim não possuí características que melhorem a propriedade de sigilo do voto em relação a sistemas tradicionais de votação.

Em um sistema E2E, o sigilo do voto depende da confiança do eleitor na entidade ou conjunto de entidades que controlam o processo. Este fato, na pior hipótese, não é diferente do atual processo eleitoral. Quanto ao Código de Rastreio entregue a cada eleitor, este código apenas permite que o eleitor encontre seu voto **cifrado** no sistema. Logo, para que uma eventual quebra de sigilo ocorra através do Código de Rastreio, seria necessário que o esquema de cifração subjacente também seja igualmente quebrado. Além disto, um sistema fim-a-fim oferece a possibilidade de inclusão de mais entidades em alguns processos, como a guarda de chaves criptográficas. Desta maneira, é necessário o conluio de múltiplas entidades para a quebra de sigilo do voto, diluindo a confiança necessária em cada entidade individualmente.

Ademais, existe uma dificuldade inerente a proteção do sigilo do voto pelo sistema pela existência de inúmeros potenciais ataques de maneira externa e física a ele, inclusive de origem do próprio eleitor. Como exemplo de um ataque ao sigilo realizado pelo próprio eleitor, é possível citar o hábito de alguns eleitores de tirarem fotos no momento da votação, com seus votos visíveis, e a posterior postagem destas fotos em mídias sociais.

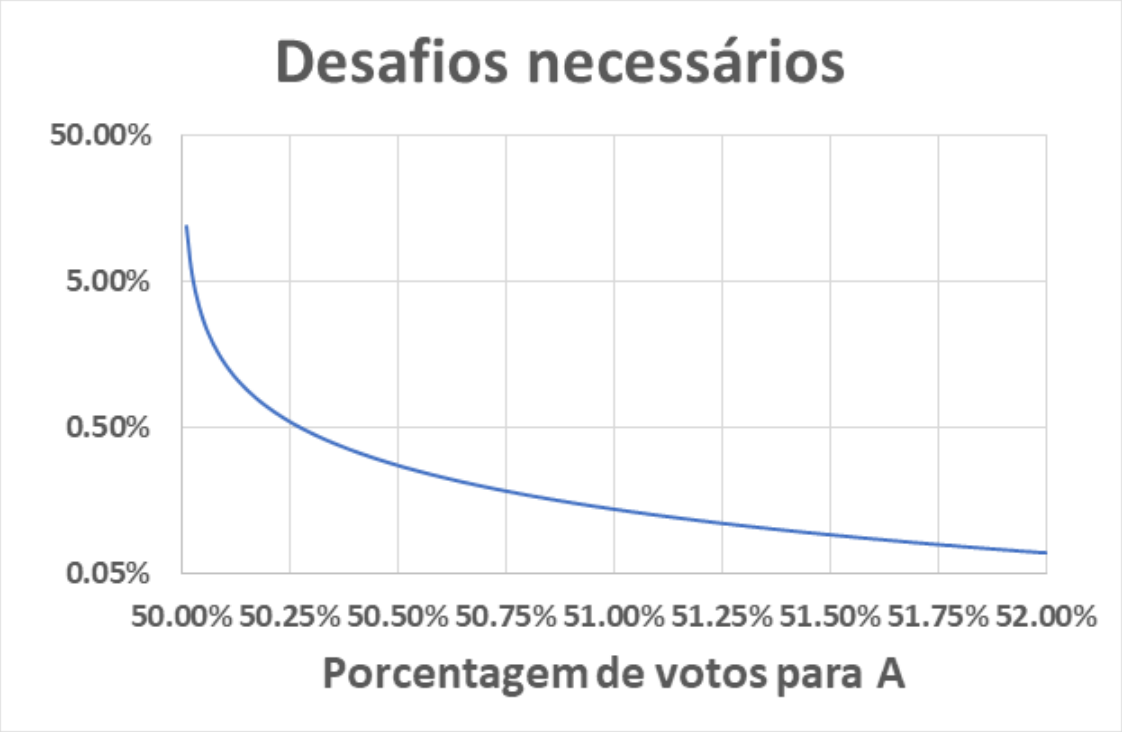


Figura 2.3: Quantidade de desafios necessária para atingir um risco de 10^{-6} com 1.000.000 de eleitores e uma porcentagem variável de votos para A.

2.3.1.7. Trilha em Papel para Auditoria Verificada pelo Eleitor - VVPAT

Um mecanismo recomendado pela literatura para ser empregado em sistemas eleitorais eletrônicos é o VVPAT, do inglês *Voter Verified Paper Audit Trail*, ou Trilha em Papel para Auditoria Verificada pelo Eleitor [Rivest 2008]. Apesar de ambos o VVPAT quanto o método fim-a-fim buscarem garantir a integridade do sistema de votação, eles atuam de maneiras diferentes, com objetivos e atores distintos.

No método VVPAT, o sistema de votação, antes de gravar o voto do eleitor, produz uma versão impressa deste voto. O eleitor, então, verifica que tanto a versão digital quanto a versão física do voto estão corretas e são iguais e, em caso positivo, permite com que o sistema grave o voto eletrônico ao mesmo tempo que deposita o voto físico (de maneira automática ou manual) em uma urna convencional. Posteriormente, o voto físico é utilizado em um processo de auditoria, chamado de Auditoria de Limitação de Risco, ou RLA (do inglês, *Risk-limiting Audit*) [Lindeman and Stark 2012].

O objetivo desta auditoria é corrigir o resultado de uma eleição caso o processo não obtenha evidências de que a apuração eletrônica esteja correta. Para isto, uma amostragem aleatória dos votos físicos é comparada com seus representantes digitais a fim de obter evidências de que uma contagem do total de votos físicos resultaria no mesmo resultado eletrônico. O "Risco Limite", presente na descrição do processo, refere-se a maior possibilidade existente de a auditoria não detectar um eventual erro no resultado da eleição, independente da origem deste erro.

Tabela 2.1: Aplicação do desafio de Benaloh em cenários reais.

Cenário	Votos para A	Total de Votos	Votos para A (%)	Desafio (Risco $< 10^{-6}$)
Eleição Presidencial Brasileira (2014)	54.501.118	105.542.273	$51,64$	$s = 0,000008$ (1/125.000 votos)
Eleição Presidencial Americana (Georgia 2020)	2.473.633	4.935.487	$50,12$	$s = 0,00235$ (1/400 votos)
Eleição Presidencial Brasileira (2022)	60.345.999	118.552.353	$50,90$	$s = 0,000013$ (1/77.435 votos)

Como esperado, quanto menor o risco que um auditor deseje obter, maior deve ser o tamanho da amostra aleatória de votos. Entretanto, a margem de vitória da eleição também influencia no tamanho da amostra, afinal, quanto menor a margem, mais evidências são necessárias, pois menor é o erro permitido. Desta maneira, é evidente que a apuração eletrônica da eleição é um processo prévio necessário ao início da auditoria.

Entretanto, como *inesperado* por grande parte da população, o tamanho da amostra aleatória é extremamente pequeno para a enorme maioria dos cenários. Mesmo eleições com margens de vitória extremamente pequenas (e.g., 1%) resultam em amostras aleatórias inferiores a 1% dos votos. Como exemplo, considerando a eleição presidencial brasileira de 2022 e utilizando a ferramenta de cálculo de número de amostras disponível em [Lindeman and Stark 2020], com risco limite de 10^{-6}, o tamanho da amostra resultante é de 1687 votos, ou aproximadamente $0,0014\%$ dos votos totais. Como curiosidade, é interessante observar que este número é bastante similar ao número de desafios necessários no processo fim-a-fim para o mesmo cenário, $0,0013\%$ dos votos totais.

Em comparação ao processo fim-a-fim, inicia-se por destacar a diferença dos atores envolvidos. No sistema E2E, o próprio eleitor é responsável por desafiar e verificar o resultado do sistema. No método VVPAT, o resultado é checado por um auditor externo, sem o envolvimento do eleitor. Desta maneira, pode-se dizer que no método VVPAT ainda é necessário, por parte do eleitor, uma confiança em algum agente, enquanto no sistema E2E o eleitor pode, em tese, executar seu próprio código para realizar a verificação do voto caso assim deseje.

Em seguida, destaca-se a diferença de objetivo entre os dois modelos. Enquanto que no VVPAT o objetivo é buscar evidências de que o resultado apurado eletronicamente está correto e corrigi-lo caso contrário, o objetivo do fim-a-fim é averiguar que o sistema está operando corretamente. Embora, em tese, os dois objetivos levem ao mesmo resultado, os modelos são submetidos a diferentes tipos de ataque. Por exemplo, não existe um mecanismo inerente ao VVPAT que impeça a substituição de um conjunto de votos eletrônicos e seus respectivos votos físicos por um conjunto falso. Já no sistema fim-a-fim, como o eleitor recebe um Código de Rastreio assinado digitalmente, ele é capaz de provar

que um voto legítimo foi trocado.

Observa-se também a facilidade de compreensão, por parte do público, entre os dois modelos. O VVPAT disponibiliza o voto às claras do eleitor para checagem, enquanto que o E2E mostra um resultado de um hash de um voto cifrado. Para um eleitor, a verificação do voto às claras é extremamente mais simples do que o cálculo matématico e garantias existentes da comparação de resultados de hashes criptográficos.

Por fim, é importante observar a questão de guarda e logística do componente impresso nos dois casos. No mecanismo VVPAT, a autoridade eleitoral é responsável por realizar a coleta, transporte e guarda dos votos físicos para posterior auditoria. Enquanto isso, no fim-a-fim, cada eleitor é responsável por seu próprio Código de Rastreio. Em votações nas quais o território e o número de eleitores é elevado, o custo inerente a logística do VVPAT pode tornar-se igualmente alto. Já o processo E2E não acarreta em custos adicionais de logística.

Como os mecanismos de VVPAT e fim-a-fim atendem a diferentes objetivos, com pontos positivos e negativos complementares, além de não serem conflitantes em termos operacionais, a implementação concomitante de ambos os modelos é possível. Ademais, caso não existam impedimentos legais ou financeiros para isto, a implementação em conjunto dos dois mecanismos é recomendada.

2.3.2. Trabalhos Relacionados de Sistemas E2E

Nesta Seção, alguns dos principais sistemas de votação fim-a-fim existentes na literatura são brevemente apresentados. Os sistemas Helios [Adida 2008] e ElectionGuard [Benaloh and Naehrig 2022] são intencionalmente omitidos, pois eles serão descritos em seções específicas.

O artigo **Prêt à Voter** [Chaum et al. 2005] descreve um esquema de votação fim-a-fim com a utilização de cédulas de votação em papel que são escaneadas. Neste esquema, cada cédula de votação é constituída de duas metades, a esquerda e a direita, e é criada individualmente. Na metade esquerda, é apresentado uma tabela com a lista de candidatos embaralhada. O embaralhamento é feito de maneira específica para cada uma das cédulas. Na metade direita, existe uma tabela em branco, alinhada com a tabela da metade esquerda, juntamente com uma linha extra, que possui um número "cebola". Este número será posteriormente utilizado para desembaralhar a cédula. Um exemplo de cédula é mostrado na Figura 2.4.

Para votar, um eleitor localiza seu candidato na lista da metade esquerda e marca o campo na metade direita (e.g., com um X). Em seguida, o eleitor separa as duas metades da cédula. A metade direita é escaneada e publicada para que o votante também possa comparar com seu recibo. O conteúdo da cédula e seu valor, o número da coluna marcada e o número cebola são, então, salvos.

Após todos os votos serem registrados, o sistema opera de maneira similar a uma rede de misturadores. Cada nó misturador, estabelecidos previamente e em uma determinada sequência, recebe o valor registrado. O nó decifra o número cebola, obtendo dois valores. O primeiro é utilizado na marcação de voto, tirando uma camada de embaralhamento. O segundo é anexado a nova marcação e enviado conjuntamente para o próximo

Basquete	
Futebol	
Natação	
Vôlei	

(a) Lista de candidatos ordenada.

Natação	X
Vôlei	
Basquete	
Futebol	
	XGJlZ2lu

(b) Exemplo de uma cédula de votação com candidatos embaralhados e voto para "Natação".

Figura 2.4: Visão do eleitor do esquema Prêt à Voter. A lista de candidatos ordenada (a) é pública e cada cédula recebe um embaralhamento diferente desta lista, com um número cebola apropriado (b).

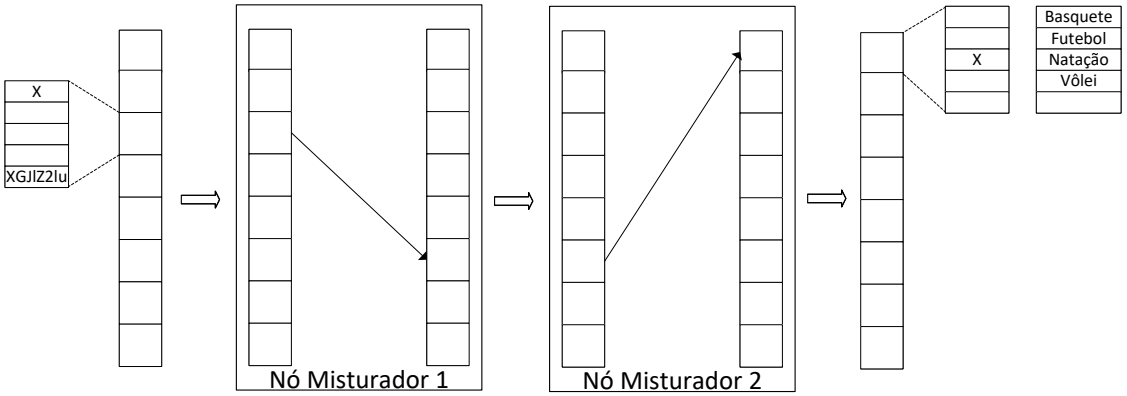


Figura 2.5: Operação simplificada do sistema Prêt à Voter. Após a coleta das cédulas de voto dos eleitores, estas são processados por uma sequência de nós misturadores, que desembaralham e alteram a ordem do conjunto de cédulas, até a produção do voto final. No exemplo, a cédula utilizada para votar em "Natação" da Figura 2.4 é desembaralhada e misturada, até a saída final, onde a marcação corresponde a lista ordenada pública de candidatos.

nó, como o novo número cebola. Adicionalmente, antes do envio do voto para o próximo nó, os votos dos eleitores são misturados, a fim de remover a associação entre a ordem de entrada e a ordem de saída do misturador. Este processo é repetido por todos os misturadores, até que a saída final do sistema corresponde a marcação do voto em uma lista ordenada dos candidatos, comum a todos os votos. Com isso, a apuração final da eleição pode ser feita.

Como todos os votos na entrada do primeiro misturador são públicos e existem processos de auditoria para garantir o correto desembaralhamento e mistura dos votos, com o comprovante de voto, o eleitor é capaz de certificar-se que seu voto foi utilizado para o cálculo do resultado final da eleição. Um resumo da operação do esquema Prêt à Voter é apresentado na Figura 2.5.

O sistema **RIES** [Hubbers et al. 2005] é um sistema de votação híbrido, que opera através da Internet ou por carta, que utiliza um mecanismo de hash associado com um de cifração. Um voto é o resultado de um hash aplicado a cifração de um identificador de

candidato.

Inicialmente, a autoridade eleitoral cria uma chave de cifração única para cada eleitor. Em seguida, esta autoridade, utilizando a chave criada para cada eleitor e para cada candidato, cifra o identificador do candidato e computa o hash deste resultado, associando o valor final com o candidato. Todos estes resultados, que constituem todos os votos possíveis, são tornados públicos. Por fim, a entidade eleitoral envia a chave de cifração para cada eleitor.

Para votar, o eleitor escolhe seu candidato e apenas cifra o identificador associado, enviando o dado cifrado a autoridade eleitoral. A autoridade eleitoral, por sua vez, publica todos os dados cifrados que recebeu e o hash destes dados. Utilizando a lista de todos os votos possíveis, publicada anteriormente, é possível associar o valor do hash calculado com o valor associado a um candidato, desta maneira contabilizando um voto para este candidato.

Como o eleitor guarda o resultado cifrado que enviou a autoridade eleitoral, ele é capaz de conferir que seu voto consta na lista final de apuração e foi utilizado. Além disto, como todos os dados cifrados são feitos públicos e qualquer pessoa é capaz de calcular o hash destes dados, a verificação do resultado da eleição pode ser feita por todos, inclusive não eleitores.

O esquema **Scantegrity II** [Chaum et al. 2008] é um sistema fim-a-fim baseado em cédulas de votação em papel com códigos de confirmação e uso de escâneres. As cédulas de votação são compostas por três partes, sendo elas o corpo principal, o voucher inferior esquerdo e o voucher inferior direito. No corpo principal, localizam-se todas as escolhas possíveis de candidato ao lado de círculos, que são preenchidos para indicar a escolha do eleitor. Estes círculos são criados com uma tinta especial e, quando marcados utilizando uma caneta específica, revelam por alguns minutos um pequeno código antes de se tornarem completamente preenchidos. O voucher inferior esquerdo contém um campo que também é criado utilizando uma tinta especial, diferente da primeira, e um espaço para anotações. O campo contém um número serial da cédula específica e o espaço para anotações pode ser utilizado pelo eleitor para copiar o código de confirmação obtido ao realizar a escolha de candidato no corpo principal. Por fim, o voucher inferior direito é composto por um campo, criado com a mesma tinta especial do voucher esquero, que contém um outro número serial associado a cédula, diferente do anterior. Um exemplo de cédula marcada do Scantegrity é apresentado na Figura 2.6.

Para votar, um eleitor recebe uma cédula do mesário e dirige-se a cabine de votação. Na cabine, o eleitor utiliza uma caneta reagente a tinta especial dos círculos do corpo principal para realizar sua escolha. Ao retornar da cabine, a cédula é inserida em um escâner, que verifica se ela foi corretamente preenchida (i.e., apenas uma escolha foi realizada). Em caso negativo, a cédula é descartada e o eleitor deve realizar a votação novamente. No descarte, é escolhido arbitrariamente o voucher esquerdo ou direito, que é retido pelo mesário. Em caso positivo, o voto é salvo, o corpo principal é depositado em uma urna e os vouchers inferiores são fornecidos ao eleitor. Em seguida, o eleitor apresenta ambos os vouchers ao mesário, que utilizando uma caneta que reage a tinta específica dos vouchers, revela ambos os números seriais associados a cédula. Por fim, o eleitor com posse de ambos os vouchers com seus números seriais revelados e sua marca-

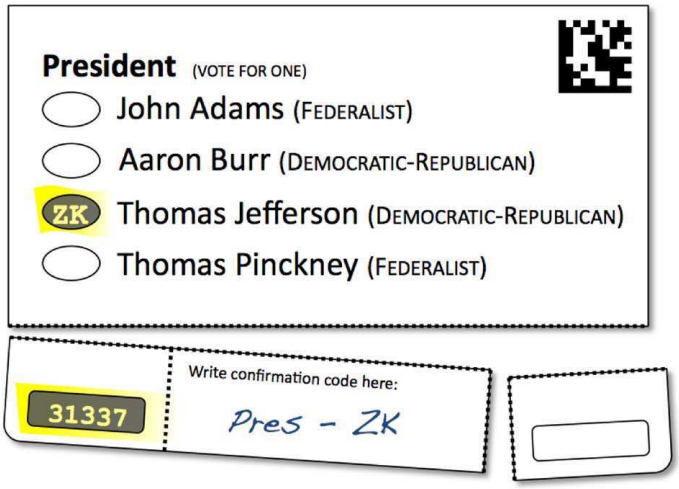


Figura 2.6: Exemplo de cédula do esquema Scantegrity II, com corpo principal e vouchers inferiores esquerdo e direito. O voto contido na cédula corresponde ao candidato Thomas Jefferson. Fonte: [Chaum et al. 2008]

ção de código de confirmação, conclui o processo de votação.

Após o encerramento da eleição, o eleitor é capaz de acessar um site específico que, ao ser informado de um dos números seriais associados a cédula, retorna o código de confirmação da escolha do eleitor. Como os códigos de confirmação são gerados de maneira criptográfica e associados previamente a cada cédula e a cada candidato, o eleitor pode confirmar que sua escolha foi corretamente capturada e seu voto utilizado para a apuração. Caso o eleitor obtenha um código de confirmação diferente do anotado, um processo de auditoria é iniciado para detectar a causa do problema. Observa-se que o caso no qual o eleitor cometa um erro, intencional ou não, é facilmente detectado. Isto é possível pois o sistema é capaz de produzir todos os códigos de confirmação de uma cédula e uma auditoria é capaz de verificar se o eleitor produziu um código válido ou não para a cédula específica, sendo a chance de acertar ao acaso um código válido parametrizável.

O sistema fim-a-fim **PunchScan** [Popoveniuc and Hosp 2010] é um esquema baseado em cédulas de papel. Neste esquema, a cédula de votação é composta por duas páginas, que são sobrepostas. Na página superior, são apresentados ao eleitor os candidatos e seus códigos de identificação, que são embaralhados para cada cédula, sendo a associação conhecida apenas pela autoridade eleitoral. Na página inferior, são apresentados apenas os códigos de identificação, em ordem aleatória, sendo esta ordem também apenas conhecida pela autoridade eleitoral. Adicionalmente, a página superior apresenta orifícios, que são alinhados com os códigos de identificação da página inferior. Desta maneira, quando o eleitor recebe a cédula de votação, ele é capaz de ver os candidatos e seus códigos na página superior e os códigos na página inferior, através dos orifícios. Um exemplo de cédula de votação é ilustrado na Figura 2.7.

Para votar, o eleitor utiliza um marcador cuja dimensão é superior ao tamanho do orifício da página superior. Consequentemente, ao marcar uma das opções da página inferior, a borda do orifício da página superior também é marcada. Após realizar a marcação, o eleitor separa as duas páginas da cédula, escaneando uma delas e destruindo a

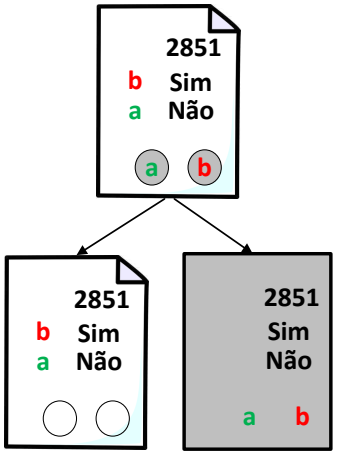


Figura 2.7: Cédula de votação do sistema Punchscan. Em cima, a visão do eleitor, com as duas páginas sobrepostas. Embaixo, à esquerda a página superior, e à direita a página inferior.

outra. A decisão de qual página é depositada é tomada antes do eleitor receber a cédula. Adicionalmente, o eleitor retém a página escaneada.

Para realizar a apuração, a autoridade eleitoral recebe as diversas páginas escaneadas e realiza a associação entre a marcação em cada página e um candidato. Isto é possível pois, como mencionado anteriormente, a autoridade sabe a associação entre candidato e identificação para cada cédula e a ordem que estas identificações são apresentadas na cédula. Por sua vez, o eleitor, com a página retida, é capaz de verificar, uma vez que a autoridade eleitoral publique as páginas recebidas, que a marcação em sua página corresponde a marcação da página recebida pela autoridade.

Para concluir esta Seção, observa-se que o artigo [Kelsey et al. 2010] cita ataques a alguns dos esquemas previamente apresentados. Por questões de espaço e por não ser o enfoque deste minicurso, estes ataques não serão comentados, deixando a análise a cargo do leitor.

2.4. Sistema E2E baseado em Redes de Misturadores

Esta Seção apresenta um sistema fim-a-fim baseado em redes de misturadores. São discutidos as técnicas utilizadas no esquema para atingir os objetivos de integridade e verificabilidade, baseando-se nos blocos criptográficos apresentados na Seção 2.2. Por fim, será apresentada a instalação da biblioteca Verificatum para a operação de uma rede de misturadores.

2.4.1. Redes de Misturadores

Redes de misturadores formam um conjunto de protocolos que garantem a comunicação segura entre duas partes [Chaum 1981, Sampigethaya and Poovendran 2006]. Originalmente proposta para uso com protocolos de roteamento, esta tecnologia permite que mensagens trafegadas entre um remetente e um destinatário não possam ser rastreadas por

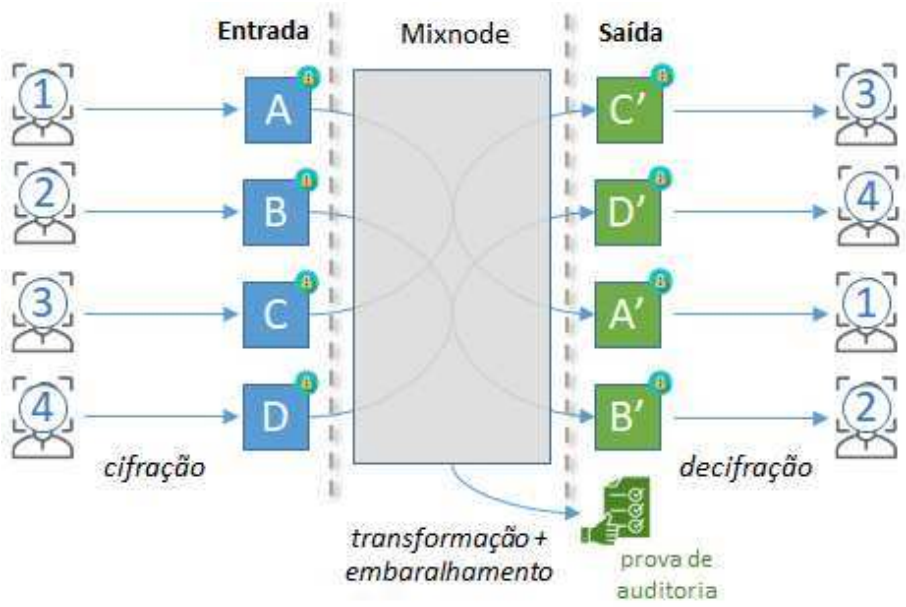


Figura 2.8: Operação de um “mixnode” ou nó misturador.

observadores na rede. A ideia consiste essencialmente no seguinte. Primeiramente, todas as mensagens dos usuários devem ser enviadas cifradas pela rede. Em alguns pontos no trajeto dessas mensagens, são incluídos então servidores proxies adicionais, chamados de “mixnodes” ou “mixservers” em inglês, ou de nós misturadores em português. Esses servidores aguardam o recebimento de um determinado volume de mensagem de diferentes usuários, e realizam algumas operações sobre essas mensagens para garantir privacidade e integridade dos dados trafegados. O resultado é uma espécie de “embaralhamento” das mensagens enviadas, não sendo possível a um observador externo ligar uma mensagem saindo do nó misturador à mensagem de origem correspondente: apenas o nó misturador conhece essa relação.

De forma geral, cada nó misturador precisa realizar três operações sobre os textos cifrados para garantir essas propriedades de segurança (vide Figura 2.8): transformação, embaralhamento e prova de auditoria. A transformação é feita para que os textos cifrados na entrada de cada nó misturador sejam modificados, mas ainda de forma que sua decifração na saída da rede de misturadores leve ao texto às claras original; o objetivo é impedir que um observador reconheça que uma mensagem de saída se relacione a uma mensagem de entrada simplesmente verificando a igualdade entre os textos cifrados em questão. Seguindo o mesmo princípio, o embaralhamento garante que também não haja relação de ordem entre as mensagens transformadas, ou seja, esconde-se a ordem entre entrada e saída. Já o mecanismo de prova de auditoria é utilizado para detectar casos em que um nó misturador malicioso tenta modificar as mensagens trafegadas (e.g., substituindo os textos cifrados correspondentes, em vez de realizar a operação de transformação que preserva o texto às claras original); em outras palavras, as provas de auditoria geradas devem ser válidas apenas para nós misturadores honestos, que não tenham tentado fazer esse tipo de modificação.

Assim, a segurança das redes de misturadores é projetada de tal forma que, mesmo

que o atacante entre em conluio com a maioria dos nós misturadores, a privacidade das mensagens é mantida se um número pequeno de nós misturadores se mantiver honesto (em alguns casos, apenas um mixnode honesto é o suficiente). Além disso, a integridade das mensagens pode facilmente ser confirmada, pois a modificação de qualquer mensagem pode ser detectada por qualquer entidade que deseje auditar o sistema.

2.4.1.1. Privacidade

Para garantir a privacidade das mensagens, os nós misturadores são capazes de transformar as mensagens cifradas de diversos usuários sem precisar revelar essas mensagens, e a ligação entre mensagens é perdida realizando um embaralhamento. A transformação da mensagem cifrada é usualmente feita por decifrações sucessivas ou por recifração dos textos cifrados.

Nas redes de misturadores de decifração [Chaum 1981], os usuários devem selecionar um conjunto de nós misturadores e, então, realizar a cifração sucessiva da mensagem utilizando as chaves de cada um deles. Quando um nó misturador receber a mensagem, ele deve decifrá-la utilizando a sua chave, embaralhar (permutar) as diferentes mensagens entre si, e só então enviar o novo conjunto de mensagens para o próximo mixnode. O último nó misturador envia a mensagem para o destinatário correto, sendo este o único capaz de decifrar a última camada da mensagem.

Formalmente, para cifrar uma mensagem m com a aplicação da cifração $E_{sk}(\cdot)$ utilizando chave de cifração pk, realiza-se a composição de cifrações:

$$C_n = E_{pk_n} \circ E_{pk_{(n-1)}} \circ \cdots \circ E_{pk_1}(m)$$

Para decifrar, cada nó misturador aplica a decifração $D_{sk}(\cdot)$ no texto cifrado C de forma iterativa, obtendo textos cifrados parciais $C_{i-1} = D_{sk_i}(C_i)$. Desta forma, a mensagem original é obtida após a composição das diversas decifrações:

$$m = C_0 = D_{sk_1} \circ D_{sk_2} \circ \cdots \circ D_{sk_n}(C_n)$$

Já redes de misturadores de recifração [Park et al. 1994] são realizadas utilizando apenas a chave pública do destinatário com uma cifra que permite recifração (e.g., a cifra ElGamal, apresentada na Seção 2.2). Quando um nó misturador recebe mensagens cifradas, ele utiliza a chave pública de cifração do destinatário para realeatorizar os textos cifrados, o que resulta em um outro texto cifrado que, ao ser decifrado, resultará na mensagem original. Em seguida, os textos recifrados são embaralhados e enviados para o próximo nó misturador. Como apenas a chave pública do destinatário é utilizada, nenhum nó misturador é capaz de decifrar as mensagens trafegadas: apenas o destinatário será capaz de fazê-lo.

Formalmente, a mensagem m é cifrada por meio da função de cifração $E_{pk}(\cdot)$ com a chave pública pk, utilizando o nonce r, apenas uma vez pelo usuário. O texto cifrado resultante, $C = E_{pk}(m;r)$, é enviado para os nós misturadores que aplicam a recifração $R_{pk}(\cdot)$, com nonce r', gerando o novo texto cifrado $C' = R_{pk}(C;r')$. Ao final da transmissão, após receber o texto cifrado $C^{(n)}$, o destinatário decifra a mensagem aplicando

a decifração $D_{sk}(\cdot)$, resultando em $m = D_{sk}(C^{(n)})$. Este processo foi detalhado na Seção 2.2.

Como mencionado anteriormente, independentemente do método de transformação que a rede de misturadores utiliza, ela precisa ser acoplada a um mecanismo de embaralhamento, misturando a ordem das diferentes mensagens para garantir privacidade. Mais precisamente, a transformação sobre os textos cifrados faz com que seja computacionalmente inviável diferenciar se a saída do nó misturador reflete a mesma mensagem recebida na entrada. Obviamente, isso assume que o nó misturador receba em sua entrada mais do que um elemento, pois caso contrário seria imediato identificar que a única mensagem de entrada e de saída pertencem ao mesmo usuário. Com diversas mensagens transformadas por iteração, o observador só conseguiria identificar quais mensagens pertencem ao mesmo usuário pela sua ordem na entrada e na saída. Portanto, ao embaralhar as mensagens antes de criar a lista de saída, esta relação deixa de existir. Logo, apenas o próprio nó misturador que conhece o embaralhamento usado seria capaz de fazer esta identificação.

2.4.1.2. Integridade

Para garantir integridade, as redes de misturadores podem utilizar mecanismos de desafio ou de provas de conhecimento. Redes de misturadores que utilizam *desafios*, por exemplo, permitem que usuários mandem mensagens marcadas de tal forma que, após serem reveladas, exigem que os nós misturadores revelem o trajeto que sua mensagem fez no sistema. Quanto mais desafios forem respondidos pela rede de misturadores, maior o grau de confiança que o processamento das mensagens não-reveladas também tenha sido feita corretamente. Já redes de misturadores que utilizam *provas de conhecimento* permitem a criação de um artefato criptográfico que pode ser verificado para garantir a corretude dos embaralhamentos, decifrações e recifrações. É importante notar que um nó misturador desonesto não seria capaz de criar um artefato caso ele tenha adulterado as mensagens, já que a verificação dessas provas resultaria em falha.

2.4.1.3. Comparação entre os tipos de redes de misturadores

A principal desvantagem de redes de misturadores baseadas em decifração é o fato de que deve haver gerenciamento de chaves de todos os nós misturadores da rede. Não somente a chave de todos os nós misturadores deve ser sincronizada no início do envio da mensagem, como o processo deve ser reiniciado caso algum nó misturador seja desconectado em operação. Já uma rede de misturadores de recifração depende apenas de uma chave pública global do sistema, e nós misturadores intermediários que falharem podem ser removidos facilmente das operações.

Com relação a integridade, a maior desvantagem do uso de desafios provém do fato de que desafios são mensagens adicionais que precisam ser enviadas em tempo real. Isso resulta no aumento da banda usada pelo sistema, e o número de desafios pode variar a cada iteração do uso. Além disso, como os desafios são realizados por usuários do sistema, o aumento de confiança só se dá por desafios confiáveis, isto é, desafios enviados

pelo próprio usuário, ou por outro usuário da sua rede de confiança. Usar provas de conhecimento, por outro lado, permite a verificação matemática de que a execução completa daquela iteração foi correta. Além disso, caso sejam utilizadas provas não-interativas, essa verificação não precisa ser feita em tempo real, mas é realizada por qualquer entidade, participante ou não do sistema.

Dadas as vantagens e desvantagens dos esquemas de redes de misturadores existentes, optou-se neste minicurso pelo uso de uma rede de misturadores de recifração (especificamente, baseada no criptosistema ElGamal), com o uso de provas de conhecimento não-interativas. As provas de conhecimento não interativas utilizadas são as provas de Wikström-Terelius, apresentadas na Seção 2.2 Nota-se que esta escolha foi voltada à eficiência, mas que o esquema sugerido pode ser substituído por outros.

2.4.2. Redes de misturadores em sistemas de votação

As propriedades de segurança voltadas à privacidade e integridade dos votos permitiu que surgissem propostas de sistemas de votação utilizando estas ferramentas [Jakobsson et al. 2002]. Isso pode ser observado em esquemas de votação online (e.g. Helios [Adida 2008]) ou esquemas presenciais (e.g., Prêt-à-Voter [Chaum et al. 2005]), inclusive em votações públicas na Noruega e Estônia, demonstrando a viabilidade deste tipo de construção.

Propõe-se aqui a aplicação de redes de misturadores em dois cenários distintos. O primeiro, chamado de **tradicional**, é semelhante a outras soluções de votação online como na primeira versão do sistema Helios. Ele refere-se ao uso de redes de misturadores de forma distribuída para garantir sigilo, inclusive em relação à autoridade eleitoral. O segundo, chamado de **modelo de auditoria**, é proposto com o uso de redes de misturadores para garantir a integridade dos votos dos eleitores de maneira mais centralizada e como ferramenta de auditoria da totalização dos votos.

No que se segue, o sistema de totalização é tratado como uma única entidade da autoridade eleitoral, sendo responsável pela decifração dos votos dos eleitores e sua conseguinte totalização. No entanto, é importante notar que este sistema pode não ser unitário, mas pode ser composto por diversos guardiões. Neste caso, cada guardião possui uma parte da chave privada de decifração relacionada a uma única chave pública (e.g., utilizando a versão limiar do criptosistema ElGamal descrito na Seção 2.2.3.4), e seria necessária a colaboração de um subconjunto destes guardiões para realizar a decifração total dos votos. Esta propriedade aumenta a confiança pois uma única autoridade eleitoral não seria capaz de quebrar o sigilo dos votos. Para isso, ela precisaria entrar em conluio com os outros guardiões do sistema.

2.4.2.1. Cenário tradicional

A arquitetura deste cenário é bastante semelhante ao cenário do sistema Helios, porém adaptado ao voto presencial. Para isso, tornam-se necessários os seguintes componentes: uma urna, como plataforma de votação; um conjunto de nós misturadores, onde pelo menos um não pertença à autoridade eleitoral; e o sistema de totalização dos votos.

Como preparação para a eleição, o sistema de totalização é responsável por criar

um par de chaves pública-privada para ser utilizada pela urna e pelos nós misturadores para (re-)cifração dos votos. A chave pública de cifração deve ser carregada previamente em todas as urnas e enviada para todos os nós misturadores, e apenas o sistema de totalização deve conhecer a chave privada de decifração.

No processo de votação, a urna funciona como plataforma de votação e é responsável por realizar corretamente a cifração dos votos dos eleitores. Quando o eleitor confirmar suas escolhas, a urna deve emitir o Código de Rastreio, como apresentado na Seção 2.3.1.4. Caso o eleitor deseje depositar este voto, o Código de Rastreio deve ser assinado e poderá ser utilizado posteriormente como material de auditoria da integridade dos votos contabilizados.

Neste momento, como o voto não é criado por dispositivo do próprio eleitor, este pode não ter a confiança de que a urna está realizando a operação de forma correta, ou seja, ele pode desconfiar que o comprovante de votação está apresentando um valor diferente do inserido na urna. Por este motivo, o eleitor pode desafiar a urna por meio do desafio de Benaloh como descrito na Seção 2.3.1.5.

Ao final do processo de votação, inicia-se o procedimento das redes de misturadores. Os votos cifrados depositados na urna são então enviados para o primeiro nó misturador da rede. Este nó pode utilizar a carga de votos de uma urna, ou aguardar o recebimento da carga de diversas urnas para realizar as operações em conjunto. Após o recebimento, a rede de misturadores faz sua operação comum: transforma e embaralha os textos cifrados, cria provas de que as operações foram feitas corretamente, e envia para o próximo nó da rede, utilizando as provas de conhecimento descritas na Seção 2.2. O último nó, por fim, envia os votos (re-)cifrados para o sistema de totalização, que deve ser capaz de decifrar os votos de maneira verificável. Os votos decifrados, agora sem qualquer vínculo com os respectivos eleitores, podem então ser totalizados.

Para a auditoria, todos os nós misturadores devem deixar públicas as tabelas de entrada e saída de votos cifrados, e suas correspondentes provas de corretude. O sistema de totalização também deve deixar públicos os votos cifrados de entrada e os decifrados de saída, juntamente com provas de correta decifração. Eleitores que desejem auditar os seus votos podem, neste momento, verificar se o voto cifrado do seu Código de Rastreio está presente nas tabelas de entradas do primeiro nó misturador, o que garante que ele trafegou por todo o sistema e entrou na totalização.

2.4.2.2. Cenário de auditoria

Ao contrário do cenário tradicional, a arquitetura deste sistema é voltada para dar poder de auditoria aos votos em uma urna eletrônica. Considera-se que esta é suficiente para dar privacidade aos votos dos eleitores contra atacantes externos, enquanto o uso de guardiões mantém a privacidade em relação à autoridade eleitoral (por prevenir que esta última decifre os comprovantes dos eleitores). Desta forma, o sistema, agora mais simples, conta com apenas dois componentes: a urna eletrônica, que faz o papel da plataforma de votação e também do único nó misturador no sistema; e a entidade de totalização.

O processo de preparação da eleição e o processo de votação ocorrem como no

cenário tradicional. O sistema de totalização gera o par de chaves pública/privada, e então carrega a chave pública de cifração na urna; já a chave privada de decifração é mantida em segredo. Os eleitores têm seus votos cifrados pela urna e seus Códigos de Rastreio assinados, podendo realizar o desafio nas urnas como no Cenário Tradicional.

A diferença ocorre ao final do processo de votação. Nesta etapa, apenas a urna realiza a função da rede de misturadores, transformando e embaralhando os votos, e criando as provas de corretude desta operação. Os votos cifrados originais e os recifrados são então enviados juntamente com as provas para o sistema de totalização. Os votos passam pelo processo de decifração verificável como no cenário tradicional, e as tabelas de votos cifrados e decifrados, bem como as provas de auditoria, são publicadas para verificação por potenciais interessados.

Cabe notar que, nesse cenário em que a urna é o único nó misturador no sistema, preserva-se essencialmente a mesma confidencialidade dos votos obtida com uma urna eletrônica tradicional brasileira. Para entender melhor essa observação, considere um cenário em que a urna eletrônica atual seja subvertida, passando então a associar os eleitores a seus respectivos votos às claras; isso poderia ser feito, por exemplo, por meio da criação de uma tabela associando o título do eleitor a seu voto, ou simplesmente guardando a ordem dos votos lançados (deixando para o atacante a tarefa de identificar a ordem dos eleitores na fila). Se isso for feito, a adição de nós misturadores em pontos seguintes do sistema, como previsto no cenário tradicional, não seria capaz de garantir o sigilo dos votos. Logo, a urna eletrônica já é um ponto crítico para garantir o sigilo das escolhas do eleitor, devendo ser protegida contra tentativas de subversão. Assumindo que essa proteção seja efetiva, a atuação da urna como o primeiro (e único) nó misturador já garante o sigilo dos votos, mesmo que não haja nós misturadores adicionais posteriormente.

Isso posto, esta abordagem parece ser uma escolha natural para o contexto de votação brasileiro, em que o dispositivo de entrada não é controlado pelo usuário (contrariamente ao cenário tradicional, em que se assume que as mensagens de entrada da rede de misturadores é gerada por um dispositivo de confiança do usuário). Como benefício adicional, pode-se considerar a possibilidade de combinar os resultados de mais de uma urna com um nó misturador adicional, com o objetivo de reduzir a possibilidade de inferir votos a partir dos resultados parciais individuais de uma única urna. Por exemplo, quando o resultado parcial de uma urna indica que todos os votos foram lançados para um mesmo candidato X, o sigilo de cada voto individual é naturalmente perdido: afinal, tem-se a certeza que cada eleitor daquela urna votou em X. Já se o resultado parcial individual dessa urna for omitido, mas apresentado apenas após sua combinação com o resultado de uma outra urna com os mesmos cargos, reduz-se a probabilidade de aparecimento de parciais nos quais um candidato é (quase) unanimidade. Entretanto, como a apresentação de resultados parciais individuais de cada urna é uma tradição do sistema eleitoral brasileiro, e considerado por muitos um mecanismo de auditoria relevante, esta possibilidade de omitir alguns resultados em favor de se apresentar apenas a combinação dessas parciais por meio de nós misturadores adicionais não é aqui explorada mais a fundo.

2.4.3. Verificatum

O Verificatum é uma implementação open-source de redes de misturadores de recifração usando cifra ElGamal. Esta biblioteca já foi utilizada em sistemas eleitorais online em Israel (Wombat), Espanha (Agora Voting), Noruega (Scytl), e Estônia. Ele foi escrito na linguagem de programação Java, com trechos de código em C por questões de eficiência. Além da funcionalidade básica de embaralhamento com recifração de textos cifrados, ele também conta com provas de conhecimento-zero interativas e não-interativas, decifração parcial verificável de textos cifrados, e capacidade de pré-processamento para aceleração de operações em tempo real.

As primitivas criptográficas de chave pública utilizadas pelo Verificatum partem do uso da biblioteca GMP (Gnu Multiple Precision), de onde partem implementações eficientes de aritmética de corpos finitos e curvas elípticas para criptografia. Essas operações foram expandidas para que operações vetoriais pudessem ser executadas simultaneamente, e utilizam a notação multiplicativa (comum para corpos finitos) pela transparência das primitivas utilizadas.

A verificabilidade das operações provém do uso de provas de conhecimento-zero, que podem ser implementadas de forma interativa ou então de forma não-interativa, utilizando a heurística de Fiat-Shamir. Essas provas de conhecimento são utilizadas para provar três argumentos: (1) que os textos-cifrados foram embaralhados e recifrados, (2) que as mensagens foram decifradas corretamente, e (3) que o pré-processamento foi executado corretamente. Assim, tais provas são as garantias matemáticas de que a operação da rede de misturadores foi feita de maneira honesta, garantindo a integridade das mensagens. Apesar do Verificatum implementar os próprios verificadores, também existem guias para implementação de verificadores por terceiros que atendam aos requisitos necessários para validar as provas criadas.

2.4.3.1. Instalação

O projeto Verificatum é composto por 6 bibliotecas:

- GMPMEE: extensão da biblioteca GMP para operações simultâneas e de base fixa mais velozes em corpos finitos.
- VEC: biblioteca de curvas elípticas com operações simultâneas e base fixa mais velozes. Possui trechos de código do OpenSSL para otimização de operações em algumas curvas padronizadas.
- VMGJ: interface das bibliotecas de baixo nível GMP e GMPMEE para aplicações em Java.
- VECJ: interface da biblioteca de baixo nível VEC para aplicações em Java.
- VCR: biblioteca contendo as principais rotinas modularizadas necessárias para a execução de uma rede de misturadores baseada na cifra ElGamal.
- VMN: implementação dos protocolos executados na rede de misturadores.

Estas bibliotecas podem ser encontradas no repositório do github do projeto (`https://github.com/verificatum`), e as versões utilizadas no teste foram clonadas das mais recentes no dia 1 de agosto de 2022. Eles foram realizados na versão 22.10 do Ubuntu e os seguintes pacotes foram instalados do repositório padrão como requisitos do projeto: libgmp3-dev, automake, libtool, e default-jdk.

Para a instalação de cada uma das bibliotecas, os seguintes comandos devem ser usados:

```
# autoreconf -vfi
# make -f Makefile.build
# ./configure
# make
# sudo make install
```

Algumas bibliotecas também exigem comandos adicionais após sua instalação:

- VMGJ: acrescentar a biblioteca no classpath do Java:

```
# export CLASSPATH=/usr/local/share/java/verificatum-vmgj-1.2.2.
    jar:${CLASSPATH}
# export LD_LIBRARY_PATH=/usr/local/lib:${LD_LIBRARY_PATH}
```

- VECJ: acrescentar a biblioteca no classpath do Java:

```
# export CLASSPATH=/usr/local/share/java/verificatum-vecj-2.1.5.
    jar:${CLASSPATH}
```

- VCR: ativar o uso das bibliotecas VMGJ e VECJ, e inicialização da fonte de aleatoriedade:

```
# ./configure --enable-vmgj --enable-vecj
# vog -rndinit RandomDevice /dev/urandom
```

Adicionalmente, as bibliotecas VMGJ, VECJ, VCR e VMN possuem testes unitários para verificação da instalação pelo comando:

```
# make check
```

2.4.3.2. Avaliação de eficiência

Para avaliar a eficiência dos métodos do Verificatum, foram analisadas suas três principais funcionalidades:

- *Shuffle*: embaralhamento e recifração dos textos cifrados, juntamente com a criação das provas de conhecimento-zero das operações.
- *Decrypt*: decifração dos textos cifrados, com a criação das provas de conhecimento-zero que garantem a correta decifração.

- *Mixing*: operação conjunta de embaralhamento, recifração e decifração dos textos cifrados, quando executados por apenas um nó.

Os testes foram feitos em um hardware com as seguintes características:

- Processador: Intel Atom Elkhart Lake Processor, J6412/X6413E
- Memória: 2 x 8GB SODIMM DDR4 3200Mhz
- Disco: SSD Crucial P2 500 GB 3D NAND NVMe PCIe M.2 (Sequential Read - 2300 MB/s, Sequential Write - 940 MB/s)
- Sistema operacional: Ubuntu 22.04.1 LTS

A biblioteca foi configurada para utilizar a cifra ElGamal baseada em curvas elípticas. Para garantir o nível de segurança de 128 bits, a curva selecionada foi a curva P-256, presente na implementação do Verificatum, e contém trecho de código em assembly da biblioteca openssl para aceleração das operações.

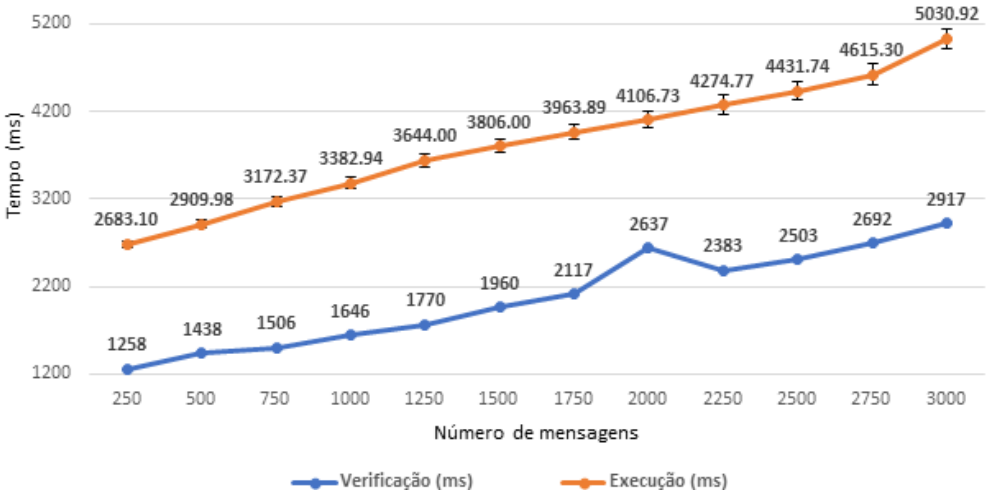
Para cada uma das funcionalidades, foram realizadas 1000 execuções independentes, com chaves geradas aleatoriamente a cada chamada para avaliar variações no tempo de execução e tamanho das provas de conhecimento-zero. Em cada execução, variou-se a dimensão dos vetores de textos cifrados entre 250 e 3000 mensagens (com passo de 250). Estes valores foram definidos considerando cenários próximos ao atualdo sistema brasileiro, em que cada urna atenda até 500 eleitores em uma eleição de até 6 cargos e cada cargo gera uma mensagem. Cabe notar, entretanto, que a separação de candidatos por cargo permite reduzir esse cenário em que 3000 mensagens são embaralhadas para um cenário em que 6 grupos distintos de 500 mensagens são embaralhados independentemente.

Para exemplificar o pior caso em questão, com 3000 mensagens, a Tabela 2.2 apresenta esse cenário para cada método executado pelo nó misturador. Pode-se observar que os tempos de execução e verificação do método "mixing" é um pouco menor que a soma dos tempos de "shuffle + decrypt". Além disso, nota-se que as provas de conhecimento-zero com custo predominante são aquelas do embaralhamento.

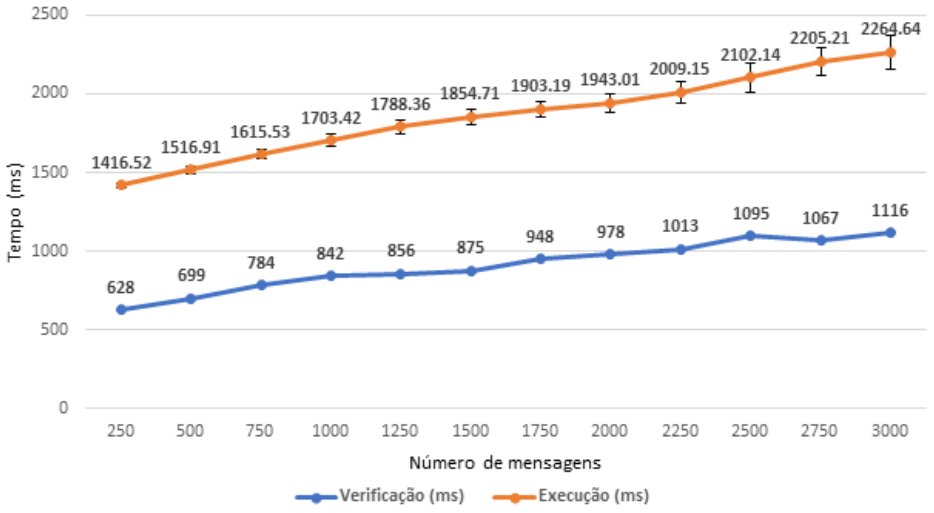
Tabela 2.2: Tempos de execução e verificação e tamanho das provas para 3000 mensagens

Método	Tempo (ms)		Tamanho (KB) Provas
	Execução	Verificação	
Shuffle	5030,92	2917	2359
Decript	2264,64	1116	949,72
Mixing	5935,27	3338	2359

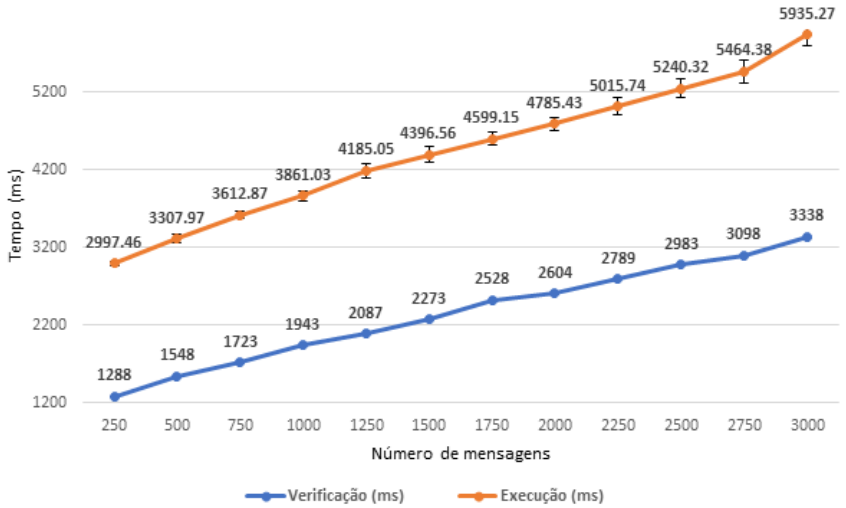
Para entender melhor como esses custos escalam, os gráficos da Figura 2.9 a seguir apresentam a média dos tempos de execução e verificação de cada método com a variação do número de textos cifrados. Já os gráficos da Figura 2.10 apresentam o tamanho das provas de conhecimento-zero que devem ser publicadas para realização da auditoria.



(a) Tempo de execução da operação Shuffle

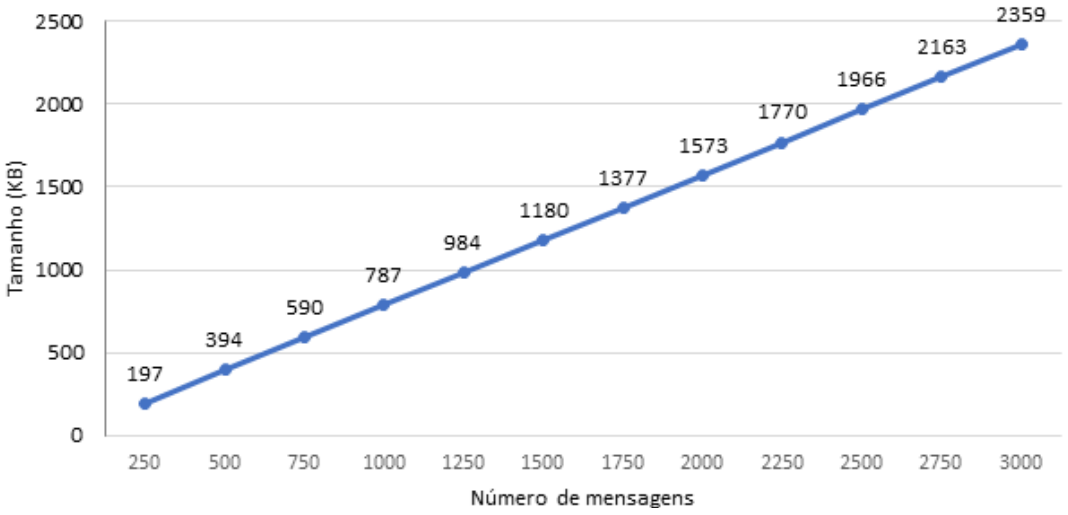


(b) Tempo de execução da operação Decrypt

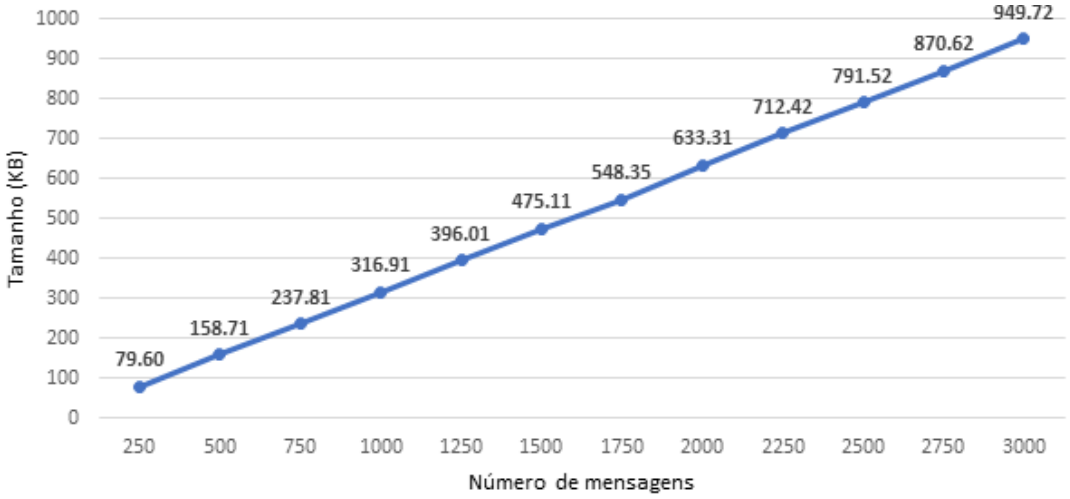


(c) Tempo de execução da operação Mixing

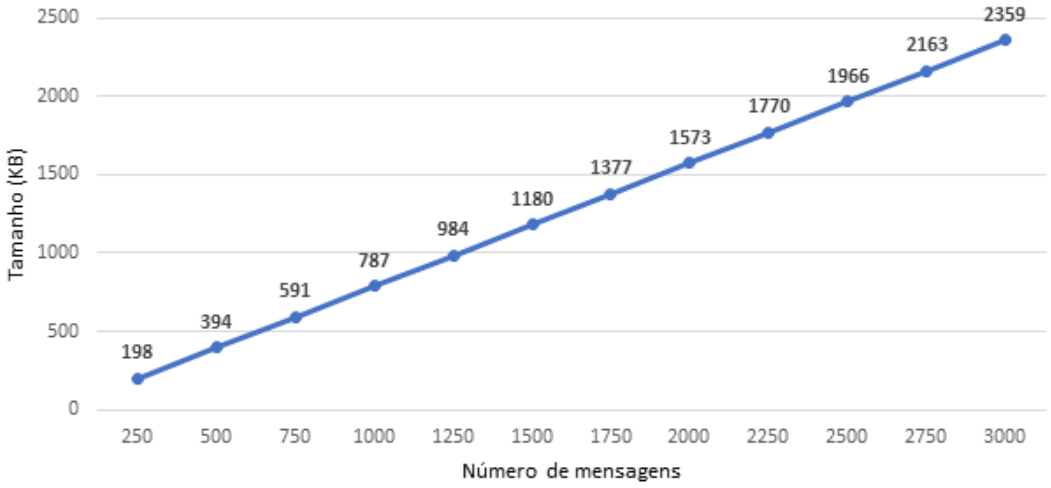
Figura 2.9: Tempo de execução das operações da biblioteca Verificatum



(a) Tamanho das provas da operação Shuffle



(b) Tamanho das provas da operação Decrypt



(c) Tamanho das provas da operação Mixing

Figura 2.10: Tamanho das provas das operações da biblioteca Verificatum

Por fim, é possível estimar o tempo de execução e verificação para quantidades diferentes de votos em dada arquitetura, além de conhecer o tamanho das provas que serão geradas. Em teoria, o tempo de execução $E(x)$ e de verificação $V(x)$ e o tamanho das provas $P(x)$ crescem linearmente com o número de votos x. Na prática, os algoritmos realizados apresentam uma pequena variação nos tempos de execução, principalmente em pontos em que é necessário gerar números aleatórios. Portanto, para obter as relações mencionadas, fez-se a regressão linear dos valores medidos, obtendo as equações aproximadas apresentadas na Tabela 2.3.

Tabela 2.3: Estimativa de desempenho e tamanho dos métodos da biblioteca Verificatum para um número x de votos. $E(x)$ corresponde ao tempo de execução, $V(x)$ ao tempo de verificação e $P(x)$ ao tamanho.

	$E(x)$ (ms)	$V(x)$ (ms)	$P(x)$ (KB)
Shuffle	2563,5 + 0,78x	1092,6 + 0,6x	0,9 + 0.78x
Decrypt	1383,6 + 0,29x	632,3 + 0,17x	0,15 + 0,32x
Mixing	2838,9 + 0,99x	1183 + 0,72x	1,1 + 0,79x

2.4.3.3. Adaptações na biblioteca

A biblioteca Verificatum foi projetada como uma implementação pura de uma mixnet. Para sua utilização no ambiente eleitoral, tornam-se necessárias algumas adaptações no código original: a exposição do método de cifração, o agrupamento de textos cifrados, e a decodificação de pontos de curvas elípticas associadas a mensagens. Esses pontos são discutidos a seguir.

Primeiramente, como o primeiro nó misturador em uma mixnet de recifração normalmente recebe as mensagens já cifradas pelos usuários, e a cifração pura não precisa ser utilizada em outros pontos do sistema, o Verificatum não expõe ao usuário o método de cifração. Além disso, o Verificatum armazena os textos cifrados de cada nó em um único arquivo. Para permitir que uma urna no cenário de auditoria cifre os votos do usuário, foram incluídos os seguintes métodos:

- A cifração de uma mensagem às claras (plaintext) com a chave pública (publicKey) produz um texto cifrado que será salvo no arquivo ciphertexts, e o número aleatório utilizado na cifração está salvo no arquivo nonce. Caso o arquivo nonce não exista, o arquivo é criado e o nonce gerado aleatoriamente é salvo nele. Cabe notar que esse mecanismo com uso de arquivo temporário não precisaria ser utilizado em uma implementação de fato, mas foi adotado no cenário de testes por ser mais próximo dos métodos já existentes na biblioteca do Verificatum.

```
vmnd -enc <publicKey> <plaintext> <ciphertexts> <nonce>
```

- O texto cifrado ciphertextin, criado com a chave pública publicKey, é acrescentado à lista de textos cifrados ciphertexts.

```
vmnd -apnd <publicKey> <ciphertexts> <ciphertextin>
```

Além disso, para fins de teste, torna-se necessário realizar a conferência dos métodos anteriores. No entanto, a cifra ElGamal construída sobre curvas elípticas codifica a mensagem a ser cifrada em um ponto da curva. Consequentemente, a decifração das mensagens retorna o ponto de curva elíptica que foi codificado. Assim, para averiguação das mensagens às claras após a decifração, também foi criado um método para decodificação das mensagens decifradas representadas por um ponto de curva elíptica:

- As mensagens decifradas messages, usado o ponto base da curva elíptica armazenado na chave publicKey, são decodificadas e salvas em decoded.

```
vmnd -dec <publicKey> <messages> <decoded>
```

2.5. Sistema E2E baseado em Criptografia Homomórfica – ElectionGuard

Esta Seção tem como objetivo apresentar a construção e a operação de um sistema de votação fim-a-fim que utiliza a biblioteca ElectionGuard, baseada em criptografia homomórfica.

O funcionamento e características do sistema serão descritos, através do uso dos blocos criptográficos previamente apresentados. Por fim, será apresentada a implementação do sistema por meio da implementação fornecida pela equipe do ElectionGuard.

2.5.1. Criptografia Homomórfica

Em 1978, Rivest, Adleman e Dertouzos propuseram uma classe de funções especiais de criptografia que eles chamaram de "homomorfismos de privacidade" [Rivest et al. 1978]. Estas funções especiais de criptografia permitem que dados cifrados possam ser operados sem a necessidade de sua decifração. A Figura 2.11 apresenta uma referência visual de criptografia homomórfica. Dados dois textos às claras m_1 e m_2 e suas cifrações c_1 e c_2, existe uma operação $\diamond$ aplicada aos textos cifrados que é equivalente a uma operação $\star$ aplicada aos textos às claras. Desta maneira, a decifração de $c_1 \diamond c_2$ é igual a $m_1 \star m_2$. Historicamente, alguns esquemas de criptografia bastante populares, como o RSA e o ElGamal, possuem uma propriedade homomórfica, como mostrado, para o último caso, na Seção 2.2.3.1.

No caso de eleições, cada eleitor tem direito a produzir um voto que representa a sua escolha e o conjunto destes votos deve ser agrupado no fim do processo, a fim de se calcular o resultado e se determinar a escolha ou escolhas vencedoras. Além disso, uma propriedade essencial de um voto é o sigilo, não devendo ser possível um terceiro determinar a escolha de um eleitor. Esta propriedade é especialmente relevante, pois ela garante ao eleitor o seu direito de livre escolha, mitigando as chances de uma eventual coerção.

Diferentemente de sistemas de votação baseados em redes de misturadores, em sistemas homomóficos os votos não são decifrados individualmente. Em outras palavras, por meio da propriedade de homomorfismo do criptosistema, um conjunto de votos cifrados são homomorficamente somados, e apenas o resultado dessa combinação de textos cifrados é então decifrado. Desta forma, um esquema criptográfico que permita a soma dos textos às claras a partir de seus textos cifrados é de especial interesse ao cenário de

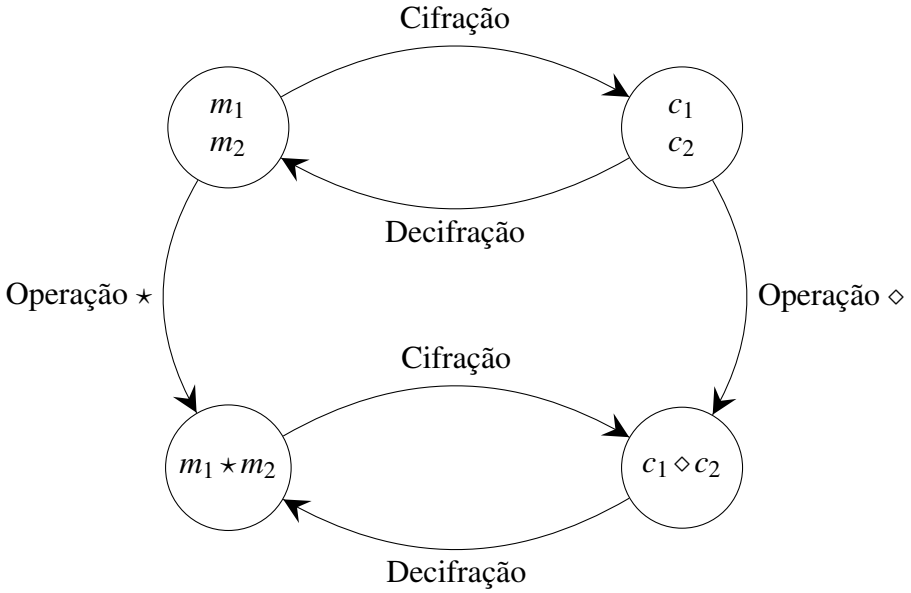


Figura 2.11: Cifração homomórfica: existe um morfismo entre a operação $\star$ realizada nos textos às claras m_1 e m_2 e a operação $\diamond$ realizada nos textos cifrados c_1 e c_2.

uma eleição. O eleitor seria capaz de criar um voto cifrado e, portanto, sigiloso, que poderia ser compartilhado publicamente. Em seguida, com todos os votos cifrados publicados, qualquer pessoa seria capaz de realizar a soma homomórfica dos votos, garantindo a utilização de todos os votos, e verificar sua igualdade com a soma cifrada publicada pela apuração oficial. Por fim, como apenas esta soma seria decifrada, o sigilo individual de cada voto seria preservado.

2.5.2. ElectionGuard

O ElectionGuard [Benaloh and Naehrig 2022] é um conjunto de ferramentas (SDK) de código aberto, cujo objetivo é o desenvolvimento de sistemas eleitorais seguros e transparentes. Para isso, o ElectionGuard utiliza um conjunto de primitivas criptográficas com a intenção de implementar a verificação fim-a-fim (E2E) ao processo. Especificamente, o ElectionGuard emprega criptografia homomórfica para garantir as propriedades fundamentais de segurança de um sistema eleitoral.

No ElectionGuard, o voto de um eleitor é constituído por um conjunto ordenado formado pela representação individual de sua escolha para cada uma das opções válidas no pleito. Caso o eleitor deseje votar em uma opção, esta opção recebe o valor 1; caso contrário, a opção recebe o valor 0. Desta maneira, em um pleito onde o eleitor pode selecionar x das n opções válidas, um voto final será representado pelo concatenamento ordenado das n opções, onde x delas apresentam o valor 1 e o restante apresentam o valor 0.

Em seguida, cada uma das opções é individualmente cifrada utilizando um esquema criptográfico homomórfico para soma, como o ElGamal Exponencial 2.2.3.2. Este voto cifrado é então adicionado a lista de votos depositados.

Por fim, uma vez finalizado o pleito, a operação homomórfica para soma é empregada em cada uma das opções individuais, para todos os votos depositados. Consequentemente, é obtido um resultado cifrado em que cada opção individual cifrada contém a soma do número de vezes que a opção foi escolhida. Portanto, a decifração única deste resultado para cada opção produz diretamente o resultado do pleito, não sendo necessária a decifração de votos individuais.

Para garantir o sigilo e integridade do processo, outros mecanismos, além do esquema criptográfico homomórfico, são necessários. Estes mecanismos serão apresentados nas seções a seguir.

2.5.2.1. Privacidade

Como apresentado anteriormente, uma opção de voto no ElectionGuard possui o valor 1 caso seja escolhida como voto ou 0 do contrário. Cada uma destas opções é individualmente cifrada e, no fim da eleição, somada homomórficamente com os outros votos para produzir o resultado cifrado final. Deste modo, somente o resultado cifrado final precisa ser decifrado para produzir o resultado da eleição. Logo, o voto individual cifrado de cada eleitor nunca é revelado, tendo sua privacidade protegida pelas garantias de segurança do criptosistema empregado.

Entretanto, uma entidade que é capaz de decifrar o resultado final também é capaz de decifrar os votos individuais, uma vez que eles estão protegidos pelo mesmo conjunto de chaves criptográficas. Portanto, esta possibilidade pode representar um risco a privacidade de um eleitor caso a entidade adote um comportamento malicioso.

A fim de mitigar este risco, o ElectionGuard faz o uso do criptosistema ElGamal com decifração limiar 2.2.3.4. Através do uso desta modalidade do esquema ElGamal, o ElectionGuard define a figura de "Guardiões da eleição". Estes Guardiões nada mais são do que entidades confiáveis, responsáveis pela geração de chaves criptográficas no criptosistema, além da decifração do resultado final. Assim, enquanto existe uma chave pública única que é utilizada para cifrar os votos durante uma eleição, a chave privada correspondente é desconhecida individualmente por cada um dos Guardiões. Como resultado, para decifrar um voto, é necessário que um número mínimo pré-estabelecido de guardiões atuem em conjunto. Portanto, para que um voto individual seja revelado, seria necessário haver um conluio entre múltiplas entidades do sistema, mitigando o risco deste ataque.

2.5.2.2. Integridade e Transparência

A fim de garantir a integridade de um processo eleitoral e ao mesmo tempo aumentar sua transparência, o ElectionGuard emprega dois mecanismos: o desafio de Benaloh e provas de conhecimento não interativas.

O desafio de Benaloh é utilizado pelo sistema para fornecer evidências ao eleitor que seu voto encontra-se corretamente representado.

Como descrito na Seção 2.3.1, no desafio de Benaloh, o eleitor pode desafiar o

sistema que cifrou o seu voto a revelar o nonce utilizado no processo. Após isso, o eleitor (ou terceiros) pode utilizar um dispositivo computacional diferente para calcular novamente o texto cifrado e o Código de Rastreio resultante. Com isto, ele é capaz de verificar se o texto cifrado resultante é igual ao produzido pelo sistema de origem.

Já as provas de conhecimento não interativas são usadas pelo sistema para fornecer evidências a todos os participantes e observadores do processo eleitoral de que as etapas envolvidas em sua operação estão sendo executadas de maneira correta. Destas provas, destacam-se as utilizadas pelo dispositivo responsável por cifrar o voto de cada eleitor e as utilizadas pelos Guardiões ao término do pleito.

Diferentemente de um sistema baseado em redes de misturadores, um sistema baseado em criptografia homomórfica nunca realiza a decifração individual de um voto. Portanto, é necessário que o dispositivo que realizou a cifração de um voto demonstre que este voto está bem formado. Isto significa provar, primeiramente, que para cada opção do voto ela foi escolhida, no máximo, uma única vez. Em seguida, prova-se que o total de opções escolhidas em um voto é igual ao total de escolhas permitidas em um voto. Como exemplo, considerando a eleição presidencial brasileira, o dispositivo precisa provar que cada candidato recebeu 0 ou 1 votos e que apenas um único candidato foi escolhido. A boa formação do voto é imprescindível para a integridade do sistema. Sem ela, seria possível ao dispositivo responsável por cifrar o voto a criação de votos com múltiplas opções escolhidas, ou múltiplas escolhas para uma única opção. Além disso, seria possível até mesmo a presença de um voto "corretor", onde números negativos são cifrados em algumas opções e múltiplos votos são cifrados em outras, produzindo um total de escolhas aparentemente correto.

Para tanto, o dispositivo que cifra o voto dos eleitores executa dois conjuntos de provas de conhecimento não interativas. O primeiro conjunto é executado para provar que cada opção de um voto recebeu o valor 0 ou 1. Para isto, o dispositivo emprega a técnica de Cramer-Darmgård-Schoenmakers 2.2.5.2 em associação com provas de Chaum-Pedersen 2.2.5.1 para valores de mensagem 0 e 1.

O segundo conjunto é executado para provar que a soma de todas as opções de um voto corresponde a um valor específico. Este valor específico é o total de escolhas permitidas em um único voto. Para isto, o dispositivo soma homomorficamente todas as opções e, utilizando novamente uma prova de Chaum-Pedersen, prova que está soma possuí o valor esperado.

Unindo os dois conjuntos de provas, qualquer participante ou observador do processo eleitoral recebe evidências de que um voto depositado é bem formado, embora não se revele o conteúdo do voto em si.

Já ao final do processo, é necessário que os guardiões provem que o resultado do pleito às claras realmente é igual ao contido no resultado final cifrado obtido pela soma homomórfica dos votos. Caso isto não fosse feito, os guadiões poderiam divulgar um resultado aleatório qualquer. Para realizar esta prova, os guardiões novamente adotam a prova de conhecimento de Chaum-Pedersen, com algumas modificações devido a decifração parcial resultante da utilização da modalidade de decifração limiar do criptosistema ElGamal.

Unindo todos as técnicas apresentadas nesta seção, o eleitor é capaz de averiguar que seu voto foi corretamente gravado. O eleitor também é capaz de observar que seu voto cifrado foi utilizado para a produção do resultado final, pois todos os votos do sistema, assim como a operação homomórfica necessária, são públicos. Logo, qualquer pessoa é capaz de produzir o resultado final utilizado pelos guardiões para a divulgação da apuração. Por fim, as provas realizadas pelo dispositivo e guardiões fornecem elementos para a garantia de que apenas votos válidos foram utilizados e de que o resultado às claras final é um produto direto do resultado cifrado final. Desta maneira, todo o processo eleitoral é verificável por qualquer observador, garantido a sua transparência.

2.5.3. Cenários para uso do ElectionGuard

Assim como no sistema baseado em redes de misturadores, propõe-se os mesmos dois cenários para o uso da ferramenta ElectionGuard, o **tradicional** e o **modelo de auditoria**.

2.5.3.1. Cenário Tradicional

Assim como no caso do uso de redes de misturadores, o cenário tradicional com o ElectionGuard também é adaptado ao voto presencial. Os componentes necessários para este cenário são uma urna eletrônica, como plataforma de votação, um sistema para a totalização dos votos, e um conjunto de entidades, que serão os guardiões do processo eleitoral.

Como preparação para a eleição, cada um dos guardiões gera seu próprio par de chaves privada e pública. As chaves públicas então são combinadas para formar a chave de cifração para a eleição. Esta chave de cifração é então previamente carregada em todas as urnas que serão utilizadas e divulgada para os eleitores e observadores.

No processo de votação, a urna funciona como plataforma de votação e é responsável por realizar corretamente a cifração dos votos dos eleitores e as provas de conhecimento descritas na Seção 2.5.2.2. Quando o eleitor confirmar suas escolhas, a urna deve emitir o Código de Rastreio, como apresentado na Seção 2.3.1.4. Caso o eleitor deseje depositar este voto, o Código de Rastreio deve ser assinado e poderá ser utilizado posteriormente como material de auditoria da integridade dos votos contabilizados.

Neste momento, como o voto não é criado por dispositivo do próprio eleitor, este pode não ter a confiança de que a urna está realizando a operação de forma correta, ou seja, ele pode desconfiar que o comprovante de votação está apresentando um valor diferente do inserido na urna. Por este motivo, o eleitor pode desafiar a urna por meio do desafio de Banaloh como descrito na Seção 2.3.1.5.

Ao final do processo de votação, inicia-se diretamente o processo de totalização. Todos os votos depositados na urna são enviados para o sistema de totalização, em conjunto com suas provas de conhecimento. Este sistema, então, verifica as provas de conhecimento de cada voto e, caso as provas sejam válidas, o soma homomorficamente ao total da eleição. Uma vez que todos os votos tenham sido somados, o resultado cifrado final é publicado e os guardiões iniciam o seu procedimento de decifração. Nesta etapa, os guardiões geram as provas de conhecimento de correta decifração do resultado e publicam o resultado às claras final em conjunto com as estas provas.

Para a auditoria, eleitores e observadores podem verificar a corretude de todas as provas de conhecimento produzidas, sejam elas referentes aos votos ou a decifração do resultado. Além disso, eleitores que desejem auditar os seus votos podem verificar que o voto associado ao seu Código de Rastreio foi utilizado para o cálculo do resultado cifrado.

2.5.3.2. Cenário de Auditoria

Assim como no caso de uso de redes de misturadores, a arquitetura do sistema neste cenário é voltada para dar poder de auditoria aos votos em uma urna eletrônica. Considera-se que esta é o suficiente para dar privacidade aos votos dos eleitores contra atacantes externos. Como resultado, o sistema é mais simplificado do que no cenário anterior, requisitando apenas dois componentes: a urna eletrônica, que executa a biblioteca ElectionGuard em sua totalidade, e um sistema de totalização para votos às claras.

O processo de preparação para eleição ocorre individualmente para cada urna. Cada uma das urnas é considerada o único guardião de uma eleição que abrange apenas os eleitores desta urna. A urna cria um par de chaves pública-privada, onde a chave pública será utilizada como a chave de cifração para a urna. Adicionalmente, a chave de cifração é divulgada para todos os interessados.

O processo de votação ocorre de maneira análoga ao cenário tradicional, os eleitores tem seus votos cifrados pela urna, que também gera as provas de conhecimento para eles. Os votos podem então serem desafiados ou depositados, neste último caso sendo realizadas as assinaturas dos Códigos de Rastreio.

A diferença principal dos cenários ocorre ao final do processo de votação. Nesta etapa, a urna soma homomorficamente todos os votos e realiza a decifração do resultado final, produzindo a apuração para aquela urna. A prova de correta decifração é igualmente produzida e todo o material é enviado ao sistema de totalização.

O sistema de totalização, por sua vez, apenas verifica as provas de conhecimento produzidas pela urna e, em caso de elas estarem corretas, soma o resultado já decifrado da urna ao total da eleição. Este processo é bastante similar ao já existente no sistema de votação brasileiro, onde o Tribunal Superior Eleitoral (TSE) soma os resultados parciais de cada urna no país, os Boletins de Urna (BU), para totalizar a eleição. Desta maneira, assim como no caso de uso de redes de misturadores, o cenário de auditoria aparenta ser uma escolha natural para o contexto de votação brasileiro.

Também de maneira análoga ao caso de uso de redes de misturadores e através da mesma argumentação, neste cenário a confidencialidade dos votos obtida é essencialmente a mesma do que em uma urna eletrônica tradicional brasileira.

2.5.3.3. Instalação

O núcleo da biblioteca possui implementações nas linguagens Python e C++. No entanto, somente a versão Python possui uma API e uma interface de usuário. Para fins de instalação do SDK e seus adicionais, são referenciados os seguintes endereços:

A versão do SDK em linguagem Python está disponível em `https://github.com/microsoft/electionguard-python` e sua versão C++ em `https://github.com/microsoft/electionguard-cpp/`. A API Python pode ser encontrada em `https://github.com/microsoft/electionguard-api-python`. Uma implementação de rederência da interface de usuário está disponível em `https://github.com/microsoft/electionguard-ui/tree/main`.

2.5.3.4. Avaliação de eficiência

Diferentemente de um sistema de votação baseado em redes de misturadores, um sistema baseado em criptografia homomórfica não necessita de um procedimento especial após o período de votação para garantir privacidade ou integridade. Estas garantias são conferidas no momento de cifração do voto e criação das provas de conhecimento associadas a ele. Deste modo, para avaliar a eficiência da biblioteca ElectionGuard, optou-se por aferir o tempo necessário para a etapa de criação do voto. Esta etapa é a que impacta fundamentalmente no tempo de resposta do sistema para o eleitor, já que este deve esperar sua conclusão para receber seu Código de Rastreio e realizar a escolha entre depósito ou desafio.

O hardware utilizado no teste é o mesmo apresentado na Seção 2.4.3.2:

- Processador: Intel Atom Elkhart Lake Processor, J6412/X6413E
- Memória: 2 x 8GB SODIMM DDR4 3200Mhz
- Disco: SSD Crucial P2 500 GB 3D NAND NVMe PCIe M.2 (Sequential Read - 2300 MB/s, Sequential Write - 940 MB/s)
- Sistema operacional: Ubuntu 22.04.1 LTS

Foi utilizada a versão 1.3.0 de 12/10/2021 da biblioteca ElectionGuard em Python, sendo executada com Python versão 3.9.2.

Para os votos em si, foram considerado votos com 100, 200, 500, 1000, 1500, 2000, 2500 e 3000 candidatos. É importante observar que o tempo de cifração e criação de provas é linearmente dependente ao número de candidatos. Isto ocorre pois cada candidato acrescenta é cifrado individualmente e requer provas igualmente individuais, com exceção da prova de total de escolhas permitidas.

Também é importante notar que o número de candidatos considerados é extremamente alto em alguns cenários. Esta escolha foi feita para considerar a eleição para vereadores na cidade de São Paulo, que apresenta esta quantidade de candidatos. Apesar de não se esperar que os tempos para estes cenários fossem factíveis para uma implementação de mundo real, eles permitem com que se faça uma interpolação dos resultados para obter o tempo esperado por candidato na disputa. Consequentemente, inicialmente os resultados destes grandes cenários são apresentados de maneira integral na Figura 2.12 Posteriormente, são apresentados os tempos interpolados para cada candidato em uma cédula de votação, na Tabela 2.4.

A fim de obter um tempo de resposta menor do que 2 segundos, o número máximo de candidatos possíveis é igual a 11. Esta quantidade de candidatos é adequada para eleições de cargo majoritário, como presidente e governador. Entretanto, para um cenário de cargos minoritários, especialmente no Brasil, esta quantidade de candidatos é insuficiente.

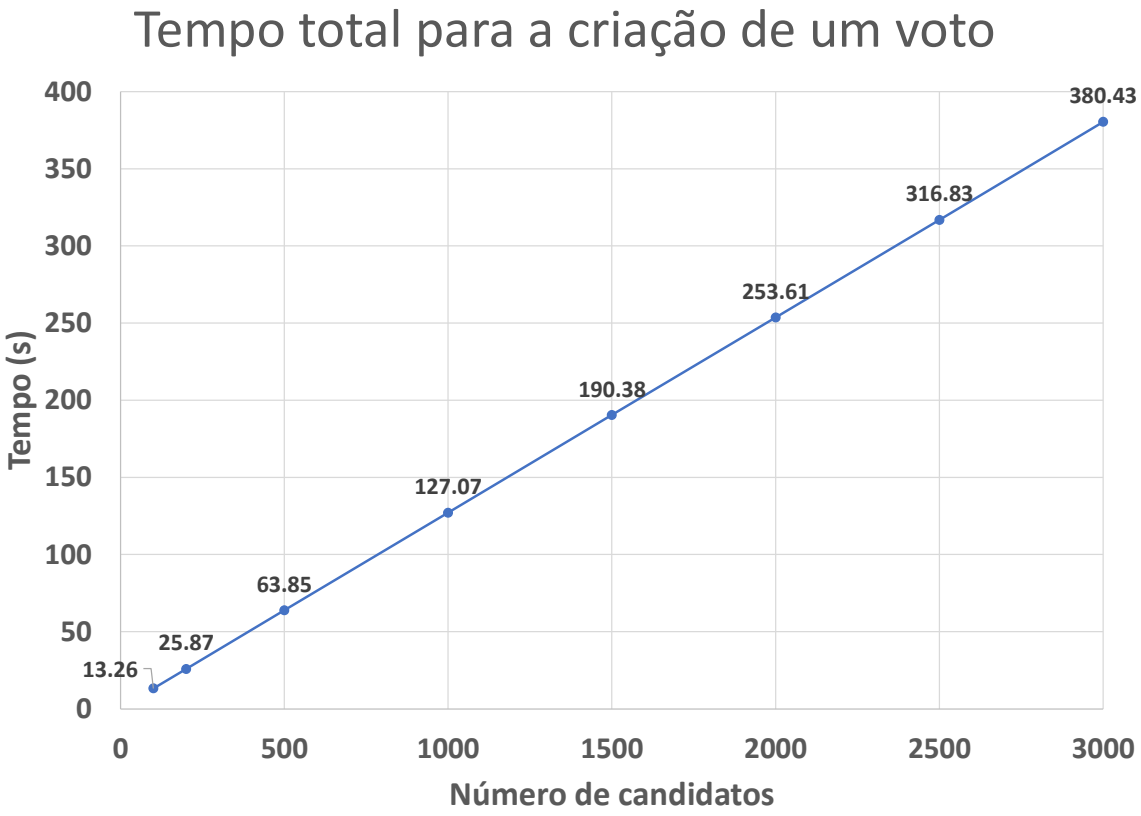


Figura 2.12: Tempo para a criação de um voto na biblioteca ElectionGuard para cédulas com múltiplos candidatos. O tempo inclui a cifração e a elaboração das provas de conhecimento.

De maneira análoga, a Figura 2.13 apresenta o tamanho de um voto gerado pela biblioteca ElectionGuard para os mesmos cenários anteriores. O tamanho dos votos para cada candidato também é interpolado e incluído na Tabela 2.4

Para o cenário de 11 candidatos abordado anteriormente, o voto de cada eleitor possui o tamanho de aproximadamente $0,219$ MB. Desta maneira, para cada 1000 votos, o sistema ElectionGuard produz cerca de 219 MB de dados.

Tabela 2.4: Interpolação do tempo necessário para a criação e do tamanho final de um voto no ElectionGuard para cada candidato.

	Tempo (s)	Tamanho (MB)
Inicial	$0,545$	$0,13427$
Por candidato	$0,127$	$0,00768$

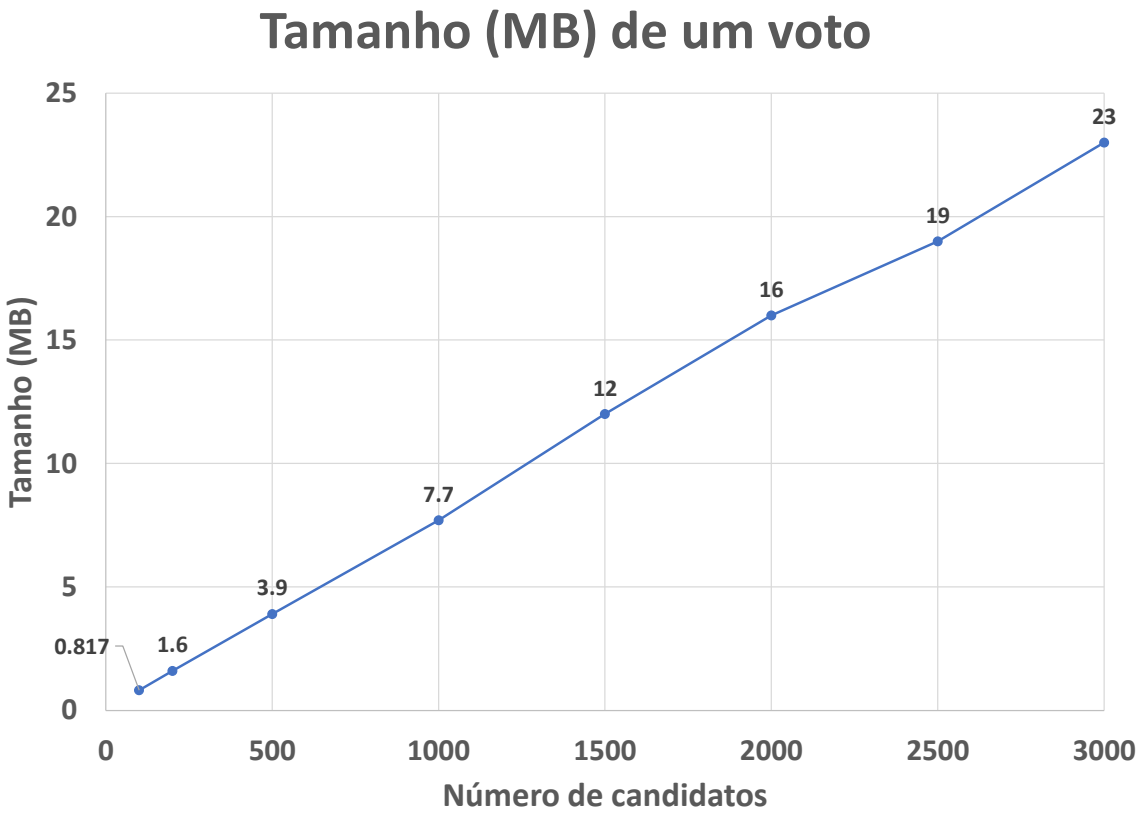


Figura 2.13: Tamanho de um voto na biblioteca ElectionGuard para cédulas com múltiplos candidatos. O tamanho inclui a cifração e as provas de conhecimento.

2.6. Comparação entre Sistemas baseados em Redes de Misturadores e Homomorfismo

Nesta Seção, é realizada uma comparação entre os dois modelos de votação fim-a-fim. Inicialmente, a necessidade de tratar os dados após o término da votação é comparada, pois esta é a principal diferença entre os dois métodos. Em seguida, são feitas comparações relativas a privacidade, integridade e desempenho obtidas pelos esquemas estudados.

2.6.1. Tratamento de dados após encerramento da votação

A principal diferença entre um esquema que utiliza redes de misturadores e criptografia homomórfica é a necessidade de tratar os dados após o encerramento da votação.

Em um sistema de redes de misturadores, quando a eleição se encerra, é necessário que os votos cifrados sejam transmitidos a um ou mais nós misturadores antes de serem entregues ao sistema de totalização. Este é um passo fundamental, pois como os votos serão individualmente decifrados para a apuração, é preciso quebrar o vínculo entre os votos inicialmente cifrados e os eleitores que o adicionaram ao sistema. Desta maneira, existe um tráfego de dados extra, além de um pós processamento relativo ao processo de mixagem dos votos pelos nós misturadores.

Já em um sistema baseado em criptografia homomórfica, os votos cifrados vinculados a eleitores nunca são individualmente decifrados. Logo, após o encerramento da

votação, estes dados podem ser transmitidos diretamente ao sistema de totalização.

Consequentemente, em um sistema que utiliza redes de misturadores, existe um tempo adicional entre o fim da votação e o início da apuração. A superfície de ataque do sistema de redes de misturadores também é maior, visto que os nós misturadores são entidades independentes e podem ser individualmente atacadas. Entretanto, o tempo de processamento extra, relativo a aplicação do método "*mixing*" aos votos cifrados, não é grande, como indicado na Seção 2.4.3.2. Da mesma maneira, como é possível existirem mais nós misturadores do que a quantidade necessária, um eventual ataque em um ou alguns nós misturadores pode ser mitigada pelo uso dos demais nós.

2.6.2. Comparação de privacidade

A privacidade do eleitor em um sistema que usa redes de misturadores é dependente de ao menos um nó misturador honesto e da confiança na entidade, ou conjunto de entidades, que realiza a decifração dos votos.

No primeiro caso, ao menos um nó misturador honesto é necessário a fim de quebrar o vínculo entre os votos cifrados antes do processamento pelos nós e os votos cifrados após a execução do processo de mixagem pelos nós. Caso haja um conluio entre todos os nós misturadores, eles são capazes de reverter o processo de mistura e, assim que os votos na saída forem decifrados pelo sistema de totalização, associar um voto às claras a um eleitor.

Além disso, é preciso confiar na entidade, ou conjunto de entidades, que realiza a decifração dos votos misturados. Como a chave de decifração é a mesma tanto para os votos na entrada da rede de misturadores quanto para os votos na saída, a entidade responsável pela decifração poderia decifrar, adicionalmente, os votos não misturados. Consequentemente, a entidade seria capaz de associar votos às claras com eleitores. Este ataque pode ser mitigado pelo uso do sistema ElGamal com decifração limiar. Deste modo, seria necessário um conluio das entidades detentoras das chaves de decifração para executar este ataque.

No caso de um sistema baseado em criptografia homomórfica, a privacidade depende da confiança nos Guardiões do sistema, ou seja, nas entidades que realizam a decifração dos votos. A razão da necessidade de confiança nestas entidades é exatamente a mesma do modelo baseado em redes de misturadores: as entidades que possuem as chaves de decifração podem decifrar os votos individuais de cada eleitor, em adição a soma homomórfica. Igualmente ao explicado anteriormente, este ataque é mitigado pelo uso do sistema ElGamal com decifração limiar, ou seja, pelo uso de múltiplos Guardiões.

Como resultado da comparação de privacidade entre os dois sistemas de votação estudados, é exigido a confiança extra no sistema com uso de redes de misturadores de que exista ao menos um nó misturador honesto. Contudo, a presença de diversos nós misturadores e a possível inclusão e troca destes nós a qualquer momento do processo eleitoral reduz um possível risco de privacidade existente no modelo.

2.6.3. Comparação de integridade

Existe uma equivalência em relação a integridade entre os dois métodos estudados neste minicurso.

No sistema baseado em redes de misturadores, a integridade depende das provas de conhecimento zero de mistura, recifração e decifração, além do uso do desafio de Benaloh e da construção do Código de Rastreio.

Já no sistema que utiliza criptografia homomórfica, a integridade depende das provas de conhecimento zero de que os votos são bem formados e das provas de correta decifração, além de, de maneira igual ao anterior, do uso do desafio de Benaloh e da construção do Código de Rastreio.

Como a segurança e confiabilidade das provas utilizadas em ambos os métodos são similares, não existe uma diferença significativa em relação a integridade do processo em razão destas primitivas. Além disso, como o desafio de Benaloh e a construção do Código de Rastreio ocorre de maneira igual nos dois sistemas, também não existe uma diferença em relação a integridade neste ponto. Consequentemente, a integridade nos dois modelos estudados para a implementação de um sistema fim-a-fim não possuí uma diferença significativa notável.

2.6.4. Comparação de desempenho

A comparação de desempenho entre os dois modelos de sistema deve levar em conta o tempo gasto e o volume de dados gerados por cada um deles.

Em relação ao tempo gasto, o sistema baseado em redes de misturadores concentra este tempo no período posterior ao encerramento da votação. Para cada **voto** lançado por um eleitor, um nó misturador leva cerca de 1ms a mais de tempo para realizar a mistura.

Já em um esquema que usa criptografia homomórfica, o tempo gasto concentra-se no período de lançamento do voto, para cada eleitor individual. Para cada **candidato** extra existente em uma cédula de votação, o sistema necessita de 127ms a mais para criar o voto.

Portanto, caso o tempo seja o parâmetro crítico na escolha entre os dois métodos de implementação de um esquema de votação fim-a-fim, uma eleição com poucos candidatos é ideal para um sistema baseado em criptografia homomórfica, enquanto que uma eleição com poucos eleitores é recomendado para o modelo de redes de misturadores. Entretanto, o número de eleitores causa um impacto menor em uma rede de misturadores do que o número de candidatos causa em um sistema com uso de criptografia homomórfica. Este fato é ainda mais considerável pois o impacto do tempo em um esquema homomórfico recai no eleitor, que necessita esperar com que o sistema encerre a cifração de seu voto. Logo, para cada cenário de eleição é necessário um estudo levando em consideração os parâmetros de candidatos existentes e de número de eleitores.

No caso da eleição brasileira, que conta com muitos candidatos e muitos eleitores, o uso de um sistema baseado em criptografia homomórfica é inviável, pois o tempo de resposta do sistema para o eleitor, durante a criação do voto, é extremamente alto, chegando a ordem de minutos no caso de uma eleição para vereador.

Em relação ao volume de dados gerados, um sistema baseado em redes de misturadores cria uma prova de conhecimento no processo de mistura, que aumenta em aproximadamente $0,79$ KB para cada **voto** extra no sistema. Como cada nó misturador produz uma prova independente, o resultado acima é multiplicado pelo número de **nó misturadores** utilizados. Por outro lado, um esquema que utiliza criptografia homomórfica cria uma prova de conhecimento no processo de votação, que aumenta em aproximadamente $7,7$ KB para cada **candidato** extra na cédula. Como cada eleitor produz um voto, o resultado acima também é multiplicado pelo número de eleitores, ou **votos**, no sistema.

Consequentemente, um sistema baseado em criptografia homomórfica gera um volume de dados maior do que um esquema que utiliza rede de misturadores, ao menos que o número de nós misturadores seja bastante elevado. No caso de uma eleição com dois candidatos, um nó misturador cria uma prova de tamanho 790 MB para 1 milhão de eleitores. Enquanto isso, um sistema homomórfico produz um voto de tamanho $0,14963$ MB para cada eleitor, totalizando 149.630 MB para 1 milhão de eleitores. Desta maneira, seriam necessários cerca de 190 nós misturadores para que o volume de dados fosse equivalente.

É importante observar que no caso de uso de um sistema de rede de misturadores, os votos cifrados, e apenas os votos cifrados, devem ser repassados entre os diversos nós misturadores, gerando um tráfego extra de rede, Entretanto, o tamanho de um voto cifrado, sem provas de conhecimento, é pequeno. Logo, mesmo levando em conta um potencial uso extra de rede no cálculo acima apresentado, um sistema baseado em redes de misturadores continua necessitando de uma banda de comunicação consideravelmente inferior a de um esquema de criptografia homomórfica.

2.7. Considerações Finais

Neste minicurso foi realizada uma apresentação dos principais conceitos de um sistema de votação fim-a-fim, ou E2E. Primeiramente, foram apresentadas as principais primitivas criptográficas utilizadas na construção dos sistemas fim-a-fim de interesse para este documento.

Em seguida, foi realizada uma apresentação geral sobre sistemas E2E, com os principais mecanismos utilizados e as principais características oferecidas por este método.

Uma vez realizada a apresentação geral da teoria de sistemas fim-a-fim, foram discutidas as duas implementações mais populares desta técnica: através do uso de redes de misturadores e através do uso de criptografia homomórfica.

Por fim, foi feita uma breve comparação entre os dois métodos de implementação, destacando suas diferenças e semelhanças em relação as principais propriedades de segurança desejadas em um sistema de votação. Uma pequena comparação em relação ao desempenho dos dois sistemas também foi realizada.

Um dos principais desafios em relação a sistemas fim-a-fim é a explicação de seus conceitos para uma população leiga. A população, para confiar no sistema de votação, precisa acreditar que os mecanismos de integridade e privacidade implementados pelas primitivas criptográficas realmente funcionam. Como estes mecanismos envolvem o uso

de matemática complexa, mecanismos de explicação mais simples, como alegorias, devem ser desenvolvidos.

Um outro desafio é o estudo do cenário de votação ao qual o sistema E2E será aplicado a fim de determinar qual implementação deve ser usada. Embora existam casos nos quais a adoção de um ou outro sistema é evidente, também existem cenários nos quais ambas as implementações podem ser consideradas. Consequentemente, cada cenário necessita ser individualmente analisado a fim de se adotar a melhor escolha possível.

Referências

[Adida 2008] Adida, B. (2008). Helios: Web-based open-audit voting. In *USENIX security symposium*, volume 17, pages 335–348.

[Benaloh 2006] Benaloh, J. (2006). Simple verifiable elections. In *Proceedings of the USENIX/Accurate Electronic Voting Technology Workshop 2006 on Electronic Voting Technology Workshop*, pages 5–5.

[Benaloh and Naehrig 2022] Benaloh, J. and Naehrig, M. (2022). Electionguard specification 1.1. Technical report, Microsoft Research. `https://www.electionguard.vote/spec/`.

[Berlinski et al. 2021] Berlinski, N., Doyle, M., Guess, A. M., Levy, G., Lyons, B., Montgomery, J. M., Nyhan, B., and Reifler, J. (2021). The effects of unsubstantiated claims of voter fraud on confidence in elections. *Journal of Experimental Political Science*, pages 1—16.

[Chaum et al. 2008] Chaum, D., Carback, R., Clark, J., Essex, A., Popoveniuc, S., Rivest, R. L., Ryan, P. Y., Shen, E., Sherman, A. T., et al. (2008). Scantegrity ii: End-to-end verifiability for optical scan election systems using invisible ink confirmation codes. *EVT*, 8(1):13.

[Chaum et al. 2021] Chaum, D., Carback, R. T., Clark, J., Liu, C., Nejadgholi, M., Preneel, B., Sherman, A. T., Yaksetig, M., Zhang, B., et al. (2021). Votexx: Coercion resistance for the real world (preliminary extended abstract). *UMBC Student Collection*.

[Chaum and Pedersen 1992] Chaum, D. and Pedersen, T. P. (1992). Wallet databases with observers. In *Annual international cryptology conference*, pages 89–105. Springer.

[Chaum et al. 2005] Chaum, D., Ryan, P. Y., and Schneider, S. (2005). A practical voter-verifiable election scheme. In *Computer Security–ESORICS 2005: 10th European Symposium on Research in Computer Security, Milan, Italy, September 12-14, 2005. Proceedings 10*, pages 118–139. Springer.

[Chaum 1981] Chaum, D. L. (1981). Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–90.

[Commission 2023] Commission, U. E. A. (2023). End to End (E2E) protocol evaluation process. https://www.eac.gov/voting-equipment/end-end-e2e-protocol-evaluation-process. Acessado em 2 de abril de 2023.

[Cramer et al. 1994] Cramer, R., Damgård, I., and Schoenmakers, B. (1994). Proofs of partial knowledge and simplified design of witness hiding protocols. In *Annual International Cryptology Conference*, pages 174–187. Springer.

[ElGamal 1985] ElGamal, T. (1985). A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE transactions on information theory*, 31(4):469–472.

[Fiat and Shamir 1986] Fiat, A. and Shamir, A. (1986). How to prove yourself: Practical solutions to identification and signature problems. In *Conference on the theory and application of cryptographic techniques*, pages 186–194. Springer.

[Haenni et al. 2017] Haenni, R., Locher, P., Koenig, R., and Dubuis, E. (2017). Pseudo-code algorithms for verifiable re-encryption mix-nets. In *Financial Cryptography and Data Security: FC 2017 International Workshops, WAHC, BITCOIN, VOTING, WTSC, and TA, Sliema, Malta, April 7, 2017, Revised Selected Papers 21*, pages 370–384. Springer.

[Hubbers et al. 2005] Hubbers, E., Jacobs, B., and Pieters, W. (2005). Ries-internet voting in action. In *29th Annual International Computer Software and Applications Conference (COMPSAC'05)*, volume 1, pages 417–424. IEEE.

[Jakobsson et al. 2002] Jakobsson, M., Juels, A., and Rivest, R. L. (2002). Making mix nets robust for electronic voting by randomized partial checking. In *11th USENIX Security Symposium (USENIX Security 02)*.

[Kelsey et al. 2010] Kelsey, J., Regenscheid, A., Moran, T., and Chaum, D. (2010). Attacking paper-based e2e voting systems. In *Towards Trustworthy Elections: New Directions in Electronic Voting*, pages 370–387. Springer.

[Lindeman and Stark 2012] Lindeman, M. and Stark, P. B. (2012). A gentle introduction to risk-limiting audits. *IEEE Security & Privacy*, 10(5):42–49.

[Lindeman and Stark 2020] Lindeman, M. and Stark, P. B. (2020). Tools for comparison risk-limiting election audits. https://www.stat.berkeley.edu/~stark/Vote/auditTools.htm. Acessado em 19 de julho de 2023.

[Nicas et al. 2022] Nicas, J., Milhorance, F., and Ionova, A. (2022). How Bolsonaro built the myth of stolen elections in Brazil. https://www.nytimes.com/interactive/2022/10/25/world/americas/brazil-bolsonaro-misinformation.html.

[Park et al. 1994] Park, C., Itoh, K., and Kurosawa, K. (1994). Efficient anonymous channel and all/nothing election scheme. In *Advances in Cryptology—EUROCRYPT'93: Workshop on the Theory and Application of Cryptographic*

Techniques Lofthus, Norway, May 23–27, 1993 Proceedings 12, pages 248–259. Springer.

[Perez 2021] Perez, E. (2021). Hacking to save democracy. *Voting Village - DEF CON*.

[Popoveniuc and Hosp 2010] Popoveniuc, S. and Hosp, B. (2010). An introduction to punchscan. In *Towards Trustworthy Elections: New Directions in Electronic Voting*, pages 242–259. Springer.

[Rivest et al. 1978] Rivest, R., Adleman, L., and Dertouzos, M. (1978). On data banks and privacy homomorphisms. *Foundations of secure computation*, 4(11):169–180.

[Rivest 2008] Rivest, R. L. (2008). On the notion of 'software independence'in voting systems. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 366(1881):3759–3767.

[Ryan et al. 2005] Ryan, P., Peacock, T., et al. (2005). Prêt à voter: a system perspective. *School of Computing Science Technical Report Series*.

[Sampigethaya and Poovendran 2006] Sampigethaya, K. and Poovendran, R. (2006). A survey on mix networks and their secure applications. *Proceedings of the IEEE*, 94(12):2142–2181.

[Terelius and Wikström 2010] Terelius, B. and Wikström, D. (2010). Proofs of restricted shuffles. In *Progress in Cryptology–AFRICACRYPT 2010: Third International Conference on Cryptology in Africa, Stellenbosch, South Africa, May 3-6, 2010. Proceedings 3*, pages 100–113. Springer.

[Zagórski et al. 2013] Zagórski, F., Carback, R. T., Chaum, D., Clark, J., Essex, A., and Vora, P. L. (2013). Remotegrity: Design and use of an end-to-end verifiable remote voting system. In *International Conference on Applied Cryptography and Network Security*, pages 441–457. Springer.

Capítulo 3

Introdução à Engenharia Social: da Psicologia Cognitiva aos Ataques Automatizados

Jéferson Campos Nobre (UFRGS), Pamela Carvalho da Silva (UFCSPA), Antônio João Gonçalves de Azambuja (UFRGS), Maurício Ariza (UFRGS), Lisandro Zambenedetti Granville (UFRGS) e Caroline Tozzi Reppold (UFCSPA)

Resumo

A Engenharia Social (ES) é uma disciplina que visa explorar a natureza humana e suas vulnerabilidades psicológicas para obter informações e acessos não autorizados a sistemas, ou então persuadir indivíduos a realizar ações indesejadas. Com base em princípios da Psicologia, a Engenharia Social utiliza uma variedade de estratégias de manipulação com a intenção de explorar aspectos humanos relacionados ao processo de tomada de decisão, bem como às vulnerabilidades nas interações humanas e caraterísticas culturais. Este minicurso tem como objetivo fornecer uma visão abrangente sobre a interseção entre a ES, a Psicologia e a Automatização Computacional, abordando as técnicas de manipulação empregadas. Serão abordados os aspectos psicológicos relacionados aos principais ataques de ES, bem como será observada a crescente preocupação com a Engenharia Social Automatizada . No decorrer do minicurso serão descritos os principais vieses cognitivos explorados nos ataques de Engenharia Social. Em seguida, serão abordados os fundamentos e técnicas aplicados no contexto da Engenharia Social Automatizada, considerando o avanço da Inteligência Artificial e do Aprendizado de Máquina e potenciais ataques em larga escala que utilizam ferramentas de automação. Ao final, espera-se que os participantes compreendam as técnicas de manipulação utilizadas na Engenharia Social, os vieses cognitivos associados e os desafios apresentados pela Engenharia Social Automatizada.

Abstract

Social Engineering (SE) is a discipline that aims to explore human nature and its psychological vulnerabilities to obtain unauthorized information and access to systems,

or therefore persuade individuals to perform unwanted actions. Based on principles of Psychology, ES uses a variety of manipulative strategies with the intention of exploring human aspects related to the decision-making process, such as vulnerabilities in human interactions and cultural characteristics. This mini-course has the objective of providing a comprehensive vision of the intersection between Social Engineering, Psychology and Computational Automation, addressing the manipulation techniques used. The psychological aspects related to the main SE attacks will be addressed, as will be observed the growing concern with Automated Social Engineering (ASE). In the course of the mini-course, the main cognitive processes explored in SE attacks will be described. Next, the fundamentals and techniques applied in the Automated Social Engineering context will be addressed, considering the advancement of Artificial Intelligence and Machine Learning and potential large-scale attacks that use automation tools. In the end, it is hoped that the participants understand the manipulation techniques used in Social Engineering, see the associated cognitive issues and the challenges presented by ASE.

3.1. Introdução

Ataques cibernéticos exploram as vulnerabilidades das estruturas de Tecnologia da Informação e Comunicação. As organizações têm empregado diversas soluções para enfrentar os ataques cibernéticos, tais como *Firewalls*, Sistema de Detecção de Intrusão (*Intrusion Detection System* - IDS), Antivírus, entre outros. No entanto, esses mecanismos de defesa muitas vezes não são suficientes para impedir as ações relacionadas com aspectos humanos no ambiente cibernético. Os ataques têm explorado a interação humana em conjunto com as brechas tecnológicas, enfraquecendo a cadeia de segurança [Salahdine and Kaabouch 2019] [Klimburg-Witjes and Wentland 2021]. As relações de confiança entre humanos no ambiente cibernético têm proporcionado um cenário para a prática de atos ilícitos. Dessa forma, alguns autores apontam que o fator humano é o elo mais fraco na cadeia de Segurança Cibernética [Mitnick and Simon 2003].

A Engenharia Social (ES) é uma disciplina que visa explorar a natureza humana e suas vulnerabilidades para auxiliar em ações como obter informações e acessos não autorizados a sistemas e persuadir indivíduos a realizarem ações indesejadas. Com base em princípios da Psicologia, a ES utiliza uma variedade de estratégias com a intenção de explorar aspectos humanos relacionados ao processo de tomada de decisão, bem como vulnerabilidades nas interações humanas e caraterísticas culturais. Os atacantes que utilizam a ES vêm diversificando os mecanismos para explorar as relações de confiança, tendo como objetivo ampliar o acesso a dados relevantes, assim como potenciais alvos. Neste contexto, a interconectividade das redes sociais e o crescimento da dimensão cognitiva do trabalho estão tornando os recursos humanos como um dos pilares da segurança [Culot et al. 2019] [Greitzer et al. 2019].

Uma das abordagens que pode ser utilizada para compreender aspectos da Es é o viés cognitivo. Este é um conceito oriundo da Psicologia Cognitiva e da Economia Comportamental que estuda as falhas comuns no pensamento humano, resultando em pensamentos distorcidos, imprecisos e incompletos, o que pode levar à tomada de decisões precipitadas e fracas. O viés cognitivo pode ser utilizado e explorado por um atacante para que o alvo do ataque seja induzido a algum erro, como a distorção de determinados julgamentos que o levem a uma tomada de decisão ruim (i.e., permitir um acesso inde-

vido). Informações sobre características psicológicas podem auxiliar em ataques que se utilizam de viés cognitivo. Tais informações podem ser coletadas, por exemplo, em redes sociais.

O crescente aumento do uso das redes sociais para estabelecer relacionamentos pessoais e profissionais abre possibilidades para as ações de ES [Shires 2018] [Klimburg-Witjes and Wentland 2021]. Assim, ao oferecerem serviços, as redes sociais coletam dados pessoais e corporativos formando bases de dados de alto valor, as quais podem ser utilizadas como ferramentas para ataques cibernéticos [Crossler and Bélanger 2014]. Muitas vezes essas bases podem ser empregadas em conjunto com mecanismos automatizados. Os ataques automatizados requerem pouca intervenção humana e podem simular o comportamento humano [Huber et al. 2009] [Shafahi et al. 2016].

Diversas tecnologias podem ser utilizadas para uma Engenharia Social Automatizada (ESA) [Huber et al. 2009]. Um exemplo de tais tecnologias são *bots*, softwares automatizados que são capazes de executar comandos de operação e controle sem a necessidade de participação humana. *Bots* podem ser utilizados para ações positivas, como por exemplo, ajudar o usuário a ter uma melhor Qualidade de Experiência (QoE). Contudo, *bots* têm sido utilizados como ferramenta para ataques de ESA. Como são escaláveis, essa ferramenta permite que um único atacante contate um grande número de potenciais vítimas simultaneamente, por exemplo, na busca de informações confidenciais [Huber et al. 2009][Dewangan and Kaushal 2016].

A ESA tem evoluindo, considerando o avanço da Inteligência Artificial e do Aprendizado de Máquina. Dessa forma, os humanos tendem a confiar nos *bots* [Dickerson et al. 2014], já que os mesmo tem a capacidade de se passar por seres humanos, imitando as atividades dos usuários reais [Shafahi et al. 2016]. Informações oriundas de redes sociais podem ser associadas a conceitos da Psicologias Cognitiva para para tornar os ataques que usam *bots* cada vez mais efetivos.

Na literatura, poucos trabalhos apresentam análises sobre a ESA com o uso de *Bots*. A maioria dos trabalhos estuda a área da Psicologia Social, com foco no comportamento humano diante das ações de ES [Huber et al. 2009]. Os autores [Al-Charchafchi et al. 2019] e [Piovesan et al. 2019] abordam as ameaças à segurança nas redes sociais, decorrentes dos ataques de ES utilizando contas falsas, roubo de identidade e *phishing*. No sentido de influenciar os usuários nas redes sociais, há trabalhos que avaliam as vulnerabilidades das redes sociais com o uso de *bots* para campanhas de convencimento nas redes. Os autores [Freitas et al. 2014] e [Messias et al. 2018] analisam o uso de *bots* no *Twitter* para influenciar os usuários e comprometer a estrutura da rede. Já [Huber et al. 2009] propõem a automação das tarefas de ES por meio de um *bot* no *Facebook*, concluindo que a persuasão é um recurso essencial no processo de ESA.

O presente capítulo apresentar os principais fundamentos de ES, assim como uma visão abrangente sobre sua interseção com a Psicologia e a Automatização Computacional. Além disso, fundamentos e técnicas aplicados no contexto da ESA são discutidos com mais profundidade, assim como questões relacionadas à utilização de redes sociais neste contexto. O objetivo é auxiliar na compreensão das técnicas utilizadas na ES, auxiliando na formação de profissionais com atuação na Segurança da Informação.

O presente capítulo está organizado da seguinte forma. Na Seção 3.2, serão apresentadas definições e conceitos básicos de Engenharia Social. Na Seção 3.3, serão apresentados os principais pontos de interseção entre a Psicologia, a Segurança da Informação e a Engenharia Social. Na Seção 3.4, será apresentada uma contextualização sobre a Engenharia Social Automatizada, bem como uma apresentação das ferramentas e técnicas utilizadas. Na Seção 3.5, serão apresentadas ações para prevenir e mitigar os ataques de Engenharia Social. Na Seção 3.6, serão apresentados estudos de casos relacionados com os temas descritos no capítulo. Na Seção 3.7, serão apresentadas reflexões sobre questões éticas implicadas na prática da Engenharia Social em contextos não maliciosos, como na pesquisa e na execução de testes de invasão. Finalmente, considerações finais e perspectivas de trabalhos futuros são apresentados na Seção 3.8.

3.2. Fundamentos de Engenharia Social

A interconectividade proporcionada pela Internet tem ampliado consideravelmente a superfície de ataques cibernéticos. A crescente dependência da tecnologia da informação (TI) por parte das empresas e organizações, juntamente com a expansão das redes e sistema conectados, resulta em uma maior exposição e ameaças digitais. Essa evolução tecnológica, no que pese tenha gerado benefícios, também abriu novas brechas de segurança que os cibercriminosos têm explorado de forma crescente [Benias and Markopoulos 2017].

Como consequência, a necessidade de uma abordagem abrangente e eficaz em relação à cibersegurança passou a ser uma prioridade para proteger a integridade dos dados, sistemas e infraestruturas das organizações em uma sociedade cada vez mais conectada digitalmente. A ES, aliada a esse cenário, é uma das estratégias utilizadas pelos atacantes para acessar informações confidenciais e comprometer a segurança das organizações [Reep-van den Bergh and Junger 2018]

3.2.1. Definição e Conceitos Básicos

A ES é definida como a arte de explorar as pessoas com o objetivo de obter acesso aos dados e informações de potenciais alvos dos sistemas de informação, independente de usar ou não a tecnologia. É uma técnica de ataque que explora o comportamento humano, por meio da persuasão e manipulação psicológica das pessoas. Na prática, o fator humano é o elo mais fraco da cadeia de cibersegurança. [Mitnick and Simon 2003] [Klimburg-Witjes and Wentland 2021].

A disponibilidade de diversos meios de comunicação de grande alcance gera um cenário propício para os ataques de ES. O avanço da tecnologia tem facilitado a automação e escalabilidade desses ataques, permitindo que os atacantes atingir um número maior de potenciais vítimas em um curto período de tempo [Pinheiro 2020].Compreender e proteger contra os ataques de ES passa, portanto, ser um requisito essencial para garantir a segurança dos dados e sistemas em ambientes conectados e dependentes da tecnologia.

O processo para proteger os dados e informações, visando assegurar a confidencialidade, integridade e disponibilidade está relacionado com a Segurança da Informação (SI). A relação entre ES e SI é reforçada pelo desenvolvimento contínuo da tecnologia tem permitido a automação e escalabilidade dos ataques de ES, tornando-os ainda mais desafiadores de combater [Beal 2005].

Considerando o contexto da ES no campo da SI emergem fatores que impactam o comportamento humano e podem tornar as pessoas mais suscetíveis a manipulação e persuasão por parte dos atacantes. Esses fatores abordam:

1. Conhecimento e conscientização: a falta de conhecimento e conscientização sobre as ameaças de ES e as táticas utilizadas pelos atacantes pode tornar as pessoas menos preparadas para identificar e evitar esses ataques.

2. Confiança: as pessoas tendem a confiar em outras pessoas, principalmente quando essas pessoas parecem ser amigáveis, prestativas ou apresentam uma autoridade aparente. Os atacantes aproveitam essa tendência para criar subterfúgios convincentes e ganhar a confiança de suas vítimas;

3. Curiosidade: a curiosidade humana pode ser explorada por meio de táticas como títulos sensacionalistas ou informações intrigantes, fazendo as pessoas a clicarem em *links* maliciosos ou a interagirem com conteúdos duvidosos, que podem representar riscos à SI;

4. Caos informacional: em um ambiente digital repleto de informações, que cria o caos informacional, as pessoas podem receber um volume grande de informações dificultando a identificação de tentativas de ataques;

5. Conexões de relacionamento: a criação de conexões pessoais ou profissionais pode ser explorada pelos atacantes para obter informações importantes. Eles podem pesquisar sobre as vítimas nas redes sociais para estabelecer conexões usando argumentos convincentes;

6. Emoções: reações à estímulos que, em um processo complexo, geram sentimentos utilizados para influenciar pessoas a revelarem informações ou clicarem em *links* maliciosos; e

7. Rotinas de comportamento: as pessoas podem agir de forma automática, seguindo rotinas estabelecidas sem questionar a legitimidade das ações diárias.

Conscientizar as pessoas sobre esses fatores e promover uma cultura de segurança é essencial para reduzir os riscos associados à ES e fortalecer a SI. Os programas de conscientização devem estar alinhados com as medidas técnicas e processuais relacionadas com a segurança nas organizações.

3.2.2. Importância da Engenharia Social na Segurança da Informação

A ES desempenha um papel fundamental na SI, utilizando a manipulação psicológica para obter acesso não autorizado aos dados, as informações e os sistemas computacionais. A SI é uma área do conhecimento dedicada a proteção dos ativos da informação contra acessos e mudanças não autorizados, falta de disponibilidade dos recursos digitais e quebra da autenticidade [Beal 2005]. Sendo assim, a SI considera os princípios da confidencialidade, integridade, disponibilidade e autenticidade, a saber [Shaabany and Anderl 2018]:

1. Confidencialidade: assegura que os dados, sistemas e as informações serão acessadas exclusivamente pelos usuários autorizadas;
2. Integridade: assegura que os dados e as informações não tenham sido modificadas ao longo do seu ciclo de transmissão entre os usuários com acesso autorizado;
3. Disponibilidade: assegura à capacidade dos dados, sistemas e informações estarem acessíveis e funcionando quando necessário, para os usuários autorizados; e
4. Autenticidade: assegura a origem, a identidade e integridade dos dados e informações, na busca de uma proteção para as transações eletrônicas, os sistemas e as informações em um ambiente digital.

O avanço tecnológico tem possibilitado medidas técnicas para incrementar a capacidade de proteção dos dados, informações e sistemas. No entanto, os atacantes fazem uso de ações de ES para burlar as técnicas de proteção. Essas ações tem como alvo o ser humano, considerado um elo fraco na cadeia de segurança. As vulnerabilidades dos elementos sociais e humanos para acesso aos sistemas e roubo de informações são utilizadas pelos engenheiros sociais [de Souza Pereira et al. 2022].

3.2.3. Tipos de Ataques de Engenharia Social

A Engenharia Social se caracteriza pela manipulação da vítima através de técnicas psicológicas, utilizando a tecnologia como recurso de suporte em diferentes níveis. Essa análise permite entender a ES a partir de duas perspectivas principais, que podemos nos referir como os aspectos psicológicos e os aspectos tecnológicos. Compreender as características e interconexões entre ambos é de vital importância para o entendimento real da ES e, consequentemente, como combater as ameaças da mesma.

O decreto-Lei que instituiu o Código Penal brasileiro em 1940 já incluía no Capítulo VI a tipificação criminal para golpes e fraudes, com destaque ao Artigo 171, que define o crime de Estelionato, descrito como a obtenção ilícita de vantagem sobre outro indivíduo através da indução ao erro, utilizando quaisquer meios fraudulentos para isso [Brasil 1940]. Portanto o engano e manipulação de outros indivíduos é anterior ao advento da internet e da ampliação do acesso à tecnologia.

Do ponto de vista da ES, podemos dizer que a tecnologia trouxe recursos que permitiram que o estelionatário tradicional pudesse aumentar suas chances de sucesso com menor necessidade de auto-exposição, diminuindo assim os seus riscos. Como exemplo, um dos golpes *online* mais popularmente conhecidos é o *e-mail* do Príncipe Nigeriano, cujo ápice ocorreu nos anos 80-90, se trata na verdade de uma variação de outro golpe datado do século 18. A fraude se baseia no pedido de auxílio para a realização de uma transação comercial, prometendo que a vítima será recompensada com uma porcentagem do valor envolvido. Apenas do ponto de vista desse caso, podemos observar algumas características do recurso tecnológico utilizado:

1. O uso de *e-mail* como canal de comunicação permite maior agilidade tanto na produção da mensagem quanto no recebimento da mesma por potenciais vítimas, comparado por exemplo ao envio por carta;

2. Enquanto a relação entre carta/mensagem e destinatário/vítima é de 1:1 por carta, o *e-mail* oferece uma relação de 1:n, oferecendo uma maior escalabilidade sem necessidade de esforço adicional proporcional por parte do atacante;

3. A busca por destinatários/vítimas ou mesmo o envio das mensagens em massa podem ser completamente automatizados, novamente gerando escalabilidade e menor necessidade de esforço.

A combinação das interações sociais e tecnológicas caracterizam portanto os ataques de ES. Os ataques normalmente seguem um estrutura de quatro fases, nomeadamente: i) obter informações sobre a vítima para realização da abordagem; ii) estabelecer uma relação de confiança entre o agressor e a vítima; iii) explorar a informação para o desenvolvimento de ações específicas; e iv) executar o ataque para atingir seu(s) objetivo(s) [Mitnick and Simon 2003] [Klimburg-Witjes and Wentland 2021].

Os ataques de ES objetivam comumente a obtenção de informações confidenciais, o acesso não autorizado a sistemas ou persuadir as pessoas a realizar ações indesejadas. Por conta dessas características, o ataque pode visar um objetivo final do atacante, ou servir como etapa para a realização de outros ataques, como a obtenção de informações sensíveis para embasar um ataque direcionado a um usuário privilegiado, ou o roubo de identidade da vítima para realização de transações comerciais fraudulentas.

Considerando diferentes linhas na literatura, os principais tipos de ataques de ES são os seguintes:

1. *Phishing*: é uma forma de ataque que utiliza o envio de mensagens falsas com um *link* de aparência legítima, por *e-mail*, para enganar pessoas a revelarem informações pessoais, como senhas e dados financeiros;

2. *Spear Phishing*: é uma forma mais direcionada de *phishing*, na qual os atacantes personalizam as mensagens de *e-mail* para um alvo específico, possibilitando um taxa de sucesso maior no ataque;

3. *Vishing*: é uma forma de ataque realizada pelos engenheiros sociais, utilizando o telefone para estabelecer uma relação de confiança com o usuário, na busca informações pessoais, corporativas e confidencias;

4. *Whaling*: é uma forma de ataque aos integrantes do alto escalão das organizações, utilizando um *spear phishing* para personalizar as mensagens de *e-mail*;

5. *Smishing*: é uma forma de ataque que utiliza mensagens de texto, como por exemplo SMS, para obter informações pessoais dos usuários alvo;

6. *Impersonation*: é uma forma de ataque, na qual o atacante busca estabelecer um relação de confiança para obter informações, utilizando *e-mails* e/ou mensagens.

7. *Pretexting*: os atacantes criam uma história fictícia para manipular as vítimas a compartilharem informações, geralmente por telefone.

8. *Quid Pro Quo*: os atacantes oferecem algo em troca das informações das vítimas, como suporte técnico falso em troca de senhas;

9. *Water Holing*: é um ataque que os engenheiros sociais buscam comprometer um site frequentado pelas vítimas esperadas, explorando a confiança nas fontes para disseminar *malware*;.

10. *Tailgating*: é uma técnica de ES usada em segurança física e cibernética. Essa abordagem envolve um indivíduo não autorizado aproveitando a entrada de um local seguro ou restrito ao seguir de perto um funcionário autorizado; e

11. *Baiting*: os atacantes oferecem um atrativo, como um *download* gratuito, para convencer as vítimas a realizar ações que comprometam a segurança.

3.3. Psicologia e Engenharia Social

A exploração de fatores humanos representa uma significativa parcela dos incidentes de segurança da informação. Desses incidentes, ataques de ES costumam ocupar papel de destaque nos relatórios de ameaças publicados anualmente e chegam a representar 98% de todos os crimes cibernéticos de phishing e de violação de dados [Martineau et al. 2023]. Tal representatividade pode ser explicada pela maior facilidade na execução dos ataques de ES, uma vez que não exigem o uso de ferramentas complexas ou conhecimentos técnicos prévios, bem como pelo seu potencial de ser bem sucedido.

Enquanto controles e tecnologias aplicadas à segurança da informação são aperfeiçoadas com o passar dos anos, tornando-se mais complexos e mais difíceis de serem comprometidos, aspectos psicológicos tornam-se uma estratégia vantajosa para os atacantes. Isto porque tendem a apresentar a mesma complexidade e são mais fáceis de presumir e de explorar. Descrevendo um contexto no qual o vetor de ataque assume, predominantemente, uma natureza psicológica, contrastando com a abordagem estritamente tecnológica, amplamente prevalente na área [Martineau et al. 2023].

3.3.1. Interface Psicologia e Segurança da Informação

Embora questões relacionadas aos fatores humanos em segurança da informação e segurança cibernética, como conscientização e comportamento, constituam elementos críticos mencionados em pesquisas, em diretrizes e em boas práticas [Robinson 2023] [Collier et al. 2023]. Dimensões culturais e comportamentais tem sido pouco enfatizadas nas abordagens de segurança da informação [Collier et al. 2023].

No contexto maior, segurança da informação tende a ser considerada uma disciplina essencialmente técnica, fortemente atrelada à tecnologia da informação. Essa característica é percebida também na formação dos profissionais, na qual observa-se uma lacuna acerca do estudo dos fatores humanos na área [Nobles 2023]. Promovendo, assim, uma perspectiva que negligencia o caráter estratégico e comportamental, e favorece uma abordagem focada na implementação de controles sobretudo tecnológicos e em políticas que, na maioria das vezes, não consideram elementos comportamentais, culturais e necessidades humanas [Collier et al. 2023].

A efetiva adoção e integração de controles e práticas de segurança da informação pelas pessoas é facilitada quando estes levam em conta o comportamento humano e as necessidades dos usuários. Regras e normas complexas ou insuficientemente justificadas afetam a capacidade de reflexão das pessoas e, frequentemente levam a uma predisposição a não segui-las e a buscar formas de contorna-las [Collier et al. 2023].

A segurança da informação é uma área que se beneficia das pesquisas em psicologia [Ancis 2020]. Reconhecer a influência que os fatores psicológicos exercem no âmbito da segurança da informação, bem como compreender os aspectos da cognição humana explorados nos ataques de ES [Montañez et al. 2020] favorece a elaboração e implementação de estratégias de segurança mais propensas de serem adotadas. Assim, torna-se imprescindível a inclusão ativa da disciplina psicológica, ampliando a perspectiva para além do humano como parte de um problema [Zimmermann and Renaud 2019]. Desta forma, segurança da informação deve igualmente ser vista a partir de uma disciplina comportamental [Martineau et al. 2023], contemplando a complexa interação entre seres humanos e tecnologia, e suas implicações.

Neste sentindo a segurança da informação também deve incorporar os avanços da ciberpsicologia. Disciplina que se dedica a compreender os processos psicológicos e os aspectos e características do comportamento humano na interação com a tecnologia [Attrill-Smith et al. 2019b] [Ancis 2020]. Essa disciplina surge a partir do caráter pervasivo da tecnologia na contemporaneidade. Tem como característica a inter e a transdisciplinaridade, incluí disciplinas como interação humano-computador, ciência da computação, engenharia e psicologia [Attrill-Smith et al. 2019b] [Ancis 2020]. Igualmente, sua aplicação é variada, contemplando áreas como saúde, educação, práticas em psicologia e também segurança [Ancis 2020] [Martineau et al. 2023].

Nessa integração, a visão tradicional de que há uma separação entre os ambientes virtuais e reais é desafiada. A perspectiva contemporânea nos leva a reconhecer que as fronteiras entre essas duas dimensões se tornaram fluidas, revelando a convergência entre as experiências *online* e *offline* sua intrincada relação. Tal como previsto no incio dos anos 2000 por [Castells 2002].

Desfazer a noção de que há distinção entre real e virtual, *online* e *offline*, e fundamentar-se em uma visão integrada das dimensões, permite reduzir a tendência de subestimar riscos e impactos das decisões e ações relacionadas as interações mediadas pela tecnologia. Nas próximas subseções ficará evidente que muitos dos fenômenos observados em ambientes *offline* estão presentes nos ambientes *online* ou mediados, destacando a importância da perspectiva integrada e sua relevância à segurança da informação.

3.3.2. Psicologia Aplicada na Engenharia Social

Apesar da direta relação com fatores humanos, a predominância de um enfoque tecnológico é observada também na compreensão e identificação de ataques de ES [Montañez et al. 2020]. Repetindo, assim, o tradicional modo de abordar segurança da informação e subestimando os aspectos psicológicos associados.

Beneficiando-se de vulnerabilidades humanas do processo de tomada de decisão, que influenciam comportamentos e impactam na motivação, a ES baseia-se, em grande

parte, na exploração da psicologia humana [Montañez et al. 2020]. O processo humano de tomada de decisão é complexo, envolve processos cognitivos como atenção e memória e é afetado por estados emocionais, bem como conhecimentos e experiências prévias.

Para facilitar a tomada de decisão ou a resolução de problemas, devido limitações relacionadas à informações disponíveis e à capacidade de processamento de informações, seres humanos podem recorrer as heurísticas[Korteling and Toet 2022]. Heurísticas são estratégias baseadas na experiência, que podem oferecer uma forma eficiente de encontrar uma solução, devido simplificações e desvios, mas que não garantem um resultado preciso [APA nd].

Quando levam a resultados abaixo do ideal ou incorretos, as heurísticas tornam-se vieses cognitivos, que podem ser facilmente manipulados e explorados. Os vieses cognitivos podem ser definidos como tendências e inclinações que enviesam ou distorcem o processamento de informações [Tversky and Kahneman 1974] [Korteling and Toet 2022]. São numerosos os vieses cognitivos conhecidos [Korteling and Toet 2022], a tabela 3.1 lista alguns dos principais vieses cognitivos que podem ser explorados ou influenciar o desfecho de ataques de ES.

No que tange aspectos psicológicos, para além dos vieses cognitivos, ataques de ES tem por característica estimular e explorar emoções como medo, excitação e surpresa. Estas emoções geram sentimentos que tendem a induzir respostas e ações rápidas, afetando a tomada de decisão e potencializando a probabilidade de sucesso de um ataque de ES.

Restrições de tempo, senso de urgência, intimidação, curiosidade, simpatia, aparente legitimidade, confiança, criação de conexão interpessoal e sobrecarga de informações também são empregados para aumentar a complexidade decisória, influenciar a tomada de decisão e levar aos vieses. Outros fatores cognitivos como carga de trabalho, estresse e vigilância, também se relacionam com ataques de engenharia social podem se relacionar com ataques de ES e influenciar o desfecho destes [Montañez et al. 2020]. Contextos situacionais e ambientes com altos níveis de tensão, tendem a influenciar e prejudicar a tomada de decisão das pessoas.

Abaixo são listados alguns exemplos de como os vieses podem ser utilizados por atacantes na aplicação da ES:

- Viés de ancoragem: dando ênfase inicialmente a uma informação que pode gerar senso de legitimidade, como a referência ao nome de algum individuo que ocupe cargo relevante na organização, o atacante pode gerar a impressão de que conhece e trabalha na organização, encorajando a tomada de decisão rápida;

Tabela 3.1: Principais vieses cognitivos - adaptado de [Korteling and Toet 2022], de [Wilke and Mata 2012] e de [APA nd].

Vies	Descrição
Viés de ancoragem	tendência a utilizar uma informação específica a qual foi exposto previamente na tomada de decisão.
Viés de autoridade	tendência a atribuir maior valor e confiabilidade à opinião de uma figura de autoridade (não relacionada ao seu conteúdo).
Viés de confirmação	tendência de selecionar, interpretar, focar e lembrar informações de uma forma que confirme as própria crenças, pontos de vista e/ou hipóteses, independentemente da veracidade.
Viés de conformidade	tendência de ajustar o pensamento e o comportamento de um indivíduo ao padrão de um grupo. Este viés está na base da dinâmica e do marketing dos influenciadores digitais.
Viés de crença	tendência de ser influenciado pelo conhecimento de alguém ao avaliar conclusões e aceitá-las como verdadeiras porque são críveis e não porque apresentam validade lógica.
Viés de disponibilidade	tendência de julgar a frequência, importância ou probabilidade de uma ocorrência pela facilidade com que exemplos imediatos vêm à mente, priorizando informações facilmente acessíveis.
Viés de enquadramento	tendência de basear decisões na forma como a informação é apresentada - conotações positivas ou negativas - e não nos fatos presentes.
Viés de escassez	tendência de atribuir maior valor subjetivo a itens mais raros, difíceis de obter ou de maior demanda.
Viés de normalidade	tendência de subestimar a probabilidade e as possíveis consequências de eventos e de acreditar que as coisas sempre funcionarão da forma como ocorrem normalmente.
Viés de otimismo	tendência de superestimar a probabilidade de eventos positivos e subestimar a probabilidade de eventos negativos.
Heurística de prioridade	tendência de tomar uma decisão baseada em apenas uma informação dominante.
Viés retrospectivo	uma distorção da memória pela qual, as pessoas tendem a perceberem eventos prévios como mais previsíveis do que foram ou como inevitáveis.
Efeito bandwagon	pode ser considerado uma forma do viés de conformidade, é a tendência em adotar crenças e comportamentos, principalmente porque já foram adotados por outras pessoas.
Reciprocidade	tendência de responder uma ação positiva com outra ação positiva e ter dificuldade em dizer não ou ficar "devendo"à outra pessoa.
Prova social	é a tendência de adaptar ou copiar ações e opiniões de outros, com o intuito de adotar comportamento considerado correto ou esperado em uma determinada situação.

- Viés de autoridade: o ataque é desenhado para simular autoridade, utilizando-se de informações, imagens e outros recursos que possam transmitir autoridade e despertar confiança. Ataques bem elaborados tendem a reunir diversos elementos na tentativa de não levantar suspeitas, incluindo o emprego de linguagem comum a área, como o uso de termos técnicos, o que pode ser facilitado pelo uso de Inteligência Artificial (IA);

- Viés de conformidade: com intuito de gerar um comportamento semelhante, o atacante informa ou da margem para que a pessoa entenda que diversas outras pessoas obtiveram benefícios ao realizar determinada ação ou resposta;

- Viés de disponibilidade: diante de uma solicitação de informação ou para executar uma ação, como desabilitar temporariamente o software de anti-vírus, o indivíduo tende a superestimar a informação de nunca ter experienciado um golpe e subestimar a probabilidade do caráter malicioso da solicitação;
- Viés de escassez: utilizando-se do senso de urgência, de restrições de tempo ou de recursos, ou da ideia de que é uma oportunidade única, atacantes tentam manipular o indivíduo para que realize uma ação;
- Reciprocidade: com a intenção de que a pessoa realize uma ação e ofereça algo em troca, o atacante inicialmente oferece alguma suposta vantagem, prêmio ou favor.

É importante destacar que os vieses cognitivos são intrínsecos à natureza humana e ataques de ES estão intimamente associados á influências situacionais e contextuais. No entanto, ter consciência acerca dos vieses e da forma como são utilizados na ES nos torna cientes da nossa própria vulnerabilidade e nos permite assumir que podemos ser manipulados, influenciados e moldados. Contribuindo para que possamos desenvolver senso crítico ao avaliar informações e interações *online*.

Aprimorando, assim, habilidades para reconhecer e questionar informações ou solicitações suspeitas, bem como facilitando a identificação de potenciais tentativas de manipulação. Além disso, do ponto de vista coletivo, melhora as competências de sensibilização de outras pessoas para os riscos, desempenhando um papel na promoção da cultura de segurança.

Ademais, o processo de tomada de decisão, assim como o comportamento, pode ser influenciado por experiências pregressas e por conhecimento prévio. Desta forma, saber sobre os vieses e como operam, pode antecipar percepções e facilitar a adoção de comportamentos mais cautelosos.

3.3.3. Mídias Sociais, Psicologia e Engenharia Social

Mídias sociais podem ser definidas como canais baseados na internet, com suporte a interações sociais síncronas e assíncronas, que permitem a transmissão de comunicações com públicos amplos e restritos [Bayer et al. 2020] [Carr and Hayes 2015]. A definição também incluí a presença de quatro elementos que caracterizam essa tecnologia [Bayer et al. 2020].

1. Perfil: elemento que permite aos usuários manter conjuntos exclusivos de atributos pessoais criados pelo próprio usuário, pelos usuários de sua rede e/ou pela plataforma [Bayer et al. 2020];
2. Rede: representa conexões sociais e pode ser compreendido como o conjunto de contatos criados por meio de "amizade"mútua ou do "seguir"unilateralmente [Bayer et al. 2020];
3. *Stream*: refere-se ao fluxo de conteúdo, é o elemento de mídia social que permite ao usuário consumir e/ou interagir com *feeds* de conteúdo gerado por outros usuários de sua rede [Bayer et al. 2020];

4. Mensagem: elemento que possibilita aos usuários o envolvimento em interação social direcionada usando texto, vídeo, foto ou qualquer outra mídia suportada pela plataforma utilizada [Bayer et al. 2020].

Em pesquisas psicológicas, uma abordagem a partir dos elementos fundamentais a essas mídias permite aos pesquisadores conceituar mídias sociais sem restringir-se em particularidades relacionadas às plataformas específicas e estabelecer uma base mais duradoura do que o tempo em que uma plataforma se mantém ativa, para assim observar seus efeitos e implicações [Bayer et al. 2020].

Além disso, esses elementos associam-se a efeitos [Bayer et al. 2020], que podem ser vistos como ações e práticas já observadas em ambientes *offline*, mas que também operam no ambiente *online*. No que tange ES, estes efeitos são relevantes porque são utilizados por atacantes, se correlacionam com alguns dos vieses explorados e podem contribuir significativamente para o êxito do ataque de ES.

Abaixo são listados e descritos três destes efeitos e o elemento relacionado, conforme a literatura [Bayer et al. 2020]. Também é realizada e exposta a associação com alguns dos vieses previamente apresentados.

- Autoapresentação: consiste no processo de controlar a percepção de outras pessoas a respeito de alguém [Leary 2019] [Attrill-Smith et al. 2019a], na tentativa de modificar uma resposta com objetivos sociais ou pessoais [Leary and Tangney 2014]. A autoapresentação é um fenômeno presente nas interações sociais, não se restringe as interações mediadas por tecnologias e é considerado também um processo de gerenciamento de impressão [Attrill-Smith et al. 2019a]. Inclui os aspectos deliberados da modificação, mas também aspectos menos conscientes, que podem estar vinculados a normas e expectativas ou componentes culturais, por exemplo. A autoapresentação, não tem necessariamente um caráter de falsidade. Em interações *offline*, indivíduos se comportam de formas diferentes em situações e ambientes variados, o mesmo ocorre nas interações mediadas por tecnologia [Attrill-Smith et al. 2019a].

  São diversas as razões pelas quais as pessoas utilizam a autoapresentação [Attrill-Smith et al. 2019a], diversas também são as motivações de uma pessoa para modificar ou esconder partes ou a totalidade de uma identidade [Leary and Tangney 2014]. Essas alterações podem ocorrer motivadas pela liberdade de expressar-se de uma maneira distinta, por exemplo, adotando uma versão menos introvertida de si [Attrill-Smith et al. 2019a] ou para atingir um objetivo [Leary and Tangney 2014], como adotar um modo de autoapresentação em uma rede profissional com intuito de conquistar uma vaga de emprego [Attrill-Smith et al. 2019a].

  Em mídias sociais é por meio do componente perfil, da sua criação e organização, que a autoapresentação se expressa de maneira direta [Bayer et al. 2020]. Na ES, a autoapresentação estaria motivada pela busca de um objetivo. E em contextos maliciosos, atacantes podem criar e organizar perfis falsos, construindo uma imagem que favoreça a forma como são vistos e gere credibilidade. Sendo capazes de simular e transmitir uma suposta autoridade, explorando assim o viés de autoridade.

- Conexões sociais: no âmbito das mídias sociais, conexão social pode ser compreendida como um efeito ligado as necessidades humanas de relacionamento com outros indivíduos [Ryan and Deci 2000] [Bayer et al. 2020]. Através da conexão social pessoas podem experimentar sentimentos de aceitação e isso favorece a sensação de vínculo. O estabelecimento de conexão e de um bom vínculo aumenta a probabilidade de êxito de ataques de ES.

  Em mídias sociais a conexão social se relaciona com diferentes elementos. Ainda que o elemento de rede seja o seu representante, a conexão social como efeito se manifesta por meio do elemento de comunicação direta, a mensagem.

  Através de contato direto, utilizando-se do recurso para envio de mensagem, o atacante estabelece comunicação com a vítima, se apresenta tanto pelo seu perfil como na comunicação direta e tenta formar um vínculo. Esse elemento pode influenciar o efeito da conexão social, bem como favorecer o viés de ancoragem. O indivíduo tende, então, a dar maior relevância as primeiras informações recebidas, assim como as primeiras percepções e basear nisso as suas decisões e ações subsequentes.

- Mobilização social: também fundamenta-se na necessidade de pertencimento e de se sentir socialmente conectado a outros indivíduos. Refere-se à princípios que podem ser usados para influenciar grupos de indivíduos a participar de determinadas atividades [Rogers et al. 2018]. A presença em mídias sociais como as redes sociais, potencializa e reforça os impactos da mobilização social [Rogers et al. 2018].

  Em nível individual, pesquisas sugerem que a transmissão de conteúdos com natureza de pedidos de ajuda, tendem a receber mais interações e respostas [Bayer et al. 2020]. A nível coletivo, esse fenômeno também pode ser observado nos eventos como a Primavera Árabe em 2011 [Gohn 2014] e os movimentos do Brasil em 2013 [Solano and Rocha 2019].

  Um atacante poderia explorar o efeito de mobilização promovido pelas mídias sociais para sensibilizar, a partir de apelos como pautas políticas e causas sociais, e induzir um individuo a executar determinada ação, como clicar em um link malicioso. Apoiado no elemento rede, com indivíduos reais ou composta por uma série de perfis falsos, o atacante poderia se valer da prova social. Encorajando, assim, um individuo ou grupo, a partir da mobilização de outros.

  Nesse exemplo, é possível identificar uma potencialização da probabilidade de êxito do ataque de ES. Uma vez que a mobilização social tende a influenciar grupos de pessoas para adoção de um comportamento ou execução de uma ação [Rogers et al. 2018].

A lista e a descrição dos efeitos e suas relações com os elementos de mídias sociais, obviamente não foram esgotadas nessa sessão. Considerando o escopo do presente capítulo, foi privilegiada uma apresentação parcial. A aplicação de alguns tópicos cobertos nessa sessão, pode ser observada de forma prática na sessão de estudos de caso.

Por fim, faz-se necessário destacar que plataformas de mídias sociais devem adotar medidas para proteger seus usuários. Implementando controles que dificultem a aplicação de ES e promovendo a conscientização e a sensibilização dos usuários por meio de campanhas e outros recursos informacionais.

3.4. Automação de Ataques de Engenharia Social

Os ataques de ES buscam estabelecer uma posição privilegiada do atacante no fluxo de informações, tendo como objetivo a construção de uma relação de confiança com a vítima, que é obtida por meio da manipulação psicológica. A automação da ES permite que esses ataques sejam realizados de forma mais eficiente e escalável, por meio de sistemas automatizados.

O uso de automação vem sendo uma tendência na área de tecnologia, permitindo que atividades normalmente repetitivas venham a ser executadas sem a necessidade de interferência humana, oferecendo diversas vantagens como a velocidade de execução das tarefas e a menor possibilidade de erros. O tempo normalmente investido na execução passa a ser aplicado no planejamento e definição das regras a serem seguidas, podendo seguir um modelo estrito de instruções passo a passo, como um *script*, ou com maior capacidade de entendimento e reação, como com o uso de inteligência artificial e aprendizado de máquina.

Da mesma forma que a evolução dessas tecnologias permite a melhoria e inovação, as mesmas podem ser também aplicadas a objetivos maliciosos. Da mesma forma que a internet e a computação trouxeram a possibilidade que golpes antes realizados pessoalmente ou através de cartas tivessem a capacidade de atingir um número maior de alvos, gerar mais falsos indícios que ajudem na credibilidade da isca e uma menor exposição do atacante, o uso de automação eleva essa capacidade a outros níveis.

Levando em conta os quatro estágios de um ataque de SE [Mitnick and Simon 2003]:

1. Obtenção de informações sobre a vítima;
2. Construção de uma relação de confiança;
3. Exploração das informações obtidas;
4. Execução do ataque;

O uso da automação permite maior agilidade na execução de tarefas associadas a cada fase, como por exemplo, retornar ao atacante as informações de interesse dos perfis de rede social do alvo. O ganho de tempo que permite maior eficiência de algum profissional traz a mesma vantagem ao agente malicioso.

O uso aprofundado dessas técnicas permite a rotulagem própria dessa forma de ataque, classificada como Engenharia Social Automatizada (ESA), onde o uso da automação se integra completamente à execução dos ataques, sendo peça fundamental para o seu ciclo, ou mesmo em casos extremos podendo rodar o ataque por inteiro sem necessidade de interferência humana, atingindo o máximo de escalabilidade com a menor exposição possível por parte do atacante.

3.4.1. Engenharia Social Automatizada

A Engenharia Social Automatizada (ESA) é uma abordagem que combina técnicas de ES com automação por meio de ferramentas e *scripts* para criar ataques eficazes em escala. Os ataques de ES demandam tempo e recursos para desenvolver uma relação de confiança

do atacante com o usuário. Sendo assim, ao automatizar os aspectos repetitivos e tediosos do processo, os atacantes utilizam a ESA para lançar ataques em larga escala de maneira mais eficiente [Guzman and Lewis 2020].

A comunicação humana tem sido a base para o desenvolvimento de interfaces homem-máquina os atacantes tem utilizado os *Bots* para automatizar os passos para estabelecer uma conexão de confiança com o usuário [Shafahi et al. 2016]. Essa relação de confiança faz uso da manipulação psicológica incentivando os usuários interagirem com ferramentas utilizadas pelos engenheiros sociais no ambiente digital. A combinação de táticas de manipulação psicológica com tecnologia avançada possibilita que atacantes atinjam múltiplos alvos com eficiência surpreendente.

Com o uso diário das redes sociais e o grande volume de dados no ciberespaço, os engenheiros sociais passaram a espalhar *Bots* com comportamento semelhante ao do ser humano para um grande número de usuários. Esses *Bots* simulam conversas humanas, conhecidos como *ChatBots* e, os que atuam nas redes sociais, os *SocialBots* [Shafahi et al. 2016].

Bot é uma ferramenta automatizada para implementar uma série de funções préprogramadas de operação e controle.. Os *Bots* podem ser autênticos para realizar tarefas úteis para os usuários. No entanto, existem *Bots* com foco malicioso, que podem realizar ataques para obter informações relevantes ou manter o controle do dispositivo acessado. *Bots* podem ser utilizados para ações de disseminação de informações falsas (*fake news*), *spam* e *phishing* [Mitnick and Simon 2003]. [Freitas et al. 2015].

SocialBot é uma ferramenta de *software* que simula o comportamento humano para realizar interações automatizadas nas redes sociais . Os *SocialBots* têm a capacidade de comprometer a estrutura das redes sociais, influenciando os usuários e aumentando o número de seguidores, para inflar os índices de popularidade de uma determinada conta de perfil. Essa ferramenta é eficaz para ataques de ES, utilizando-se de informações sensíveis de possíveis vítimas, como o roubo de identidade [Rouse 2013] [Camisani-Calzolari 2012] [Dewangan and Kaushal 2016].

ChatBot é a integração de sistemas, ferramentas e roteiros que promovem conversas por mensagens instantâneas com ou sem a participação de humanos. São desenvolvidos para ajudar usuários humanos em situações de serviços específicos, não sendo exaustivo. Por exemplo: atendimento ao cliente, atendimento por telefone e serviço de educação digital [Grimme et al. 2017]. O uso da linguagem natural nos *ChatBots* é um desafio a ser superado para o desenvolvimento dessa ferramenta [Khan and Das 2018] [Stoeckli et al. 2018].

3.4.2. Ferramentas e Técnicas Utilizadas na Engenharia Social Automatizada

À medida que o avanço tecnológico gera um crescimento exponencial do volume de dados no ciberespaço, tem-se como resultado uma dependência cada vez maior dos recursos tecnológicos. Por decorrência disso, a superfície de ataque para os engenheiros sociais tem aumentado, considerando o uso da automação das ferramentas e técnicas no campo da ES.

Ao explorar o comportamento humano e o uso de sistemas de informações, os

atacantes se beneficiam de uma compreensão profunda dos processos cognitivos, a fim de obter dados e informações relevantes de potenciais alvos. Utilizando ferramentas e técnicas de automação, os engenheiros sociais buscam estabelecer relações de confiança, manipulando psicologicamente as pessoas para que realizem ações específicas.

Algumas das principais ferramentas e técnicas utilizadas nessa abordagem são as seguintes:

1. Perfis falsos: é uma técnica amplamente adotada na Engenharia Social Automatizada é a criação de perfis falsos em plataformas de redes sociais. Esses perfis fictícios são meticulosamente construídos com o objetivo de estabelecer relações de confiança com possíveis alvos, explorando suas fraquezas e vulnerabilidades. Essa abordagem permite aos engenheiros sociais automatizar a criação e a manutenção de múltiplos perfis falsos, ampliando significativamente o alcance de seus ataques.

2. Análise e Mineração de Dados: a ESA depende de uma análise e obtenção eficaz de dados sobre potenciais vítimas. Nesse sentido, técnicas de mineração de dados, análise de redes sociais e obtenção de informações pessoais publicamente disponíveis são aplicadas. Essas técnicas permitem identificar alvos em potencial, compreender suas preferências, hábitos e padrões de comportamento, aumentando assim a eficácia dos ataques.

3. Manipulação Psicológica e Persuasão Automatizada: explorando as fragilidades do ser humano a ESA também abarca o uso de técnicas de manipulação psicológica e persuasão automatizada. Utilizando algoritmos e Inteligência Artificial (IA), é possível personalizar mensagens e interações para se adequarem ao perfil de cada potencial vítima. Essas técnicas visam explorar os mecanismos cognitivos e emocionais dos usuários, persuadindo esses usuários a realizar ações desejadas pelos engenheiros sociais.

4. *Bot* é o termo resumido da palavra da língua inglesa *robot*, que na tradução livre significa robô. É uma ferramenta automatizada que realiza uma série de funções pré-programadas de operação e controle. Os *Bots* podem ser autênticos, que têm como objetivo realizar atividades úteis para os usuários, por outro lado também existem *Bots* de cunho malicioso, que podem realizar ataques para obter informações relevantes ou manter o controle do dispositivo acessado. *Bots* podem ser utilizados para ações de disseminação de informações falsas (*fake news*), *spam* e *phishing* [Freitas et al. 2015].

   No contexto de ESA os cibercriminosos usam os *Bots* maliciosos para simular o comportamento humano, burlando os mecanismos de segurança. Com o crescimento das redes sociais e o grande volume de dados no ciberespaço, os engenheiros sociais passaram a espalhar *Bots* com comportamento semelhante ao do ser humano para um grande número de usuários. Esses *Bots* simulam conversas humanas, conhecidos como *ChatBots* e, os que atuam nas redes sociais, os *SocialBots* [Shafahi et al. 2016].

5. *ChatBot* é a integração de sistemas, ferramentas e roteiros que promovem conversas por mensagens instantâneas com ou sem a participação de humanos [Stoeckli et al. 2018]. São desenvolvidos para ajudar usuários humanos em situações de serviços específicos, não sendo exaustivo. Por exemplo: atendimento ao cliente, atendimento por telefone e serviço de educação digital [Grimme et al. 2017]. O uso da linguagem natural nos *ChatBots* é um desafio a ser superado para o desenvolvimento dessa ferramenta [Khan and Das 2018].

6. *SocialBot* é uma ferramenta de *software* que simula o comportamento humano para realizar interações automatizadas nas redes sociais [Rouse 2013]. Os *SocialBots* têm a capacidade de comprometer a estrutura das redes sociais, influenciando os usuários e aumentando o número de seguidores, para inflar os índices de popularidade de uma determinada conta de perfil [Camisani-Calzolari 2012]. Essa ferramenta é eficaz para ataques de ES, utilizando-se de informações sensíveis de possíveis vítimas, como o roubo de identidade [Dewangan and Kaushal 2016].

Essas ferramentas e técnicas têm sido desenvolvidas com a ajuda de mecanismos de IA que interagem com os usuários [Freitas et al. 2015]. A IA é similar a inteligência humana, desenvolvida com a automatização conforme a necessidade da aplicação [Ferrara et al. 2016]. Na medida que um certo grau de inteligência é incorporado nas ferramentas para simular o comportamento humano, aumenta a capacidade e escalabilidade dos ataques.

3.5. Prevenção e Mitigação de Ataques de Engenharia Social

A prevenção e mitigação de ataques de ES têm sido objeto de estudo e preocupação nas empresas, setor público e acadêmica. Esses ataques exploram a manipulação psicológica e a falta de conscientização das pessoas para obter acesso não autorizado a informações sensíveis ou comprometer sistemas de segurança. Abordagens acadêmicas para combater esse problema incluem a análise de técnicas e táticas empregadas pelos invasores, o desenvolvimento de métodos de detecção e alerta precoce, bem como a conscientização e treinamento dos usuários para identificar e evitar armadilhas de ES.

Como já discutido, a ES se baseia em dois principais pilares, os aspectos tecnológicos e as questões psicológicas humanas. A fim de obter resultados mais concretos na prevenção desses ataques é portanto necessário uma visão completa do ciclo de ataque e controles em ambos os pilares, aumentando não apenas as chances de evitar essas formas de ataque mas também que em caso de falhas em um ativo, que o ataque possa ser detectado ou impedido em outros níveis.

Enquanto a parte humana é normalmente abordada com o uso de treinamento e conscientização, auxiliando os usuários a identificarem e reportarem ações suspeitas, os controles técnicos atuam para evitar que os ataques cheguem nos usuários ou, em caso de falha das vítimas, minimizar os danos. Engenharia Social é muitas vezes utilizada apenas como forma de entrada para execução de outros ataques, portanto não basta que os riscos de ES sejam analisados individualmente, mas devem fazer parte de um programa maior de Segurança da Informação.

3.5.1. Conscientização e Treinamento dos Usuários

A conscientização e o treinamento dos usuários são elementos essenciais para na prevenção de ataques de ES, considerando que a manipulação psicológica é uma característica intrínseca desses ataques, que tem foco nas emoções humanas como o medo, a curiosidade e a confiança. A conscientização busca capacitar os usuários para reconhecerem as técnicas utilizadas pelos engenheiros sociais e, com isso, o conhecimento oferecido aos usuários permite uma ação defensiva em relação as tentativas de ataques. Um programa de conscientização e treinamento deve incluir:

1. Educação sobre táticas de ataque: os usuários devem ser informados sobre as táticas comuns de ES, como por exemplo: *phishing* e *spear phishing*. O entendimento como essas táticas operam ajuda os usuários a identificar sinais de alerta;

2. Simulações de ataque: simulações controladas de ataques de ES podem ser realizadas para testar a prontidão dos usuários. Com isso, é possível identificar as vulnerabilidades do ambiente computacional, bem como as oportunidades de aprendizado;

3. Desenvolvimento de habilidades críticas: os usuários devem ser capacitados para tomar decisões informadas sobre a divulgação de informações ou o clique em *links*, possibilitando avaliar a legitimidade das demandas e verificar a fonte;

4. Compartilhamento de exemplos reais: estudo de caso reais de ataques bem-sucedidos podem ilustrar as consequências da ES, tornando mais tangíveis os riscos envolvidos;

5. Atualizações regulares: as táticas de ataque são dinâmicas, é importante manter os usuários atualizados sobre as novas ameaças e métodos de defesa; e

6. Cultura de segurança: promover uma cultura de segurança onde os funcionários se sintam encorajados a relatar tentativas de Engenharia Social e outras formas de ataque ou suspeitas ajuda a criar uma abordagem coletiva para a prevenção.

Um importante fator a ser considerado nas ações de conscientização e treinamento é o da aplicação de uma abordagem social, orientada ao coletivo [Collier et al. 2023]. Tal orientação fomenta que os usuários se percebam como integrantes de cenário maior e cujo foco não seja o indivíduo [Zimmermann and Renaud 2019]. Essa perspectiva evita responsabilização excessiva e individualizada, além de promover comportamento pró segurança.

Abordagens individualizantes tendem a se basear no medo, desconsideram comportamento e motivações dos seres humanos e não contribuem para mudanças comportamentais efetivas [Collier et al. 2023]. Além disso não promovem o desenvolvimento de uma cultura de segurança, em contrapartida tendem a prejudicar o contexto e o senso crítico dos indivíduos e interferir nos processos de tomada de decisão, uma vez que elevam preocupações e tensões.

3.5.2. Técnicas de Detecção de Ataques de Engenharia Social Automatizada

Detectar ataques de ES realizados de maneira automatizada é uma abordagem complementar à conscientização e treinamento dos usuários. Esses ataques são conduzidos em grande escala, dificultando a identificação manual. Técnicas de detecção automatizada podem incluir:

1. Análise de Conteúdo: O uso de algoritmos para analisar e identificar padrões em mensagens de phishing ou em links maliciosos pode ajudar a identificar tentativas de Engenharia Social.
2. Análise Comportamental: observar o comportamento do usuário, como cliques rápidos e atípicos ou solicitações incomuns de informações, pode sinalizar atividades suspeitas.
3. Aprendizado de Máquina e Inteligência Artificial: os algoritmos de aprendizado de máquina podem ser treinados para reconhecer padrões de ataques de Engenharia Social com base em dados históricos.
4. Monitoramento de Tráfego: Observar o tráfego de rede em busca de atividades suspeitas, como tentativas de redirecionamento, pode ajudar a identificar ataques.
5. Análise de Metadados: Examinar metadados em emails e links pode revelar informações ocultas, como a verdadeira origem de uma mensagem.
6. Lista de Domínios Maliciosos: Manter uma lista atualizada de domínios conhecidos por hospedar ataques de Engenharia Social pode ser usado para bloqueio preventivo.

3.6. Estudos de Caso

Engenharia Social é definida como a técnica de explorar pessoas com o objetivo de acessar informações sobre potenciais alvos utilizando combinações de interações tecnológicas e sociais [Libicki 2018] [Klimburg-Witjes and Wentland 2021]. Já é um conceito comum a classificação dos usuários/humanos como o "elo fraco"da corrente em Segurança da Informação, visto que ataques explorando as suas falhas tem tido maiores taxas de sucesso e, muitas vezes, necessitando menores habilidades técnicas ou exposição à risco por parte do atacante [Darwish et al. 2012].

3.6.1. Estudo de Caso 1 - LinkedIn

As redes sociais possuem o objetivo de permitir interação virtual entre humanos. Porém além de oferecer um espaço onde as distâncias físicas possam ser ultrapassadas para permitir essas conexões, as redes também trazem para o ambiente virtual muitos dos riscos e ameaças existentes no mundo real. Uma diferença pertinente porém está no fato dos usuários desses espaços de convivência não terem o mesmo nível de conscientização, alerta ou capacidade de reconhecer riscos que teriam no mundo físico, o que aliado a maior capacidade de anonimização e personificação torna esses ambientes um espaço de grande interesse para Engenharia Social [Crossler and Bélanger 2014].

Comparando com outras redes sociais como *Facebook*, *Twitter* e *Instagram*, redes sociais profissionais replicam um ambiente corporativo, focado em conexões profissionais e crescimento de carreira. Esses cenários ligados ao mercado de trabalho criam um sentimento de seriedade, confiança e credibilidade, não tendo um foco em entretenimento ou distração. Esses ambientes atraem recrutadores em busca de candidatos, assim como empresas em busca de potenciais clientes. As relações existentes em redes sociais profissionais já são exploradas por engenheiros sociais, em especial a personificação de recrutadores, utilizando vagas de trabalho atrativas como isca para roubar informações confidenciais de empresas ou dados pessoais das vítimas. [1]

Atualmente o *LinkedIn* é a rede social profissional mais popular, com mais de 930 milhões de usuários em mais de 200 países de acordo com dados da própria rede (Maio de 2023)[2]. As Políticas de Uso do LinkedIn definem na Seção 8.2[3] as ações que não são permitidas aos usuários, destacando-se a proibição do uso de informações falsas ou personificação no perfil, e o uso de *Bots* e automação para realizar ações na plataforma.

Levando em conta as referências de ataques de ES já existentes no próprio LinkedIn, é possível encontrar referências ao uso de perfis e informações falsas para os mais diversos motivos. Analisando especificamente sobre automação, realizando uma busca em sites de repositórios de código é possível encontrar facilmente dezenas de *Bots* e *Scripts* especificamente para uso no *LinkedIn*. Se as políticas da rede social proíbem o uso de dados falsos e automação, porém sendo possível identificar o uso dos mesmos, há indicação de uma potencial falta ou insuficiência na implementação desses controles por parte da plataforma, ou a aplicação das regras é feita apenas baseando-se em reclamações feitas por outros usuários. Além dessas observações, o estudo de caso analisado realizou uma avaliação do capacidade do LinkedIn em detectar e/ou bloquear o uso de automação e informações falsas, sendo estes os requisitos básicos para executar um ataque de ESA contra os usuários da plataforma.

3.6.1.1. Metodologia

O teste foi realizado em um cenário de prova de conceito com 2 *Bots* para avaliar o ataque. O primeiro interagia com a rede social para buscar e contatar alvos - o *Bot* Plataforma. O segundo é um serviço de *ChatBot* que executaria uma entrevista de emprego com as vítimas - o *Bot* Recrutador. Também para dar suporte à execução das ações foi criado um perfil falso no LinkedIn, personificando um recrutador.

Para o *Bot* Plataforma foi desenvolvido um código *Python* para conexão com o LinkedIn. O LinkedIn também oferece uma API[4] para interações via *software*. Entendendo porém que usuários regulares não navegam em uma rede social através de uma API, utilizar esse canal para os testes potencialmente afetaria os resultados buscados. Para melhor reproduzir a mesma interação que um humano, foi utilizada a biblioteca Selenium[5],

[1] https://www.ft.com/content/a8d262f4-5d52-4464-8714-e21a457aab33
[2] https://about.linkedin.com
[3] https://www.linkedin.com/legal/user-agreement#dos
[4] https://developer.linkedin.com/product-catalog
[5] https://www.selenium.dev/

para permitir que o *bot* se comunicasse em formato padrão via navegador.

Para o *Bot* Recrutador, existiam diversas opções disponíveis que poderiam executar as ações necessárias sem que fosse preciso desenvolver uma nova aplicação. Usando um grupo pré-definido de questões padrão de entrevistas de emprego, junto a informações coletadas dos perfis das vítimas no *LinkedIn*, o *Bot* Recrutador basicamente conduz uma falsa emprevista de emprego com a vítima, tendo como objetivo porém obter informações sensíveis. Sendo possível executar tanto questões mais genéricas/padronizadas feitas por times de RH, quanto questões mais técnicas sobre as experiências profissionais atuais e anteriores, o processo inteiro foi executado utilizando o mesmo *bot*. Conforme o objetivo do atacante o *bot* pode incluir questões sobre empresas ou experiências específicas da vítima - coletadas do seu próprio perfil na rede social - buscando obter informações sensíveis sobre projetos, clientes, etc. Também, ao final da entrevista, o atacante poderia "selecionar"a vítima para a vaga, roubando dados pessoais através da assinatura de um falso contrato de trabalho e solicitando documentos como o passaporte, por exemplo, permitindo a realização de ataques posteriores de roubo de identidade.

De forma similar a estrutura de quatro estágios de um ataque de ES [Mitnick and Simon 2003], a proposta do projeto em questão segue um formato de etapas similar: i) Autenticação, ii) Busca, iii) Abordagem e iv) Entrevista. Esse formato permitiu a quebra do ataque em estágios e a verificação individual de cada etapa. A Figure 3.1 indica as fases do ataque testadas por cada um dos *Bots* de prova de conceito propostos.

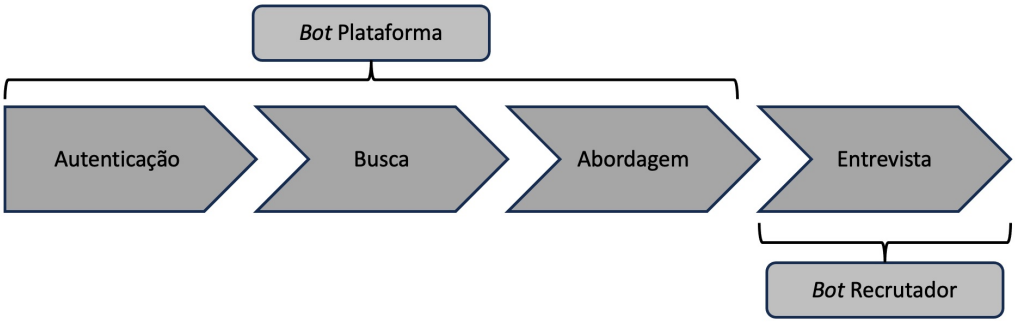


Figura 3.1: Relação entre os *Bots* propostos e as fases de ataque.

1. Autenticação: Como ilustrado na Figura 3.2, o objetivo dessa etapa é verificar se a rede social detecta ou tem diferentes comportamentos quando a autenticação do usuário é realizado através de automação. Para essa avaliação, o *Bot* Plataforma abre a página do *LinkedIn* no navegador, mapeia o código-fonte da página principal para identificar os campos das credenciais, preenche os mesmos com os valores recebidos e então submete as credenciais ao servidor e conclui o processo de autenticação, acessando a página principal do usuário.

2. Busca: Esta etapa busca verificar a detecção de uma busca automatizada de perfis. Similar a primeira etapa, o *Bot* Plataforma mapeia o código-fonte da página, identifica o campo de busca, realiza a busca utilizando os termos recebidos e então armazena temporariamente os perfis retornados. A Figura 3.7 também se refere a essa etapa.

3. Abordagem: O objetivo dessa etapa é iniciar a interação com os perfis de

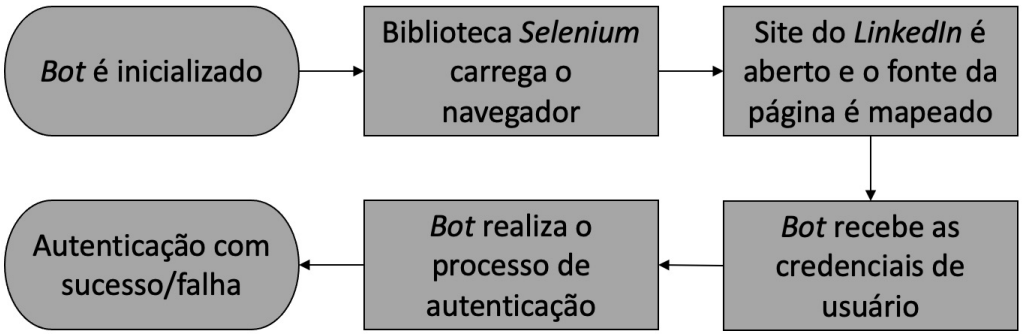


Figura 3.2: Fase de Autenticação.

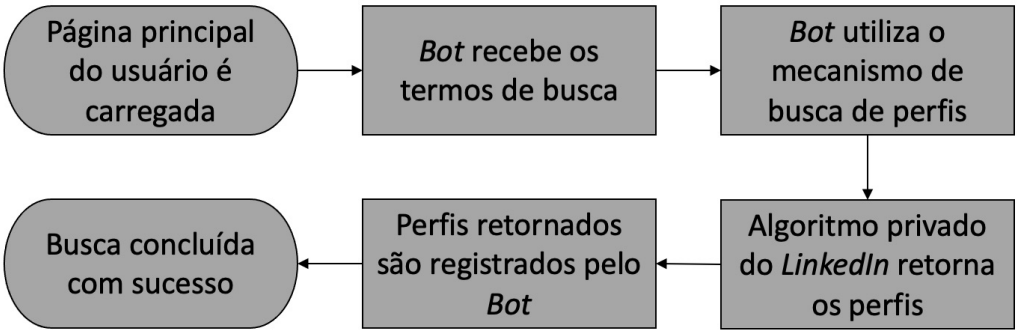


Figura 3.3: Fase de Busca.

potenciais vítimas coletados na etapa anterior. Como visto na Figura 3.4, usando os perfis capturados, o *Bot* Plataforma adiciona as vítimas como contatos e envia uma mensagem customizada, que serve como isca para as interações. Essas ações são realizadas com todos os perfis capturados.

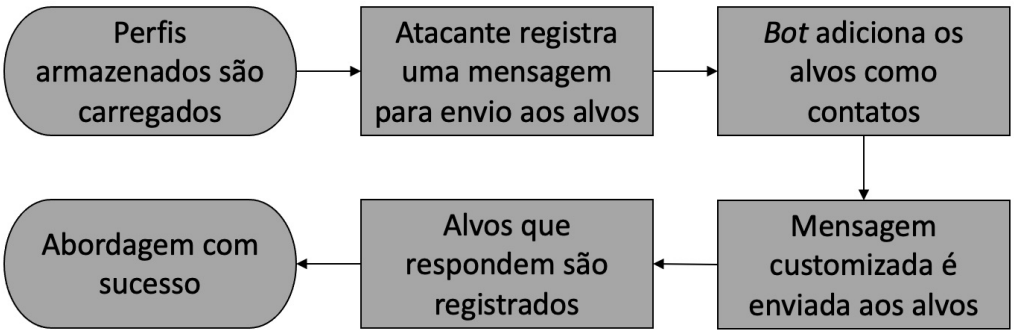


Figura 3.4: Fase de Abordagem.

4. Entrevista: Baseado nos resultados da abordagem realizada na etapa 3, um *script* captura as informações do perfil da vítima no *LinkedIn* para alimentar o banco de dados do *Bot* Recrutador, o qual terá então informações suficientes para executar uma entrevista de emprego com a vítima. A Figura 3.5 também ilustra o ciclo completo desta etapa. Como em geral é comum que um recrutador real aborde indivíduos através do *Linkedin* e então realize a entrevista e outra etapas em diferentes canais de comunicação, é esperado que a isca inclua um convite para realizar uma entrevista em um canal diferente

do próprio sistema de mensagens do *LinkedIn*.

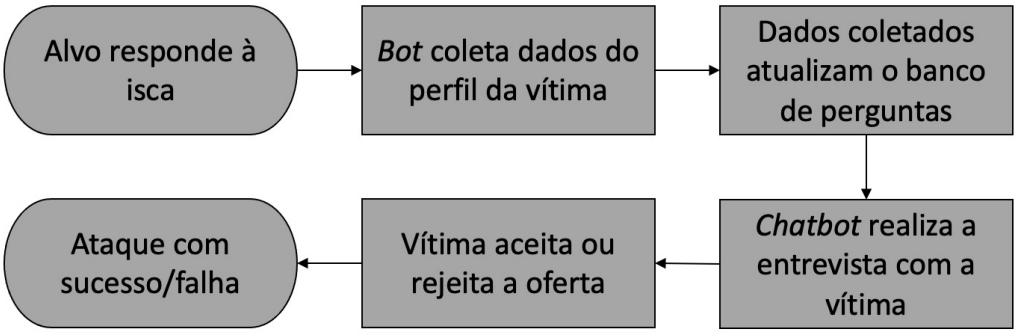


Figura 3.5: Fase de Entrevista.

3.6.1.2. Limitações

A Engenharia Social nasce no campo da psicologia, pois embora utilize da tecnologia como suporte, para atingir seus principais objetivos os atacantes exploram fraquezas humanas e características comportamentais, tópicos de pesquisa das ciências sociais. Para um completo entendimento do impacto de um ataque de ES, seria necessário não apenas validar os aspectos técnicos envolvidos, mas também enganar indivíduos e observar seu comportamento e ações. O campo da psicologia social, especialmente por conta dos diversos cenários envolvendo pesquisas com seres humanos, vem enfrentando desafios e discutindo os limites de ética em pesquisa há um longo tempo. A história nos traz exemplos extremos como os famosos experimentos de Milgran nos anos 70 [Riecken 1974], resultando em grande impacto e trauma nos participantes, situações que a pesquisa moderna entende como anti-éticas.

Expôr pessoas a situações onde elas devem ser enganadas, ter suas vulnerabilidades exploradas sem o seu consentimento (ou ao menos seu completo entendimento da situação) viola diversos dilemas éticos. Como resultado, posteriormente pode-se criar frustração e estresse por conta das expectativas criadas, promessas quebradas ou o sentimento de ter sido enganado ou manipulado. Entender e respeitar esses limites foi um dos guias desse estudo, e mesmo que isso não seja um tópico técnico não é possível executar uma pesquisa no campo de ES ou quaisquer outros tipos de ataques em cibersegurança sem discutir ou levantar questões sobre ética em pesquisa.

O primeiro desafio surgiu em como validar a proposta de ataque respeitando as políticas éticas. Como forma de atingir esses objetivos, foi decidido quebrar o ciclo de ataque em diferentes etapas e testá-las individulamente. Os resultados permitiriam um entendimento mínimo da resposta da aplicação, e cruzando os dados entre as diferentes fases permitiria ter conclusões quanto ao potencial de um ataque completo.

Especialmente a fase de Abordagem foi testada em uma linha tênue entre manter as premissas e violar barreiras éticas. Como seria necessário enviar requisições para usuários reais da plataforma, foi decidido limitar a menor quantidade possível de usuários recebendo a requisição e a mensagem. Essa quantidade foi demarcada pelos perfis

retornados na primeira página de resultados (normalmente entre 15 e 21 perfis), sendo que abordar todos eles simultaneamente ou numa janela de tempo bastante curta já indicaria um mínimo de uso de automação ou envio ativo de *SPAM* ou similares. Como o objetivo não era medir a resposta dos usuários à isca, e entendendo os requisitos para ter pessoas participando da pesquisa, após o envio das requisições e mensagens e validando que não houveram bloqueios ou similares ocorrendo na plataforma, todas as interações eram imediatamente canceladas/excluídas, a fim de evitar a chance que fossem vistas ou respondidas por usuários reais.

Entender a diferença entre quantas requisições por segundo um usuário testando a plataforma poderia fazer, comparado a um usuário apenas navegando de forma normal na rede permite identificar comportamentos que caracterizem automação sem a necessidade de identificar o limite máximo da aplicação ou gerar negação de serviço. Mesmo que potencialmente algum tipo de controle possa ser ativado após centenas ou milhares de requisições serem feitas, isso indicaria muito mais um controle contra negação de serviço ou controle de tráfego do que uma proteção contra automação. Portanto números altos não necessariamente indicam comportamento automatizado.

Para a fase de Entrevista, a avaliação focou na principal funcionalidade do *chatbot* de recrutamento: uma entrevista de emprego. A única diferença entre uma entrevista real e uma maliciosa são os objetivos, pois ao invés de tentar avaliar a capacidade e habilidades de um indivíduo para um certo cargo, o foco do atacante seria na obtenção de dados através de perguntas ou pela assinatura de um contrato de trabalho e apresentação de documentos, numa contratação falsa. O uso de um *chatbot* recrutador já existente permitiu apenas coletar informações sobre uma vítima para alimentar o *bot* e observar se as questões maliciosas propostas eram corretamente distribuídas na entrevista, sem necessidade de validar o *chatbot* em si e sua capacidade.

Considerando todos os mecanismos implementados e as decisões em como testar cada fase, seria possível concluir que suficientes resultados poderiam ser obtidos para validar o potencial do ataque proposta sem necessidade de violar nenhum requisito ético. Essa precaução é um tópico vital para qualquer tipo de pesquisa similar, e uma análise mais profunda do impacto de pesquisa ética, em especial no campo da ES, é um assunto importante para trabalhos futuros.

3.6.1.3. Experimentos

A primeira etapa foi a criação de um perfil falso que deu suporte a realização do ataque. Foi utilizada uma imagem de um banco de imagens gratuito da internet e dados aleatórios para simular experiência de trabalho prévias e formação acadêmica. Foi possível associar o perfil à empresas e universidades reais sem quaisquer verificações ou checagens necessárias. Não foi identificado nenhum impacto por conta do uso de dados falsos, ou pelo fato que desde a criação do perfil todas as interações com a plataforma do *LinkedIn* foram realizadas utilizando alguma forma de automação. A Figura 3.6 mostra algumas informações destacadas do perfil falso criado.

Os experimentos de simulação seguiram as etapas propostas no fluxo de ataque. O *Bot* Plataforma foi executado em uma máquina Windows rodando *Python*, a biblioteca

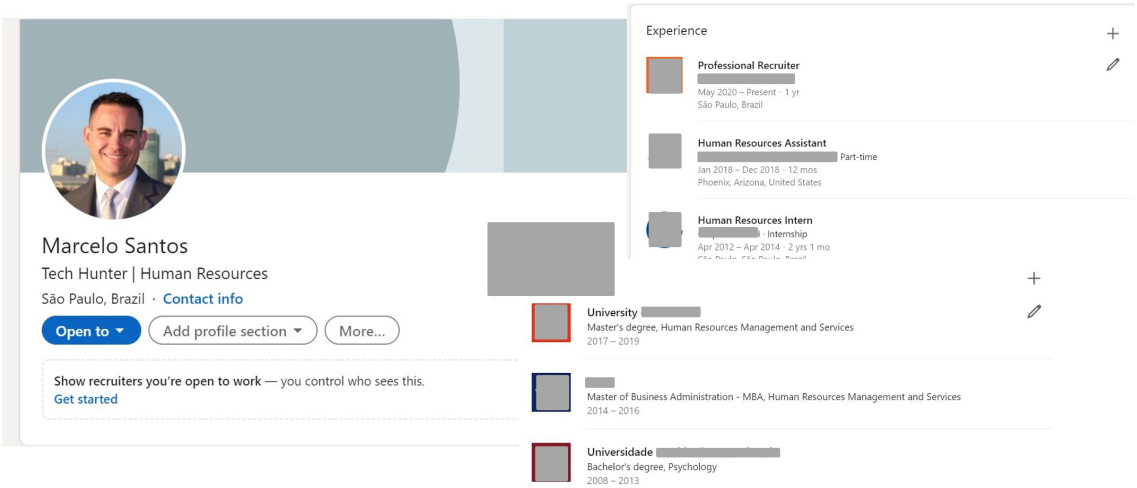


Figura 3.6: Perfil falso de recrutador criado nos testes.

Selenium e o navegador *Google Chrome*. Para o *Bot* Recrutador, foi utilizado a plataforma *SAP Conversational AI*.

Teste Etapa 1 - Autenticação:

O critério de sucesso desta etapa era executar a autenticação na plataforma seguindo diferentes comportamentos e observando se alguma das simulações ativaria controles ou bloqueios na aplicação por conta das características da automação. Como critério comparativo, foram definidos três padrões básicos de comportamento para os testes: (1) Realizar o processo de *logon* 10 vezes simultâneas; (2) Realizar o processo de *logon* 10 vezes com cinco segundos de intervalo entre cada tentativa; e (3) Realizar o processo de *logon* 10 vezes com dez segundos de intervalo entre cada tentativa. Esses padrões tentavam replicar comportamentos não esperados de um usuário humano quanto a quantidade ou velocidade das tentativas, especialmente como a execução ocorre através do navegador. Para os três padrões propostos, foram também testadas as seguintes variações para observar se as mesmas impactavam os resultados de alguma forma:

- receber as credenciais do perfil falso em tempo de execução via *script*;
- ler as credenciais do perfil falso de arquivo;
- uso de credenciais erradas/inválidas;
- uso de um *proxy* público para executar a tentativa de autenticação de um país aleatório, diferente do definido originalmente na criação do perfil.

Analisando os resultados dos testes, não foi possível observar quaisquer diferenças no comportamento da rede social, além de após algumas tentativas com as credenciais incorretas. Os testes de cada formato foram realizados em dias diferentes, de forma a garantir que a execução de um teste não interferisse nos demais. A critério de comparação, foram também executadas 10 tentativas sequenciais de *logon* utilizando as credenciais válidas,

repetindo com credenciais inválidas, porém todos realizados de forma manual (sem uso do *bot*), através do navegador, onde também não foram verificadas diferenças nos resultados.

Quando os testes foram realizados utilizando credenciais inválidas, tanto nos testes automatizados usando o *bot* quando nos manuais, após a sexta tentativas o *LinkedIn* passava a solicitar uma checagem de *puzzle* (similar a verificação *Captcha*) e/ou solicitava uma validação adicional, como um código enviado por email ou mensagem, para prosseguir com o *logon*. Esse padrão indicou que comportamentos de força bruta são identificados e bloqueados, o que não ocorre porém com outras formas de tentativa de acesso automatizadas.

Um ponto de discussão em potencial seria a quantidade de requisições utilizada. Apesar de uma quantidade similar de tentativas em alguns cenários possa ser reproduzida por um ser humano, este padrão ocorreria apenas em caso proposital para testes, não para uso regular da rede social. O processo padrão de entrada envolve inserir as credenciais, realizar a autenticação e então navegar na rede social, com alguns eventuais erros de acesso causados por confusão ou erros de digitação em algumas tentativas. Da mesma forma que após algumas tentativas sem sucesso (seis, no caso específico do *LinkedIn*, um controle adicional (*puzzle/Captcha*) é requisitado pois esse comportamento é considerado suspeito, mesmo que ele possa também ser reproduzido por um ser humano. Da mesma forma, entende-se que valores ao redor de 10 tentativas simultâneas ou em um espaço de tempo muito curso se caracterizariam como suspeitos e potencialmente de caráter automatizado.

Teste Etapa 2 - Busca:

Nessa etapa foi avaliada a capacidade do *Bot* Plataforma de executar consultas na rede social de forma automatizada sem detecção. Para realizar as buscas, foram definidas uma série de palavras-chave. É importante ressaltar que os resultados das buscas, tanto a quantidade de perfis retornados quando a ordem em que aparecem e informações similares tem relação direta com o algoritmo de busca proprietário do *LinkedIn*, e entender ou manipular os seus resultados não fizeram parte do escopo desse trabalho. Os termos de busca utilizados foram apenas uma maneira de avaliar a resposta da rede à consultas automatizadas através do navegador. A Figura 3.7 demonstra o *bot* executando a fase de busca.

Para o teste dessa fase foram listados 10 termos, baseados em alguns conhecimentos comuns na área de computação relacionados apenas para referência. Os termos usados foram os seguintes: *test* ("teste"), *Social Engineering* ("Engenharia Social"), *Bot*, *ChatBots*, *Social Networks* ("Redes Sociais"), *Information Security* ("Segurança da Informação"), *Python*, *Automation* ("Automação"), *GitHub* e *API*. De forma similar a Etapa 1, foram utilizadas as seguintes variações:

- Buscar o mesmo termo 10 vezes de forma simultânea;
- Buscar o mesmo termo 10 vezes sequenciais, com 5 segundos de intervalo entre cada consulta;
- Realizar 10 consultas simultâneas, cada uma utilizando um termo diferente;

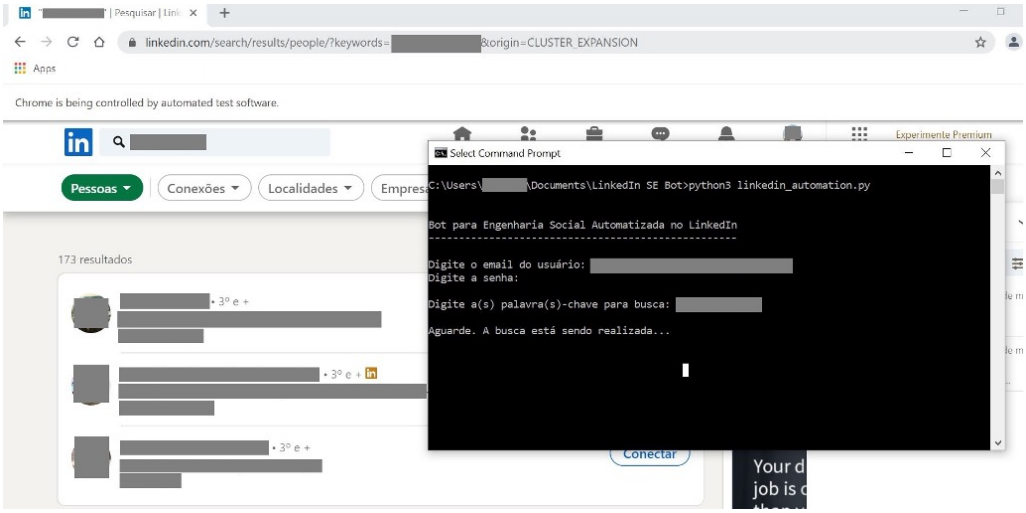


Figura 3.7: Fase de Busca sendo executada pelo *Bot* Plataforma.

- Buscar os 10 diferentes termos na mesma sessão, em sequência, com 5 segundos de intervalo entre cada consulta.

Não foi o objetivo dessa etapa realizar testes de carga ou estresse nem causar negação de serviço na aplicação. Os comportamentos testados analisam a quantidade ou velocidade das consultas em valores acima dos esperados de um usuário humano, especialmente se feitas via navegador. Apesar das diferenças esperadas nos resultados das consultas (considerando os diferentes termos utilizados e o algoritmo da rede social, o qual esteve fora do escopo de análise), não foram observadas diferenças nas variações, e todas as consultas retornaram nos resultados perfis com alguma associação com a palavra-chave utilizada.

Teste Etapa 3 - Abordagem: Na fase anterior, após realizar as consultas, o *Bot* Plataforma armazenava referências aos perfis retornados para que os mesmos forem utilizados nessa próxima fase. Aqui o principal desafio seria não violar os requisitos éticos de pesquisa, interagindo com usuários reais sem autorização ou conhecimento dos mesmos. Buscando limitar o impacto da pesquisa com a menor interferência possível nos resultados, os seguintes controles foram aplicados no *bot*:

- Para cada consulta, ao invés de armazenar as diversas páginas de resultados, foram guardados apenas os perfis da primeira página, normalmente entre 15-21 de acordo com o termo utilizado;
- A abordagem foi realizada em apenas 10 grupos, um por palavra-chave, independente das variações utilizadas nas consultas;
- Após a execução da abordagem, o *bot* registrava o sucesso do envio e então apagava/cancelava todas as ações de interação realizadas com o alvo.

A abordagem ocorria com um pedido de conexão e o envio de uma mensagem customizada, utilizando *tags* para personalizar o uso do nome real do usuário em vez de termos

genéricos. Como já citado, assim que o envio do pedido e da mensagem eram confirmados, a requisição era cancelada e a mensagem apagada, evitando quaisquer interações futuras com os usuários. Novamente nenhum impacto ou ação por parte da rede social pode ser percebida durante os testes. A Figura 3.8 mostra um exemplo de mensagem durante os testes.

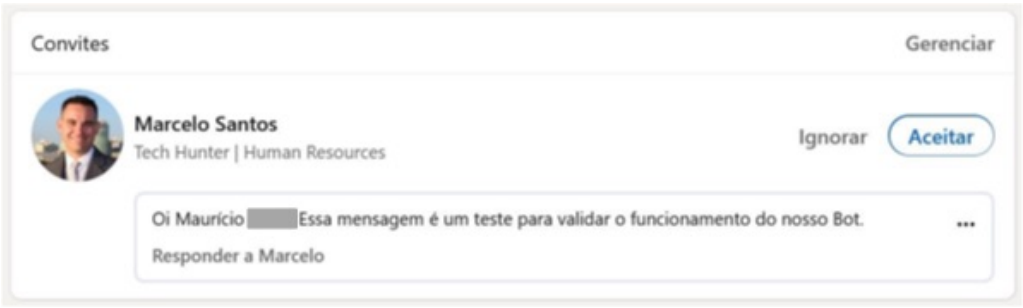


Figura 3.8: Exemplo de mensagem customizada.

Teste Etapa 4 - Entrevista:

O teste dessa etapa seguiu uma direção diferente. Foi utilizado o *SAP Conversational IA*[6], uma plataforma de criação de *chatbots*. Ao invés de criar um próprio, foi utilizado um *chatbot* recrutador já existente na plataforma, chamado *Smart Recruiter*. Utilizando essa abordagem, além de algumas verificações básicas, não seria necessário de validar a capacidade do mesmo em realizar uma entrevista, mas sim apenas de prover os valores maliciosos que seriam de interesse de um atacante e verificar os resultados. Baseado em um banco de dados inicial com questões padrão de uma entrevista de emprego já disponíveis no *chatbot*, foi utilizado um *script* para puxar os dados do perfil da vítima e utilizar os mesmos na criação de perguntas adicionais, permitindo questões mais específicas como "Conte sobre sua experiência na E=empresa X?", "Você poderia falar mais sobre suas habilidades na tecnologia Y?"ou mesmo "Você poderia mencionar os principais clientes e projetos nos quais você teve um papel chave enquanto esteve na empresa X?".

Baseando-se na observação da capacidade do *chatbot* em realizar entrevistas utilizando informações de diferentes perfis selecionados aleatoriamente da etapa anterior, o mesmo teve capacidade de realizar a entrevista completa sem necessidade de intervenção. Esses resultados permitem demonstrar a capacidade do mesmo em ser utilizado no ataque proposto sem a necessidade de abordar pessoas reais, podendo afetar os resultados buscados ou violar princípios éticos de pesquisa. A Figura 3.9 mostra parte da interação com o usuário durante uma entrevista. Apesar da tela padrão de conversa do *chatbot* ter sido utilizada durante os testes, a plataforma oferecia ferramentas que permitiram a execução da entrevista através de diferentes canais de comunicação, utilizando APIs ou *webhooks*. Seria possível inclusive ao atacante criar um *website* de uma falsa empresa e integrar o *chatbot* a mesma para criar um cenário que inspire ainda mais confiança da vítima.

[6]https://cai.tools.sap

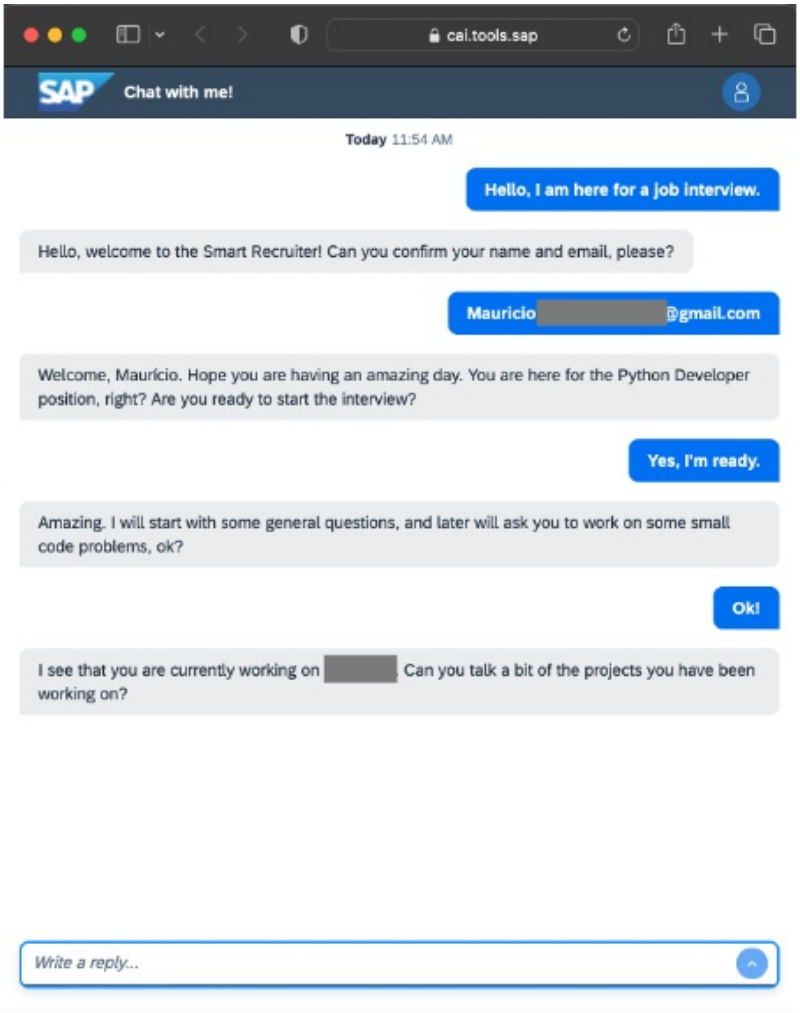


Figura 3.9: *Bot* Recrutador realizando uma entrevista.

3.6.1.4. Análise dos Resultados e Recomendações

O principal objetivo desse caso foi validar a hipótese da falta ou insuficiência de controles implementados nas redes sociais. Mesmo que esses resultados fossem esperados por algumas pessoas do campo da tecnologia, não foram encontratos pesquisas oficiais ou referências acadêmicas que pudessem embasar cientificamente essas conclusões. Não foi o objetivo desse teste específico avaliar o fator humano, que é normalmente o foco em boa parte dos trabalhos no campo de ES, mas de observar como canais tecnológicos permitem ou não oferecem suficientes barreiras para evitar ou diminuir os riscos aos seus usuários, especialmente pois em casos como o analisado o atacante pode atingir grande escalabilidade.

Certamente, o principal impacto de implementar controles rígidos em uma rede social seria a própria experiência do usuário, que como resultado por tornar a navegação menos fácil e fluída, causando potencial migração dos usuários para serviços concorrentes e causar efeitos negativos nos indicadores da plataforma. É, porém, possível encontrar um equilíbrio entre o aumento dos níveis de segurança com pouco ou nenhum impacto à

usabilidade.

Considerando que os Termos de Uso atuais do *LinkedIn* já proíbem o uso de automação, controles que permitam a detecção de comportamento automatizado poderiam ser aplicados. Eles não apenas ajudariam a evitar ESA, mas também outros padrões já proibidos. Esse controle porém poderia ser implementado com um nível de flexibilidade. Um usuário regular, conectado via navegador, possui algumas limitações humanas na sua velocidade e quantidade de requisições - acima de um certo limite, não apenas há caracterização de uso de automação, como há altas chances de *SPAM* e outras formas de interação não-solicitadas.

Baseando-se nos resultados desse estudo de caso, alguns exemplos de controles simples para identificar comportamente automatizado seriam:

- Mais de um *logon* simultâneo do mesmo usuário (com algumas variações, como considerar o caso de um usuário conectado através do computador e do *smartphone* ao mesmo tempo), ou uma grande quantidade de *logons* feita num curto espaço de tempo;
- Diversas requisições simultâneas e/ou contínuas (não apenas de busca, mas para qualquer ação) em uma quantidade ou faixa de tempo acima da capacidade normal de um ser humano;
- Diversos pedidos de conexão e/ou envio de mensagens para diferentes usuários simultâneamente ou numa curta faixa de tempo (potencialmente também indicando *SPAM*.

A aplicação dos controles propostos não precisa causar o bloqueio da requisição ou penalidades ao usuário. Exigir dados adicionais como uma validação do tipo *Captcha*, similar ao que já é implementado para evitar ataques de força bruta, já seria uma excelente forma de evitar a automação, já que o mesmo seria necessário apenas em certos cenários, não afetando a grande maioria dos usuários em seu uso regular.

Indo um pouco além, é possível imaginar que alguns usuários ou em certos cenários algum nível de automação seja útil ou necessário - para recrutadores reais, por exemplo, a aplicação desses controles pode ser mais rígida nas conexões via navegador (onde usuários humanos são esperados, não automação), e mais flexível nas conexões via API, por exemplo. Isso permitiria um melhor monitoramento e controle por parte da plataforma, podendo ser inclusive uma oportunidade de negócios oferecer uma certa quantidade de requisições para clientes pequenos (como recrutadores independentes), ou soluções mais robustas vendidas como serviço pela plataforma, como o já existente serviço *LinkedIn Recruiter*.

Os controles propostos permitiram resolver a questão da automação, aplicando políticas que já existem com o menor impacto possível aos usuários. Um desafio diferente porém é a questão dos perfis e dados falsos, problema que não afeta apenas o *LinkedIn* mas também todas as demais redes sociais nos dias de hoje. É possível estabelecer uma base de interações e contatos que crie uma sensação de legitimidade, organicamente através de usuários reais ou mesmo através de uma rede de perfis falsos.

Verificação de usuários envolve validações mais complexas. Porém o impacto dos perfis falsos tem crescido tão rápido que discussões sobre validação obrigatória de usuários já surgiram até mesmo em outras redes como o Twitter[7]. O projeto de pesquisa analisado nesse estudo de caso vem recomendando alguma funcionalidade de validação desde o seu início, e recentemente o *LinkedIn* anunciou seu programa de verificação[8]. O programa ainda tem suas limitações, como a validação através do email corporativo para algumas empresas registradas ou através de documentos de identidade (por enquanto disponível apenas nos EUA), mas certamente é um grande avanço. Considerando o objetivo do *LinkedIn* como rede social profissional, credibilidade e veracidade devem ser um ponto de interesse de todos os seus usuários. Potencialmente mesmo sem obrigatoriedade muitos usuários devem buscar a validação como forma de reconhecer seu trabalho e responsabilidade - ou mesmo alguns grupos-alvo podem ser priorizados, como recrutadores. Existem diversas oportunidades, cada com seus prós e contras, mas certamente algum controle nesse sentido é necessário para ao menos dificultar os ataques de personificação.

3.6.2. Estudo de Caso 2 - Testes Padronizados de ES

Esse estudo de caso propôs uma metodologia para realização de testes de *phishing* e executou testes com grupos de participantes em diferentes cenários para observar sua resposta e comportamentos. Um dos objetivos do trabalho foi oferecer guia para a produção de futuros testes, devendo os passos serem acessíveis, podendo ser facilmente reproduzidos, e a infraestrutura necessária com o menor custo possível, para que os testes não dependam de grandes orçamentos.

O processo se divide em cinco etapas: autorização, planejamento, montagem da infraestrutura *web*, montagem e envio dos *e-mails* e coleta e análise dos resultados. Antes de qualquer teste de Engenharia Social, por envolver informações e os colaboradores da empresa, deve-se iniciar pela autorização explícita de um gestor ou pessoa responsável, sendo uma boa prática que o cargo do mesmo seja superior ao do alvo com maior nível de hierarquia. Jamais realize um teste do tipo sem autorização formal. A etapa de planejamento então envolve a análise dos alvos e a escolha do serviço/sistema a ser clonado, diretrizes iniciais que irão nortear o restante do teste. A escolha do serviço ou sistema a ser clonado deve dar preferência à páginas que solicitem credenciais de acesso, pois assim é possível diferenciar usuários que apenas acessaram a página dos que realmente foram vítimas da fraude.

A etapa de montagem da infraestrutura *web* é a que envolve mais passos técnicos. Em primeiro lugar é necessário uma infraestrutura de servidor para hospedagem do *phishing*. Nos testes executados foi criada uma máquina virtual com acesso IP público na nuvem do serviço *Amazon Web Services* (AWS). A máquina utiliza sistema operacional Ubuntu Linux Server, seguindo a configuração básica disponível, classificada dentro do plano de *Free Tier*, o qual, dentro de determinados critérios de utilização não possui custos, apresentando-se então como uma plataforma acessível para criação de testes. Foi possível realizar os dois experimentos práticos deste trabalho sem ultrapassar os limites estabelecidos, portanto sem nenhuma cobrança. Nesse servidor foi então instalado o

[7]https://help.twitter.com/en/managing-your-account/about-twitter-verified-accounts

[8]https://www.linkedin.com/help/linkedin/answer/a1359065/verifications-on-your-linkedin-profile

Social Engineering Toolkit (SET) para clonagem dos *websites* válidos.

Um fator importante para maior credibilidade do teste é o registro de um domínio. No primeiro teste foi utilizado um domínio *.br devidamente registrado, com grafia semelhante ao original, ao custo de 40 reais anuais, um valor acessível. Para o segundo teste utilizou-se um registro gratuito, tendo, porém, uma URL mais chamativa para identificação do *phishing*. A escolha por um domínio registrado ou gratuito pode ser equilibrada com o nível de dificuldade para identificação da fraude nos demais aspectos do teste. A não utilização de um domínio pode causar problemas com filtros *anti-SPAM* na etapa de envio dos *e-mails*. Os domínios devem ser configurados no arquivo de *VirtualHosts* do Apache no servidor, procedimento bastante simples e amplamente documentado na internet.

Através do SET é feita então a clonagem dos *websites* verdadeiros, através da ferramenta *Web Cloner*, disponível no serviço de *Credential Harvester*. É importante no momento da criação do clone o endereço de retorno ser apontado para o domínio criado anteriormente. Para garantia de que não ocorra cruzamento dos dados que pudessem comprometer a identificação dos resultados da pesquisa, utiliza-se arquivos individuais para cada alvo do teste. A Figura 3.10 apresenta a estrutura de arquivos utilizada.

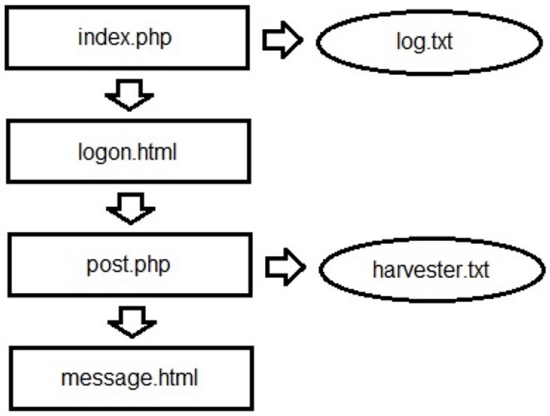


Figura 3.10: Estrutura de arquivos do *phishing*.

O primeiro arquivo, *index.php*, captura o IP do usuário, a data e a hora do acesso, o endereço de origem (*HTTP Referer*, para garantir a origem do usuário) e as informações do navegador utilizado. As informações são então salvas no arquivo *log.txt* e o usuário é automaticamente redirecionado para o arquivo *logon.html*, que é a página clonada com o auxílio do SET. A Figura 3.11 demonstra o conteúdo do arquivo *index.php*.

O arquivo *logon.html* é o *phishing* em si, a página que deve capturar as credenciais do usuário. O meio de validar que um usuário realmente foi capturado seria a coleta das credenciais, e o acesso ao sistema/serviço verdadeiro com as mesmas. Esse tipo de teste, porém, entra em conflito com boas práticas de segurança, não sendo sugerida a sua realização. O teste de Engenharia Social visa uma visão da resposta dos colaboradores à um ataque, não podendo ser um vetor de incidentes, devendo a segurança das informações do usuário ser uma prioridade. Em virtude disso, um importante fator a ser considerado é o fato do *phishing* não utilizar criptografia (HTTPS), portanto as credenciais do usuá-

```php
<?php

$arquivo = 'log.txt';
$novalinha = "\n";

$timestamp = date("d/m : H:i");
$ip = $_SERVER['REMOTE_ADDR'];
$referer = $_SERVER['HTTP_REFERER'];
$browser = $_SERVER['HTTP_USER_AGENT'];

file_put_contents($arquivo, $timestamp.$data.PHP_EOL, FILE_APPEND);
file_put_contents($arquivo, $ip.$novalinha, FILE_APPEND);
file_put_contents($arquivo, $referer.$novalinha, FILE_APPEND);
file_put_contents($arquivo, $browser.$novalinha.$novalinha, FILE_APPEND);

?>

<meta http-equiv="refresh" content="0; url=http://www.site.com/logon.html" />
```

Figura 3.11: Arquivo *index.php*.

rio que for vítima serão enviadas ao servidor sem qualquer tipo de proteção. Para evitar esse tipo de problema, uma boa prática é adicionar na própria página do *phishing* um pequeno código que apague a senha antes mesmo dela ser enviada, de forma completamente transparente para o usuário, capturando apenas o nome de usuário. Para isso é necessário abrir o código da página e identificar o ID relacionado ao campo de senha, normalmente *password*. Depois localizar a linha do formulário, a responsável pelo envio das informações, iniciada pelo código *<form name=*. Verificar se a linha possui o parâmetro *onsubmit=*, se sim, o mesmo deve ser substituído pelo código, caso não, basta adicionar o mesmo código, lembrando de substituir *id_senha* pelo ID identificado no corpo da página: *onsubmit="document.getElementById('id_senha').value='';"*. A utilização desse código elimina a senha antes dos dados serem tratados pela página, portanto garantindo a mínima segurança para as informações do usuário.

Uma vez submetidas as credenciais, o próximo arquivo chamado é o *post.php*, responsável pela captura das informações. No momento que a página é clonada através do SET é gerado automaticamente um arquivo *post.php*, porém o mesmo não captura todas as informações necessárias e redireciona o usuário vitimado para a página verdadeira, fugindo do objetivo do teste. A Figura 3.12 apresenta o código utilizado no *post.php*. Além das credenciais, ele captura as mesmas informações que o arquivo *index.php*, de forma que seja possível comparar as mesmas e garantir que é o mesmo usuário que acessou e forneceu as informações.

As informações do usuário são então enviadas para o arquivo *harvester.txt*, e o usuário é automaticamente redirecionado para o arquivo *message.html*. Esse arquivo contém uma mensagem para o usuário que falhar no teste. Recomenda-se iniciar com logo ou cabeçalho oficial da instituição. Em seguida, solicitar ao usuário que evite comentar com seus colegas a respeito do teste, evitando assim que seja gerado um alerta que comprometa os resultados. O texto deve então reiterar que a senha do usuário não foi comprometida, e apresentar os objetivos do teste, que são avaliar o nível de capacidade dos colaboradores para identificação de fraudes e, baseado nisso, montar futuras estratégias de treinamento e capacitação. Alertar o usuário que o teste não visa punir quem for vítima da fraude é uma prática extremamente recomendável, visto que o mesmo pode se caracterizar como assédio moral dentro de determinadas circunstâncias e a prática do medo de punição tem

```php
<?php

$file = 'harvester.txt';
$quebra = "\n";

$dados = $_POST;
$date = date("d/m : H:i");
$ip = $_SERVER['REMOTE_ADDR'];
$referer = $_SERVER['HTTP_REFERER'];
$navegador = $_SERVER['HTTP_USER_AGENT'];

file_put_contents($file, $dados, FILE_APPEND);
file_put_contents($file, $quebra.$date.$data.PHP_EOL, FILE_APPEND);
file_put_contents($file, $ip.$quebra, FILE_APPEND);
file_put_contents($file, $referer.$quebra, FILE_APPEND);
file_put_contents($file, $navegador.$quebra.$quebra, FILE_APPEND);
?>

<meta http-equiv="refresh" content="0; url=http://www.site.com/message.html" />
```

Figura 3.12: Arquivo *post.php*.

um efeito contrário aos objetivos buscados com essa forma de teste.

O próximo passo na mensagem é auxiliar o usuário a perceber as falhas da fraude enviada. Revelar os exemplos que possam ser observados no próprio teste criado, como *e-mail* de origem de um domínio inválido, a URL com grafia incorreta, a ausência de criptografia na página, etc. Outros exemplos relevante podem também ser adicionados. Esse é o mais importante item do teste, pois é a ferramenta que vai auxiliar os usuários a fixar os conhecimentos necessários para identificar um ataque verdadeiro. É importante também apontar ao usuário como proceder em caso de suspeita de ES, e quais canais devem ser utilizados para notificação. A mensagem pode terminar oferecendo *links* e outras informações que auxiliem os usuários a descobrir mais sobre o assunto, aprofundando o conhecimento gerado, como treinamentos, alguma política ou diretriz interna, e afins.

A fim de criar os arquivos individuais, sugere-se a utilização de alguma forma de codificação nos nomes dos arquivos, para garantir que cada usuário seja direcionado para um fluxo separado e não consiga alterar os resultados dos demais usuários. É importante lembrar que além de criar os arquivos é necessário alterar o conteúdo dos mesmos, pois os mesmos possuem *links* para o próximo arquivo. A codificação errada no conteúdo irá misturar os dados capturados e afetar os resultados do teste. É importante também conferir as permissões de acesso dos arquivos *log.txt* e *harvester.txt* para garantir que seja possível gravar as informações capturadas neles.

A próxima etapa é a criação do e-mail. Tendo um domínio, é possível criar um *e-mail* no mesmo domínio gratuitamente durante uma fase de testes através do serviço *Google Workspace*, o que foi utilizado no primeiro teste realizado. Para o segundo teste foi criado um *e-mail* simples no serviço *Gmail*, o qual não tem custos. Para a criação do corpo do *e-mail*, é ideal seguir um modelo de mensagem verdadeiro, lembrando-se de utilizar a URL correta para cada usuário. Uma boa prática é realizar alguns testes de envio do *e-mail* antes de encaminhar aos alvos, a fim de evitar que o mesmo seja identificado por filtros *anti-SPAM* e bloqueado.

Uma vez enviados os e-mails, os resultados capturados podem ser verificados conferindo o conteúdo dos arquivos *log.txt* e *harvester.txt*. Deve-se aguardar um período suficiente para que todos os usuários possam ter visto a mensagem, sendo sugerido cerca

de 48 horas. Os usuários que forem vitimados recebem o material informativo no mesmo momento que caírem na fraude, porém a informação deve ser repassada mesmo aos demais envolvidos. Quando o teste se der por encerrado, recomenda-se enviar um *e-mail* de um endereço válido da instituição para todos os participantes no escopo do teste, detalhando que no período foi realizado um teste de Engenharia Social, e apresentando as informações presentes na mensagem vista pelos usuários vitimados. É importante mais uma vez frisar que o objetivo do teste não deve ser o teste em si, mas sim o treinamento e conscientização dos usuários para que sua capacidade de reconhecer fraudes aumente.

3.6.2.1. Experimentação Prática

O primeiro teste foi realizado em três ondas diferentes, envolvendo um total de 179 usuários com variados níveis de conhecimento. O ambiente envolvia colaboradores do setor de TI de uma instituição de nível superior, sendo o teste devidamente autorizado junto aos responsáveis e gestores do setor. A proposta de *phishing* envolveu um clone da página de acesso do serviço de *email*, sendo então enviada aos mesmos uma mensagem de correio eletrônico simulando as mensagens reais do serviço enviadas quando a cota de armazenamento da conta está próxima do limite. A Figura 3.13 mostra o modelo de e-mail enviado aos usuários, sendo que o link *Log in here* direcionava o usuário para uma página individual. As informações que permitam identificar a instituição foram suprimidas por questões de privacidade.

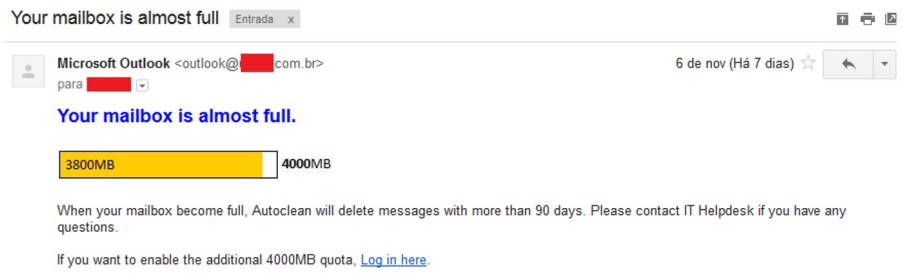


Figura 3.13: Modelo de *e-mail* utilizado no primeiro teste.

A primeira onda foi enviada para um grupo pré-selecionado de 05 usuários, todos de nível técnico mais avançado. O objetivo era ter uma ideia inicial do teste, e obter sugestões desses usuários sobre a qualidade do teste. A segunda onda por sua vez ocorreu para um primeiro grupo de 13 usuários, envolvendo diretores, responsáveis de área, líderes de equipe e outros cargos de gestão. Esse teste serviu como validação para que a direção da instituição autorizasse a última onda de testes. A terceira e última onda então foi enviada para os demais 161 usuários do setor, sendo a grande maioria destes usuários regulares com nível de conhecimento básico a respeito de *phishing*.

O segundo teste foi realizado em uma única onda, envolvendo um total de 47 usuários, sendo a grande maioria pessoal técnico. O ambiente envolvia colaboradores dos setores de TI, comercial e administrativo de uma empresa privada da área de tecnologia, sendo o teste devidamente autorizado junto aos responsáveis e gestores do setor.

A proposta de *phishing* envolveu um clone da página de serviço de calendário da companhia, utilizado para marcação de reuniões e compromissos, hospedado na plataforma *WebCalendar*, sendo então enviada aos colaboradores uma mensagem de correio eletrônico simulando as mensagens reais de agendamento de reunião enviadas pelo serviço.

A Figura 3.14 apresenta o modelo de *e-mail* enviado aos colaboradores. Como o modelo original apresentava uma URL completa, quando se configurava a mesma para direcionar para outra URL, porém mostrando a original, os filtros *anti-SPAM* identificavam a mensagem como fraude. O recurso utilizado foi substituir a URL no corpo do *e-mail* por uma imagem da mesma, com o link para o *phishing* associado à imagem, sendo assim possível burlar os filtros.

Figura 3.14: Modelo de *e-mail* utilizado no segundo teste.

3.6.2.2. Análise dos Resultados

O primeiro teste realizado envolveu 179 usuários, sendo destes cinco usuários técnicos e capacitados, 13 usuários gerenciais e 161 usuários com treinamento básico. Do total do grupo, apenas 12 acessaram a página do *phishing*, quatro usuários técnicos, três gerenciais e cinco usuários padrão. Nenhum usuário foi vitimado, sendo que sete entraram com dados apenas de teste. Também foi verificado que 13 usuários reportaram a tentativa de fraude ao setor responsável. Depois do evento, através de uma amostra de entrevistas individuais, houve a confirmação de que o teste de fato ampliou o nível de preocupação dos usuários, que reportaram estar mais atentos a partir daquele momento.

O segundo teste envolveu 47 usuários, sendo destes 42 usuários técnicos e capacitados, e cinco usuários padrão, porém com treinamentos na área de segurança. Do total do grupo, 14 acessaram a página do *phishing*, sendo todos com nível técnico de conhecimento. Desses, dez foram vitimados pelo teste, incluindo dois líderes de equipe técnica. Nenhum usuário utilizou os canais formais da empresa para notificar o incidente, mesmo os que perceberam a fraude. A equipe de segurança da informação elogiou a montagem e análise dos resultados do teste, sendo que os testes passarão a ser realizados periodicamente como parte das campanhas de segurança da empresa da companhia, e o teste e treinamentos específicos serão adicionados aos treinamentos periódicos já realizados com

as equipes.

Uma situação observada nos testes chamou a atenção durante a análise dos resultados. Um usuário que foi vítima da fraude cerca de cinco minutos após ter submetido suas credenciais submeteu em sequência os nomes de usuário verdadeiros de outros sete colaboradores, inclusive de gestores e executivos da empresa, possivelmente temendo os resultados do teste. Essa atitude traz à tona a importância de conscientizar os usuários quanto aos objetivos do teste, que não envolvem a punição, e sim o treinamento. Também foi assim validada a importância da utilização de arquivos individuais para cada alvo do teste, sendo possível diferenciar a identidade e informações fornecidas por cada usuário, com menores riscos de cruzamento de informações ou tentativas de manipular os resultados do teste por parte dos usuários.

Após a realização de dois testes em ambientes reais foi possível verificar que a estrutura proposta para a criação do *phishing* e *e-mail* atende aos requisitos de ser acessível, de baixo custo e simples de ser reproduzida, atingindo portanto os objetivos esperados. A montagem da estrutura não apresentou dificuldades, e o retorno recebido dos envolvidos foi positivo.

Uma das propostas iniciais foi a modificação do código do SET para automatizar a montagem da estrutura de arquivos. Após longa análise do código e diversas tentativas sem sucesso foi concluído que a abordagem não atenderia aos objetivos do trabalho. Optou-se então por utilizar apenas o mecanismo de clonagem de *websites*, o qual funcionou com sucesso em todos os testes realizados. Outra situação identificada neste trabalho foi um problema na captura credenciais quando o *phishing* padrão criado pelo SET era utilizado. Em alguns *websites* testados, ao submeter as credenciais, as mesmas não eram capturadas. Além de não fornecer todas as informações da vítima necessárias, em alguns casos o arquivo preparado pelo SET nem mesmo capturava as credenciais, devendo portanto sua utilização em configuração padrão ser feita com cautela, sendo sugerida a modificação manual dos arquivos.

Um possível problema verificado no primeiro teste, onde nenhum usuário foi vítima do teste, é o correto planejamento da montagem da página e do *e-mail*. No caso, foi opção de comum acordo com os responsáveis o envio das mensagens falsas em inglês, enquanto o sistema de *e-mail* da instituição estava configurado em português. Não há como garantir que esse tenha sido o real motivo da ausência de vítimas, porém a utilização de fraudes excessivamente fáceis de serem percebidas irão prejudicar os resultados esperados no trabalho.

A configuração da captura das informações dos usuários que tanto apenas acessavam a página quanto dos que submetiam as credenciais em arquivos individuais permitiu a correta identificação dos usuários, sem cruzamento de informações. Foi possível também diferenciar o foco de futuros treinamentos a partir das situações onde a maioria dos usuários caiu na fraude, o que seria o pior cenário, e os usuários que apenas acessaram a URL, situação também não recomendável em muitas situações.

O teste também mostrou a importância de campanhas que incentivem os usuários a reportarem mensagens suspeitas. A ocorrência de um ataque direcionado à companhia ou instituição deve gerar um alerta imediato nas equipes e acionamento dos procedimentos

de resposta a incidentes.

3.7. Ética e Engenharia Social

Em segurança cibernética, o conceito de *hacking* ético refere-se ao uso de conhecimentos, técnicas, ferramentas e outros recursos para identificar, analisar e apoiar a mitigação de falhas e vulnerabilidades utilizadas por atacantes [Hatfield 2019] [Peake 2003]. Tem como objetivo proteger contra ataques e aprimorar a segurança [NIST 2019].

No *hacking* ético a execução das ações para explorar falhas e vulnerabilidades ocorre após consentimento [Hatfield 2019] e respeita diretrizes estabelecidas e aprovadas previamente [Peake 2003]. Adicionalmente, o processo completo é registrado, detalhando os procedimentos adotados, para que possam ser reproduzidos caso seja necessário [Peake 2003].

A utilização de ES também é aplicada por *hackers* éticos, entretanto, por operar nos fatores humanos da segurança, na tentativa de manipular indivíduos através de outros indivíduos ou de maneira automatizada, levanta questões éticas importantes [Hatfield 2019]. Apesar disso, observa-se a ausência de qualquer formalização, ainda que mínima, relacionada ao impacto ético de um ataque de ES em contextos não maliciosos [Mouton et al. 2015], como nas pesquisas ou na prática de *hacking* ético.

Na pesquisa e na prática em segurança da informação, a ética no campo da ES é pouco estudada. Ainda que aspectos éticos relacionados a execução de testes de invasão e técnicas de *hacking* sejam abordados, discussões específicas acerca da ética da ES e ESA não são facilmente identificadas [Hatfield 2019]. Segundo [Hatfield 2019], além de serem escassas pesquisas que se dedicam explicitamente à ética da ES, estas apresentam limitações relacionadas a abordagem do tema, que tende utilizar abordagens teóricas reduzidas.

Dentre os fatores que podem contribuir para a pouca produção no tema, merece destaque as complexidades ligadas à ética coletiva e individualizada e suas respectivas restrições [Mouton et al. 2015]. E a necessidade de abordagens plurais e de um diálogo interdisciplinar, no qual pesquisadores e especialistas em psicologia, ética e segurança da informação possam colaborar de maneira conjunta para o debate.

Dado o cenário de escassa produção acadêmica sobre o tema, bem como a ausência de formalizações, esta seção não tem objetivo apresentar um consenso sobre a ética e a ES em contextos não maliciosos. No entanto, busca colaborar para abertura do debate acerca da ética em ES, apresentando possíveis contribuições à discussão, a partir da ética em pesquisa com seres humanos, da ética em pesquisas na internet e das orientações e códigos de ética profissionais do campo da tecnologia e da internet. Áreas nas quais já há conceituações, formalizações e cujo debate já está amplamente difundindo.

Ainda que as técnicas de ES possam ser aplicadas fora do contexto de mediação tecnológica, é consenso que a ES costuma utilizar predominantemente tecnologias baseadas em internet. Principalmente pelo poder amplificador destas. Desta forma, justifica-se a inclusão no debate sobre ética, das considerações sobre a condução de pesquisas éticas baseadas em internet. Além disso, a investigação baseada em internet deixou de ser uma metodologia e passou a uma prática quase onipresente, exigindo consideração cui-

dadosa no que diz respeito aos conceitos tradicionais da pesquisa com seres humanos [Buchanan and Zimmer 2021][.

Em pesquisas que relacionam-se com tecnologia e internet, a ética compreende questões relacionadas a consentimento dos participantes, a privacidade, confidencialidade e integridade de dados, ao anonimato, à propriedade intelectual, a aspectos coletivos e códigos e condutas profissionais [Buchanan and Zimmer 2021]. É definida como sendo a análise de questões éticas e a aplicação de princípios de ética em pesquisa no que se refere à pesquisas conduzidas ou mediadas pela internet [Buchanan and Zimmer 2021]. Abrange, entre outras, quaisquer pesquisas e estudos que coletem dados ou realizem a análise de atividades a partir ou em ambientes *online*, bem como pesquisas relacionadas com os efeitos da mediação pela internet em comportamentos de indivíduos [Buchanan and Zimmer 2021].

Independente do cenário de pesquisa incluir intervenção tecnológica ou mediada, qualquer investigação deve considerar e ser orientada por princípios éticos fundamentais [Buchanan and Zimmer 2021]. Na pesquisa com seres humanos, há princípios bem estabelecidos. Uma pesquisa ética é aquela que respeita o participante, levando em conta a sua vulnerabilidade e ponderando diligentemente sobre os riscos e benefícios, tanto os estabelecidos quanto os possíveis. Considerando impactos individuais e coletivos. Além de comprometer-se em não causar dano, em maximizar possíveis benefícios e em minimizar possíveis danos e prejuízos. Essas características são resumidas pelos princípios da autonomia, beneficência, não maleficência e justiça.

No que se refere a ética profissional, códigos como o IEEE Código de ética [IEEE 2020] apontam , entre outras orientações: (a) priorizar a segurança, a saúde e o bem-estar dos indivíduos, esforçar-se para cumprir o design ético e práticas de desenvolvimento sustentável, proteger a privacidade e divulgar fatores que possam gerar riscos à indivíduos ao ao meio ambiente; (b) melhorar a compreensão, dos indivíduos e da sociedade, sobre as capacidades e as implicações sociais das tecnologias convencionais e emergentes, incluindo os sistemas inteligentes; (c) evitar condutas ilícitas nas atividades profissionais e rejeitar o suborno em todas as suas formas; (d) reconhecer e corrigir erros, ser honesto e realista ao declarar afirmações ou estimativas baseadas em dados disponíveis; (e) evitar prejudicar terceiros, a sua propriedade, reputação ou emprego através de ações falsas ou maliciosas, rumores ou quaisquer outros abusos verbais ou físicos; (f)tratar todas as pessoas de forma justa e respeitosa e não praticar discriminação; e (g) esforçar-se para garantir que o código seja respeitado[IEEE 2020].

A partir de tais ponderações e de alguns pontos de convergência, podemos considerar que as práticas de ES em contextos não maliciosos devem analisar e avaliar questões éticas relacionadas principalmente: a informação e consentimento dos indivíduos, à privacidade, á confidencialidade e à integridade de dados, à anonimização de dados, a minimização de riscos e prejuízos aos indivíduos e a não geração de danos. Adicionalmente, códigos de ética e condutas profissionais específicas do contexto devem ser seguidos.

No âmbito da informação e do consentimento, dada as características de um teste de invasão e do *hacking* ético, um forma de garantir que as pessoas estejam cientes da possibilidade de participarem de ações que poderão submetê-las a práticas de ES, são comunicados coletivos e gerais. Informando que eventualmente podem ser realizados

pela organização testes ou rotinas para checar a eficácia dos controles de segurança da informação implementados. Sem individualizar os testes e sem especificar o momento no qual irá ocorrer, o que poderia gerar viés nos resultados.

Essa já é uma prática comum em organizações com áreas de segurança da informação estabelecidas e com rotinas para testar a probabilidade de seus colaboradores caírem em ataques de *phishing*. Nestes comunicados, devem estar explícitos que os indivíduos não sofrerão sanções de qualquer natureza baseado nos resultados ou nas informações coletadas. Além de informar que poderão ocorrer ações do gênero, devem ficar claros também os objetivos da ação. Sugere-se ainda que os participantes sejam informados acerca da natureza dos dados coletados, bem como do compromisso e das medidas adotadas para garantir privacidade.

No que tange a privacidade e confidencialidade em ES de caráter não malicioso, os dados que permitem identificar participantes, podendo gerar constrangimento, exposição ou qualquer tipo de prejuízo podem e devem ser considerados de acesso restrito. No caso de organizações, restritos a equipe interna ou contratada para execução dos testes e no caso de pesquisas acadêmicas, restritos aos pesquisadores envolvidos.

Os resultados devem ser tratados de maneira coletiva, fazendo referência a organização inteira, a setores ou áreas de uma organização. E sempre que possível, sugere-se a anonimização dos dados ou descarte de dados que possam identificar pessoas. Em pesquisas acadêmicas, os dados devem ter sua utilização restrita a pesquisa e as produções científicas relacionadas. Quando não necessária a identificação dos participantes não recomenda-se a coleta de dados que a permitam. Nos casos em que se faz necessária, estes devem ser anonimizados. A guarda dos dados deve obedecer as orientações e regulamentações dos comitês de ética em pesquisa e das legislações vigentes.

Uma abordagem coletiva e ampla, reduz a probabilidade de individualizar a responsabilidade e de gerar sobrecarga. Isso, por sua vez, reduz a autorresponsabilização e os efeitos negativos que a tomada de consciência sobre a própria susceptibilidade a um ataque de ES poderia gerar em uma pessoa ou a um pequeno grupo. Minimizando possíveis prejuízos aos quais a pessoa estaria vulnerável.

É importante lembrar que qualquer pessoa é susceptível a ataques de ES. Assim, a responsabilização de indivíduos, para além de promover mal-estar e condutas antiéticas, não tem potencial para promover uma cultura de segurança e não se relaciona com práticas efetivas em segurança da informação.

Em práticas éticas, seja qual for sua natureza, debater questões relativas aos objetivos, à eficácia e os benefícios de determinadas ações é imprescindível. Não seria diferente na execução de ES em cenários não maliciosos.

O objetivo de qualquer ação ou pesquisa em ES é garantir e promover segurança da informação, protegendo assim usuários e organizações. Dessa forma, tais práticas não podem perder de vista tal objetivo. O fator humano e suas vulnerabilidades são explorados, entretanto, o foco das práticas são os controles implementados ou necessários e as possíveis melhorias nas tecnologias.

O debate é extenso e merece atenção e produções futuras que incluam novas defi-

nições e consensos a partir de regulamentações e de orientações às pesquisas que utilizam tecnologias. Suas particularidades também podem exigir novas orientações regulatórias e profissionais. Por fim, a exigência do debate sobre ética em cibersegurança torna mais evidente o quanto segurança da informação se inscreve para além do campo tecnológico, abrangendo campos sociais e comportamentais.

3.8. Considerações Finais

Neste capítulo, exploramos a interseção entre a ES, a Psicologia Cognitiva e a ESA. O conteúdo apresentado busca entender os impactos da ES no comportamento humano e suas vulnerabilidades psicológicas para obter informações pessoais e corporativas, bem como formas de persuasão dos usuários para realizarem ações indesejadas.

É indiscutivelmente desafiador abordar de maneira completa todos os ângulos da ciência cognitiva, da psicologia e de suas intersecções com a segurança da informação em apenas um capítulo, aprofundar-se nesses campos certamente proporciona uma perspectiva mais abrangente sobre os ataques de ES. Ao invés de buscar uma compreensão exaustivamente técnica, permite direcionar esforços para compreender e, consequentemente, proteger vulnerabilidades inerentes à condição humana.

A relevância dos fatores humanos, já intrínseca no pilar pessoas da segurança da informação, ganha proeminência no contexto da ES. A compreensão da interação entre seres humanos e tecnologia, juntamente com a análise dos elementos psicológicos envolvidos, também deve ser incorporada no âmbito da segurança da informação.

Com o estudo realizado emerge o impacto da Psicologia relacionadas com as ações de ES. A manipulação de vieses cognitivos e a compreensão das características humanas envolvidas no processo de tomada de decisão têm sido a base para os ataques de ES. Com a análise realizada fica evidente que as estratégias de manipulação dos usuários no contexto da ES, estão ganhando escala no mundo digital com a ESA.

A ESA utiliza *Bots* e técnicas automatizadas no ecossistema digital, impulsionando exponencialmente a capacidade de ataques em larga escala. Essa transformação reforça a necessidade de abordar a segurança não apenas com medidas tecnológicas. A segurança deve considerar os aspectos psicológicos envolvidos.

À medida que as redes sociais e outras plataformas digitais continuam a moldar nossa interação *online* e *offline*, se faz necessário reconhecer os desafios e riscos associados à ESA. A confiança nas interações cibernéticas pode ser utilizada por atores maliciosos, comprometendo a segurança e a privacidade dos usuários.

Por fim, este capítulo destaca a importância de uma abordagem multidisciplinar para enfrentar os desafios da ES. A colaboração entre especialistas em Psicologia e SI é fundamental para desenvolver estratégias abrangentes de defesa contra os ataques de ES e ESA. Essa integração busca implementar o uso ético da automação para interação das ferramentas automatizadas com os usuários.

Referências

[APA nd] (n.d.). Apa dictionary of psychology - american psychological association.

[Al-Charchafchi et al. 2019] Al-Charchafchi, A., Manickam, S., and Alqattan, Z. N. (2019). Threats against information privacy and security in social networks: A review. In *International Conference on Advances in Cyber Security*, pages 358–372. Springer.

[Ancis 2020] Ancis, J. R. (2020). The Age of Cyberpsychology: An Overview. *Technology, Mind, and Behavior*, 1(1). https://tmb.apaopen.org/pub/2yn6jhyv.

[Attrill-Smith et al. 2019a] Attrill-Smith, A., Fullwood, C., Keep, M., and Kuss, D. J. (2019a). The online self. In *The Oxford Handbook of Cyberpsychology*, pages 17–34. Oxford University Press.

[Attrill-Smith et al. 2019b] Attrill-Smith, A., Fullwood, C., Keep, M., and Kuss, D. J. (2019b). *The Oxford Handbook of Cyberpsychology*. Oxford University Press.

[Bayer et al. 2020] Bayer, J. B., Triêu, P., and Ellison, N. B. (2020). Social media elements, ecologies, and effects. *Annual Review of Psychology*, 71(1):471–497.

[Beal 2005] Beal, A. (2005). Segurança da informação: Princípios e melhores práticas para a proteção dos ativos de informação nas organizações. *Atlas*.

[Benias and Markopoulos 2017] Benias, N. and Markopoulos, A. P. (2017). A review on the readiness level and cyber-security challenges in industry 4.0. In *2017 South Eastern European Design Automation, Computer Engineering, Computer Networks and Social Media Conference (SEEDA-CECNSM)*, pages 1–5. IEEE.

[Brasil 1940] Brasil (1940). Lei nº 2.848, de 7 de dezembro de 1940. *Diário Oficial [da] República Federativa do Brasil*.

[Buchanan and Zimmer 2021] Buchanan, E. A. and Zimmer, M. (2021). Internet research ethics. In Zalta, E. N., editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, summer 2021 edition.

[Camisani-Calzolari 2012] Camisani-Calzolari, M. (2012). Analysis of twitter followers of the us presidential election candidates: Barack obama and mitt romney. *Online). http://digitalevaluations. com*.

[Carr and Hayes 2015] Carr, C. T. and Hayes, R. A. (2015). Social media: defining, developing, and divining. *Atlantic Journal of Communication*, 23(1):46–65.

[Castells 2002] Castells, M. (2002). *A sociedade em rede*. Editora Paz e Terra.

[Collier et al. 2023] Collier, H., Morton, C., Alharthi, D., and Kleiner, J. (2023). Cultural influences on information security. In *European Conference on Cyber Warfare and Security*, volume 22, pages 143–150.

[Crossler and Bélanger 2014] Crossler, R. and Bélanger, F. (2014). An extended perspective on individual security behaviors. *ACM SIGMIS Database*, 45(4):51–71.

[Culot et al. 2019] Culot, G., Fattori, F., Podrecca, M., and Sartor, M. (2019). Addressing industry 4.0 cybersecurity challenges. *IEEE Engineering Management Review*, 47(3):79–86.

[Darwish et al. 2012] Darwish, A., Zarka, A. E., and Aloul, F. (2012). Towards understanding phishing victims' profile. In *2012 International Conference on Computer Systems and Industrial Informatics*, pages 1–5.

[de Souza Pereira et al. 2022] de Souza Pereira, L. A., Vicentine, A. L., and Rizo, A. C. (2022). Impactos da engenharia social na segurança da informação. *Revista Brasileira em Tecnologia da Informação*, 4(1):48–58.

[Dewangan and Kaushal 2016] Dewangan, M. and Kaushal, R. (2016). Socialbot: Behavioral analysis and detection. In *International Symposium on Security in Computing and Communication*, pages 450–460. Springer.

[Dickerson et al. 2014] Dickerson, J. P., Kagan, V., and Subrahmanian, V. (2014). Using sentiment to detect bots on twitter: Are humans more opinionated than bots? In *2014 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2014)*, pages 620–627.

[Ferrara et al. 2016] Ferrara, E., Varol, O., Davis, C., Menczer, F., and Flammini, A. (2016). The rise of social bots. *Communications of the ACM*, 59(7):96–104.

[Freitas et al. 2015] Freitas, C., Benevenuto, F., Ghosh, S., and Veloso, A. (2015). Reverse engineering socialbot infiltration strategies in twitter. In *2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 25–32. IEEE.

[Freitas et al. 2014] Freitas, C., Benevenuto, F., and Veloso, A. (2014). Socialbots: Implicações na segurança e na credibilidade de serviços baseados no twitter. *SBRC, Santa Catarina, Brasil*, pages 603–616.

[Gohn 2014] Gohn, M. d. G. M. (2014). *Sociologia dos movimentos sociais*. Number 47 in Questões da nossa época Sociologia. Cortez Editora, São Paulo, 2. ed edition.

[Greitzer et al. 2019] Greitzer, F. L., Purl, J., Leong, Y. M., and Sticha, P. J. (2019). Positioning your organization to respond to insider threats. *IEEE Engineering Management Review*, 47(2):75–83.

[Grimme et al. 2017] Grimme, C., Preuss, M., Adam, L., and Trautmann, H. (2017). Social bots: Human-like by means of human control? *Big data*, 5(4):279–293.

[Guzman and Lewis 2020] Guzman, A. L. and Lewis, S. C. (2020). Artificial intelligence and communication: A human–machine communication research agenda. *New Media & Society*, 22(1):70–86.

[Hatfield 2019] Hatfield, J. M. (2019). Virtuous human hacking: The ethics of social engineering in penetration-testing. *Computers & Security*, 83:354–366.

[Huber et al. 2009] Huber, M., Kowalski, S., Nohlberg, M., and Tjoa, S. (2009). Towards automating social engineering using social networking sites. In *2009 International Conference on Computational Science and Engineering*, volume 3, pages 117–124. IEEE.

[IEEE 2020] IEEE (2020). *Code of Ethics*. IEEE - The Institute of Electrical and Electronics Engineers, Inc.

[Khan and Das 2018] Khan, R. and Das, A. (2018). Build better chatbots. *A complete guide to getting started with chatbots*.

[Klimburg-Witjes and Wentland 2021] Klimburg-Witjes, N. and Wentland, A. (2021). Hacking humans? social engineering and the construction of the "deficient user" in cybersecurity discourses. *Science, Technology, & Human Values*, 46(6):1316–1339.

[Korteling and Toet 2022] Korteling, J. and Toet, A. (2022). Cognitive Biases. In *Encyclopedia of Behavioral Neuroscience, 2nd edition*, pages 610–619. Elsevier.

[Leary and Tangney 2014] Leary, M. and Tangney, J. P., editors (2014). *Handbook of self and identity*. Guilford Press, New York London, second edition edition.

[Leary 2019] Leary, M. R. (2019). *Self-Presentation: Impression Management and Interpersonal Behavior*. Routledge, 1 edition.

[Libicki 2018] Libicki, M. (2018). Could the issue of dprk hacking benefit from benign neglect? *Georgetown Journal of International Affairs*, 19:83–89.

[Martineau et al. 2023] Martineau, M., Spiridon, E., and Aiken, M. (2023). A comprehensive framework for cyber behavioral analysis based on a systematic review of cyber profiling literature. *Forensic Sciences*, 3(3):452–477.

[Messias et al. 2018] Messias, J., Benevenuto, F., and Oliveira, R. (2018). Bots sociais: Como robôs podem se tornar pessoas influentes no twitter? *Revista Eletrônica de Iniciação Científica em Computação*, 16(1).

[Mitnick and Simon 2003] Mitnick, K. D. and Simon, W. L. (2003). *The art of deception: Controlling the human element of security*. John Wiley & Sons.

[Montañez et al. 2020] Montañez, R., Golob, E., and Xu, S. (2020). Human cognition through the lens of social engineering cyberattacks. *Frontiers in Psychology*, 11.

[Mouton et al. 2015] Mouton, F., Malan, M. M., K., K. K., and Venter (2015). Necessity for ethics in social engineering research. *Computers Security*, 55:114–127.

[NIST 2019] NIST (2019). Glossary. *National Institute of Standards and Technology*.

[Nobles 2023] Nobles, C. (2023). Human factors in cybersecurity: academia's missed opportunity. *MWAIS 2023 Proceedings*.

[Peake 2003] Peake, C. (2003). Red teaming: the art of ethical hacking | sans institute.

[Pinheiro 2020] Pinheiro, P. P. (2020). Segurança digital: Proteção de dados nas empresas. *1ª edição. São Paulo, SP: Grupo GEN*.

[Piovesan et al. 2019] Piovesan, L. G., Silva, E. R. C., de Sousa, J. F., and Turibus, S. N. (2019). Engenharia social: Uma abordagem sobre phishing. *REVISTA CIENTÍFICA DA FACULDADE DE BALSAS*, 10(1):45–59.

[Reep-van den Bergh and Junger 2018] Reep-van den Bergh, C. M. and Junger, M. (2018). Victims of cybercrime in europe: a review of victim surveys. *Crime science*, 7(1):1–15.

[Riecken 1974] Riecken, H. W. (1974). Obedience to authority. an experimental view. stanley milgram. harper and row, new york, 1974. xx, 224 pp., illus. 10. *Science*, 184(4137):667–669.

[Robinson 2023] Robinson, N. (2023). Human factors security engineering: The future of cybersecurity teams. *EDPACS*, 67(5):1–17.

[Rogers et al. 2018] Rogers, T., Goldstein, N. J., and Fox, C. R. (2018). Social Mobilization. *Annual Review of Psychology*, 69(1):357–381.

[Rouse 2013] Rouse, M. (2013). What is socialbot? *WhatIs.com*.

[Ryan and Deci 2000] Ryan, R. M. and Deci, E. L. (2000). Self-determination theory and the facilitation of intrinsic motivation, social development, and well-being. *American Psychologist*, 55(1):68–78.

[Salahdine and Kaabouch 2019] Salahdine, F. and Kaabouch, N. (2019). Social engineering attacks: a survey. *Future Internet*, 11(4):89.

[Shaabany and Anderl 2018] Shaabany, G. and Anderl, R. (2018). Security by design as an approach to design a secure industry 4.0-capable machine enabling online-trading of technology data. In *2018 International Conference on System Science and Engineering (ICSSE)*, pages 1–5. IEEE.

[Shafahi et al. 2016] Shafahi, M., Kempers, L., and Afsarmanesh, H. (2016). Phishing through social bots on twitter. In *2016 IEEE International Conference on Big Data*, pages 3703–3712. IEEE.

[Shires 2018] Shires, J. (2018). Enacting expertise: Ritual and risk in cybersecurity. *Politics and Governance*, 6(2):31–40.

[Solano and Rocha 2019] Solano, E. and Rocha, C., editors (2019). *As direitas nas redes e nas ruas: a crise politica no Brasil*. Expressao Popular, Sao Paulo, 1a edicao edition. OCLC: on1126542066.

[Stoeckli et al. 2018] Stoeckli, E., Uebernickel, F., and Brenner, W. (2018). Exploring affordances of slack integrations and their actualization within enterprises-towards an understanding of how chatbots create value. In *Proceedings of the 51st Hawaii International Conference on System Sciences*.

[Tversky and Kahneman 1974] Tversky, A. and Kahneman, D. (1974). Judgment under Uncertainty: Heuristics and Biases: Biases in judgments reveal some heuristics of thinking under uncertainty. *Science*, 185(4157):1124–1131.

[Wilke and Mata 2012] Wilke, A. and Mata, R. (2012). Cognitive Bias. In *Encyclopedia of Human Behavior*, pages 531–535. Elsevier.

[Zimmermann and Renaud 2019] Zimmermann, V. and Renaud, K. (2019). Moving from a 'human-as-problem" to a 'human-as-solution" cybersecurity mindset. *International Journal of Human-Computer Studies*, 131:169–187. 50 years of the International Journal of Human-Computer Studies. Reflections on the past, present and future of human-centred technologies.

Capítulo 4

Ameaças e Vulnerabilidades em Open RAN: Desafios e Soluções

Diogo Menezes Ferrazani Mattos (UFF), Dianne Scherly Varela de Medeiros (UFF), Rodrigo de Souza Couto (UFRJ), Pedro Henrique Cruz Caminha (UFRJ), Lucas Airam Castro de Souza (UFRJ), Felipe Gomes Táparo (UFRJ), Guilherme Araujo Thomaz (UFRJ), João Vitor Valle (UFF), Franciele Batista de Oliveira (UFF), Miguel Elias Mitre Campista (UFRJ), Luís Henrique Maciel Kosmalski Costa (UFRJ), Igor Monteiro Moraes (UFF)

Resumo

A experiência com as redes de acesso via rádio atuais demonstra que o uso de arquiteturas monolíticas e de código fechado não favorece a customização de serviços nem a interoperabilidade dos componentes da rede. Tais obstáculos trazem concentração de mercado, dificuldade de inovação e maior custo de operação. Nesse sentido, a Rede de Acesso via Rádio Aberta (Open Radio Access Network – Open RAN) traz um modelo totalmente disruptivo que abre as interfaces de software e hardware para a desagregação de componentes, permitindo a entrada de novos participantes e serviços. As interfaces abertas permitem a "softwarização" da rede e a introdução de gerenciamento e controle baseado em inteligência artificial. A abertura, porém, tem um custo em segurança, tendo em vista a expansão da superfície de ataque da RAN. Este capítulo discute as ameaças e vulnerabilidades relativos à Open RAN e os desafios e oportunidades de pesquisa para o desenvolvimento de soluções de segurança em redes móveis de próxima geração.

4.1. Introdução

Estimativas do mercado global relacionado às redes 5G apontam uma taxa de crescimento anual de 21,1% e um volume de capital de aproximadamente 57 bilhões de dólares até 2030 [Research e Markets, 2022]. Esses números demonstram a importância das redes de acesso móvel, tanto pelos serviços de interconexão prestados quanto pelo ponto de vista econômico que mantém grandes empresas no setor de telecomunicações. Dessa forma, a concentração do mercado fornecedor de tecnologia de acesso sem fio móvel em poucos participantes, com o uso de soluções fechadas e monolíticas, não é conveniente em um ambiente altamente disputado e estratégico. Esse entendimento leva à tendência

de abertura das soluções empregadas nas redes de acesso móvel ao longo das gerações. Tais ações culminam na abertura de interfaces, que tem por objetivo promover a interoperabilidade de *hardware* e de *software* de múltiplos fabricantes, além de possibilitar a criação e inserção de novas funções à rede.

As Redes de Acesso via Rádio (*Radio Access Networks* – RANs) são compostas por um conjunto de componentes que interagem entre si para possibilitar a comunicação entre o equipamento do usuário (*User Equipment* – UE) e a rede de núcleo do sistema de comunicação móvel celular [Couto et al., 2023b, de Oliveira et al., 2023, Lopez et al., 2022]. Essa interação é intermediada pela Estação Rádio-Base (*Radio Base Station* – RBS) [Couto et al., 2023b], principal afetada pela abertura de interfaces de *hardware* e *software*. As condições necessárias para abertura de interfaces podem ser percebidas ao longo das diferentes gerações das redes móveis. Nas primeiras gerações, a RBS era implementada como um componente monolítico que acumulava funções de múltiplas camadas. Na terceira geração, houve a separação das funções de comunicação via rádio em funções de transmissão e recepção, executadas por uma nova RBS, chamada de NodeB, e em funções de gerenciamento de recursos de rádio e processamento relacionado aos usuários, executadas pelo Controlador da Rede via Rádio (*Radio Network Controller* – RNC) [Arnaz et al., 2022]. A quarta geração agregou funcionalidades do RNC à nova RBS, chamada de *Evolved Node B* (eNB), extinguindo a entidade de controle separada. Tanto a terceira quanto a quarta geração desagregaram as funções da RBS em Unidade de Rádio Remota (*Remote Radio Head* – RRH), que executava as funções de rádio, e em Unidade de Banda Base (*BaseBand Unit* – BBU), que realizava o processamento de sinal em banda base. Tal desagregação permitiu que a BBU e a RRH fossem fisicamente separadas, sendo a RRH mantida mais próxima da RBS. Essa arquitetura introduz a RAN distribuída (*Distributed RAN* – D-RAN) [Brik et al., 2022] utilizada nas gerações atuais.

A quinta geração das redes celulares deu um passo a mais na direção da desagregação de funcionalidades, propondo a separação da RNB, eNB do 4G que evoluiu para gNB (*Next Generation Node* B), em três unidades: central (*Central Unit* -– CU), distribuída (*Distributed Unit* -– DU) e de rádio (*Radio Unit* – RU) [Polese et al., 2023]. As funcionalidades das camadas física, de enlace e de rede são então divididas entre essas três unidades, que ainda mantêm um projeto composto por componentes monolíticos com interfaces fechadas. Tal característica ainda favorece a concentração do mercado em poucos fabricantes, criando obstáculos tecnológicos tanto para customização da rede quanto para interoperabilidade das unidades. A arquitetura desagregada da rede e os entraves das soluções fechadas, portanto, motivaram o movimento atual em direção ao uso de interfaces abertas. Nesse sentido, a *O-RAN Alliance*[1] lidera a proposta de uma arquitetura e um conjunto de interfaces que concretizem a RAN aberta (*Open* RAN) [O-RAN Working Group 1, 2023]. A arquitetura O-RAN para a RAN aberta segue os seguintes princípios fundamentais: desagregação dos componentes da RAN, controle inteligente, virtualização e interfaces abertas [Polese et al., 2023]. A arquitetura O-RAN define um novo padrão industrial para redes móveis de quinta geração e além (*Beyond 5G* - B5G). As interfaces abertas e padronizadas permitem a interoperabilidade entre equipamentos de diferentes fornecedores, oferecendo flexibilidade de rede a um custo reduzido

[1]Disponível em `https://www.o-ran.org/`.

de capital e operação. A arquitetura O-RAN combina os benefícios da execução de funções de rede em *software* e inteligência artificial para tornar a operação de dispositivos e da rede de acesso mais eficiente e segura.

A arquitetura O-RAN introduz componentes, interfaces e tecnologias que possibilitam a participação de novos atores e viabilizam novos modelos de negócios. A abertura das interfaces e desagregação dos componente trazem benefícios para a segurança da rede. Enquanto as interfaces abertas O-RAN introduzem transparência à implementação dos componentes, tornando mais viável o alinhamento com os padrões de segurança e as melhores práticas, a desagregação dos componentes aumenta a agilidade e a adaptabilidade. Além disso, a O-RAN suporta o uso de microsserviços hospedados em nuvens, permitindo empregar mecanismos de segurança nativos da nuvem, como isolamento de recursos de *hardware*, reconfiguração automática e automação de testes de segurança, que reforçam o gerenciamento de vulnerabilidades de código aberto e a configuração de segurança. No entanto, novas tecnologias implicam aumento da superfície de ataque, trazendo desafios de segurança e riscos associados às novas interfaces e componentes específicos da O-RAN, às técnicas de virtualização e conteinerização, ao suporte a código-fonte aberto e ao suporte de modelos de aprendizado de máquina de terceiros. O *software* de código-fonte aberto aumenta a necessidade de práticas de desenvolvimento seguras nas comunidades de código aberto. A incorporação de Inteligência Artificial (IA) na RAN apresenta riscos como a falta de transparência e explicabilidade dos modelos de IA, que pode levar a ações imprevisíveis na RAN [de Oliveira et al., 2023]. O acesso não-autenticado e não-autorizado aos componentes da arquitetura pode ser alcançado através de diferentes interfaces. Tal acesso depende do projeto de *hardware* e *software* da O-RAN e de como as diferentes funções são desacopladas dentro da própria arquitetura. Assim, interfaces de gerenciamento devem seguir as melhores práticas de segurança do setor, incluindo criptografia forte, autenticação mútua, controle de acesso, registro robusto e validação de entrada.

Os componentes da O-RAN podem estar vulneráveis caso estejam desatualizados por falta de gerenciamento de remendos (*patches*), tenham um projeto de arquitetura deficiente em termos de segurança, não tenham proteção apropriada, usem uma função ou protocolo que não sejam seguros ou que não sejam mais necessários. Nesse caso, um atacante pode tanto injetar *malwares* e manipular *software* existente, quanto manipular parâmetros ou reconfigurar os componentes O-RAN para reduzir seu desempenho e desativar os recursos de segurança. Esse último com o objetivo de espionar ou interceptar planos de controle e de dados, atingir interfaces externas, atacar uma rede para disparar um ataque de negação de serviço mais amplo ou roubar chaves privadas desprotegidas, certificados e valores de *hash*. Ademais, os componentes O-RAN podem ser componentes de *software* que fornecem funções de rede, sendo possivelmente vulneráveis a falhas como, por exemplo, o estouro de *buffer* para execução de comandos arbitrários. A fim de aumentar a segurança da O-RAN, microsserviços integrados aos Controladores Inteligentes da RAN (RAN *Intelligent Controllers* – RICs) podem empregar dados e análises em tempo real, avaliando a integridade da RAN e reforçando os recursos de segurança e gerenciamento. Para tanto a plataforma de *software* O-RAN segue as referências de segurança do setor de telecomunicações e os requisitos de garantia de segurança do 3GPP. Contudo, é importante destacar que a implementação de controles de segurança requer

atenção devido às rigorosas demandas de latência [NIS Cooperation Group, 2022] das redes móveis de quinta geração e além.

Este capítulo aborda os desafios de segurança da Open RAN, incluindo seus conceitos básicos e soluções. O objetivo é apresentar as principais ameaças, vulnerabilidades e vetores de ataque, e discutir os desafios de pesquisa para prover segurança a essas redes. Inicialmente, apresenta-se a arquitetura de referência O-RAN e os requisitos de segurança especificados pela O-RAN Alliance. Em seguida, as principais ameaças e vulnerabilidades da O-RAN são explicadas. Este capítulo ainda introduz as principais técnicas e estratégias para defesa e mitigação de ataques às RANs. Discute-se também os principais ataques baseados em aprendizado de máquina para RANs. Por fim, os desafios e oportunidades de pesquisa para o desenvolvimento de mecanismos de segurança e preservação da privacidade em redes móveis de próxima geração são elencados.

4.2. A Arquitetura de Referência O-RAN e os Requisitos de Segurança

A arquitetura O-RAN é proposta pela *O-RAN Alliance* para a RAN aberta com o objetivo de oferecer maior flexibilidade na implantação das RANs e facilidade para gerenciamento e orquestração dos diversos serviços e componentes da rede. A arquitetura O-RAN permite às operadoras de telecomunicações independência de soluções monolíticas de um único fornecedor, devido ao suporte à desagregação, virtualização e "softwarização" de componentes que são conectados através de interfaces abertas padronizadas. Essa padronização permite a interoperabilidade entre equipamentos de fornecedores distintos e possibilita às operadoras utilizarem *hardware* não proprietário. Ademais, a arquitetura O-RAN permite aproveitar os princípios das soluções nativas em nuvem e integrar inteligência no controle da RAN [Polese et al., 2023]. Para tanto, a arquitetura segue a desagregação das três unidades funcionais da gNodeB proposta pelo 3GPP. Essas unidades se interconectam e estão conectadas a outros componentes da arquitetura O-RAN por meio de interfaces abertas.As três unidades definidas são nós lógicos, denominados Unidade Central O-RAN (O-RAN *Central Unit* – O-CU), Unidade Distribuída O-RAN (O-RAN *Distributed Unit* – O-DU) e Unidade de Rádio O-RAN (O-RAN *Radio Unit* – O-RU).

A inovação da arquitetura O-RAN em relação à RAN tradicional são os Controladores Inteligentes da RAN (*RAN Intelligent Controllers* – RICs). Os RICs introduzem componentes programáveis que podem executar rotinas de otimização com um laço de controle fechado e orquestrar serviços na RAN de forma eficiente, possuindo uma visão abstrata e centralizada da rede. Para isso, os RICs acessam informações de medidas de desempenho e contexto provenientes da RAN e de fontes externas à RAN, que podem ser utilizadas para produzir modelos de aprendizado de máquina com o objetivo de criar políticas e ações de controle sobre a RAN [O-RAN Working Group 1, 2023]. Dessa forma, é possível automatizar procedimentos de otimização da rede com foco no fatiamento dos recursos, balanceamento de carga e mudança de células (*handovers*) [Polese et al., 2023]. As especificações da O-RAN Alliance [O-RAN Working Group 3, 2023a, O-RAN Working Group 2, 2021] descrevem requisitos e funcionalidades de diferentes componentes dos RICs para que as implementações, em conformidade com o padrão, forneçam os mesmos conjuntos de serviços e sejam interoperáveis.

Apesar de trazer benefícios para as RANs, a nova arquitetura sofre com vulnerabilidades que precisam ser tratadas. Por um lado, a desagregação da arquitetura facilita a atualização e a introdução de funções de rede por utilizar redes definidas por *software* (*Software Defined Networking* – SDN) e virtualização de funções de rede (*Network Function Virtualization* – NFV). Por outro lado, a "softwarização" da RAN traz desafios de segurança relacionados a arquiteturas virtualizadas, englobando riscos de segurança em nuvem e contêineres. A desagregação também aumenta a superfície de ataque da RAN devido aos novos componentes criados [Soltani et al., 2023]. A separação da O-DU e da O-RU, por exemplo, expõe uma das interfaces abertas padronizadas, a Open Fronthaul [Dik e Berger, 2023]. O uso de *software* de código-fonte aberto aumenta a dependência de práticas de desenvolvimento seguro, ficando exposto a vulnerabilidades de "dia zero" (*zero-day*). Além disso, com a introdução dos RICs e a consequente dependência de fluxos de aprendizado de máquina e inteligência artificial (*Machine Learning/Artificial Intelligence* – ML/AI) levam a vulnerabilidades não mapeadas inicialmente. A possibilidade de diversas entidades poderem desenvolver aplicações que operam nos RICs, consumindo dados da RAN, é um fator crítico, visto que essas aplicações podem inserir vulnerabilidades. Existe, ainda, o aumento da superfície de ataque devido ao crescimento exacerbado de dispositivos conectados, exigindo que a RAN seja protegida contra dispositivos que foram violados. É importante destacar que nem todos os problemas citados são exclusivos da RAN aberta, mas se tornam mais significativos devido à adição de novos componentes não existentes nas RANs contemporâneas, de novas interfaces abertas e ao uso da opção de divisão 7.2x [O-RAN Working Group 11, 2023c]. Nesse contexto, a O-RAN Alliance especifica requisitos de segurança e pilhas de protocolos para configuração e criptografia que devem ser usados nas interfaces O-RAN para garantir confidencialidade, autenticidade, integridade, autorização e proteção contra ataques de repetição [O-RAN Working Group 11, 2023c]. O objetivo é adequar a RAN aberta ao padrão de qualidade e segurança que os sistemas de telecomunicações exigem, de forma que o risco à segurança seja mitigado na implantação da RAN aberta pelos Provedores de Serviços de Comunicação (*Communication Service Providers* – CSPs).

Esta seção apresenta primeiramente uma visão geral da arquitetura de referência O-RAN proposta pela O-RAN Alliance, destacando em seguida os requisitos de segurança especificados pela aliança.

4.2.1. Arquitetura O-RAN

A desagregação da gNodeB tem como base a Opção de Divisão 7.2x (*Split Option* 7.2x) definida no conjunto de especificações 3GPP *New Radio* (3GPP NR), que separa as funcionalidades de camada física para serem implementadas em duas unidades distintas. Os componentes desagregados estão representados na Figura 4.1. A O-RU passa a ser responsável apenas pelo processamento de sinais de radiofrequência e pelas funcionalidades de camada física inferior (*Low*-PHY), como transformada rápida de Fourier direta e inversa, remoção e adição de prefixo cíclico e conformação de feixe (*beamforming*). A O-DU é responsável pelas funcionalidades de camada física superior (*High*-PHY), como embaralhamento, modulação, mapeamento de camada e mapeamento de elementos de recursos. Dessa forma, a O-RU torna-se uma unidade de baixo custo e de fácil implantação, reduzindo a largura de banda necessária para comunicação no *fronthaul* e o atraso.

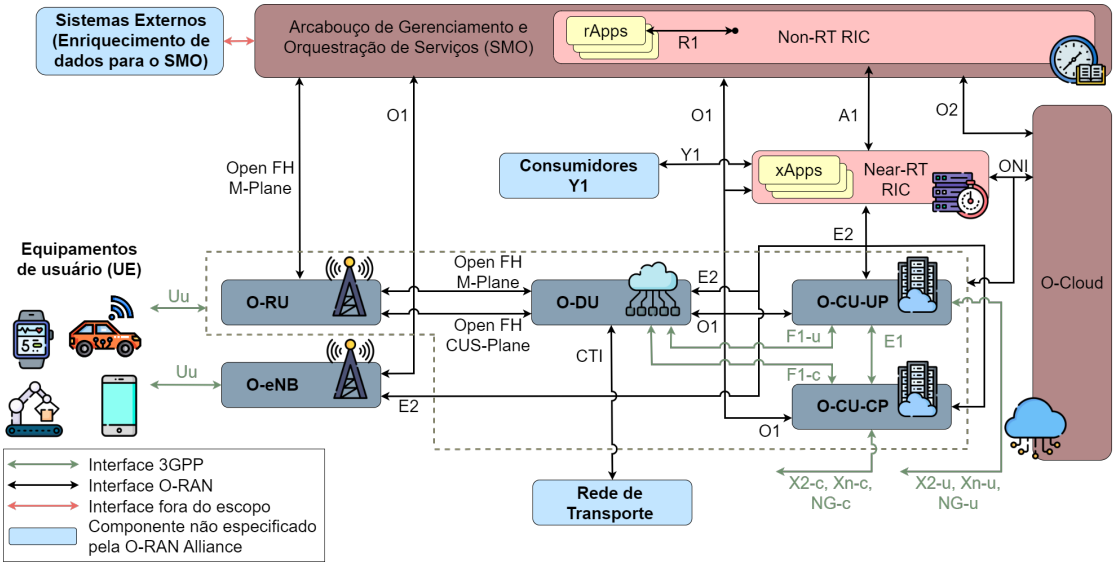


Figura 4.1. Componentes da arquitetura O-RAN. A arquitetura divide as funcionalidades da estação rádio base em três componentes, O-CU, O-DU e O-RU, além de definir o SMO, os RICs e a O-Cloud. Os componentes da arquitetura interagem por meio de interfaces especificadas pela O-RAN Alliance (em preto) e pelo 3GPP (em verde). Adaptado de [O-RAN Working Group 1, 2023], com ícones de Freepik (flaticon.com).

A O-DU também é responsável pelas funcionalidades de camada de enlace, implementando a subcamada de Controle de Acesso ao Meio (*Medium Access Control* – MAC) e a subcamada de Controle de Enlace de Rádio (*Radio Link Control* – RLC). As operações realizadas pela camada física superior e pelas subcamadas MAC e RLC devem ser fortemente sincronizadas, porque a subcamada MAC gera Blocos de Transporte (*Transport Blocks* – TBs) para serem enviados pela camada física usando dados que são enfileirados pela subcamada RLC. A O-CU é responsável por funcionalidades como controle de mobilidade, compartilhamento da RAN, gerenciamento de sessão e transferência de dados do usuário. Para isso, a O-CU implementa a camada de Controle de Recursos de Rádio (*Radio Resource Control* – RRC), que gerencia o ciclo de vida das conexões; a camada de Protocolo de Adaptação de Serviços de Dados (*Service Data Adaptation Protocol* – SDAP), que gerencia a qualidade de serviço dos fluxos de tráfego; e a camada de Protocolo de Convergência de Pacotes de Dados (*Packet Data Convergence Protocol* – PDCP), responsável pela reordenação de pacotes, tratamento de pacotes duplicados, criptografia dos dados para a interface aérea, dentre outras funções [Arnaz et al., 2022]. A O-CU é formada por um componente lógico para o plano de controle (O-CU *Control Plane* – O-CU-CP) e outro para o plano de usuário (O-CU *User Plane* – O-CU-UP) para permitir a implementação de funcionalidades distintas em diferentes locais da rede e plataformas de *hardware* [O-RAN Working Group 1, 2023].

A O-RAN Alliance especifica o RIC não tempo-real (*Non-Real-Time* RIC – Non-RT RIC) e o RIC quase tempo-real (*Near-Real-Time* RIC – Near-RT RIC) [O-RAN Working Group 1, 2023], apresentados na Figura 4.1. Enquanto o Non-RT RIC fornece operação inteligente e otimização da RAN em uma escala de tempo maior do que 1 segundo, o Near-RT RIC opera em escala de tempo entre 10 milissegundos e 1 segundo. Para operar nessa escala de tempo, o Near-RT RIC é implan-

tado na borda da rede, estando conectado às O-DUS, O-CUs e O-eNBs. Uma O-eNB é uma eNB ou *Next-Generation* eNB (gNodeB) que suporta as interfaces E2 e O1 [O-RAN Working Group 1, 2023].

O Non-RT RIC suporta a execução de rApps, aplicações de terceiros que fornecem serviços de valor agregado para facilitar ou aprimorar as operações da RAN [Polese et al., 2023, Arnaz et al., 2022]. Por sua vez, o Near-RT RIC suporta a execução de xApps, que são microsserviços usados para gerenciar recursos de rádio através de interfaces padronizadas e modelos de serviços [Polese et al., 2023, Arnaz et al., 2022], que também podem ser fornecidas por terceiros. Por meio das xApps e rApps, é possível realizar o controle inteligente da RAN. Enquanto as xApps implementam uma lógica personalizada que pode utilizar informações de telemetria da RAN e enviar ações de controle para serem executadas por elementos da RAN, as rApps permitem a implantação de serviços de valor agregado a fim de suportar e facilitar a otimização e operação da RAN, oferecendo serviços de orientação de políticas, enriquecimento de informação, gerenciamento de configuração e análise de dados [Polese et al., 2023].

Outro componente fundamental da arquitetura O-RAN, responsável pelo gerenciamento do domínio da RAN, é o componente de Orquestração e Gerenciamento de Serviços (*Service Management and Orchestration* – SMO). O SMO fornece uma interface com as funções de rede O-RAN, seguindo o modelo FCAPS (*Fault, Configuration, Accounting, Performance, Security*). O SMO é composto pelo Non-RT RIC e por um conjunto de funções e serviços para gerenciamento, orquestração e automação da RAN, tendo um papel semelhante à entidade de Gerenciamento e Orquestração (*Management and Orchestration* – MANO) definida na arquitetura NFV [Thiruvasagam et al., 2023].

O Near-RT RIC é formado por [O-RAN Working Group 3, 2023a]:

- terminações de interfaces (O1, A1, Y1, E2), que permitem a comunicação por meio das interfaces conectadas ao Near-RT RIC;
- APIs do Near-RT RIC para as xApps, um conjunto de APIs que fornecem serviços à plataforma Near-RT RIC, definindo explicitamente os possíveis tipos de fluxos de informação e modelos de dados. São definidas APIs relacionadas às interfaces A1 e E2, APIs de gerenciamento, APIs para a camada de compartilhamento de dados e as APIs de ativação;
- infraestrutura de mensagens internas, responsável por interconectar os componentes internos do Near-RT RIC;
- componente para mitigação de conflitos, que lida com possíveis conflitos entre requisições de configurações geradas por diferentes xApps;
- gerenciador de assinaturas de xApps, que permite que as xApps se conectem a funções expostas pela interface E2, controlando também o acesso individual das xApps às mensagens nessa interface;
- componente de funções de gerenciamento, fornece funções de gerenciamento de falha, configuração e desempenho, registro e rastreio, coleta de métricas para capturar, monitorar e obter o estado dos componentes internos do Near-RT RIC;

- componente de segurança, responsável por prevenir a exploração abusiva de informações por xApps e o controle de funções da RAN com intenção maliciosa;
- suporte ao fluxo de trabalho de AI/ML, que oferece treinamento de modelos nas xApps e preparação dos dados para consumo pelas xApps;
- função de repositório de xApps, permite a seleção de xApps para roteamento de mensagens da interface A1 com base nas políticas adotadas, e controle de acesso das xApps ao serviço de enriquecimento de informação com base nas políticas da operadora;
- Base de Informações de Rede (*Network Information Base* – NIB) e camada de compartilhamento de dados, que permitem às xApps e ao Near-RT RIC modificarem informações armazenadas no banco de dados e obterem informações sobre os componentes conectados à interface E2 e sobre os UEs; e
- API de ativação, que suporta a operação da API do Near-RT RIC, como registro, autenticação, descoberta de APIs, dentre outras operações.

O Non-RT RIC implementa um subconjunto de funcionalidades do arcabouço SMO com o objetivo de realizar a otimização inteligente da RAN por meio de orientação baseada em políticas, gerenciamento de modelos de aprendizado de máquina e enriquecimento de informação para o Near-RT RIC. Dessa forma, o Non-RT RIC é responsável pelos procedimentos de orquestração, gerenciamento e automação usados para monitorar e controlar os componentes da RAN. A comunicação entre os componentes do SMO e do Non-RT RIC é feita por meio de uma infraestrutura de mensagens interna. O Non-RT RIC oferece dois serviços de gerenciamento e orquestração de alto nível, permitindo que a arquitetura O-RAN seja suficientemente flexível para que o comportamento de cada componente da rede e funcionalidade possa ser ajustado em tempo real, atendendo aos objetivos e intenções das operadoras. O primeiro serviço é o gerenciamento de rede baseado em intenção, que permite às operadoras especificarem intenções utilizando uma linguagem de alto nível. A intenção é automaticamente analisada pelo Non-RT RIC que determina a política e o conjunto de rApps e xApps que devem ser implantadas e executadas para satisfazer as políticas. O segundo serviço é a orquestração inteligente, que permite coordenar e orquestrar as diferentes xApps e rApps que executam em diferentes RICs e locais da rede. Assim, o Non-RT RIC é responsável pela orquestração da inteligência da rede para garantir que as aplicações selecionadas sejam adequadas para satisfazer as intenções da operadora e atender aos requisitos impostos. Além disso, o Non-RT RIC deve garantir que as aplicações sejam instanciadas no local apropriado para garantir o controle sobre os elementos da RAN especificados na intenção, sejam alimentadas com dados relevantes, e sejam robustas o suficiente para não gerarem conflitos por condição de corrida entre as aplicações [Polese et al., 2023]. As rApps devem ser capazes de se comunicarem por meio da interface R1, interna ao Non-RT RIC, para que o arcabouço Non-RT RIC possa oferecer serviços às rApps [O-RAN Working Group 1, 2023].

Além das rApps, o Non-RT RIC é composto pelo arcabouço Non-RT RIC, formado por funções ancoradas e não ancoradas no arcabouço. Dentre as não ancoradas estão [O-RAN Working Group 2, 2023]:

- as funções de gerenciamento e exposição de serviços da interface R1, que permitem registrar e descobrir serviços, enviar notificações sobre serviços, autenticar e autorizar acessos, dentre outras funções;
- a função de gerenciamento de rApp, responsável pelo gerenciamento das rApps no contexto do arcabouço, permitindo a configuração das aplicações, acesso a informações de desempenho e falha, dentre outras funções;
- as funções relacionadas à interface A1, que fornecem acesso às funcionalidades oferecidas pela interface A1 como descoberta de políticas suportadas pela interface e criação, atualização e remoção de políticas, e suporte ao enriquecimento de informação;
- as funções de gerenciamento e exposição de dados, que permitem consumir dados do arcabouço Non-RT RIC e do SMO;
- as funções do fluxo de trabalho AI/ML, que permitem acessar serviços como treinamento de modelos de acordo com pré-requisitos estabelecidos, registro e descoberta de modelos, atualização e armazenamento de modelos, monitoramento do desempenho de modelos implantados; e
- as terminações externas, que permitem ao SMO e ao arcabouço Non-RT RIC trocarem mensagens com entidades externas por meio de interfaces que estejam fora do escopo da arquitetura O-RAN.

As funções ancoradas no arcabouço Non-RT RIC são [O-RAN Working Group 2, 2023]:

- a terminação da interface R1, que possibilita a troca de mensagens entre o arcabouço Non-RT RIC e as rApps para acessar os serviços da interface R1;
- a terminação da interface A1, que permite a troca de mensagens entre o Non-RT RIC e o Near-RT RIC pela interface A1; e
- outras funções do Non-RT RIC, como funções específicas da RAN para gerenciamento de fatias de rede.

A O-Cloud combina nós físicos, componentes de *software* e funcionalidades de gerenciamento e orquestração com o intuito de desacoplar componentes de *hardware* e *software*. A O-Cloud permite o compartilhamento de *hardware* entre diferentes inquilinos e automatiza a implantação e instanciação de funcionalidades da RAN, como Funções Virtuais de Rede (*Virtual Network Functions* – VNFs) encontradas na O-CU e as rApps do Non-RT RIC [Arnaz et al., 2022, Polese et al., 2023]. A Tabela 4.1 resume as funções dos componentes definidos pela O-RAN Alliance na arquitetura O-RAN.

O objetivo das interfaces abertas especificadas pela O-RAN Alliance é padronizar e flexibilizar o acesso aos componentes da RAN, permitindo a conexão entre os diversos componentes da arquitetura de forma interoperável [Arnaz et al., 2022]. A arquitetura O-RAN utiliza as interfaces A1, E2, *Open FrontHaul* (Open FH), O1, O2, R1, ONI (*O-Cloud*

Tabela 4.1. Resumo das funções dos componentes da arquitetura O-RAN.

Componentes	Descrição
SMO	Hospeda o Non-RT RIC e é responsável pelo monitoramento e orquestração da RAN
Non-RT RIC	Suporta rApps, atua em laços de controle maiores que 1 s
Near-RT RIC	Suporta xApps, atua em laços de controle entre 10 ms e 1 s
O-CU	Implementa as camadas superiores da pilha 3GPP: RRC, SDAP, PDCP
O-CU-CP	Componente lógico do plano de controle da O-CU
O-CU-UP	Componente lógico do plano de usuário da O-CU
O-DU	Implementa funções da High-PHY e subcamadas MAC e RLC
O-RU	Implementa funções da Low-PHY e de processamento de sinais de radiofrequência
O-eNB	Estação rádio base 4G/LTE compatível com O-RAN
O-Cloud	Plataforma de computação em nuvem híbrida formada por um conjunto de recursos computacionais e infraestrutura virtualizados reunidos em um ou mais centros de dados

Notification Interface), Y1 e CTI (*Cooperative Transport Interface*) padronizadas pela O-RAN Alliance, e as interfaces E1, F1, X2, Xn, NG e UU, padronizadas pelo 3GPP. Em conjunto com a interface Open FH, as interfaces 3GPP permitem desagregar a gNodeB. Por meio das interfaces padronizadas pela O-RAN Alliance, os RICs obtêm informações sobre a RAN usadas para definir as ações de controle e automação a serem executadas na RAN [Polese et al., 2023]. A Tabela 4.2 resume as interfaces da arquitetura O-RAN e suas terminações.

A interface O1 conecta o SMO aos RICs, à O-eNB, à O-CU e à O-DU, permitindo o gerenciamento e orquestração das funcionalidades de rede [O-RAN Working Group 10, 2023]. A interface A1 é usada para comunicação entre o Non-RT RIC e o Near-RT RIC, permitindo que o Non-RT RIC envie para o Near-RT RIC orientações baseadas em políticas, gerencie modelos de aprendizado de máquina e envie informações para o Near-RT RIC com o objetivo de otimizar a RAN. Os modelos de aprendizado implantados no Near-RT RIC para otimizar a RAN podem ser refinados com informações enriquecidas fornecidas pelo Non-RT RIC ao Near-RT RIC. A comunicação é feita por meio de mecanismos padronizados baseados em uma sintaxe específica que pode expressar intenções de alto nível e políticas. Dessa forma, permite-se a implementação do controle de não tempo-real e de políticas e modelos inteligen-

Tabela 4.2. Resumo das interfaces O-RAN e 3GPP presentes na RAN aberta.

Interface	Terminação	Tipo
O1	Non-RT RIC, O-eNB, Near-RT RIC, O-CU-CP , O-CU-UP, O-DU e O-RU	O-RAN
A1	Non-RT RIC e Near-RT RIC	O-RAN
E2	Near-RT RC e Nós E2	O-RAN
Open FH	O-DU e O-RU	O-RAN
O2	SMO e O-Cloud	O-RAN
ONI	O-Cloud, Near-RT RIC, O-RU, O-DU e O-CU	O-RAN
Y1	Near-RT RIC e consumidor Y1	O-RAN
CTI	O-DU e nó da rede de transporte	O-RAN
R1	rApp e arcabouço Non-RT RIC	O-RAN
E1	O-CU-CP e O-CU-UP	3GPP
F1	O-CU-CP, O-CU-UP e O-DU	3GPP
X2, Xn, NG	O-CU-CP, O-CU-UP e funções 3GPP	3GPP
Uu	UE, O-eNB, O-RU, O-DU e O-CU	3GPP

tes no Near-RT RIC [Polese et al., 2023]. O Near-RT RIC usa a interface A1 com o Non-RT RIC também para descoberta, requisição e entrega de informações enriquecidas, além de descoberta de informações de enriquecimento provenientes de fontes externas [Polese et al., 2023]. A interface E2 permite a comunicação entre o Near-RT RIC e os componentes lógicos denominados Nós E2, isto é, os componentes que são pontos de terminação para essa interface. Assim, a O-CU, a O-DU e a O-eNB compõem os Nós E2. Por meio da interface E2 são transmitidos dados de telemetria da RAN e as respostas de controle do Near-RT RIC [Polese et al., 2023]. O Near-RT RIC pode executar ações sobre os Nós E2, como monitoramento, suspensão, parada e controle do comportamento do nó [O-RAN Working Group 3, 2023b].

A interface Open FH possibilita a interação entre O-RUs e O-DUs, permitindo o controle das operações da O-RU a partir da O-DU e a distribuição das funcionalidades de camada física entre a O-DU e a O-RU. Como deve existir forte sincronização entre as duas unidades, a interface Open FH oferece uma referência de relógio compartilhada [Polese et al., 2023]. Para ofertar esses serviços, a interface Open FH inclui quatro planos: controle (C-Plane), usuário (U-Plane), gerenciamento (M-Plane) e sincronização (S-Plane). A interface O2 conecta o SMO à O-Cloud para suportar as funcionalidades que executam na nuvem. Essa interface oferece funções tanto para gerenciar a infraestrutura em nuvem quanto para gerenciar implantações na infraestrutura [O-RAN Working Group 6, 2023]. A O-RAN Alliance considera adotar para a interface O2 padrões e soluções abertas, como os padrões da *European Telecommunications Standards Institute* (ETSI) para *Network Function Virtualization* (NFV), interfaces baseadas em serviços do 3GPP e os projetos Kubernetes, OpenStack e ONAP/OSM [Polese et al., 2023]. A ONI permite que componentes da arquitetura implantados na O-Cloud recebam notificações com informações críticas de eventos e estado da O-Cloud que outrora seriam conhecidos apenas pela infraestrutura de nuvem. A interface CTI permite a comunicação com nós da rede de transporte para controle dinâmico da alocação de largura de banda quando se utiliza uma rede de transporte ponto-a-multiponto. A interface Y1 permite o consumo de informações de análise (*analytics*) do Near-RT RIC por entidades que estão dentro ou fora de um domínio de confiança PLMN (*Public Land Mobile Network*), denominadas consumidores Y1.

As interfaces E1, F1, X2, Xn, NG e Uu possibilitam a interoperabilidade entre componentes da RAN aberta e componentes herdados de outras gerações da RAN. A interface E1 permite realizar a conexão entre O-CU-CP e O-CU-UP. A interface F1 conecta elementos da O-DU e O-CU para troca de informação sobre o compartilhamento de recursos de rádio e sobre outros estados da rede. As interfaces X2 e Xn ajudam com a interoperabilidade entre nós de diferentes gerações. A interface NG conecta nós 5G à rede de núcleo quando está operando no modo *standalone*. Por fim, a interface Uu permite a conexão dos UEs às O-eNBs e às O-RUs [O-RAN Working Group 1, 2023].

4.2.2. Requisitos de Segurança

As especificações de segurança da O-RAN Alliance buscam alcançar os objetivos de uma arquitetura de "confiança zero" (*Zero-Trust Architecture* – ZTA) [O-RAN Working Group 1, 2023]. A confiança zero (*Zero Trust* – ZT) é um paradigma de cibersegurança focado na proteção de recursos e tem como premissa nunca

confiar em nenhum dispositivo ou usuário, sejam eles internos ou externos ao perímetro de segurança definido. Assim, todos os dispositivos e usuários devem ser avaliados antes de terem acesso a qualquer recurso [Rose et al., 2020]. Portanto, no paradigma ZT, mesmo que um usuário ou dispositivo esteja autenticado em um sistema, não há garantia de que terá autorização a acessar os recursos daquele sistema. Cada solicitação de acesso a um recurso é autorizada e monitorada individualmente durante o período de acesso para verificar a conformidade com as regras da política de segurança, autenticação e autorização iniciais [Ramezanpour e Jagannath, 2022]. Assim, ao aplicar ZT à RAN, não há confiança implícita de um usuário ou recurso, mesmo levando em consideração a localização física, localização de rede ou posse do recurso.

A ZTA é uma arquitetura baseada no paradigma ZT, permitindo aplicar a confiança zero no sistema de forma fim-a-fim, ou seja, abrangendo identidade (humana e não-humana), credenciais, acesso de gerenciamento, operação, pontos de terminação (*endpoints*), ambientes de hospedagem e interconexões de infraestrutura. Tanto as ameaças internas quanto as externas devem ser consideradas na ZTA e, dessa forma, seu uso na RAN aberta reduz os riscos associados à superfície de ataque aumentada [O-RAN Working Group 1, 2023]. A ZTA é definida na especificação NIST 800-27 e deve incluir suporte para uma Infraestrutura de Chave Pública (*Public Key Infrastructure* – PKI) com autenticação mútua baseada em certificados. A especificação também prevê a utilização de ZTA em infraestrutura de rede, com um controlador capaz de configurar e reconfigurar a rede de acordo com a concessão de acesso a um usuário ou dispositivo [Rose et al., 2020].

A arquitetura O-RAN segue os princípios de segurança do 3GPP e as melhores práticas da indústria, trabalhando em direção ao ZT como princípio orientador para que a RAN aberta ofereça o nível de segurança esperado pelos operadores e usuários das redes móveis celulares. A O-RAN Alliance especifica requisitos de segurança para cada componente e interface, com a intenção de proteger ativos críticos. A segurança na arquitetura é fundamentada em criptografia, certificados X.509v3 e IKEv2 (*Internet Key Exchange version 2*) e nos protocolos mTLS (*mutual Transport Layer Security*), TLS, IPsec (*Internet Protocol Security*), SSH (*Secure Shell*), DTLS (*Datagram Transport Layer Security*) e CMPv2(*Certificate Management Protocol version 2*). Os requisitos de segurança são agrupados em três categorias: (i) funções de rede e aplicações, (ii) interfaces abertas e (iii) requisitos transversais [O-RAN Working Group 11, 2023c], discutidas a seguir.

4.2.2.1. Funções de Rede e Aplicações

Na primeira categoria incluem-se todos os componentes da arquitetura O-RAN e as aplicações que executam nos RICs. De forma geral, exige-se que esses componentes e aplicações suportem a autenticação e a autorização de funções internas e de sistemas externos e sejam capazes de se recuperarem de ataques DDoS (*Distributed Denial of Service*) massivos que cheguem por uma interface interna ou externa. As comunicações internas e externas devem suportar autenticação mútua, confidencialidade, integridade e proteção contra reprodução. A Tabela 4.3 apresenta uma lista não exaustiva dos requisitos de segurança especificados para os componentes da O-RAN, exceto O-CU, O-DU, O-RU e O-eNB [O-RAN Working Group 11, 2023c, O-RAN Working Group 11, 2023b,

Abdalla e Marojevic, 2023]. Esses quatro componentes devem suportar os mesmos requisitos de segurança especificados para CU, DU, RU e O-eNB definidos pelo 3GPP nas especificações TS 33.501 e TS 33.401. A Tabela 4.4 mostra os protocolos utilizados para suportar os requisitos.

Tabela 4.3. Requisitos de segurança para componentes da arquitetura O-RAN.

Requisito	Near-RT RIC	xApp	Non-RT RIC	rApp	SMO	O-Cloud
Armazenamento seguro						✓
Atualização segura					✓	✓
Autenticação multifator					✓	✓
Autenticação mútua	✓	✓	✓	✓	✓	✓
Auto-configuração segura	✓		✓	✓		✓
Autorização	✓		✓	✓	✓	✓
Boot seguro	✓		✓	✓		✓
Capacidade de recuperação e *backup*	✓	✓	✓	✓	✓	✓
Computação em nuvem segura	✓	✓	✓	✓		
Comunicação confiável						✓
Confidencialidade	✓					✓
Confidencialidade de registros					✓	✓
Controle de acesso		✓	✓	✓	✓	✓
Criptografia	✓	✓	✓		✓	✓
Gerenciamento de chaves		✓	✓	✓		✓
Gerenciamento de segurança de software de código aberto	✓		✓			
Integridade de registros					✓	
Isolamento robusto	✓	✓		✓		✓
Monitoramento contínuo	✓			✓	✓	✓
PKI		✓	✓	✓		
Privacidade	✓		✓	✓	✓	✓
Proteção contra reprodução						✓
Recuperação contra ataques DDoS	✓		✓	✓	✓	
Registro contínuo	✓			✓	✓	✓
Transferência segura de registros					✓	
Tratamento de vulnerabilidades contínuo	✓			✓	✓	✓
Virtualização segura	✓	✓	✓	✓		✓

Tabela 4.4. Protocolos de segurança recomendados pela O-RAN Alliance para componentes da arquitetura O-RAN. (MFA: *Multi-Factor Authentication*)

Requisito	Near-RT RIC	Non-RT RIC	SMO	O-Cloud	O-DU	O-RU	rApps	xApps
Autenticação	TLS, mTLS, X.509v3, IPsec, IKEv2		mTLS, X.509v3, TLS, PSK	TLS, mTLS, X.509v3, MFA	802.1X	802.1X		TLS, mTLS, X.509v3, IPsec, IKEv2
Confidencialidade	TLS, IPsec		TLS	TLS, Criptografia⋆				
Integridade	TLS, IPsec		TLS	TLS, X.509v3				
Autorização	OAuth 2.0	OAuth 2.0	OAuth 2.0	OAuth 2.0			OAuth 2.0	OAuth 2.0
Proteção contra reprodução				TLS, Resumo criptográfico⋆				
Exportação de registros			FTPES, TLS, SSH, mTLS, X.509v3					

⋆Conforme algoritmos especificados em [O-RAN Working Group 11, 2023b].

Os requisitos de segurança para a O-Cloud protegem os pacotes de aplicações e funções virtualizadas na camada de aplicação, isto é, xApps, rApps, O-CU, O-DU e Near-RT RIC; da camada de infraestrutura. A proteção da camada de infraestrutura ainda não está completamente especificada. Em relação à camada de aplicação, a O-Cloud deve suportar autenticação e autorização de usuários, recomendando-se o uso de autenticação por múltiplos fatores. As aplicações e funções que se comunicam entre si devem estar mutuamente autenticadas e autorizadas. A autenticidade e integridade das imagens das aplicações e funções que executam na O-Cloud devem ser garantida e, para tal, utilizam-se resumos criptográficos e assinaturas com a chave privada do fornecedor daquela aplicação ou função. As imagens armazenadas no repositório de imagens da O-Cloud devem ser protegidas em relação à confidencialidade e integridade, e podem ser acessadas apenas por entidades autorizadas. Os pacotes devem ser testados pelos fornecedores em relação a vulnerabilidades conhecidas antes de serem instanciados na O-Cloud. As informações sensíveis existentes nesses elementos devem estar criptografadas, de forma que a arquitetura deve suportar criptografia simétrica ou assimétrica. Deve existir registro contínuo de modificações realizadas em cada aplicação ou função entre versões e monitoramento contínuo do repositório para verificar se modificações, exclusões ou adições não autorizadas foram realizadas no repositório. É importante prover um isolamento robusto dos dados em trânsito, usados e armazenados, e controle de acesso aos recursos da infraestrutura. Em relação à camada de infraestrutura, são especificados requisitos e controle para prover atualização segura. As imagens dos pacotes devem estar sempre atualizadas e antes de serem atualizadas devem ser assinadas pelo fornecedor para assegurar a autenticidade e integridade. Deve haver capacidade da O-Cloud de se recuperar de incidentes na atualização ou instalação de pacotes [O-RAN Working Group 11, 2023c].

O Non-RT RIC deve suportar autorização e ser capaz de se recuperar de ataques DDoS massivos provenientes das interfaces A1 e R1. A autorização ocorre via OAuth2.0.

As rApps também devem ser capazes de suportar autorização e se recuperarem de ataques DDoS, porém provenientes da interface R1. O Near-RT RIC deve oferecer serviço de autenticação e autorização, de forma que apenas xApps autenticadas e autorizadas possam acessar a NIB. As xApps devem ser autenticadas com a assinatura do fornecedor antes de serem instanciadas e devem ter sua integridade verificada ao serem registradas no Near-RT RIC, usando assinaturas tanto do provedor de serviço quanto do fornecedor da xApp. As APIs do Near-RT RIC devem suportar autenticação mútua para que as xApps possam ser autenticadas. As APIs também devem suportar autorização, para que apenas xApps autorizadas segundo as políticas da operadora possam acessá-las. Ademais, o Near-RT RIC deve ser capaz de se recuperar de ataques DDoS massivos provenientes da interface A1. A autenticação mútua é realizada por meio do protocolo mTLS com certificados X.509v3. Atualmente as APIs relacionadas aos nós E2 são especificadas para executarem sobre SCTP (*Stream Control Transmission Protocol*) com Protobuf como protocolo de codificação. Essas APIs são consideradas de tempo crítico e não são suportadas pelo protocolo TLS. A autenticação de APIs relacionadas aos nós E2 é baseada em IPsec com certificado IKEv2. A autorização é feita por meio de OAuth2.0. A confidencialidade e a integridade são suportadas por TLS, exceto para APIs relacionadas aos nós E2, que utilizam IPsec nesse caso [O-RAN Working Group 11, 2023c].

O SMO deve suportar autenticação e autorização de funções internas e sistemas externos, além de confidencialidade, integridade, autenticação mútua e proteção contra reprodução para comunicações internas e externas. A comunicação externa deve adicionalmente ser autorizada. O SMO deve ser capaz de se recuperar de ataques DDoS massivos tanto internos quanto externos, seguindo o princípio de ZT. A autorização é feita por meio de OAuth 2.0 (*Open Authorization* 2.0), enquanto a autenticação é suportada por mTLS com certificados X.509v3. Opcionalmente, o SMO também pode suportar autenticação usando TLS com chave pré-compartilhada (*Pre-Shared Key* – PSK). A segurança dos registros de segurança produzidos pelo SMO também é destacada pela O-RAN Alliance. Os registros devem ser acessados apenas por agentes autorizados, autenticados mutuamente, e deve-se garantir a confidencialidade e integridade do conteúdo. Os registros de segurança gerados pelo SMO podem ser armazenados tanto local quanto remotamente. No caso de armazenamento remoto, deve haver suporte à escolha de servidores para transferência segura. Os registros de segurança devem ser armazenados separadamente dos registros do sistema. Além de mTLS e X.509v3, a segurança dos registros de segurança deve suportar FTPES (*File Transfer Protocol over explicit transport layer security*) e pode suportar o uso de PSK (*Pre-Shared Key*), sendo também sugerido o suporte a SSH e SFTP (*Secure File Transfer Protocol*), para transferência segura de arquivos [O-RAN Working Group 11, 2023c].

4.2.2.2. Interfaces Abertas

Na segunda categoria de requisitos estão as interfaces abertas, que de forma geral, devem suportar autenticação, autorização, confidencialidade, integridade e proteção contra reprodução. As interfaces devem também suportar o uso de NACM (*Network Configuration Access Control Model*) quando o NETCONF (*Network Configuration Protocol*) for usado pelas operadoras da rede para gerenciar as funções O-RAN. O NACM

fornece os meios para restringir o acesso dos usuários a um subconjunto pré-configurado de todas as operações e conteúdos disponíveis do NETCONF. O NACM também permite que as operadoras integrem autenticação e autorização com uma plataforma centralizada de gerenciamento de acesso, protejam a configuração em execução contra modificação e exclusão não autorizadas, e ofereçam suporte a procedimentos de gerenciamento de mudanças para atualizar as configurações das funções de rede e as alterações na instância do NACM em uma função de rede [O-RAN Working Group 11, 2023c]. A Tabela 4.5 resume os requisitos das interfaces e a Tabela 4.6 mostra os protocolos usados para suportar os requisitos.

Tabela 4.5. Requisitos de segurança para as interfaces especificadas pela O-RAN Alliance.

Requisito	A1	O1	O2	E2	R1	Open FH			
						U-Plane	M-Plane	C-Plane	S-Plane
Autenticidade		✓							
Autenticação	✓		✓	✓	✓		✓	✓	✓
Confidencialidade	✓	✓	✓	✓	✓	✓	✓		
Integridade	✓	✓	✓	✓	✓	✓	✓		
Autorização	✓	✓		✓	✓			✓	✓
Proteção contra reprodução	✓		✓	✓	✓				
Proteção contra *spoofing* de relógio mestre									✓
Proteção contra homem no meio									✓
Proteção contra remoção									✓

Tabela 4.6. Protocolos de segurança recomendados pela O-RAN Alliance para as interfaces abertas.

Requisito	A1	O1	O2	E2	R1	Open FH			
						U-Plane	M-Plane	C-Plane	S-Plane
Autenticidade		TLS							
Autenticação	TLS, mTLS		TLS, mTLS, X.509v3	IPsec	TLS, mTLS	802.1X, TLS, SSH	TLS, SSH, mTLS, X.509v3	802.1X	
Confidencialidade	TLS	TLS	TLS	IPsec		PDCP	TLS, SSH		
Integridade	TLS	TLS	TLS	IPsec		PDCP	TLS, SSH		
Autorização	OAuth 2.0	NACM	OAuth 2.0		OAuth 2.0	802.1X		802.1X	802.1X
Proteção contra reprodução	TLS	TLS	TLS	IPsec		PDCP	TLS, SSH		

A interface A1 deve suportar autenticação mútua, autorização, confidencialidade, integridade e proteção contra reprodução. A autenticação é feita por mTLS e autorização por OAuth2.0. Os outros requisitos são suportados pelo protocolo TLS. A interface O1

pode revelar informações sensíveis a usuários não autorizados caso seja implementada inadequadamente. A interface deve prover confidencialidade, integridade e autenticidade por meio de TLS e autorização usando NACM [O-RAN Working Group 11, 2023c]. As interfaces O2 e E2 devem suportar confidencialidade, integridade, proteção contra reprodução e autenticação da origem dos dados. A interface O2 suporta TLS, enquanto a E2 suporta IPsec. A autenticação mútua na interface O2 ocorre por meio de mTLS com certificados X.509v3. A interface Open FH é dividida em planos. O plano de controle deve suportar autenticação e autorização das O-DUs, usando o protocolo IEEE 802.1X. O plano de usuário deve suportar confidencialidade e integridade, o que é feito por meio do protocolo PDCP (*Packet Data Convergence Protocol*). O plano de sincronização deve suportar autenticação e autorização, proteção contra ataques de *spoofing* de relógio mestre e homem no meio. A arquitetura de sincronização deve prover redundância, suportando múltiplos relógios mestres. A autenticação e autorização ocorrem por meio do protocolo IEEE 802.1X. O plano de gerenciamento deve prover integridade, confidencialidade e autenticação por meio de SSH e TLS. As sessões NETCONF devem ser protegidas por conexões TLS ou túneis SSH. Podem ser utilizadas senhas e certificados X.509 para autenticação. A interface R1 deve suportar autorização via OAuth2.0, autenticação mútua via mTLS, e confidencialidade, integridade e proteção contra reprodução via TLS. A interface ONI deve suportar autenticação mútua por meio de mTLS com certificados X.509v3 e autorização por meio de OAuth2.0 [O-RAN Working Group 11, 2023c].

4.2.3. Requisitos Transversais

Na última categoria estão os requisitos transversais, formulados de forma mais genérica e que se aplicam a todo o sistema O-RAN. Existem grupos de requisitos transversais, que incluem, por exemplo, protocolos e serviços de rede e gerenciamento do ciclo de vida das aplicações. Os protocolos e serviços devem ser robustos o suficiente para lidar com entradas não esperadas, ataques DDoS massivos e ataques contra autenticação baseada em senha, caso esse tipo de autenticação seja usado. Também deve haver robustez do sistema operacional e das aplicações instaladas, com identificação e documentação clara das vulnerabilidades conhecidas. O gerenciamento de ciclo de vida das aplicações requer a garantia de autenticidade e integridade de xApps, rApps, O-CU, O-DU, O-RU e Near-RT RIC; proteção da integridade das aplicações e funções virtualizadas; capacidade de atualização segura dessas aplicações e funções; monitoramento do consumo e disponibilidade de recursos dessas aplicações e funções; controle de acesso e a divulgação imediata de vulnerabilidades descobertas com atualizações rápidas com intuito de mitigá-las. No entanto, a especificação não detalha concretamente como prover a segurança nessa categoria [O-RAN Working Group 11, 2023c].

4.3. Vulnerabilidades e Ameaças de Segurança

Esta seção discute ameaças e vulnerabilidades de segurança dos componentes e interfaces da RAN aberta e das tecnologias relacionadas. Destacam-se riscos tecnológicos referentes à adição de novos componentes à RAN e ao uso de interfaces abertas, de *software* de código-fonte aberto, de técnicas de virtualização e de inteligência artificial.

4.3.1. Vulnerabilidades

Existem potenciais vulnerabilidades na RAN aberta que podem ser exploradas por ataques contra a confidencialidade, integridade e disponibilidade. Essas vulnerabilidades podem ser específicas da arquitetura O-RAN ou gerais [O-RAN Working Group 11, 2023a].

Entre as vulnerabilidades específicas da arquitetura O-RAN estão (i) o acesso não-autorizado aos componentes O-DU, O-CU-CP, O-CU-UP e O-RU para degradar o desempenho da RAN ou executar um ataque mais abrangente à rede, comprometendo a disponibilidade da RAN; (ii) a falta de proteção à sincronização e ao tráfego de controle na interface Open FH, que pode afetar a integridade e disponibilidade; (iii) a desabilitação da cifragem na transmissão aérea (*over-the-air*) para bisbilhotagem e, assim, comprometer a confidencialidade dos dados transmitidos; (iv) os conflitos entre o Near-RT RIC e a O-eNB e entre as xApps e rApps, que podem comprometer a disponibilidade dos serviços da RAN; (v) o acesso das xApps e rApps a dados da rede e de seus usuários, o que caracteriza quebra da confidencialidade; (vi) a interface de gerenciamento não protegida que implica problemas de confidencialidade, integridade e disponibilidade e, por fim, (vii) o ataque por injeção de mensagens do O-CU-CP ao O-CU-UP, comprometendo sua disponibilidade.

Entre as vulnerabilidades gerais estão (i) o desacoplamento de funções sem uma Raiz de Confiança protegida por *hardware* ou uma Cadeia de Confiança em *software*, o que pode levar a problemas de integridade; (ii) a exposição a explorações públicas e conhecidas em função do uso de *software* com código-fonte aberto e, por fim, (iii) a configuração incorreta, o isolamento fraco ou o gerenciamento de acesso insuficiente na plataforma O-Cloud. Tanto (ii) quanto (iii) podem trazer problemas de confidencialidade, integridade e disponibilidade.

4.3.2. Ameaças

O modelo de ameaças, adotado neste capítulo, determina a superfície de ataque e identifica os pontos de entrada e os agentes de ameaças à RAN aberta. Sabe-se que a RAN aberta introduz novos componentes e adota interfaces padronizadas e abertas entre eles. Tal característica, aliada à desagregação de *hardware* e *software*, à virtualização, ao uso de componentes de *software* de código e aberto e ao uso de inteligência artificial, expande a superfície de ataque da rede, como mostra a Figura 4.2. No caso da arquitetura O-RAN, a superfície de ataque é definida pelos (i) novos componentes, SMO, Non-RT RIC e Near-RT RIC; (ii) as novas interfaces, A1, E2, O1, O2 e Open FH; (iii) o uso de virtualização para desagregação do *software* e *hardware*; (iv) o uso de *software* de código-fonte aberto e (v) o uso de técnicas de inteligência artificial. Ameaças para cada um dos pontos da superfície de ataque da O-RAN são apresentadas nas próximas seções. São considerados, ainda, pontos de entrada: as APIs entre os planos que facilitam a propagação de ameaças, as ameaças vindas de dentro da RAN aberta e as ameaças vindas de fora da RAN aberta. São considerados agentes de ameaças cibercriminosos, atacantes internos, ativistas, ciberterroristas, *script kiddies* e nações-estado [O-RAN Working Group 11, 2023a].

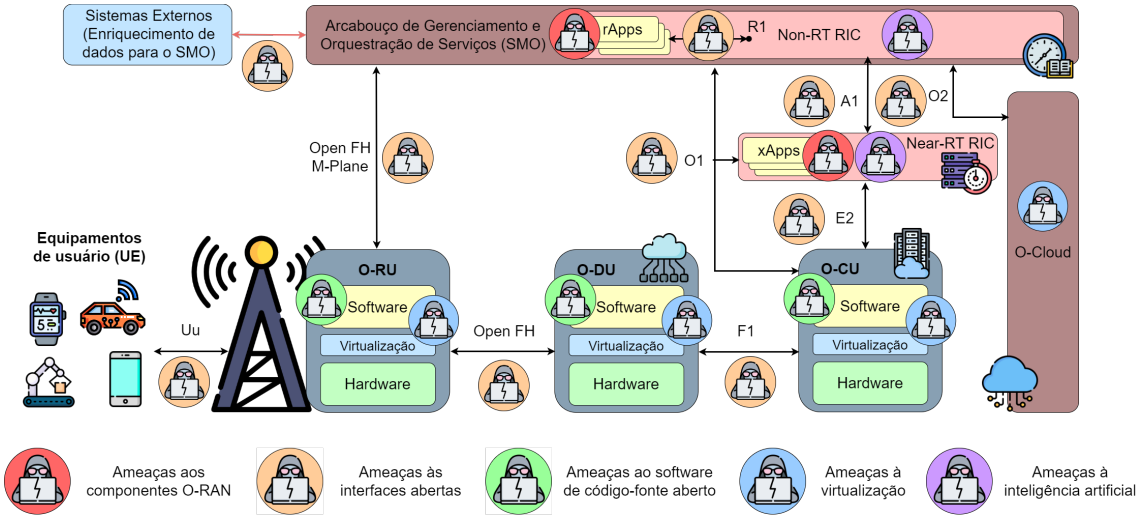


Figura 4.2. A superfície aumentada de ataque da RAN aberta com base na arquitetura O-RAN.

4.3.2.1. Novos Componentes: Controladores Inteligentes e Arcabouço de Orquestração e Gerenciamento de Serviços

Qualquer xApp pode possuir vulnerabilidades. Porém, se a xApp for desenvolvida por uma fonte não confiável ou por uma fonte que não a mantenha de forma adequada, as chances de vulnerabilidades aumentam [Open RAN Policy Coalition, 2021]. Se atacantes identificam uma xApp que pode ser explorada, eles podem interromper o serviço oferecido pela rede e potencialmente assumir o controle de outra xApp ou de todo Near-RT RIC. Nesse cenário, um atacante pode ganhar a habilidade de alterar dados transmitidos através das interfaces A1 e E2, extrair informação sensível e impactar as funções do Near-RT RIC para degradar o seu desempenho [Abdalla et al., 2022].

Uma xApp maliciosa em execução no Near-RT RIC pode explorar a identificação de um equipamento de usuário, rastrear a localização de um UE e alterar a prioridade do UE [O-RAN Working Group 11, 2023a]. As xApps no Near-RT RIC podem manipular o comportamento de uma célula, de um grupo de UEs e até mesmo de um UE específico. A ausência de uma raiz de confiança ou uma raiz de confiança com mau funcionamento pode causar problemas na rede e comprometer o desempenho da RAN e a privacidade dos usuários. Uma xApp pode, por exemplo, rastrear um dado usuário ou impactar o serviço para um assinante ou o serviço para uma determinada área coberta pela RAN. As xApps podem receber uma ordem através da Interface A1 para priorizar um determinado UE. Se uma xApp for maliciosa e receber tal ordem, então o proprietário da xApp maliciosa saberá que há um usuário privilegiado em uma determinada área. Assim, a partir dessa exposição de comandos, a xApp maliciosa poderá rastrear o usuário privilegiado ou até mesmo alterar seu nível de privilégio.

A Interface E2, que faz a comunicação do Near-RT RIC e os nós E2, também expõe a identificação de um UE que pode ser explorada por uma xApp maliciosa. Assim como a A1, a Interface E2 pode apontar para um UE específico na rede, criando uma correlação entre as identidades aleatórias (anonimizadas) dos UEs entre os nós da RAN.

Por exemplo, uma xApp pode ser usada como um farejador de rede para identificar um UE, como mostra a Figura 4.3. Essa vulnerabilidade também está presente na comunicação Non-RT RIC-A1. Entretanto, o desafio para a comunicação Near-RT RIC-E2 é maior do que para a comunicação Non-RT RIC-A1, porque espera-se que a frequência de sinalização na E2 será maior para possibilitar a operação em quase tempo real. Assim, o identificador do UE será mais frequentemente enviado pela Interface E2 do que pela A1.

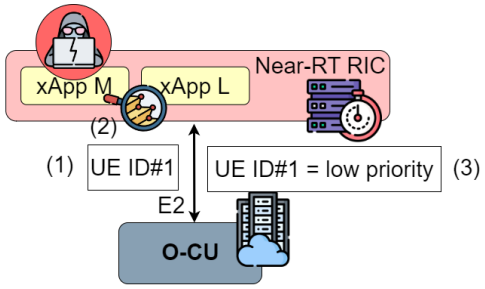


Figura 4.3. Uma xApp maliciosa atua como farejador de rede. A xApp legítima L recebe a identificação de um UE via interface E2 (Passo 1). A xApp maliciosa M tem a capacidade de escutar todas as mensagens que são enviadas pela interface E2. Por isso, também recebe o identificador do UE (Passo 2). Em seguida, a xApp M pode enviar uma mensagem à O-CU para reduzir a prioridade do UE #1 (Passo 3).

Outra ameaça às xApps é a criação de políticas A1 falsas que podem impactar o desempenho da RAN. Isso é possível no cenário em que há acesso não autorizado ao Non-RT RIC. É esse controlador que cria as políticas A1 e as envia ao Near-RT RIC para sua aplicação. Além disso, políticas enviadas ao Near-RT RIC, falsas ou legítimas, são persistentes até que sejam modificadas ou excluídas pelo Non-RT RIC ou até que o Near-RT RIC seja desligado. Políticas A1 falsas podem ser criadas do zero ou a partir da modificação de uma política A1 legítima existente para se chegar a uma política A1 falsa. Uma política A1 falsa pode, por exemplo, ter como alvo um UE específico, grupos de UEs ou uma célula inteira. Em outro exemplo, uma política falsa A1 pode fazer com que o Near-RT RIC configure funções da O-DU e da O-RU para iniciar um ataque de negação de serviço usando dados de realimentação para degradar o desempenho da RAN. As políticas A1 falsas também podem ser usadas para localizar um UE ou um grupo de UEs. Nesse caso, a política A1 falsa faz com que o Near-RT RIC isole um UE na O-CU. O Near-RT RIC também pode usar a conformação de feixe do MIMO (*Multiple Input Multiple Output)* na O-DU e na O-RU para isolar o UE em um único feixe. Os dados de realimentação da RAN podem incluir a localização do UE ou a informação de trajetória obtidas com os dados de GPS. A localização do usuário seria obtida a partir do acesso ao Non-RT RIC.

Outras ameaças ao Near-RT RIC são xApps maliciosas que obtenham acesso não-autorizado ao próprio Near-RT RIC e aos nós E2, que abusem de informações de rádio e recursos de controle sobre funções da RAN, que impactem o serviço de um usuário ou de uma dada área, que explorem a identificação de um UE, rastreiem a localização do UE e mudem a prioridade da fatia de rede do UE.

Assim como podem existir vulnerabilidades nas xApps, também podem existir vulnerabilidades nas rApps. Caso consiga explorar uma rApp, um atacante pode alterar

dados transmitidos pela Interface A1, extrair informações sensíveis, interromper o serviço oferecido pela rede e assumir o controle de outra rApp ou até mesmo do Non-RT RIC. Um atacante pode penetrar o Non-RT RIC através do SMO para disparar um ataque de negação de serviço e, assim, degradar o desempenho desse controlador. Nesse caso, o Non-RT RIC não seria confiável para garantir (i) o monitoramento da rede para entender o efeito de uma política A1 no desempenho do Near-RT RIC; (ii) a atualização de uma política A1; (iii) a exposição e a entrega segura da informação enriquecida A1 para o Near-RT RIC e, por fim, (iv) a implantação de regras de controle de acesso. Além disso, um atacante que consiga penetrar o Non-RT RIC através do SMO pode rastrear um UE e modificar e corromper dados.

Podem existir também conflitos entre rApps, causados de forma não intencional ou de forma maliciosa, que impactem o desempenho da RAN. Esses conflitos são possíveis porque as rApps são fornecidas por diferentes vendedores. Por exemplo, um vendedor fornece a rApp para escalonamento de licença de operadora, outro vendedor fornece a rApp para gerenciamento do consumo de energia etc. Com isso, há o risco de diferentes rApps tomarem decisões conflitantes ao mesmo tempo para um dado usuário. Os conflitos podem ser (i) diretos, nos quais diferentes rApps solicitam alteração de um mesmo parâmetro, (ii) indiretos, quando diferentes rApps solicitam alteração de diferentes parâmetros que criam efeitos opostos, e (iii) implícitos, quando diferentes rApps solicitam alteração de diferentes parâmetros que não criam efeitos opostos óbvios, mas que resultam em degradação do desempenho de toda a rede, instabilidades etc.. Os conflitos implícitos são difíceis de serem resolvidos por conta da dificuldade de se observar e identificar as dependências entre os parâmetros modificados pelas rApps.

Atacantes externos e internos podem explorar mecanismos fracos de autenticação e autorização no SMO. Se mecanismos de autenticação não são implementados corretamente ou não são suportados pelas Interfaces A1, O1, O2 e interfaces externas no SMO, um atacante externo pode explorar tais interfaces sem credenciais apropriadas para ganhar acesso ao SMO. Um atacante externo também pode explorar a ausência ou deficiência dos mecanismos de autorização do SMO. Nesse caso, um atacante externo que acesse as Interfaces A1, O1, O2 e as interfaces externas do SMO, mesmo sem autorização ou com um *token* de acesso incorreto, pode invocar uma função do SMO. Dados relacionados a tal função serão, então, vazados para o atacante. Além disso, o atacante pode executar determinadas ações como divulgar informações sensíveis da O-RAN ou alterar os componentes da O-RAN. Da mesma forma, atacantes internos que acessem a Interface R1 e o barramento interno de mensagens podem invocar funções e executar ações. Por fim, uma vez que tenha acesso ao SMO, um atacante pode ver, modificar ou apagar arquivos de registros (*log*) armazenados no arcabouço, bem como envenenar dados de treinamento ou os modelos de inteligência artificial armazenados no SMO para influenciá-los.

4.3.2.2. Interfaces Abertas

Existem ameaças às diferentes interfaces que interconectam os componentes da arquitetura O-RAN, como a O-CU, O-DU e O-RU, e também à interface de rádio que dá acesso aos usuários da RAN.

Um atacante pode penetrar e comprometer a RAN aberta através das Interfaces Open FH, O1, O2, A1 e E2 da RAN. Tais interfaces permitem a programabilidade da rede e podem não estar protegidas de acordo com as melhores práticas de segurança da indústria, por exemplo, não implementando mecanismos de autenticação e processos de autorização adequados, cifragem e verificações de integridade, proteção contra ataques de repetição, prevenção de reutilização de chaves, validação de entradas, resposta a condições de erro, entre outros [Open RAN Policy Coalition, 2021]. As interfaces O-RAN permitem o uso do TLS ou SSH. As atuais melhores práticas da indústria obrigam o uso de TLS versão 1.2 ou superior ou SSH com autenticação baseada em cerificados [O-RAN Working Group 11, 2023a]. Uma interface que em sua implementação use uma versão do TLS inferior a 1.2 ou SSH com autenticação baseada em senha pode ser o ponto-chave a ser explorado por um atacante que deseja comprometer a RAN aberta.

Se forem usados mecanismos de autenticação e controle de acesso fracos, é possível para um atacante criar uma estação-base falsa para enganar a O-DU e as UEs para se associarem a ela e não a uma estação-base legítima. Para tanto, um atacante se passa por uma rede móvel legítima e emprega um ataque de homem-no-meio entre o UE e a rede móvel, como mostra a Figura 4.4. Um atacante sequestra a rede *fronthaul*, isto é, o atacante (i) desabilita o acesso da O-RU legítima e operacional à Interface Open FH, (ii) conecta a estação-base falsa na Interface Open FH da O-RU legítima e, em seguida, (iii) inicia o ataque da estação-base falsa com a O-RU fornecendo a interface aérea para os UE. Para um UE, a estação-base falsa é legítima e ele se conectará a ela. Assim, a estação-base falsa é capaz de interceptar e divulgar a identidade de um usuário e registrar de forma não autorizada sua localização e movimentação. Para o operador da rede móvel legítima, a O-RU legítima simplesmente deixou de servir os UEs em sua área de cobertura, desde que o atacante desabilitou a Interface Open FH. Ataques que implementam estações-base falsas são conhecidos desde as primeira gerações de redes móveis e, apesar dos incrementos de segurança, ainda estão presentes nas redes 5G [O-RAN Working Group 11, 2023a].

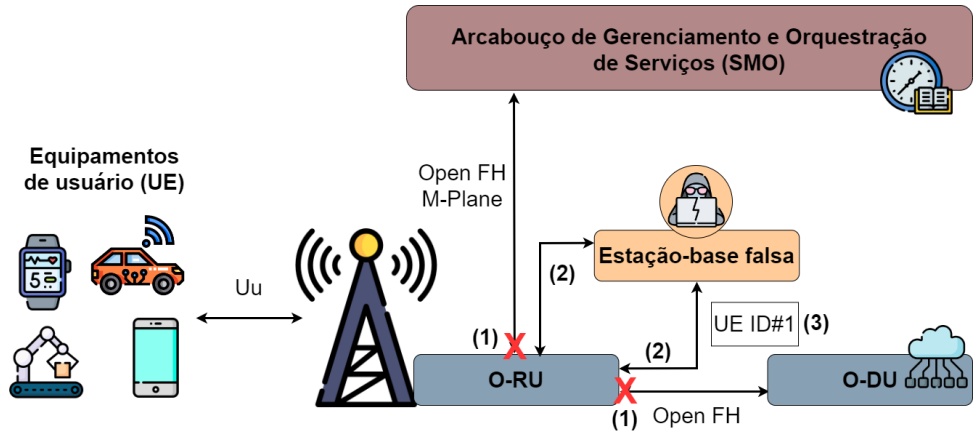


Figura 4.4. O ataque da estação-base falsa. Um atacante desabilita o acesso da O-RU legítima à Interface Open FH (Passo 1). Em seguida, o atacante conecta a estação-base falsa na Interface Open FH da O-RU legítima (Passo 2) e inicia o ataque com a O-RU ainda fornecendo a interface aérea para os UEs. O atacante pode, por exemplo, obter um identificador de um UE, conectado à O-RU legítima (Passo 3).

A Interface A1 é usada para comunicação entre o Non-RT RIC e o Near-RT RIC. Se um mecanismo fraco de autenticação mútua é usado entre esses controladores, um Non-RT RIC malicioso pode se tornar par de um Near-RT RIC legítimo através da Interface A1 ou um Near-RT malicioso pode se tornar par de um Non-RT RIC legítimo, também através dessa interface. Além disso, se um ataque de homem-no-meio é implementado entre os controladores, as políticas enviadas pela Interface A1 podem ser lidas e modificadas e falsas políticas podem ser injetadas. Com isso, o Near-RT RIC pode receber políticas maliciosas.

Ataques à interface de rádio que dá acesso aos usuários da RAN são ataques clássicos a sistemas de comunicação sem fio, como o *jamming*, farejamento e falsificação (RAN *sniffing and spoofing*). Esses ataques estão fora do escopo deste capítulo.

4.3.2.3. Virtualização

A virtualização envolve funções físicas de rede (*Physical Network Function* – PNF), funções virtuais de rede (*Virtual Network Function* – VNF), funções em nuvem de rede (*Cloud Network Function* – CNF), o SMO, o hipervisor, as máquinas virtuais e os contêineres. Todos esses elementos podem sofrer com ameaças e vulnerabilidades. São considerados cinco tipos de ameaças à virtualização: (i) comprometimento de imagens VNF/CNF, (ii) configurações fracas do orquestrador, controles de acesso e isolamento fracos, (iii) uso indevido de uma máquina virtual ou contêiner para atacar outra máquina virtual ou contêiner, hipervisor ou motor de contêiner, e outros sistemas finais, (iv) bisbilhotagem do tráfego de rede para acessar todos os dados da RAN aberta processados na carga de trabalho e (v) comprometimento dos serviços de rede auxiliares e de suporte. Um atacante pode, por exemplo, explorar uma falha de configuração ou uso de mecanismos fracos de controle de acesso e isolamento para ganhar acesso não-autorizado ao SMO. Tal orquestrador pode executar diferentes máquinas virtuais/contêineres, cada uma gerenciada por diferentes usuários e com diferentes níveis de sensibilidade. Se o acesso fornecido aos usuários não estiver em conformidade com seus requisitos específicos, um atacante ou usuário descuidado pode afetar a operação de outra máquina virtual/contêiner gerenciada pelo SMO. Dados sensíveis de usuários também podem ser acessados e vazados caso implementações da RAN aberta nativas em nuvem sofram com mecanismos de isolamento insuficientes. Ao se implementar uma O-CU em nuvem, é possível comprometê-la por meio de vetores de ameaças, como a migração de serviço, descarregamento de tráfego ou mecanismos de *handover* [Ranaweera et al., 2021]. Uma O-CU comprometida é capaz de prejudicar as direções *fronthaul* e *backhaul* se aproveitando das interfaces abertas da Open RAN.

Um exemplo de ameaça pela falta de isolamento forte é o ataque de fuga de máquina virtual ou contêiner (*VM/Container escape attack*) [O-RAN Working Group 11, 2023a], ilustrado na Figura 4.5. VNFes/CNFes em execução na mesma máquina física como inquilinos compartilham o mesmo núcleo e os recursos do sistema operacional do hospedeiro. A falta de isolamento forte entre as máquinas virtuais ou contêineres e o hospedeiro traz o risco de uma máquina virtual ou contêiner não-autorizado escapar do confinamento e impactar outras máquinas virtuais

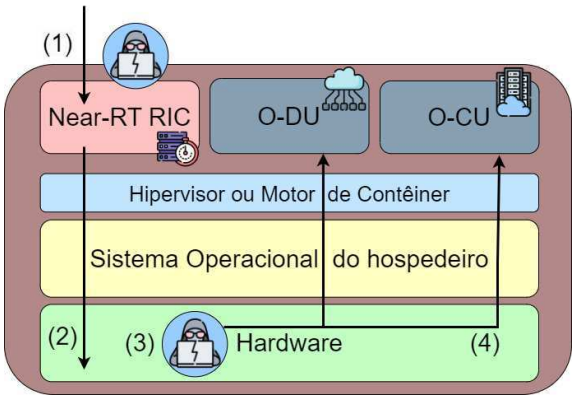


Figura 4.5. O ataque de fuga de uma máquina virtual ou contêiner. Nesse exemplo, o Near-RT RIC, a O-DU e a O-CU são inquilinos de um mesmo hospedeiro. Um atacante consegue acesso não-autorizado ao Near-RT RIC (Passo 1). Em seguida, por falta de isolamento forte, o Near-RT RIC malicioso escapa do isolamento (Passo 2) e ganha controle do hospedeiro (Passo 3). O atacante incia um ataque de negação de serviço contra a O-DU e a O-CU (Passo 4).

ou contêineres co-hospedados. Se uma máquina virtual ou contêiner malicioso escapa do isolamento, ela pode ganhar controle total do seu hospedeiro e o atacante pode: (i) se tornar capaz de iniciar ataques do hospedeiro ou comprometer o hospedeiro, (ii) comprometer a confidencialidade e a integridade das máquinas virtuais co-hospedadas e dos inquilinos, (iii) iniciar um ataque de negação de serviço às máquinas virtuais ou aos contêineres co-hospedados e aos serviços do hospedeiro para degradar seu desempenho e, por fim, (iv) introduzir novas vulnerabilidades no hospedeiro para serem exploradas em ataques futuros.

Um atacante também pode criar uma nova máquina virtual ou contêiner malicioso configurado sem regras de rede, limitações de usuários etc. para contornar as defesas existente na infraestrutura da O-Cloud. Se essa máquina virtual ou contêiner malicioso escapa do hospedeiro e alcança o *hardware* do hospedeiro, ela pode ganhar acesso de super-usuário de todo o hospedeiro no qual reside. Isso dá à máquina virtual ou ao contêiner malicioso total controle sobre as máquinas virtuais ou contêineres hospedados no mesmo hospedeiro comprometido. Com isso, o atacante pode impactar a confidencialidade, a integridade e a disponibilidade dos recursos das VNFes/CNFes.

4.3.2.4. Software de Código-Fonte Aberto

Espera-se que os componentes da RAN aberta sejam implementados baseados em *software* de código-fonte aberto e, por isso, herdem as vulnerabilidades conhecidas pela comunidade pelo uso desse tipo de *software* [Liyanage et al., 2023]. A O-RAN Software Community[2] (OSC) é responsável pela implementação das especificações O-RAN em código aberto. É um projeto da Linux Foundation, apoiado pela O-RAN Alliance.

Tanto a academia quanto a indústria reconhecem que o uso de *software* de código-fonte aberto apresenta riscos de segurança. Por falta de conhecimento, desenvolvedores podem usar componentes de *software* com vulnerabilidades conhecidas em listas públi-

[2]Disponível em: `https://www.o-ran.org/software`

cas, por exemplo, a *National Vulnerability Database* (NVD) [3], mantida pela agência NIST do governo norte-americano. Embora tais listas se destinem a desenvolvedores para divulgar vulnerabilidades, elas também podem ser usadas por atacantes para explorar as vulnerabilidades divulgadas. Sabe-se ainda que as vulnerabilidades se propagam mais rapidamente à medida que há reuso de *software* de código-fonte aberto mais frequentemente. Desenvolvedores podem também usar bibliotecas não confiáveis, sem o devido cuidado com o gerenciamento de dependências e correções. Da mesma forma, desenvolvedores podem obter componentes de *software* de código-fonte aberto em repositórios não confiáveis. Tais problemas se agravam porque fornecedores e operadores da RAN aberta podem não ter inventários precisos de dependências de *software* de código-fonte aberto usadas por suas diferentes aplicações ou um processo para receber e gerenciar notificações sobre vulnerabilidades descobertas ou remendos disponíveis da comunidade que oferece suporte ao código-fonte aberto.

Outra vulnerabilidade é a introdução de *backdoors* por desenvolvedores confiáveis, que inserem intencionalmente linhas de código malicioso em um componente de código-fonte aberto a ser usado na RAN aberta. Enquanto esse componente é usado e a vulnerabilidade não é detectada, o desenvolvedor confiável, mas, na verdade, malicioso, pode acessar o componente, comprometer dados sensíveis e iniciar ataques de negação de serviço para reduzir o desempenho da RAN.

4.3.2.5. Inteligência Artificial

Há ameaças específicas aos algoritmos e modelos de aprendizado de máquina usados pelos controladores RIC e pelo arcabouço SMO. Uma das ameaças são os ataques de envenenamento de dados, nos quais o atacante altera os conjuntos de dados destinados a treinamento, teste ou validação [Sun et al., 2022]. No caso da alteração dos dados de treinamento, serão usados dados incorretos na modelagem, resultando em um modelo não-confiável. Assim, decisões, previsões, classificações e detecções com base nesse modelo não serão apropriadas. Outro cenário pode ser uma situação em que um modelo está *online* e continua aprendendo durante o uso operacional, modificando seu comportamento ao longo do tempo. Nesse caso, um invasor pode alimentar o modelo com dados incorretos e o modelo pode aprender com esses dados incorretos e, como resultado, afetar negativamente seu desempenho e treinar novamente o modelo para tomar decisões erradas.O acesso para realizar alterações nos dados pode ser obtido via penetração através das redes *fronthaul* ou *mid-haul*, xApps ou rApps.

Outra possível ameaça é o ataque de alteração de modelo, relacionado à integridade da predição. O ataque ocorre quando um atacante obtém acesso não-autorizado ao modelo em produção e altera os parâmetros do modelo, influenciando os resultados produzidos pelo modelo. Consequentemente, isso pode levar a predições erradas e a tomadas de decisão erradas pelo operador da rede. Por fim, ataques de transferência de aprendizado podem ocorrer quando um atacante ajusta de forma maliciosa um modelo pré-treinado já disponível, mascarando seu comportamento malicioso. Mais detalhes sobre ataques aos sistemas de aprendizado de máquina são discutidos na Seção 4.4.

[3]Disponível em `https://nvd.nist.gov/`

4.4. Ataques baseados em Aprendizado de Máquina para RAN

Um modelo de aprendizado de máquina / inteligência artificial desenvolvido de maneira legítima pode ser comprometido posteriormente [Habler et al., 2022]. O comprometimento pode ocorrer por diversos motivos, incluindo um desenvolvedor de aplicações inteligentes malicioso, *hardware* ou *software* da infraestrutura que sofreu um ataque ou um usuário malicioso que fabrica dados para alterar o modelo ou forçar determinadas respostas na sua inferência.

Os ataques aos sistema de aprendizado de máquina são classificados de acordo com três características principais: (i) capacidade do atacante acessar os parâmetros do modelo; (ii) momento no qual o atacante possui acesso ao modelo; e (iii) acesso ao conjunto de dados utilizado pelo modelo.

A característica (i) é subdividida em três: caixa-branca, caixa-preta e caixa-preta interativa. Em ataques do tipo caixa-branca, o agente malicioso possui acesso ao modelo e seus parâmetros de forma direta. Por outro lado, um ataque caixa-preta identifica que o atacante não possui nenhuma informação sobre o modelo e seus parâmetros. O modelo de atacante caixa-preta interativa assume que o atacante desconhece os parâmetros do modelo, porém é possível interagir com o modelo, recebendo resultados de classificação após o envio de uma amostra. Além disso, quando se considera que o atacante tem acesso ao modelo, o acesso pode ocorrer durante o treinamento ou no momento de inferência, conforme a característica (ii). Por fim, a característica (iii) identifica se o atacante tem acesso ao conjunto de dados de treinamento, podendo ajustar indiretamente os parâmetros do modelo, ou de teste, alterando a entrada sem modificar seus parâmetros internos.

McGraw et al. identificam, a partir da Análise de Risco Arquitetural (*Architectural Risk Analysis* – ARA), dez principais riscos de segurança na implementação de qualquer sistema que possui modelos de aprendizado de máquina em sua linha de execução [McGraw et al., 2020]. Esses riscos podem ser explorados por atacantes para afetar os modelos de aprendizado de máquina incluídos na arquitetura O-RAN:

- **Exemplos adversariais**, em que a entrada do modelo de aprendizado de máquina é alterada intencionalmente por atacantes. O objetivo é produzir entradas parecidas, porém que gerem erros de classificação. Dessa forma, o atacante interfere nos resultados de predição sem modificar os parâmetros do modelo;
- **Envenenamento de dados**, em que um atacante que possua acesso ao conjunto de dados de treinamento introduz pertubações controladas em amostras. O padrão adicionado às amostras faz com que o processo de classificação de amostras futuras seja controlado. Assim, o atacante modifica a predição do modelo de aprendizado de máquina por meio do ajuste de seus parâmetros diretamente;
- **Manipulação de sistemas *online***, em que sistemas que possuem aprendizado contínuo são ajustados ao longo de sua execução. Esse ajuste pode ocorrer de forma controlada por um atacante, com a intenção de prejudicar o sistema de classificação;
- **Ataque à transferência de aprendizado**, em que o atacante pode atacar um modelo base pré-treinado e ajustado, causando vulnerabilidades no modelo final após o reajuste de pesos;

- **Confidencialidade dos dados**, em que é possível inferir informações sobre a entrada de dados a partir de resultados parciais do modelo, gerando risco à privacidade. Dessa forma, é essencial proteger a confidencialidade dos dados e do modelo utilizado, impedindo que o atacante possa executar ataques do tipo caixa-branca. Porém, a interação com o modelo e a observação de sua classificação permite a execução de ataques de extração de modelo;
- **Confiabilidade dos dados**, em que é necessário assegurar que o conjunto de dados é coletado de forma bem definida e com baixo erro de medida. A falta de garantia de integridade pode resultar em ataques de envenenamento de dados. Além disso, erros de captura geram modelos com alta interferência de ruídos, dificultando a predição de comportamentos em novos dados analisados;
- **Reprodutibilidade**, em que é necessário assegurar que os resultados obtidos por modelos de aprendizado de máquina implantados em sistemas reais sejam reprodutíveis em condições semelhantes. Dessa forma, a documentação dos processos desenvolvidos, como a coleta do conjunto de dados, etapas de pré-processamento, hiperparâmetros utilizados, têm que ser apresentados de forma clara e objetiva;
- **Sobreajuste**, em que um atacante com acesso aos hiperparâmetros do modelo ou a grande parte do conjunto de dados de treinamento é capaz de enviesar resultados com a intenção de criar um modelo sobreajustado;
- **Integridade de codificação**, já que a codificação dos dados de entrada é uma parte essencial do problema de otimização de modelos de aprendizado de máquina. A representação utilizada para as amostras é capaz de aumentar a acurácia do modelo final se utilizada da melhor forma. Porém, representações incorretas podem enviesar os resultados de classificação. É necessário garantir que as características de entrada do modelo sejam íntegras, para evitar queda de desempenho em ambiente de produção e o lançamento de ataques através da inserção de valores controlados;
- **Integridade de predição**, o resultado de predição de amostras por um modelo de aprendizado de máquina pode ser alterado tanto a partir da produção de dados falsificados de entrada ou dados falsificados na saída, quanto na alteração direta de parâmetros do modelo ou sua completa substituição. Assim, é necessário garantir a integridade do modelo e de sua predição.

A privacidade do modelo de aprendizado de máquina também é um fator crucial no sistema. Ataques do tipo caixa-branca costumam ser mais efetivos do que ataques nos quais o atacante desconhece os parâmetros do modelo. Contudo, uma forma de obter conhecimento sobre um modelo S sem acesso direto a ele é criar um novo modelo S', com hiperparâmetros similares ao modelo original. Isso é feito através do ataque de extração de modelo, no qual o atacante realiza requisições ao modelo S e o agente malicioso, então, gera um conjunto de dados estimado, $\hat{X}$ e com amostras $\hat{x}_i$ não rotuladas. Os rótulos são obtidos a partir do modelo S, que recebe $\hat{x}_i$ e retorna a sua classe $\hat{y}_i$. Após iterar sobre todas as amostras do conjunto de dados $\hat{X}$, o atacante obtém um conjunto de dados rotulado $(\hat{X}, \hat{Y})$. Então, o atacante treina um modelo S' a partir do conjunto de dados rotulado gerado. O ataque de extração de modelo está relacionado com a confidencialidade dos

dados. Espera-se que a fronteira de decisão do modelo S' seja a mesma ou a mais próxima possível do modelo S. Uma vez que o modelo S' é conhecido pelo atacante, é possível utilizá-lo para estimar os atributos mais discriminantes de S.

Diversas funcionalidades da RAN aberta dependem do funcionamento adequado de modelos de aprendizado de máquina, como direcionamento de tráfego e otimização de recursos. Entretanto, a RAN aberta permite que desenvolvedores, potencialmente maliciosos, implementem as aplicações inteligentes [Groen et al., 2023]. Adicionalmente, equipamentos podem gerar tráfego malicioso. Por exemplo, ataques com exemplos adversariais em caixa preta [Ilyas et al., 2018] podem ser preocupantes. Nesses ataques, assume-se que o atacante pode realizar consultas ao modelo de classificação. No entanto, nos ataques em caixa preta, o atacante não é capaz de conhecer o modelo por completo e nem de derivar outras informações a partir do ataque. Ainda assim, o atacante é capaz de propositalmente gerar amostras que serão classificadas de maneira errônea pelo modelo. Esses erros de classificação podem então ser explorados pelo atacante para causar perturbações a todo o sistema. Dessa forma, um usuário de um sistema de RAN aberta é capaz de enganar o modelo de aprendizado de máquina para obter recursos de rede indevidos a partir da geração de indicadores de desempenho falsificados. Esse problema é uma preocupação comum em infraestruturas gerenciadas por aprendizado de máquina. Por exemplo, Usama *et al.* propõem a combinação do ataque de extração de modelo com exemplos adversariais para enganar classificadores de tráfego [Usama et al., 2019]. Um classificador baseado em redes neurais profundas (*Deep Neural Networks* – DNNs) classifica o tráfego Tor nas classes "*browsing*", "*chat*", "*streaming* de áudio", "*streaming* de vídeo", "email", "transferência de arquivos", "voz sobre IP" e "*peer-to-peer*". Com as entradas maliciosas, a acurácia da classificação decai de 96,3% para 2%. Uma vez que a tarefa de classificação de tráfego orienta diversos mecanismos da RAN aberta, uma classificação com baixa acurácia pode causar sérios prejuízos à rede. Assim, é necessário conhecer os ataques ao aprendizado de máquina que realiza as tarefas, a fim de mitigar seus impactos.

Os ataques mencionados anteriormente fazem parte do grupo geral de ataques a sistemas de aprendizado de máquina. A O-RAN Alliance identifica como ameaças relacionadas aos modelos de aprendizado de máquina o envenenamento de dados do modelo de aprendizado de máquina, a alteração de modelo e o ataque de transferência de aprendizado [O-RAN Working Group 11, 2023a], discutidos na Seção 4.3.2.5. As ameaças da especificação são divididas de uma maneira mais genérica [O-RAN Working Group 11, 2023a], pois visam abranger diversos cenários de implementação. Um dos cenários considerados é aquele no qual ambos o Non-RT RIC e o SMO atuam no treinamento e inferência dos modelos. Em outro cenário, considera-se que o Non-RT RIC atua no treinamento e o Near-RT RIC atua na inferência do modelo. Por fim, considera-se o cenário no qual o Non-RT RIC atua no treinamento e o O-DU e o O-CU atuam na inferência do modelo.

Inicialmente, esta seção apresenta os trabalhos que mencionam componentes O-RAN. Em seguida, apresentam-se trabalhos mais gerais que abordam os ataques baseados em aprendizado de máquina em RAN. Apesar de gerais, esses ataques podem ocorrer na arquitetura O-RAN e em outras arquiteturas de RAN aberta, por serem inerentes ao uso de aprendizado de máquina na interface sem fio.

4.4.1. Ataques na Arquitetura O-RAN

Habler *et al.* fornecem uma análise sistemática de aprendizado de máquina adversarial na arquitetura O-RAN [Habler et al., 2022]. Assim, os autores avaliam as ameaças na arquitetura utilizando uma ontologia de avaliação de riscos e propondo uma taxonomia para tal. O modelo de ameaças proposto define o modelo do atacante, os agentes de ameaça e seus objetivos. O modelo do atacante de Habler *et al.* lista diferentes capacidades que o atacante precisa conhecer ou acessar para realizar um ataque bem-sucedido. O trabalho propõe uma nomenclatura para classificar as diferentes capacidades, que podem ser de acesso ou de conhecimento. Na capacidade de acesso, o atacante possui acesso não-autorizado a componentes, como RICs, O-CUs e O-DUs, e consegue acessar componentes utilizados no fluxo de trabalho de aprendizado de máquina. Na capacidade de conhecimento, o atacante não possui acesso ao sistema, mas possui algum tipo de informação sobre os alvos. A Figura 4.6 mostra a nomenclatura proposta por Habler *et al.* [Habler et al., 2022].

As capacidades de acesso podem ser subdivididas em acesso ao modelo e acesso aos dados, como mostra a Figura 4.6. Na primeira categoria, o atacante tem o conhecimento da saída do modelo. Assim, é possível realizar uma requisição ao modelo e obter uma resposta. A saída pode ser o vetor de probabilidades da inferência, como a saída da

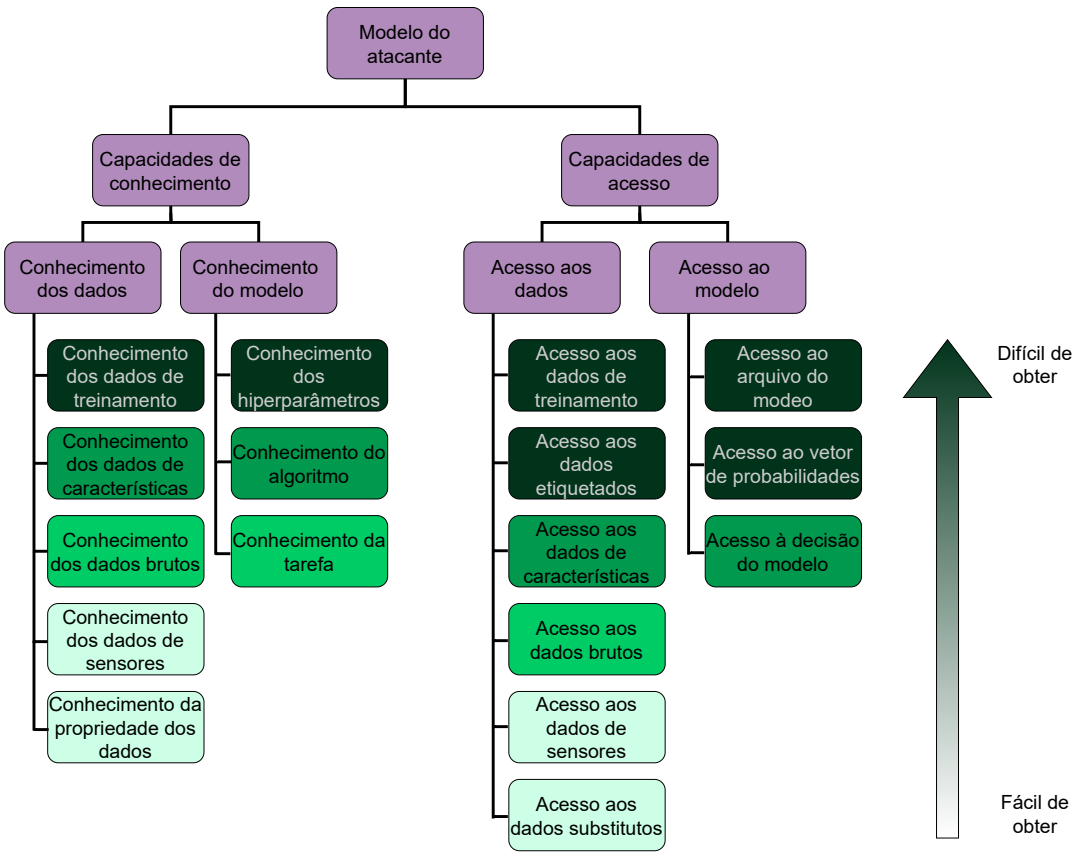


Figura 4.6. Modelo de atacante de Habler *et al.*, com as diferentes capacidades que o atacante pode ter. Os blocos referentes a cada capacidade estão marcados em verde. A cor mais escura representa as capacidades mais difíceis de o atacante obter. Adaptado de [Habler et al., 2022].

função *softmax* em uma DNN, ou apenas sua decisão, isto é, a classe inferida pelo modelo. No acesso aos dados, o atacante obtém os dados usados em alguma fase do fluxo de trabalho de aprendizado de máquina. Como exemplo, é possível citar dados brutos ou pré-processados usados no treinamento, vetores de características retirados do modelo, dados obtidos dos sensores, como UEs conectados à rede, e acesso a um conjunto de dados substituto, com as mesmas características e distribuição do utilizado no modelo.

As capacidades de conhecimento, por sua vez, consistem em conhecer informações da arquitetura e da implantação da infraestrutura O-RAN. Essas capacidades podem ser de conhecimento do modelo e conhecimento dos dados. No conhecimento do modelo, é possível conhecer seus hiperparâmetros, tais como a arquitetura da rede neural e a taxa de aprendizado, o algoritmo usado para o treinamento ou a tarefa realizada na inferência, por exemplo, classificação de tráfego. No conhecimento dos dados, o atacante pode conhecer informações sobre os dados de treinamento, os vetores de características, os dados brutos ou propriedades dos dados, como sua distribuição.

Os agentes de ameaça do modelo, que podem ter as capacidades mencionadas anteriormente, são [Habler et al., 2022]:

- **Desenvolvedor de aplicações de *software* O-RAN.** Responsável por desenvolver o software implantado nos componentes O-RAN, como rApps, xApps e mecanismos da O-DU e O-CU. O modelo assume que esse agente possui controle total do contêiner que executa o componente atacado. Entretanto, por atuar a nível de aplicação, suas ações e capacidades estão limitadas à aplicação específica desenvolvida pelo agente;
- **Desenvolvedor de infraestrutura de *software* O-RAN.** Responsável por desenvolver a infraestrutura O-RAN, por exemplo, seus mecanismos de escalonamento, de troca de mensagens, de banco de dados, e arcabouços de desenvolvimento de xApps e rApps. As ações e capacidades desse tipo de agente podem comprometer qualquer aplicação executada na infraestrutura;
- **Provedor da infraestrutura de *software* de conteinerização.** A infraestrutura de *software* consiste em implantações de contêineres, como as fornecidas pelo Kubernetes. Esse agente pode comprometer qualquer um dos contêineres que controla e, consequentemente, suas aplicações;
- **Provedor da infraestrutura de *hardware*.** Esse agente pode comprometer todo o *software* que executa na infraestrutura, visto que controla o seu *hardware*, como os servidores que hospedam os contêineres;
- **Equipamento de Usuário (UE).** Esse agente pode manipular seu próprio comportamento para atacar os modelos de aprendizado de máquina, por exemplo, manipulando dados de seus sensores para enganar sistemas de aprendizado contínuo;
- **Dados de terceiros e/ou provedores de modelo.** Modelos de O-RAN podem ser baseados em dados de terceiros [Couto et al., 2023a]. Esses dados podem ser maliciosos, comprometendo os modelos usados na infraestrutura. Além disso, modelos obtidos via provedores podem ser comprometidos.

Dentre os objetivos do atacante definidos por Habler *et al.*, é possível citar a manipulação, que compromete a integridade da rede. Como exemplo, é possível citar a geração de amostras adversariais para enganar os modelos. Outro objetivo é a negação de serviço, comprometendo a disponibilidade da rede. Nesse caso, o exemplo pode ser comprometer um modelo que classifica uma O-RU como a melhor a ser escolhida por todos os UEs. Dessa forma, o enlace de rádio será sobrecarregado, comprometendo todos os UEs que acessam essa O-RU. O terceiro objetivo definido é a divulgação de informações, comprometendo a privacidade na rede. Um exemplo é a exposição dos padrões de *handover* de um usuário a partir de análise das saídas ou vetores de características dos modelos.

O modelo de ameaças é proposto para mapear quais tipos de ataques, também chamados de famílias de ataques, são possíveis para cada modelo do atacante e agente de ameaça. Esses tipos são listados a seguir:

- **Evasão.** Consiste em induzir o modelo a fornecer saídas incorretas para uma entrada específica, comprometendo a integridade da RAN aberta. Como exemplo, um modelo de QoE (*Quality of Experience*) pode classificar um sinal de uma O-RU como excelente que, na verdade, seria classificado como ruim;
- **Envenenamento.** Similar à evasão, mas que gera uma saída errada para um conjunto de amostras, comprometendo a integridade e disponibilidade da RAN aberta. Por exemplo, um modelo de QoE pode alocar todos os UEs para um determinado O-RU, tornando-a indisponível;
- **Inferência de associação.** O atacante pode verificar se uma determinada amostra está no conjunto de treinamento, comprometendo a privacidade na RAN aberta. Ou seja, é possível saber, por exemplo, a localização de um UE por meio da identificação de padrões específicos do conjunto de treinamento;
- **Reconstrução dos dados.** O vazamento de informações do modelo, como vetores de características, pode permitir a reconstrução dos dados usados no seu treinamento, comprometendo a privacidade. Por exemplo, pode ser possível inferir um padrão de mobilidade entre os usuários de uma localidade da rede;
- **Extração de modelo.** Esse tipo de ataque compromete a privacidade por meio de extração de informações sobre o modelo. Essas informações são usadas para construir uma réplica do modelo e, assim, obter informações do ambiente. Por exemplo, é possível replicar um modelo de QoE e conseguir classificar o tráfego de uma determinada O-RU;
- **Exaustão de recursos.** Nesse tipo de ataque, que compromete a disponibilidade, o atacante torna a inferência do modelo mais lenta para um conjunto de amostras. Por exemplo, um modelo de QoE pode demorar um tempo proibitivo para classificar o tráfego e tomar decisões.

Shi e Sagduyu propõem um ataque a tarefas de fatiamento de rede executadas por uma xApp que utilize aprendizado por reforço [Shi e Sagduyu, 2021]. Os autores utilizam aprendizado por reforço em um UE malicioso para realizar pedidos por recursos de uma

RBS até que seus recursos sejam exauridos. Os autores assumem que o atacante conhece a recompensa do modelo utilizado pela RBS para oferecer recursos aos UEs. Assim, o trabalho aplica aprendizado por reforço para construir pedidos falsos que maximizem a recompensa do algoritmo de gerenciamento ao mesmo tempo, de forma a ocupar o máximo de recursos da RBS. Os autores demonstram uma queda significativa na recompensa obtida pelos algoritmos de gerenciamento sob ataque. Em outro trabalho, Shi e Sagduyu descrevem um ataque de inferência de associação em um modelo de classificação de sinais da rede sem fio [Shi e Sagduyu, 2023]. Nesse trabalho, a classificação é utilizada para autorizar o acesso de dispositivos em uma rede sem fio por meio de sinais da camada física. De acordo com os autores, esse tipo de classificação pode executar como uma xApp no Near-RT RIC. Assim, a xApp legítima pode utilizar um modelo treinado para classificar UEs como autorizados ou não. A autenticação pela camada física é uma estratégia que pode ser utilizada com dispositivos computacionalmente limitados, como os dispositivos da Internet das Coisas (*Internet of Things* – IoT), que não suportam mecanismos sofisticados de segurança. Para treinar o modelo, os UEs autorizados enviam previamente sinais para o provedor de serviços, por exemplo, uma O-RU. Devido às características do *hardware* de rádio, o sinal é enviado com um deslocamento de fase específico de cada equipamento [Shi e Sagduyu, 2023]. Além disso, o canal utilizado entre o usuário e a O-RU possui características únicas de ganho e deslocamento de fase, o que também diferencia os UEs. Dessa forma, um processo de autenticação pelos sinais da camada física consiste em um modelo receber o deslocamento de fase e potência do sinal de um UE e classificá-lo como "autorizado" e "não-autorizado".

O ataque descrito no trabalho de Shi e Sagduyu é do tipo caixa preta, uma vez que consiste em o atacante observar diferentes resultados de classificação para diversos sinais do meio sem fio [Shi e Sagduyu, 2023]. A partir dessa observação, treina-se um modelo substituto, que pode ser utilizado para o atacante produzir sinais que possam burlar o mecanismo de autenticação. Para construir o modelo, o atacante se beneficia do sobreajuste do modelo original. Como o modelo original não é perfeitamente generalista, é possível diferenciar se uma determinada amostra está presente no conjunto de treinamento, isto é, se é oriunda de um usuário legítimo, e então construir o substituto. Para evitar o ataque, Shi e Sagduyu propõem um esquema de defesa que tenta construir um modelo próximo ao do atacante, um modelo sombra. A ideia é adicionar ruído ao sinal para diminuir a acurácia do modelo sombra e, consequentemente, do modelo do atacante.

O Envenenamento de Migração de Portadores (*Bearer Migration Poisoning* – BMP) [Soltani et al., 2023] é um ataque ao RIC quase tempo-real que dispara um processo de migração de portadora de forma maliciosa. O contexto de portadoras se refere a um processo de sinalização transmitido por meio da interface E1. Essa informação é necessária para estabelecer requisitos de recursos e encaminhamento de mensagens dos serviços de plano de usuário. O RIC quase tempo-real é o elemento da arquitetura O-RAN responsável por estabelecer as informações e alterar o contexto de portadoras. O objetivo do atacante é modificar o caminho do plano de tráfego e causar anomalias na rede, atacando o modelo de aprendizado de máquina do RIC, que toma decisões automáticas sobre o plano de tráfego da rede. Os autores que propõem o ataque assumem que um modelo no qual o atacante é capaz de comprometer dispositivos com vírus, cavalos de troia e *malwares* ou ser um usuário interno ao sistema. Assim, controlando dispositi-

vos internos, o adversário falsifica mensagens do protocolo de descoberta da camada de enlace (*Link Layer Discovery Protocol* – LLDP), que são propagadas para o RIC, que é forçado a tomar decisões incorretas. Esse ataque utiliza o princípio de exemplos adversariais para gerar mensagens incorretas, utilizadas como entrada do modelo de aprendizado de máquina, para prevenir decisões corretas do controlador de rádio inteligente.

4.4.2. Outros Ataques à Interface Sem Fio

Ataques à interface sem fio podem ocorrer em diversas arquiteturas RAN. Assim, há na literatura trabalhos que, independentemente da existência de uma RAN aberta ou não, abordam ataques baseados em aprendizado de máquina na interface sem fio. A camada física da RAN aberta pode utilizar inteligência artificial para sensoriamento de espectro, classificação automática de sinais e alocação ótima de blocos de recurso. Entretanto, um atacante que observa o meio e envia um sinal malicioso gera perturbações na entrada de modelos em treinamento ou em produção para efetuar ataques. Os ataques e defesas utilizados em outros problemas de aprendizado de máquina não são necessariamente aplicáveis em comunicações sem fio, devido à natureza dinâmica do canal de comunicação [Adesina et al., 2023]. Os trabalhos relacionados a esse tópico são muito heterogêneos quanto ao tipo de tarefa de aprendizado, aos tipos de ataque, aos mecanismos de defesa e às tecnologias de radiofrequência (RF) utilizadas. A discussão a seguir oferece um panorama dos tipos mais comuns de ataques e defesas em diversas aplicações de aprendizado de máquina na interface sem fio.

Restuccia *et al.* formulam um problema de otimização para calcular um sinal modulado em fase e quadratura (I/Q) malicioso [Restuccia et al., 2020]. O atacante envia sinais que interferem com os sinais dos usuários legítimos para trocar o resultado de um modelo para classificação de espectro no receptor. Os autores realizam experimentos com conjuntos de dados reais e demonstram a eficácia do treinamento adversarial, um mecanismo de defesa baseado em treinar o modelo já usando amostras maliciosas. Karunaratne *et al.* considera um receptor que utiliza um modelo de aprendizado profundo, denominado autenticador, para decidir se as amostras I/Q vêm de um transmissor autorizado a transmitir [Karunaratne et al., 2021]. O atacante utiliza uma Rede Generativa Adversarial (*Generative Adversarial Network* – GAN), na qual o discriminador é o autenticador e o gerador é uma rede que deve aprender a gerar amostras que imitem as de um usuário autêntico. O trabalho realiza experimentos com um Rádio Definido por *Software* comercial (*Software Defined Radio* – SDR), mas não apresenta nenhum mecanismo de defesa.

Os ataques de cavalo de troia à interface sem fio consistem em realizar um envenenamento de dados ao longo do treinamento do modelo de aprendizado de máquina associado ao gerenciamento da interface sem fio [Davaslioglu e Sagduyu, 2019]. O objetivo do ataque é ativar um comportamento desejado pelo atacante durante a execução do modelo. Para isso, o atacante necessita ter acesso ao modelo de treinamento e ao conjunto de dados utilizado para alterar rótulos das amostras de treinamento e criar os padrões que mudam o comportamento do modelo. Davaslioglu *et al.* apresentam um ataque de cavalo de troia a uma rede neural que classifica a técnica de modulação a partir do sinal I/Q recebido [Davaslioglu e Sagduyu, 2019]. O atacante adiciona amostras envenenadas ao conjunto de treinamento e, posteriormente, transmite sinais com a mesma fase para provocar erros. O aumento da base de dados e o uso de testes estatísticos na inferên-

cia reduzem a eficácia do ataque. No trabalho de Wang *et al.*, os transmissores utilizam um modelo de aprendizado por reforço profundo para escolher a frequência do melhor canal na hora de acessar o meio [Wang et al., 2020b]. O atacante também utiliza um mecanismo adaptativo para ocupar o canal selecionado pelo modelo da vítima e bloquear sua transmissão, reduzindo a acurácia do aprendizado por reforço. Uma das contribuições do trabalho é propor mecanismos de defesa que forçam a escolha de um canal pior, dificultando o atacante em encontrar padrões de ataque.

A Tabela 4.7 compara os trabalhos discutidos nessa seção, classificando-os entre os tipos definidos por Habler *et al.* e descritos na Seção 4.4.1.

Tabela 4.7. Trabalhos sobres ataques à RAN baseados em aprendizado de máquina.

Trabalho	Fase do ataque	Acesso ao modelo	Tarefa de aprendizado	Tipo de ataque	Defesa
[Shi e Sagduyu, 2021]	Treinamento Inferência	Caixa branca	Alocação de recursos	Exemplos Adversariais	-
[Shi e Sagduyu, 2023]	Inferência	Caixa preta	Autenticação de sinal	Inferência da associação	Aprendizado adversarial
[Soltani et al., 2023]	Inferência	Caixa preta	Disparo de migração de portadores	Exemplos adversariais	Aprendizado adversarial
[Restuccia et al., 2020]	Inferência	Caixa branca	Classificação de espectro	Evasão	Aprendizado adversarial
[Karunaratne et al., 2021]	Inferência	Caixa preta	Autenticação de sinal	Extração de modelo Evasão	-
[Davaslioglu e Sagduyu, 2019]	Treinamento	Caixa preta	Classificação de modulação	Envenenamento	Testes estatísticos
[Wang et al., 2020b]	Treinamento Inferência	Caixa preta	Seleção de canal para MAC	Evasão	Teoria de controle, políticas ortogonais

4.5. Desafios e Tendências de Pesquisa

Esta seção foca desafios de segurança em aberto[4] e tendências de pesquisa. O primeiro desafio está relacionado à falta de um arcabouço de segurança abrangente e universal para RAN aberta que atenda a todos os requisitos de segurança e permita o controle da segurança durante o ciclo de vida da RAN aberta[5]. As soluções existentes concentram-se em questões específicas de segurança em diferentes planos da arquitetura lógica, mas nenhuma é capaz de defender contra todas as ameaças ou satisfazer todos os requisitos. Outro desafio está relacionado à necessidade de aprimorar as soluções de alocação e gerenciamento de recursos de rádio para assegurar a disponibilidade do sistema da RAN aberta [Wang et al., 2021]. O gerenciamento seguro de recursos de espectro, incluindo detecção, compartilhamento e alocação de espectro, representa um desafio significativo. As técnicas atuais de detecção de espectro, como métodos de detecção de energia, são insuficientes para lidar com todas as ameaças de espectro no complexo ambiente de comunicação da RAN aberta. O terceiro desafio é a preservação da privacidade na RAN aberta [Singh e Khoa Nguyen, 2022]. A preservação da privacidade é essencial para garantir a conformidade com as leis de proteção de dados pessoais, como a Lei Geral de Proteção de Dados (LGPD) no Brasil e o Regulamento Geral de Proteção de Dados (*General Data Protection Regulation* – GDPR) na União Europeia. A RAN aberta emprega

[4]Disponível em `https://ntia.gov/sites/default/files/publications/open_ran_security_report_full_report_0.pdf`.

[5]Disponível em `https://www.vodafone.com/sites/default/files/2023-02/joint-mou-white-paper-mwc-2023.pdf`.

dados pessoais dos usuários para a prestação de serviços personalizados e otimizados, como localização e identidades dos usuários. O gerenciamento da confiança na RAN aberta [Ramezanpour e Jagannath, 2022] é um desafio importante, uma vez que a rede aberta estabelece a interoperação entre *hardware* e *software* de diferentes fornecedores. Estabelecer um ambiente confiável é crucial para a segurança da cooperação entre diferentes operadoras e fornecedores. O quinto desafio de segurança que se destaca na RAN aberta é a segurança da camada física devido à natureza aberta e desacoplada dessa camada [Polese et al., 2023]. Por fim, ainda existem desafios de segurança para a camada de virtualização e contêineres, como garantir o isolamento entre ambientes e assegurar o desempenho adequado, além de desafios relacionados às interfaces abertas. As interfaces abertas implementam serviços de mensageria que requerem atenção para evitar a injeção de mensagens e para garantir a integridade e confidencialidade das mensagens trocadas. Para abordar essas lacunas de pesquisa, tendências promissoras incluem: (i) desenvolver um arcabouço de segurança abrangente e universal para a RAN aberta; (ii) desenvolver um mecanismo de autenticação contínua, eficiente e seguro, baseado no conceito de "confiança-zero" (zero-trust)[Ramezanpour e Jagannath, 2022], para acesso e comutação de usuários na RAN aberta, considerando cenários de migração de usuários entre operadoras; (iii) desenvolver mecanismos de reputação que permitam o compartilhamento seguro de recursos entre diferentes operadoras; (iv) projetar um mecanismo de preservação de privacidade para RAN aberta, com o objetivo de evitar vazamento de dados e garantir a conformidade com as leis de proteção de dados pessoais; e (v) propor mecanismos de isolamento seguros para ambientes de virtualização e contêineres, visando garantir a segurança geral da infraestrutura virtualizada. As tendências de pesquisa têm como objetivo o desenvolvimento de uma RAN aberta abrangente e segura, melhorando sua resiliência contra ameaças de segurança e atendendo aos requisitos das arquiteturas de referência para os casos de uso previstos. A seguir, os desafios, tendências e oportunidades de pesquisa são detalhados.

4.5.1. Desafios de Pesquisa

Embora existam iniciativas com o intuito de gerar especificações e viabilizar a criação de RANs abertas, interoperáveis, virtualizadas e inteligentes, existem desafios a serem superados, conforme ilustrados na Figura 4.7. Esta seção discute os desafios de pesquisa no gerenciamento e orquestração de serviços em RAN aberta.

Virtualização da Infraestrutura

A arquitetura de referência O-RAN para as RANs abertas enfrenta o desafio de virtualizar a infraestrutura para fornecer serviços mais avançados e ágeis aos usuários finais [Niknam et al., 2022]. A virtualização é uma técnica que permite criar ambientes com interfaces semelhantes às reais, isolados, sobre uma infraestrutura física compartilhada, permitindo que diferentes serviços e aplicativos sejam executados. A virtualização permite que diferentes serviços e aplicativos sejam executados em um ambiente isolado dos demais, melhorando a eficiência de alocação de recursos e a escalabilidade da rede, possibilitando à rede oferecer os serviços avançados e ágeis. No entanto, a virtualização da infraestrutura pode ser um desafio significativo na arquitetura O-RAN. Isso ocorre porque a virtualização requer uma infraestrutura de *hardware* e *software* que possa suportar a execução de diferentes serviços nos ambientes virtuais. Além disso, a virtualização pode

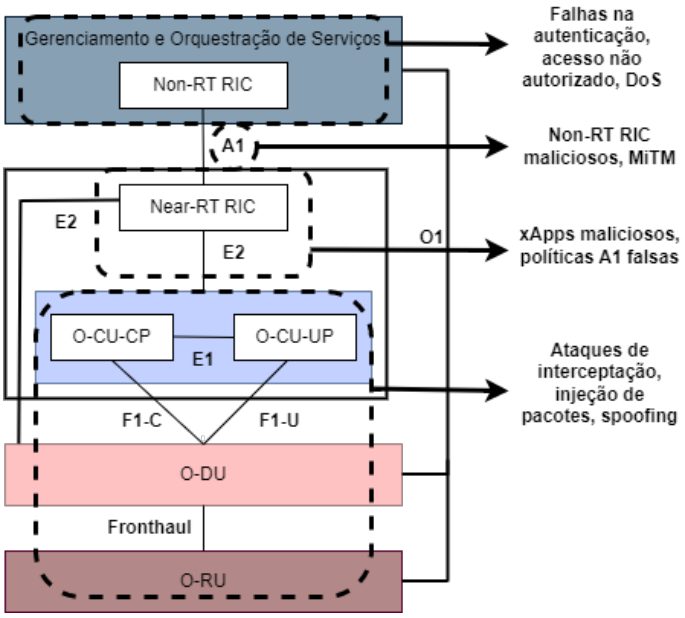


Figura 4.7. Desafios de segurança na arquitetura O-RAN. As aplicações e interfaces em uma rede O-RAN estão sujeitas a sofrerem certos tipos de ataque como de injeção de pacotes ou *spoofing*, mas também podem sofrer com vulnerabilidades na má autenticação, que podem levar usuários maliciosos a comprometerem a rede.

aumentar a complexidade da rede, o que pode tornar a implantação e a manutenção mais difíceis. Para superar esse desafio, é importante desenvolver novas técnicas e tecnologias para virtualizar a infraestrutura de forma eficiente e eficaz. Isso inclui o uso de tecnologias de virtualização de rede, como NFV. Além disso, é importante desenvolver novas técnicas de gerenciamento e orquestração para garantir que diferentes serviços e aplicativos possam ser executados de forma eficiente e segura em um ambiente virtualizado. Contudo, ressalta-se que a virtualização amplia a superfície de ataque da RAN, já que introduz novos componentes de *software* e realiza o compartilhamento de *hardware*. Assim, é também essencial o desenvolvimento de ferramentas de acompanhamento do ciclo de vida de funções virtuais de rede para mitigar possíveis riscos de segurança durante a execução da função.

Suporte a Diferentes Requisitos de Qualidade de Serviço (QoS)

Um dos desafios associados à implementação da O-RAN é projetar uma arquitetura autônoma orientada a serviços que possa suportar diferentes requisitos de Qualidade de Serviço (QoS). Isso é importante porque diferentes aplicativos têm requisitos de QoS diferentes e a rede deve ser capaz de atender a esses requisitos de forma eficiente [Xu et al., 2021b].

Para enfrentar esse desafio, a arquitetura O-RAN deve ser projetada para ser flexível e adaptável. Isso pode ser alcançado por meio da virtualização da rede, através da execução de diferentes funções da rede em ambientes virtuais, em vez de *hardware* dedicado e de propósito específico. Nesse sentido, adicionar novas funções à rede e atualizá-las conforme necessário são tarefas facilitadas pela virtualização. A arquitetura O-RAN deve ser orientada a serviços, o que significa que a rede deve ser projetada para fornecer

serviços específicos para diferentes aplicativos [Xu et al., 2021b]. Isso pode ser alcançado por meio da segmentação da rede em diferentes fatias de rede (*slices*), cada uma projetada para atender a requisitos específicos de QoS de diferentes serviços, conforme apresentado na Figura 4.8, que mostra um exemplo de rede separada em fatias que receberão recursos de acordo com o perfil de usuários, otimizando a utilização da rede. Outro desafio é garantir que a rede possa gerenciar e controlar o tráfego de forma eficiente, para garantir que os requisitos de QoS sejam atendidos. Isso pode ser alcançado por meio de algoritmos de gerenciamento de tráfego inteligentes, que priorizam o tráfego de serviços críticos e garantem que a largura de banda seja alocada de forma eficiente [Xu et al., 2021b, Das et al., 2017]. Assim, projetar uma arquitetura autônoma orientada a serviços, que possa suportar diferentes requisitos de Qualidade de Serviço (QoS), é um desafio importante na implementação O-RAN que impacta diretamente a segurança da RAN. Isso pode ser alcançado por meio da virtualização da rede, segmentação da rede em diferentes fatias e algoritmos de gerenciamento de tráfego inteligentes.

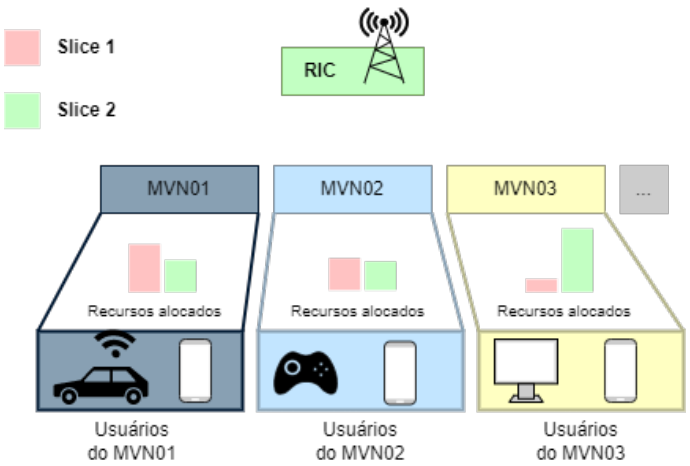


Figura 4.8. Representação de uma forma de segmentação da rede em diferentes fatias, cada uma projetada para atender a um propósito específico. Adaptado de [Xu et al., 2021b].

Controle Inteligente em Tempo Real

O desafio do controle em tempo real na arquitetura O-RAN refere-se à necessidade de garantir que as decisões de controle sejam tomadas em tempo hábil para atender aos requisitos de desempenho das aplicações em tempo real. Isso é particularmente importante para aplicações que exigem uma resposta rápida e confiável da rede, como a automação industrial, a realidade virtual, holografia e a telemedicina. Para atender a esses requisitos, propõe-se a implementação de um terceiro laço de controle em tempo real (RT RIC) a ser situado no O-CU, O-DU ou O-RU. Esse laço de controle em tempo real pode hospedar qualquer tipo de funcionalidade de controle da RAN de camada inferior, que são chamadas de zApps, aplicações de controle de terceiros hospedadas no RT RIC. A introdução de inteligência artificial (IA) permitirá lidar com a complexidade da camada física, dos recursos heterogêneos e dos ambientes operacionais para construir RANs altamente configuráveis e gerenciáveis [Azariah et al., 2022]. No entanto, a implementação de um laço de controle em tempo real apresenta vários desafios de segurança. Um dos principais desafios à disponibilidade é a capacidade de processamento dos nós (O-CU/O-DU) que

precisam ser capazes de lidar com grandes quantidades de dados em tempo real. Além disso, a eficiência energética é uma preocupação importante, pois a implementação de um laço de controle em tempo real pode aumentar o consumo de energia. Outro desafio é a capacidade dos modelos de IA de fornecer decisões em uma escala de tempo de submilissegundos, devido a restrições de tempo de resposta, o que pode ser difícil de alcançar em ambientes de rede em tempo real. Finalmente, a quantidade de sobrecarga de sinalização para o controle de camada física de baixo nível também é um fator crítico que pode ter um impacto adverso no cumprimento dos requisitos de redes celulares de próxima geração com baixa latência [Azariah et al., 2022]. Dessa forma, o desafio do controle em tempo real na arquitetura O-RAN refere-se à necessidade de garantir que as decisões de controle sejam tomadas em tempo hábil para atender aos requisitos de disponibilidade e desempenho das aplicações em tempo real, enquanto se lida com a complexidade da camada física, dos recursos heterogêneos e dos ambientes operacionais. A implementação de um laço de controle em tempo real apresenta vários desafios. Superar esses desafios é fundamental para garantir que a arquitetura O-RAN possa fornecer redes celulares de próxima geração com baixa latência e alto desempenho para aplicações em tempo real. Assim, o controle em tempo real impacta a segurança da rede em duas vertentes principais: a disponibilidade e a confiabilidade. A disponibilidade é afetada pelos desafios referentes ao atendimento dos requisitos de tempo de resposta em escala de submilissegundos. A confiabilidade é afetada pelas necessidade de validação dos modelos de IA para que as ações tomadas em tempo real estejam de acordo com o comportamento esperado da rede. Por fim, ao se considerar o controle em tempo real, a confiabilidade também é diretamente impactada pela explicabilidade dos modelos de IA, já que muitos modelos não são possíveis de serem compreendidos devido à sua complexidade e ao grande número de parâmetros.

Interoperabilidade

O desafio de garantir a interoperabilidade entre diferentes fornecedores e a integração de diferentes tecnologias na arquitetura O-RAN é um dos principais enfrentados na implementação da arquitetura. A interoperabilidade é a capacidade de diferentes sistemas e tecnologias trabalharem juntos de forma eficiente e eficaz, enquanto a integração é a capacidade de diferentes sistemas e tecnologias trabalharem juntos de forma harmoniosa e sem problemas [Wang et al., 2020a, Thiruvasagam et al., 2023]. A interoperabilidade é essencial para permitir que diferentes componentes da arquitetura O-RAN, como as O-RUs, O-DUs e O-CU possam se comunicar e trabalhar juntas de forma eficiente e confiável [Thiruvasagam et al., 2023]. Sem padrões claros e precisos para as interfaces entre esses componentes, pode haver problemas de compatibilidade, interoperabilidade e vulnerabilidades, o que pode levar a atrasos na implantação da arquitetura O-RAN, a custos mais elevados e riscos de segurança. No entanto, garantir a interoperabilidade e a integração segura entre diferentes fornecedores e tecnologias pode ser um desafio significativo. Isso, porque diferentes fornecedores podem ter diferentes abordagens e tecnologias para implementar os componentes da rede. Além disso, as tecnologias podem ter diferentes requisitos de desempenho e segurança, o que pode tornar a integração mais difícil. A falta de padronização de interfaces abertas pode levar ao problema de fornecedor único, em que os fornecedores de equipamentos de rede têm um controle excessivo sobre a arquitetura O-RAN e, então, cobram preços mais elevados por seus produtos e serviços

[Thiruvasagam et al., 2023], além de gerar o aprisionamento (*lock in*) a seus produtos. Isso pode levar a um mercado menos competitivo e menos inovador, o que prejudica a adoção da arquitetura O-RAN e a evolução da segurança da arquitetura. Para superar esse desafio, é importante estabelecer padrões e protocolos comuns para garantir a interoperabilidade e a integração bem sucedidas entre diferentes fornecedores e tecnologias. Além disso, é importante promover a colaboração e a padronização entre diferentes fornecedores para garantir que componentes da rede possam trabalhar juntos de forma harmoniosa. É importante que as organizações de padronização trabalhem em conjunto com outras organizações, como a 3GPP, para garantir a harmonização dos padrões e a interoperabilidade entre as diferentes arquiteturas [O-RAN Alliance, 2021]. Isso é essencial para garantir que a arquitetura O-RAN possa ser integrada com outras arquiteturas de rede existentes e futuras, permitindo que as operadoras de rede possam escolher a melhor solução para suas necessidades específicas [O-RAN Alliance, 2021]. A padronização de interfaces abertas é um desafio importante para a arquitetura O-RAN, mas pode ser abordada por meio do estabelecimento de padrões abertos e acessíveis a todos os fornecedores e desenvolvedores de *software* [Thiruvasagam et al., 2023]. As organizações de padronização devem trabalhar em conjunto para garantir a harmonização dos padrões e a interoperabilidade entre as diferentes arquiteturas [Thiruvasagam et al., 2023, O-RAN Alliance, 2021]. O uso de tecnologias de código aberto também pode ser uma solução para esse desafio [O-RAN Alliance, 2021]. O uso de padrões abertos permitem a escrutinação do código e validação por uma comunidade de desenvolvedores das arquiteturas e tecnologias adotadas. Com isso, há uma mitigação de riscos à segurança e de interoperabilidade.

Privacidade e Proteção dos Dados

A arquitetura O-RAN enfrenta o desafio de garantir a privacidade do usuário e a proteção de dados sensíveis. Isso ocorre porque a arquitetura O-RAN é baseada em uma abordagem aberta e virtualizada, o que significa que diferentes componentes da rede podem ser fornecidos por diferentes fornecedores e podem ser executados em um ambiente virtualizado sobre *hardware* de propósito geral [Firoozjaei et al., 2017]. O Grupo de Trabalho 11 da O-RAN Alliance tem estudado os aspectos de segurança da arquitetura O-RAN, incluindo modelagem de ameaças, avaliação de riscos, requisitos de segurança, mecanismos e protocolos de segurança para atender aos requisitos, e testes de segurança para validação e avaliação [O-RAN Alliance, 2020]. No entanto, apenas algumas interfaces e funções de rede relacionadas à segurança foram estudadas, e há muito espaço para explorar e analisar os aspectos de segurança para várias entidades e interfaces [O-RAN Alliance, 2020]. Por exemplo, as funções de rede, como O-CU-CP, O-CU-UP, O-DU, O-RU, e as interfaces, como E2, R1, Y1, O-Cloud Notification e Cooperative Transport Interface, precisam ser estudadas em relação à segurança [Thiruvasagam et al., 2023]. Como mostrado na Figura 4.7, diferentes serviços e interfaces de rede estão sujeitos a ataques e outros tipos de vulnerabilidades. Os aspectos de segurança da infraestrutura de nuvem compartilhada, integração de nós e funções de rede, *software* de código aberto e gerenciamento seguro do ciclo de vida das funções de rede também precisam ser estudados [O-RAN Alliance, 2020]. Com isso, a virtualização e a abertura da arquitetura O-RAN podem aumentar o risco de violações de privacidade e segurança, especialmente quando se trata de dados sensíveis do usuário. Por exemplo, a implantação de uma rede O-RAN pode envolver o compartilhamento de recursos de

processamento e armazenamento entre diferentes usuários, o que pode aumentar o risco de acesso não autorizado a dados sensíveis. Para superar esse desafio, é importante implementar medidas de segurança e privacidade robustas na arquitetura O-RAN. Isso pode incluir o uso de técnicas de criptografia para proteger dados sensíveis em trânsito e em repouso, bem como o uso de técnicas de autenticação e autorização para garantir que apenas usuários autorizados possam acessar dados sensíveis. É importante implementar medidas de segurança e privacidade em todos os componentes da rede, incluindo rádios, controladores e elementos de orquestração. Isso pode incluir o uso de técnicas de *software* de segurança, como *firewalls* e detecção de intrusão, bem como o uso de técnicas de *hardware* de segurança, como módulos de segurança de *hardware*. Por fim, é importante estabelecer padrões e protocolos comuns para garantir a segurança e a privacidade em toda a arquitetura O-RAN. Isso pode incluir o desenvolvimento de padrões de segurança e privacidade para diferentes componentes da rede, bem como o desenvolvimento de protocolos de segurança e privacidade para garantir a interoperabilidade entre diferentes componentes da rede. A segurança e privacidade são desafios importantes para a arquitetura O-RAN, e é necessário continuar a estudar e desenvolver mecanismos de segurança para a arquitetura. É essencial que as organizações que implementam a arquitetura adotem melhores práticas de segurança e privacidade adequadas e implementem políticas de segurança e privacidade para garantir a proteção das informações confidenciais e o controle dos usuários sobre suas informações pessoais.

Gerenciamento de recursos e otimização de desempenho

O gerenciamento de recursos e a otimização de desempenho são desafios importantes em muitos sistemas computacionais, incluindo sistemas de rede e telecomunicações. Em sistemas de rede, como a arquitetura O-RAN, o gerenciamento de recursos refere-se à alocação e utilização eficiente de recursos de rede, como largura de banda, capacidade de processamento e armazenamento. A otimização de desempenho refere-se à melhoria do desempenho do sistema, como a redução do tempo de latência e a melhoria da qualidade de serviço [Niknam et al., 2022, Kavehmadavani et al., 2023]. Nas redes de acesso via rádio, o gerenciamento de recursos e a otimização de desempenho são desafios complexos devido à natureza distribuída e heterogênea dos sistemas. A arquitetura O-RAN, por exemplo, é composta por vários componentes que têm requisitos de recursos, segurança e desempenho diferentes, e a alocação e utilização eficiente desses recursos é essencial para garantir o desempenho adequado do sistema. Para garantir a qualidade de serviço, o desempenho e a disponibilidade, é necessário garantir que as aplicações críticas tenham latência baixa e previsível [Abdalla et al., 2022]. Isso é particularmente importante para aplicações em tempo real que exigem uma resposta rápida e confiável da rede. A latência determinística é a capacidade de garantir que um pacote de dados chegue ao seu destino dentro de um tempo previsível e consistente. Na RAN aberta, isso requer a implementação de mecanismos de controle de latência em toda a rede, incluindo a RAN, a rede de transporte e a rede de núcleo [Abdalla et al., 2022, O-RAN Alliance, 2021]. Para garantir a latência determinística, é necessário minimizar a variação da latência em toda a rede. Isso pode ser alcançado por meio de técnicas como a alocação de recursos de rede dedicados para aplicações críticas, a priorização de tráfego em tempo real e a implementação de mecanismos de controle de congestionamento [Abdalla et al., 2022]. Contudo, ataques de injeção de tráfego podem interferir na previsibilidade do sistema e,

portanto, são uma ameaça ao gerenciamento otimizado de recursos, pois desviam do modelo de funcionamento do sistema. Por fim, o gerenciamento de recursos e a otimização de desempenho são desafios para a segurança da RAN, pois introduzem meios de controle automatizados que podem ser subvertidos por injeção de tráfego e comportamentos anômalos da rede. Para enfrentar esses desafios, é necessário desenvolver soluções adaptáveis, escaláveis e eficientes que possam lidar com a natureza distribuída e heterogênea dos sistemas e com as mudanças dinâmicas nas demandas de recursos e desempenho.

4.5.2. Tendências de Pesquisa

Tendências atuais de pesquisa focada em segurança das arquiteturas de RAN aberta incluem o desenvolvimento de novas técnicas de transporte (*fronthaul*) para atender aos requisitos de ultra-redução de latência e comunicação confiável (uRLLC)e aprimorar a virtualização da infraestrutura para fornecer serviços mais avançados e ágeis aos usuários finais. Continuar a explorar soluções de segurança para garantir a privacidade do usuário e a proteção de dados sensíveis na arquitetura O-RAN também despontam como tendências de pesquisa [Niknam et al., 2022, Wang et al., 2020a].

Novas Técnicas para a Rede de Transporte (*Fronthaul*)

Fronthaul é a conexão entre a estação base e o processador de sinalização, que é responsável por processar os sinais de rádio, sendo crítica para a qualidade de serviço, pois o transporte do sinal deve ser feito com baixa latência e alta confiabilidade.No entanto, a *fronthaul* baseada em *Common Public Radio Interface* (CPRI) tem limitações em termos de largura de banda e latência, o que pode dificultar a implantação de aplicativos uRLLC. Para superar esse desafio, é necessário desenvolver novas técnicas de transporte que possam atender aos requisitos de largura de banda e latência para aplicações uRLLC. Uma das soluções propostas é o uso de Ethernet como uma alternativa ao CPRI para a rede de transporte, por ter a capacidade de atender aos requisitos de largura de banda e latência para aplicações uRLLC. No entanto, a rede de transporte baseada em Ethernet também apresenta desafios, como a necessidade de garantir a qualidade do serviço e a segurança dos dados. Portanto, é necessário desenvolver novas técnicas de gerenciamento e orquestração para garantir que o transporte baseado em Ethernet possa atender aos requisitos de largura de banda e latência para aplicativos uRLLC, mas também solucionar desafios clássicos do Ethernet em prover qualidade de serviço e segurança aos dados.

Aprimoramento da Virtualização

Aprimorar a virtualização da infraestrutura contribui para fornecer serviços mais avançados e ágeis aos usuários finais na arquitetura O-RAN [Niknam et al., 2022, Singh e Khoa Nguyen, 2022]. Para aprimorar a virtualização da infraestrutura na arquitetura O-RAN, é necessário desenvolver novas técnicas e tecnologias para virtualizar a infraestrutura de forma eficiente e eficaz, capazes de abarcar o uso de tecnologias de NFV. O desenvolvimento de mecanismos para aprimorar o desempenho e a segurança da virtualização de redes é uma tendência de pesquisa atual. Kawahara *et al.* propõem um método para melhorar a taxa de transferência e minimizar a sobrecarga do plano de controle em plataformas de nuvem virtualizadas. Ao enfrentar os desafios relacionados ao encapsulamento e ao roteamento de comunicação, a proposta oferece instâncias de rede virtual eficientes em centros de dados, garantindo desempenho ideal em dispositivos

padrão e em interfaces de redes virtuais [Kawahara et al., 2019]. Sadok *et al.* propõem Ensō, uma nova interface por fluxo para a ligação da interface de rede à aplicação. Ensō evita *buffers* de tamanho fixo e estrutura a comunicação como um fluxo que pode ser usado para enviar tamanhos de dados arbitrários.

Análises da Arquitetura de RAN Aberta

A possibilidade de pesquisa sobre análise da arquitetura de RAN aberta e suas principais características envolve a investigação detalhada da arquitetura O-RAN e inclui a evolução da RAN, que é a base para a arquitetura O-RAN, bem como as funções da RAN desagregadas e as interfaces entre elas. A análise da evolução da arquitetura da RAN inclui a revisão de padrões anteriores, como 3GPP, e a comparação com a arquitetura O-RAN. A revisão da arquitetura visa entender as principais diferenças entre as arquiteturas e as vantagens e desvantagens de cada uma. A análise das funções RAN desagregadas consiste na revisão das principais funções RAN, como processamento de sinalização, processamento de dados, gerenciamento de recursos de rádio e gerenciamento de mobilidade, e como essas funções podem ser desagregadas em componentes menores e independentes. Isso pode ajudar a entender como a arquitetura O-RAN pode ser mais flexível e escalável do que as arquiteturas RAN tradicionais. A análise das interfaces entre as funções da RAN desagregadas revisa as interfaces definidas pela O-RAN Alliance. O objetivo dessa área de pesquisa é entender como as funções da RAN desagregadas podem ser interconectadas de forma eficiente e eficaz.

Atividades de Padronização da O-RAN Alliance

A possibilidade de pesquisa sobre o estudo das atividades de padronização da O-RAN Alliance envolve uma análise detalhada das atividades de padronização em andamento na organização, bem como suas implicações para a indústria de telecomunicações. Essa pesquisa pode ser realizada por meio da revisão de documentos e especificações técnicas publicados pela O-RAN Alliance, que descrevem as atividades de padronização em andamento e as especificações técnicas que estão sendo desenvolvidas para promover a interoperabilidade e a implementação de RANs abertas. A pesquisa envolve a análise das atividades de padronização em andamento, como a definição de interfaces abertas e a especificação de requisitos de implementação. Paralelamente, a pesquisa sobre a revisão das iniciativas de implementação em andamento visa identificar as implicações regulatórias e de mercado da adoção dessas tecnologias e identificar as melhores práticas e lições aprendidas na implementação de tecnologias de RAN aberta.

Utilização de Aprendizado Federado

O aprendizado federado é uma técnica de aprendizado de máquina distribuída que permite que várias instância de computação de borda cooperem para treinar um modelo de aprendizado de máquina sem transferir seus dados de treinamento [Konečnỳ et al., 2016]. Essa técnica é particularmente útil em sistemas de RAN aberta, nos quais a computação de borda é distribuída em várias instâncias e a transferência de dados é limitada devido a restrições de comunicação. No contexto do problema de alocação ótima de recursos, o aprendizado federado pode ser utilizado para permitir que várias instâncias de computação de borda cooperem para treinar um modelo sem transferir dados privados de treinamento [Chen e et al., 2021, Abouaomar et al., 2022]. Isso permite que o modelo seja

treinado de forma mais eficiente e privada, pois os dados de treinamento permanecem na borda e não são transferidos para um servidor centralizado. Para implementar o aprendizado federado no contexto do problema de alocação ótima de recursos, é necessário selecionar um conjunto de nós de treinamento locais em cada iteração global de aprendizado federado. Esses nós de treinamento locais são responsáveis por treinar o modelo em seus próprios dados de treinamento e enviar as atualizações do modelo para um servidor centralizado para agregação. A seleção dos nós de treinamento locais deve ser feita de forma a maximizar a eficiência do processo de treinamento e minimizar o custo de recursos. É necessário alocar recursos de forma ótima para os nós de treinamento locais selecionados. Isso inclui a alocação de recursos de computação, armazenamento e comunicação. A alocação ótima de recursos deve levar em consideração as restrições de comunicação e as limitações de recursos de cada nuvem de borda. Dessa forma, o aprendizado federado é uma tendência de pesquisa para solucionar o problema de alocação ótima de recursos em sistemas de RAN aberta, permitindo que várias instâncias de computação de borda cooperem para treinar um modelo de aprendizado de máquina de forma eficiente e privada. A seleção ótima de nós de treinamento locais e a alocação ótima de recursos são fundamentais para o sucesso do processo de treinamento.

Estudo de Casos de Uso

Uma tendência de pesquisa em segurança em RAN aberta é o estudo de casos de uso específicos de RAN aberta em diferentes cenários, explicitando a análise de como as tecnologias de RAN aberta estão sendo implementadas em diferentes contextos. Essa tendência consiste na revisão de estudos de caso e relatórios que descrevem a implementação de tecnologias de RAN aberta em cenários como redes móveis 5G, redes de acesso fixo sem fio (*Fixed Wireless Access* – FWA), redes privadas e redes de IoT. A implementação de RAN aberta nesses cenários envolve a utilização de tecnologias de virtualização e desagregação para permitir a implementação de funções de rede em *hardware* genérico. Isso permite que as operadoras de rede móvel reduzam os custos de *hardware* e aumentem a flexibilidade e escalabilidade da rede.

4.5.3. Oportunidades de Segurança

É importante continuar a explorar soluções de segurança para garantir a privacidade do usuário e a proteção de dados sensíveis na arquitetura O-RAN porque a arquitetura O-RAN é baseada em uma abordagem aberta e virtualizada, o que pode aumentar o risco de ataques cibernéticos e violações de segurança. Para garantir a privacidade do usuário e a proteção de dados sensíveis na arquitetura O-RAN, é necessário explorar soluções de segurança que possam mitigar esses riscos. Isso pode incluir o uso de técnicas de criptografia para proteger os dados em trânsito e armazenados, bem como o uso de técnicas de autenticação e autorização para garantir que apenas usuários autorizados possam acessar os dados [Wang et al., 2020a, Wang et al., 2021]. Logo, é importante desenvolver soluções de segurança que possam detectar e responder a ameaças de segurança em tempo real. Isso inclui o uso de técnicas de detecção de intrusão e análise de comportamento para identificar atividades suspeitas na rede e o uso de técnicas de resposta a incidentes para mitigar os efeitos de um ataque cibernético ou violação de segurança [Singh e Khoa Nguyen, 2022, Kavehmadavani et al., 2023]. Ao explorar soluções de segurança para garantir a privacidade do usuário e a proteção de dados sensíveis na

arquitetura O-RAN, é possível mitigar os riscos de ataques cibernéticos e violações de segurança, o que pode melhorar a confiança dos usuários finais na rede e promover a adoção da arquitetura O-RAN. A segurança é um requisito fundamental para a implantação de serviços críticos, como serviços de saúde e serviços financeiros, que exigem um alto nível de proteção de dados sensíveis.

Como a RAN aberta é derivada de princípios de virtualização, herda alguns desafios relacionados à segurança. Esses desafios incluem autenticação e autorização de migrações de máquinas virtuais, instanciações de ambientes virtuais, segurança dos hipervisores e orquestração [Firoozjaei et al., 2017]. Embora a RAN aberta possibilite a criação de serviços flexíveis sob medida para cada usuário, é importante considerar os benefícios sob a ótica dos desafios relacionados à segurança trazidos por redes abertas virtualizadas [Niknam et al., 2022]. As principais oportunidades para aprimorar a segurança da RAN aberta são listadas a seguir [NIS Cooperation Group, 2022]:

- **Diversidade de Fornecedores na RAN**. A Open RAN possibilita o surgimento e uso de mais fornecedores na RAN juntamente com uma RAN desagregada, interfaces interoperáveis e o aumento do uso de *hardware* de código aberto e comercial (COTS). Assim, é possível reduzir os riscos relacionados à dependência de um único fornecedor. Contudo, ainda há o risco de centralização do mercado em torno de um número reduzido de fornecedores, integradores de sistemas e provedores nuvem, indo contra a diversificação;

- **Visibilidade e Auditoria**. O uso de padrões abertos para as interfaces entre componentes da RAN implica que componentes diferentes fornecedores se conectam de maneira semelhante, melhorando a visibilidade e a transparência. Quanto à auditoria, o uso de interfaces de padrão aberto torna mais fácil para auditores de segurança entenderem como uma determinada implementação de RAN está funcionando e se está funcionando corretamente. O aumento do uso de código aberto na RAN aberta tende a permitir uma maior visibilidade e transparência sobre como os componentes funcionam internamente. Contudo, o código aberto não é uma garantia de melhor segurança, pois também pode conter vulnerabilidades;

- **Interoperabilidade**. Para manter a estabilidade e confiabilidade em uma RAN aberta, produtos de diferentes operadoras devem interoperar. Para isso, estratégias de mitigação de risco devem ser aplicadas em caso de conflitos [Niknam et al., 2022]. É crucial identificar os riscos das incompatibilidades entre produtos de rádio e controle de diferentes provedores de serviço;

- **Desempenho**. Um desafio relacionado ao desempenho decorre da virtualização das RANs abertas. Diferentes funções de rede e módulos podem ser migrados dinamicamente de uma infraestrutura para outra. Porém, prever congestionamentos ou falhas iminentes se torna uma tarefa complexa, especialmente em sistemas RAN dinâmicos e escaláveis [Thiruvasagam et al., 2023];

- **Inteligência Artificial**. A iniciativa do O-RAN Alliance prioriza o uso de RICs para operar e manter o desenvolvimento de redes escaláveis e altamente dependentes de modelos de inteligência artificial e aprendizado de máquina

[Thiruvasagam et al., 2023]. Esses modelos são usados para realizar análise de dados e entregar resultados em quase-tempo real. O desafio técnico é garantir que a acurácia e outras métricas de desempenho dos modelos estejam em um patamar aceitável do ponto de vista de garantir a qualidade dos serviços. Caso contrário, os modelos de inteligência artificial se tornam altamente danoso para a rede se não forem capazes de desempenhar dentro dos limites aceitáveis. Técnicas de aprendizado profundo (*deep learning*) também são tendências para o treinamento de modelos em não-tempo real para funcionalidades da RAN, mas com o compartilhamento dos modelos em tempo próximo ao tempo real. Porém, certos modelos são considerados modelos de caixa-preta, logo, torna-se difícil entender como as operações ocorrem neles e como as decisões e ações são tomadas [Fiandrino et al., 2022]. A automação consolidada por modelos de inteligência artificial pode trazer riscos adicionais de segurança, responsabilidade e disponibilidade, e as operadoras de rede podem perder o controle sobre processos críticos. A falta de transparência nos modelos pode levar a vulnerabilidade a ataques. Para oferecer maior segurança aos serviços, os modelos de inteligência artificial devem ser explicáveis, robustos a ataques adversariais e verificáveis.

- **Confiança Distribuída e *Blockchain***: A segurança atual da RAN aberta depende de soluções de criptografia e autenticação baseadas em Infraestrutura de Chave Pública (*Public Key Infraestructure* – PKI) de adesão voluntária, transferindo a confiança para uma Autoridade de Certificação (AC) centralizada como parte confiável. Isso faz com que a rede falhe catastroficamente caso ocorra uma falha de comunicação com a AC ou ocorra um ataque que comprometa o seu funcionamento. Para superar essa vulnerabilidade, segurança e autenticação garantidos por cadeia de blocos (*blockchain*) se tornam atrativos para o desenvolvimento da rede [Giupponi e Wilhelmi, 2022]. Novas arquiteturas estão sendo desenvolvidas nesse sentido, como a BE-RAN, da O-RAN Alliance [Xu et al., 2021a]. Estudos futuros são necessários para compreender o impacto dessa abordagem nas interfaces de controle, usuário e no plano de sincronização.

4.6. Considerações Finais

As redes de acesso via rádio abertas (*Open RAN*) introduzem interfaces abertas e viabilizam a desagregação de componentes, o que suscita considerações de segurança, abrangendo tanto as interfaces abertas quanto a adoção de modelos de inteligência artificial para a automação do gerenciamento e orquestração da rede. A implementação das interfaces abertas promove maior transparência e facilita a adoção de padrões de segurança. Paralelamente, a desagregação dos componentes permite que atualizações de segurança e novas funções possam ser implementadas de forma eficiente em componentes de *software* individuais. Modelos de inteligência artificial asseguram o gerenciamento e a orquestração da rede autônomos através de laços fechados de controle e gerenciamento. Entretanto, implantações da *Open RAN* implicam novos desafios de segurança para operadoras de redes móveis (*Mobile Network Operators* – MNOs). Um ecossistema aberto e multi-fornecedor demanda um foco específico no aumento da superfície de ataque nas interfaces entre tecnologias integradas. Além das considerações de segurança relacionadas

à integração de componentes provenientes de diversos fornecedores, as operadoras de serviços também enfrentarão desafios referentes ao uso de *software* de código-fonte aberto e à implementação de novas funções da rede 5G, bem como interfaces cujos padrões ainda estão em desenvolvimento. As MNOs devem, então, lidar com questões de segurança que abarcam, mas não se restringem, à *Open RAN*, tais como infraestrutura de nuvem, ataques à virtualização/conteinerização e ataques de negação de serviço distribuídos.

Para enfrentar esses desafios, a arquitetura de referência O-RAN enfatiza a adoção de melhores práticas de segurança. A proteção das interfaces de gerenciamento engloba a utilização de protocolos criptográficos, cifras robustas, autenticação mútua, rigorosos controles de acesso e registros detalhados. A separação entre O-DU e O-RU exige uma compreensão ampla das possíveis ameaças para desenvolver controles de segurança eficazes. A integração de microsserviços em controladores inteligentes da RAN requer uma arquitetura de segurança sólida, que garanta análises em tempo real e potencialize as capacidades de segurança e gerenciamento. A adoção da arquitetura de confiança zero (*Zero-Trust Architecture* – ZTA) enfatiza a proteção de recursos e a avaliação contínua de dispositivos e usuários, aumentando a segurança ao eliminar a confiança implícita. A adoção da ZTA na O-RAN visa reduzir os riscos associados à ampliada superfície de ataque, mantendo a arquitetura da RAN segura e aberta.

É importante ressaltar que a arquitetura O-RAN negligencia o conceito de "segurança/privacidade por *design*/padrão" [Köpsell et al., 2022], o que resulta em um sistema que apresenta vários riscos de segurança. Soluções de segurança tradicionais conseguem mitigar os riscos das redes de acesso via rádio abertas e podem ser implementadas sem grande esforço, e com baixo investimento de capital, combatendo de forma eficaz as ameaças individuais. No entanto, para reduzir riscos de segurança específicos, como os relacionados às soluções de inteligência artificial, são necessários esforços para adaptar as especificações e implementar as salvaguardas de segurança correspondentes.

As RANs abertas e, particularmente, a arquitetura de referência O-RAN, combinam a abertura e a desagregação com sólidas práticas de segurança. Esse capítulo abordou os desafios de segurança intrínsecos às RANs e discutiu as considerações decorrentes da natureza aberta e desagregada da rede. Mediante a aderência aos padrões da indústria, à arquitetura *Zero-Trust* e outros requisitos de segurança, o capítulo demonstrou que a RAN aberta busca proporcionar um ambiente seguro, aproveitando os benefícios das interfaces abertas e da maior flexibilidade. Por fim, o capítulo elencou desafios em aberto e oportunidades de pesquisa para o desenvolvimento de soluções de segurança adaptadas à nova geração de redes de acesso via rádio com interfaces abertas e componentes desagregados, executando código aberto em *hardware* de propósito geral.

Agradecimentos

Este capítulo foi realizado com recursos do CNPq, FAPERJ e RNP. Além disso, o presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001.

Referências

[Abdalla e Marojevic, 2023] Abdalla, A. S. e Marojevic, V. (2023). End-to-end O-RAN security architecture, threat surface, coverage, and the case of the open fronthaul. *arXiv preprint arXiv:2304.05513*.

[Abdalla et al., 2022] Abdalla, A. S., Upadhyaya, P. S., Shah, V. K. e Marojevic, V. (2022). Toward next generation open radio access networks: What O-RAN can and cannot do! *IEEE Network*, 36(6):206–213.

[Abouaomar et al., 2022] Abouaomar, A., Taik, A., Filali, A. e Cherkaoui, S. (2022). Federated learning for RAN slicing in beyond 5G networks. *arXiv preprint arXiv:2206.11328*.

[Adesina et al., 2023] Adesina, D., Hsieh, C.-C., Sagduyu, Y. E. e Qian, L. (2023). Adversarial machine learning in wireless communications using RF data: A review. *IEEE Communications Surveys & Tutorials*, 25(1):77–100.

[Arnaz et al., 2022] Arnaz, A., Lipman, J., Abolhasan, M. e Hiltunen, M. (2022). Toward Integrating Intelligence and Programmability in Open Radio Access Networks: A Comprehensive Survey. *IEEE Access*, 10:67747–67770.

[Azariah et al., 2022] Azariah, W., Bimo, F. A., Lin, C.-W., Cheng, R.-G., Jana, R. e Nikaein, N. (2022). A survey on open radio access networks: Challenges, research directions, and open source approaches. *arXiv preprint arXiv:2208.09125*.

[Brik et al., 2022] Brik, B., Boutiba, K. e Ksentini, A. (2022). Deep Learning for B5G Open Radio Access Network: Evolution, Survey, Case Studies, and Challenges. *IEEE Open Journal of the Communications Society*, 3:228–250.

[Chen e et al., 2021] Chen, M. e et al. (2021). A joint learning and communications framework for federated learning over wireless networks. *IEEE Transactions on Wireless Communications*, 20(1):269–283.

[Couto et al., 2023a] Couto, R. S., Cruz, P., Campista, M. E. M. e Costa, L. H. M. K. (2023a). Using public datasets to train O-RAN deep learning models. Em *2st International Conference on 6G Networking (6GNet)*, p. 1–8. Artigo aceito para publicação (convidado).

[Couto et al., 2023b] Couto, R. S., Mattos, D. M. F., Moraes, I. M., Cruz, P., Medeiros, D. S. V., Souza, L. A. C., Táparo, F. G., Campista, M. E. M. e Costa, L. H. M. K. (2023b). Gerenciamento e orquestração de serviços em O-RAN: Inteligência, tendências e desafios. Em *Minicursos do XLI Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC 2023)*, p. 1–52.

[Das et al., 2017] Das, S. K., Chowdhury, S. S. e Das, S. K. (2017). A survey on resource allocation in cloud computing: Issues and challenges. *IEEE Transactions on Cloud Computing*, 5(2):358–378.

[Davaslioglu e Sagduyu, 2019] Davaslioglu, K. e Sagduyu, Y. E. (2019). Trojan attacks on wireless signal classification with adversarial machine learning. Em *IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN)*, p. 1–6.

[de Oliveira et al., 2023] de Oliveira, N. R., Moraes, I. M., Medeiros, D. S. V., Andreoni, M. e Mattos, D. M. F. (2023). An agile conflict-solving framework for intent-based management of service level agreement. Em *2st International Conference on 6G Networking (6GNet)*, p. 1–8. Artigo aceito para publicação (convidado).

[Dik e Berger, 2023] Dik, D. e Berger, M. S. (2023). Open-RAN fronthaul transport security architecture and implementation. *IEEE Access*, 11:46185–46203.

[Fiandrino et al., 2022] Fiandrino, C., Attanasio, G., Fiore, M. e Widmer, J. (2022). Toward native explainable and robust AI in 6G networks: Current state, challenges and road ahead. *Computer Communications*, 193:47–52.

[Firoozjaei et al., 2017] Firoozjaei, M. D., Jeong, J. P., Ko, H. e Kim, H. (2017). Security challenges with network functions virtualization. *Future Generation Computer Systems*, 67:315–324.

[Giupponi e Wilhelmi, 2022] Giupponi, L. e Wilhelmi, F. (2022). Blockchain-enabled network sharing for O-RAN in 5G and beyond. *Netwrk. Mag. of Global Internetwkg.*, 36(4):218–225.

[Groen et al., 2023] Groen, J., Doro, S., Demir, U., Bonati, L., Polese, M., Melodia, T. e Chowdhury, K. (2023). Implementing and Evaluating Security in O-RAN: Interfaces, Intelligence, and Platforms. *arXiv preprint arXiv:2304.11125*.

[Habler et al., 2022] Habler, E., Bitton, R., Avraham, D., Klevansky, E., Mimran, D., Brodt, O., Lehmann, H., Elovici, Y. e Shabtai, A. (2022). Adversarial machine learning threat analysis in open radio access networks. *arXiv preprint arXiv:2201.06093*.

[Ilyas et al., 2018] Ilyas, A., Engstrom, L., Athalye, A. e Lin, J. (2018). Black-box adversarial attacks with limited queries and information. Em *International conference on machine learning*, p. 2137–2146. PMLR.

[Karunaratne et al., 2021] Karunaratne, S., Krijestorac, E. e Cabric, D. (2021). Penetrating RF fingerprinting-based authentication with a generative adversarial attack. Em *ICC 2021-IEEE International Conference on Communications*, p. 1–6. IEEE.

[Kavehmadavani et al., 2023] Kavehmadavani, F., Nguyen, V.-D., Vu, T. X. e Chatzinotas, S. (2023). Intelligent traffic steering in beyond 5G Open RAN based on LSTM traffic prediction. *IEEE Transactions on Wireless Communications*.

[Kawahara et al., 2019] Kawahara, H., Yamamoto, R., Ohzahata, S. e Kato, T. (2019). Throughput enhancement with overlay network virtualization using commodity devices. Em *Proceedings of the 12th IEEE/ACM International Conference on Utility and Cloud Computing Companion*, UCC '19 Companion, p. 147–148, New York, NY, USA. Association for Computing Machinery.

[Konečnỳ et al., 2016] Konečnỳ, J., McMahan, H. B., Yu, F. X., Richtárik, P., Suresh, A. e Bacon, D. (2016). Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*.

[Köpsell et al., 2022] Köpsell, S., Ruzhanskiy, A., Hecker, A., Stachorra, D. e Franchi, N. (2022). Open RAN risk analysis. Relatório técnico, Federal Office for Information Security (German). Disponível em https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/Studies/5G/5GRAN-Risk-Analysis.pdf?__blob=publicationFile&v=5, acessado em 20 de agosto de 2023.

[Liyanage et al., 2023] Liyanage, M., Braeken, A., Shahabuddin, S. e Ranaweera, P. (2023). Open RAN Security: Challenges and Opportunities. *Journal of Network and Computer Applications*, 214:103621.

[Lopez et al., 2022] Lopez, M. A., Barbosa, G. N. N. e Mattos, D. M. F. (2022). New Barriers on 6G Networking: An Exploratory Study on the Security, Privacy and Opportunities for Aerial Networks. Em *International Conference on 6G Networking (6GNet)*, p. 1–6.

[McGraw et al., 2020] McGraw, G., Figueroa, H., Shepardson, V. e Bonett, R. (2020). An Architectural Risk Analysis of Machine Learning Systems: Toward More Secure Machine Learning. Relatório técnico, Berryville Institute of Machine Learning.

[Niknam et al., 2022] Niknam, S., Roy, A., Dhillon, H. S., Singh, S., Banerji, R., Reed, J. H., Saxena, N. e Yoon, S. (2022). Intelligent O-RAN for beyond 5G and 6G wireless networks. Em *2022 IEEE Globecom Workshops (GC Wkshps)*, p. 215–220. IEEE.

[NIS Cooperation Group, 2022] NIS Cooperation Group (2022). Report on the cybersecurity of Open RAN. Relatório técnico, European Union. Disponível em https://digital-strategy.ec.europa.eu/en/library/cybersecurity-open-radio-access-networks, acessado em 20 de agosto de 2023.

[O-RAN Alliance, 2020] O-RAN Alliance (2020). External open source projects. https://www.o-ran.org/resources/external-open-source-projects.

[O-RAN Alliance, 2021] O-RAN Alliance (2021). O-RAN architecture. https://www.o-ran.org/technical-specifications.

[O-RAN Working Group 1, 2023] O-RAN Working Group 1 (2023). O-RAN architecture description 9.0. Especificação Técnica v09.00, O-RAN Alliance. Disponível em https://orandownloadsweb.azurewebsites.net/specifications.

[O-RAN Working Group 10, 2023] O-RAN Working Group 10 (2023). O-RAN operations and maintenance interface specification. Especificação Técnica v10.00, O-RAN Alliance. Disponível em https://orandownloadsweb.azurewebsites.net/specifications.

[O-RAN Working Group 11, 2023a] O-RAN Working Group 11 (2023a). O-ran.wg11.threat-model.o-r003-v06.00. Especificação técnica, O-RAN Alliance. Disponível em https://orandownloadsweb.azurewebsites.net/specifications, Acessado em 15 de agosto de 2023.

[O-RAN Working Group 11, 2023b] O-RAN Working Group 11 (2023b). Security protocols specifications. Relatório Técnico v06.00, O-RAN Alliance. Disponível em https://orandownloadsweb.azurewebsites.net/specifications.

[O-RAN Working Group 11, 2023c] O-RAN Working Group 11 (2023c). Security requirements specifications. Especificação Técnica v06.00, O-RAN Alliance. Disponível em https://orandownloadsweb.azurewebsites.net/specifications.

[O-RAN Working Group 2, 2021] O-RAN Working Group 2 (2021). Non-RT RIC: Functional Architecture. Relatório Técnico v01.01, O-RAN Alliance. Disponível em https://orandownloadsweb.azurewebsites.net/specifications.

[O-RAN Working Group 2, 2023] O-RAN Working Group 2 (2023). Non-RT RIC architecture. Especificação Técnica v03.00, O-RAN Alliance. Disponível em https://orandownloadsweb.azurewebsites.net/specifications.

[O-RAN Working Group 3, 2023a] O-RAN Working Group 3 (2023a). Near-rt ric architecture. Especificação Técnica v04.00, O-RAN Alliance. Disponível em https://orandownloadsweb.azurewebsites.net/specifications.

[O-RAN Working Group 3, 2023b] O-RAN Working Group 3 (2023b). O-RAN e2 service model (e2sm). Especificação Técnica v03.01, O-RAN Alliance. Disponível em https://orandownloadsweb.azurewebsites.net/specifications.

[O-RAN Working Group 6, 2023] O-RAN Working Group 6 (2023). O2 Interface General Aspects and Principles. Especificação Técnica v04.00, O-RAN Alliance. Disponível em https://orandownloadsweb.azurewebsites.net/specifications.

[Open RAN Policy Coalition, 2021] Open RAN Policy Coalition (2021). Open RAN security in 5G. Relatório técnico, Open RAN Policy Coalition. Disponível em https://www.openranpolicy.org/wp-content/uploads/2021/04/Open-RAN-Security-in-5G-4.29.21.pdf.

[Polese et al., 2023] Polese, M., Bonati, L., D'Oro, S., Basagni, S. e Melodia, T. (2023). Understanding O-RAN: Architecture, Interfaces, Algorithms, Security, and Research Challenges. *IEEE Communications Surveys & Tutorials*, 25(2):1376–1411.

[Ramezanpour e Jagannath, 2022] Ramezanpour, K. e Jagannath, J. (2022). Intelligent zero trust architecture for 5G/6G networks: Principles, challenges, and the role of machine learning in the context of O-RAN. *Computer Networks*, 217:109358.

[Ranaweera et al., 2021] Ranaweera, P., Jurcut, A. D. e Liyanage, M. (2021). Survey on multi-access edge computing security and privacy. *IEEE Communications Surveys & Tutorials*, 23(2):1078–1124.

[Research e Markets, 2022] Research e Markets (2022). 5G Radio Access Network Market Size, Share & Trends Analysis Report By Component (Hardware, Software, Services), By Architecture Type, By Deployment, By End-user, By Region, And Segment Forecasts, 2022 - 2030. Relatório técnico. Disponível em `https://www.researchandmarkets.com/reports/5702152`.

[Restuccia et al., 2020] Restuccia, F., D'Oro, S., Al-Shawabka, A., Rendon, B. C., Chowdhury, K., Ioannidis, S. e Melodia, T. (2020). Generalized wireless adversarial deep learning. Em *Proceedings of the 2nd ACM Workshop on Wireless Security and Machine Learning*, p. 49–54.

[Rose et al., 2020] Rose, S., Borchert, O., Mitchell, S. e Connelly, S. (2020). Zero trust architecture. Relatório Técnico NIST Special Publication 800-207, National Institute of Standards and Technology - U.S. Department of Commerce. Disponível em `https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-207.pdf`.

[Shi e Sagduyu, 2021] Shi, Y. e Sagduyu, Y. E. (2021). Adversarial machine learning for flooding attacks on 5G radio access network slicing. Em *IEEE International Conference on Communications Workshops (ICC Workshops)*, p. 1–6.

[Shi e Sagduyu, 2023] Shi, Y. e Sagduyu, Y. E. (2023). Membership inference attack and defense for wireless signal classifiers with deep learning. *IEEE Transactions on Mobile Computing*, 22(7):4032–4043.

[Singh e Khoa Nguyen, 2022] Singh, A. K. e Khoa Nguyen, K. (2022). Joint selection of local trainers and resource allocation for federated learning in Open RAN intelligent controllers. Em *2022 IEEE Wireless Communications and Networking Conference (WCNC)*, p. 1874–1879.

[Soltani et al., 2023] Soltani, S., Shojafar, M., Brighente, A., Conti, M. e Tafazolli, R. (2023). Poisoning Bearer Context Migration in O-RAN 5G Network. *IEEE Wireless Communications Letters*, 12(3):401–405.

[Sun et al., 2022] Sun, G., Cong, Y., Dong, J., Wang, Q., Lyu, L. e Liu, J. (2022). Data poisoning attacks on federated machine learning. *IEEE Internet of Things Journal*, 9(13):11365–11375.

[Thiruvasagam et al., 2023] Thiruvasagam, P. K., Venkataram, V., Ilangovan, V. R., Perapalla, M., Payyanur, R., Kumar, V. et al. (2023). Open RAN: Evolution of architecture, deployment aspects, and future directions. *arXiv preprint arXiv:2301.06713*.

[Usama et al., 2019] Usama, M., Qayyum, A., Qadir, J. e Al-Fuqaha, A. (2019). Black-box adversarial machine learning attack on network traffic classification. Em *2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC)*, p. 84–89. IEEE.

[Wang et al., 2020a] Wang, C.-X., Di Renzo, M., Stanczak, S., Wang, S. e Larsson, E. G. (2020a). Artificial intelligence enabled wireless networking for 5G and beyond: Recent advances and future challenges. *IEEE Wireless Communications*, 27(1):16–23.

[Wang et al., 2020b] Wang, F., Zhong, C., Gursoy, M. C. e Velipasalar, S. (2020b). Defense strategies against adversarial jamming attacks via deep reinforcement learning. Em *2020 54th annual conference on information sciences and systems (CISS)*, p. 1–6. IEEE.

[Wang et al., 2021] Wang, T.-H., Chen, Y.-C., Huang, S.-J., Hsu, K.-S. e Hu, C.-H. (2021). Design of a network management system for 5G Open RAN. Em *2021 22nd Asia-Pacific Network Operations and Management Symposium (APNOMS)*, p. 138–141.

[Xu et al., 2021a] Xu, H., Zhang, L., Sun, Y. e I, C.-L. (2021a). BE-RAN: blockchain-enabled Open RAN with decentralized identity management and privacy-preserving communication. *arXiv preprint arXiv:2101.10856*.

[Xu et al., 2021b] Xu, X., Zhang, Y., Li, X., Zhang, Y. e Zhang, H. (2021b). Open RAN: Challenges and opportunities. *IEEE Communications Magazine*, 59(4):34–39.

Capítulo 5

Desenvolvimento ágil de software seguro e a cultura *DevSecOps*

Alexandre Braga[1] (CPQD)

Abstract

Software insecurity is a growing and recurring problem in society nowadays. All the time, vulnerabilities related to common defects and poor habits of insecure programming are exploited in attacks, affecting everyone's lives and leading to financial losses and various damages. As social activities, software security and development depend not only on technology but also on the engagement of everyone involved with it, since from the beginning of development to deployment and proper usage. This course shows how a combined approach of techniques, practices, and tools can enhance software security awareness, even during the early stages of agile development, incorporating a culture of prevention into the DevSecOps culture, where prevails the practices of automated testing, early detection of issues, and fast correction. To achieve this, a variety of complementary tools is necessary, distributed along a development pipeline, in a layered protection architecture, with scalable and repeatable results along multiple secure development cycles.

Resumo

A insegurança de software é um problema crescente e recorrente na sociedade hoje em dia. O tempo todo, vulnerabilidades relacionadas a defeitos comuns e maus hábitos de programação insegura são exploradas em ataques, afetando a vida de todos e levando a prejuízos financeiros e perdas diversas. Em sendo atividades sociais, segurança e desenvolvimento de software dependem não apenas da tecnologia, mas também do engajamento de todos os envolvidos no processo desde o início do desenvolvimento até a implantação e uso correto do software. Este minicurso mostra como a aplicação combinada de técnicas, práticas e ferramentas pode aumentar a conscientização sobre a segurança do software já durante os primeiros estágios de desenvolvimento ágil, incorporando a cultura de prevenção à cultura DevSecOps, em que imperam as práticas de testar automaticamente, detectar os problemas cedo e corrigi-los rápido.

[1]ambraga@cpqd.com.br

Para tal, é necessária uma variedade de ferramentas complementares, distribuídas ao longo de uma esteira de desenvolvimento, em uma arquitetura de proteção em camadas, com resultados escaláveis e repetíveis por vários ciclos de desenvolvimento seguro.

5.1. Introdução

À medida que aumenta a dependência das pessoas em relação à infraestrutura tecnológica, também aumentam a complexidade e a conectividade dos sistemas de computação subjacentes, porém, sem um aumento correspondente da segurança com que os softwares são construídos, configurados e implantados. Por isso, a insegurança dos sistemas de software é um problema crescente e recorrente. O tempo todo, vulnerabilidades relacionadas a defeitos comuns e maus hábitos de programação insegura são exploradas em ataques, afetando a vida de todos e levando a prejuízos financeiros e outras perdas.

Do ponto de vista do desenvolvedor de software, a segurança é um aspecto da qualidade e o desenvolvimento de software é uma atividade tanto técnica quanto social. Por outro lado, a garantia de segurança é uma responsabilidade coletiva. Logo, qualquer solução para a insegurança dos sistemas de software não é uma questão tecnológica apenas e deve considerar o engajamento das pessoas e a ajuda de todos os envolvidos no processo de construção desde o início do desenvolvimento até a implantação e uso correto dos sistemas.

Além disso, a escassez de mão de obra qualificada em segurança de software e a grande escala do problema dificultam o seu tratamento por times de segurança dedicados. Em um cenário comum, nas empresas de tecnologia, onde dezenas ou centenas de times de desenvolvimento atuam simultaneamente, uma equipe de segurança dedicada e reduzida tem dificuldades em atender a todas as demandas por mais segurança. A boa prática atual dita que esse time de segurança precisa dividir a responsabilidade e o protagonismo em segurança de software com os times de desenvolvimento, que devem ser capazes de resolver de forma autônoma os problemas simples, a fim de ganhar escala e velocidade no tratamento de vulnerabilidades técnicas, antes de acionar os especialistas.

Nesse contexto, existe um interesse crescente na academia e na indústria pelos aspectos práticos relacionados ao engajamento continuado de equipes de desenvolvedores de software em atividades de desenvolvimento de software seguro. Assim, este minicurso busca atender a demanda por conteúdo voltado para a utilização, diretamente por um esquadrão de desenvolvimento de software, de técnicas de análise de segurança apoiadas por ferramentas, tais como *Static Application Security Test* (SAST) e *Software Composition Analysis* (SCA), assim como também de testes de segurança em aplicações com auxílio de ferramentas *Dynamic Application Security Test* (DAST), em um contexto de *DevSecOps*, assim como também o uso de técnicas de segurança *by design* e de proteção da aplicação em produção, com o uso de *Web Application Firewalls* (WAFs).

O restante do texto está organizado da seguinte forma. A Seção 5.2 apresenta ao leitor a cultura de segurança de software ágil e aspectos de *DevSecOps*. A Seção 5.3 trata especificamente as avaliações e testes de segurança auxiliadas por ferramentas: revisão de código com SAST (subseção 5.3.2), avaliação de segurança com SCA (subseção 5.3.3) e testes de segurança com DAST (subseção 5.3.4). Já a Seção 5.4 trata vulnerabilidades de projeto com testes de segurança em APIs REST, enquanto a Seção 5.5 mostra como proteger aplicações web com WAF. Finalmente, a Seção 5.6 compara as ferramentas em relação aos falsos positivos e falsos

negativos e a Seção 5.7 tece considerações finais.

5.2. Segurança de software ágil e *DevSecOps*

Esta seção aborda as práticas de segurança de software adotadas pelas empresas de tecnologia que desenvolvem software conforme os métodos adaptativos [Institute 2023] e que incluem segurança em seus ciclos de desenvolvimento ágil. A seção discute como as práticas de segurança de software têm sido adotadas junto aos métodos ágeis e às esteiras ou pipelines *DevSecOps*.

A principal contribuição desta seção é a elaboração da ideia de esteira ou *pipeline DevSecOps* na qual as diversas técnicas e ferramentas de segurança são distribuídas de acordo com as necessidades de verificações manuais ou automatizadas, estáticas ou dinâmicas, antecipadas ou tardias, no ciclo de desenvolvimento de software.

Primeiramente, a seção discute os papéis principais envolvidos no desenvolvimento de software seguro, em particular, o *Security Champion*, e os aspectos de conscientização em segurança e cultura de desenvolvimento seguro. Em seguida, a seção discute o uso de ferramentas automáticas para acelerar e ganhar escala na execução de testes de segurança em esteiras de desenvolvimento *Continuous Integration*/*Continuous Delivery* (CI/CD) e a participação ativa de programadores na solução do débito de segurança. As técnicas e ferramentas são apresentadas e explicadas nesta seção.

5.2.1. O *Security Champion* de desenvolvimento de software seguro

No mundo atual repleto de incertezas e mudanças constantes, o desenvolvedor de software é um peregrino na sua jornada pessoal de aperfeiçoamento e aprendizado contínuo. De acordo com [Institute 2023], para sobreviver em um ambiente onde a incerteza de requisitos é grande e a velocidade da mudança tecnológica e cada vez maior, o desenvolvedor de software precisa seguir os princípios de desenvolvimento ágil, respondendo rápido às mudanças e sendo adaptativo no desenvolvimento de software para mitigar as incertezas.

Por outro lado, de acordo com [Pressman 2018], a flexibilidade de adaptação decorrente da assimilação rápida das mudanças leva naturalmente a uma maior instabilidade do software, com mais defeitos e um maior número de falhas. Somente o esforço deliberado do time de desenvolvimento pode trazer o software para próximo do estado estável anterior a mudança. Porém, uma vez que o software não é mais o mesmo porque a superfície de ataque mudou, o estado estável anterior é inatingível. Logo, há potencialmente mais defeitos que podem ser explotados como vulnerabilidades em ataques reais.

Segurança envolve pessoas realizando tarefas com eficiência por meio da tecnologia. O problema do estado atual de insegurança do software não é apenas uma questão tecnológica. As pessoas sempre fizeram coisas ruins, mesmo antes da haver tecnologias de computação e comunicação. A tecnologia facilitou o acesso aos meios para cometer crimes (cibernéticos) e compartilhar seus resultados. Quando a sociedade dependia menos da tecnologia, a necessidade por segurança cibernética também era menor. A partir do momento em que a sociedade ficou mais dependente da tecnologia, também aumentou a necessita de mais segurança cibernética e tanto o impacto quanto a percepção da insegurança do software aumentaram.

Segurança é importante para todo mundo! Aos usuário da tecnologia, as falhas de software trazem prejuízos pessoais e coletivos. Aos que constroem tecnologia, exige-se a respon-

sabilidade por mantê-la estável e segura para uso nos negócios e na sociedade. Quando os desenvolvedores de software não assumem a sua parte da responsabilidade pela segurança cibernética, aumenta a chance da tecnologia se tornar inadequada para exercer sua função na sociedade.

Toda pessoa desenvolvedora de software tem um papel a desempenhar na construção de software seguro, porque, no mínimo, a segurança é um aspecto de qualidade. Sendo assim, o desenvolvedor perfeiçoa-se em segurança de software, assume responsabilidade pela segurança do seu código, colabora com o grupo de segurança de aplicações da organização onde atua, colabora com o seu time na promoção da segurança de software e corrige vulnerabilidades simples diretamente.

O *Security Champion* [Miller 2022, saf 2019] vai além destas responsabilidades gerais, porque ele é a ponte entre os times de desenvolvimento e de segurança. Ele promove dentro do seu time a segurança *by design* desde cedo e facilita a comunicação entre desenvolvedores e especialistas em segurança de aplicações (e vice-versa). O *Security Champion* é um desenvolvedor que também sabe segurança. Como ele pertence ao time de desenvolvimento, ele é capaz de acelerar o andamento das tarefas, tratando as questões simples de segurança de software localmente e garantindo que verificações e testes são feitos direito e no tempo certo. Ele também leva problemas difíceis da equipe de software para os especialistas em segurança de aplicações.

Um *Security Champion* de desenvolvimento de software seguro não nasce pronto. Segurança cibernética é geralmente uma lacuna na formação das pessoas de tecnologia. Por isso, na prática, o desenvolvedor adquire habilidades e conhecimentos sobre segurança cibernética à medida em que supera os desafios de construir software seguro durante o desenvolvimento de software. O *Security Champion* não substitui o especialista em segurança de aplicações, mas deve conhecer o suficiente para fazer bem seu trabalho. Em uma espiral de aperfeiçoamento contínuo, no início, a vontade de aprender e de ajudar são mais importantes que conhecimento formal.

No geral, o *Security Champion* experiente precisa conhecer engenharia de software (métodos e ferramentas), modelagem de ameaças, tratamento de incidentes e vulnerabilidades, programação segura e testes de segurança. No papel de mediador entre desenvolvimento e segurança, muitas vezes, *soft skills* são importantes, tais como as habilidades em negociação, resolução de conflitos, apresentação, argumentação, motivação e engajamento de pessoas.

Em termos de cultura de segurança em times de software [Miller 2022, saf 2019], há papéis dentro do time de software mais alinhados às responsabilidades do *Security Champion*: arquitetos, engenheiros, desenvolvedores e testadores. Além disso, gerentes de projeto, agilistas, *product owners*, *scrum masters*, entre outros, também podem atuar como *Security Champions*, mas conflitos de interesse com outros papéis devem ser evitados.

Valem lembrar que, infelizmente, nem sempre o clima organizacional em um time de software é saudável. Em ambientes tóxicos, o *Security Champion* é menos eficaz em promover bons hábitos de segurança *by design*. Nesses ambientes, há três sinais de que o *Champion* não consegue mais ajudar sua equipe: *(i)* quando ele faz o trabalho do policial mau, aponta defeitos e vai embora, vigiando todo mundo e procurando culpados; *(ii)* quando ele é o bombeiro dos projetos, salva a segurança do projeto sozinho e age como o fiscal das regras de segurança; *(iii)* quando ele é o exército de uma pessoa só, tem inimigos declarados e luta uma batalha perdida. Por isso, é muito importante que o *Security Champion* tenha o apoio quase incondicional dos executivos, gestores e patrocinadores dos projetos, além do apoio do time.

5.2.2. Esquadrões, guildas, capítulos e tribos de segurança de software

Na cultura de software ágil, os desenvolvedores de software se organizam em diversas estruturas sociais. As organizações mais comuns são o Esquadrão, a Guilda, o Capítulo e a Tribo.

O time de desenvolvimento ágil é formado por todos os especialistas necessários e profissionais interdisciplinares, funcionando como um **esquadrão** autônomo e autocontido, com pouca dependência externa, capaz de resolver suas questões internamente de maneira rápida, direta e sem burocracia processual, que se recorrente, é logo automatizada. O esquadrão é geralmente uma equipe pequena (por exemplo, com até nove membros) multifuncional e auto-organizada, cujos membros possuem responsabilidades de ponta a ponta na esteira e trabalham juntos em direção a uma missão de longo prazo.

A **Tribo** pode ser entendida como um conjunto de esquadrões alinhados a um negócio, área de atuação, ou produto. Já o **Capítulo** lembra a organização de uma gerência e é um grupo formado por pessoas de competências comuns, como suporte de qualidade, mentoria ágil, segurança de aplicações, desenvolvimento web, etc. A **guilda**, por outro lado, é uma comunidade de interesse em que pessoas de toda a empresa se reúnem e compartilham conhecimento sobre uma área específica, como por exemplo, desenvolvimento de software seguro. Qualquer pessoa pode entrar ou sair de uma guilda a qualquer momento.

Os *Security Champions* geralmente se agrupam em uma guilda. Estabelecer uma função de *Security Champion* nos esquadrões empodera a segurança da aplicação enquanto o *Champion* mantém e aprimora suas habilidades e competências. Porém, o *Champion* precisa de uma comunidade de apoio. A guilda de *Security Champions* serve de plataforma para os *champions* desenvolverem habilidades com apoio do grupo e compartilharem conhecimento. Em sendo uma comunidade de vínculo fraco, a guilda de *Security Champions* ainda precisa de motivação e engajamento dos membros, além de alguma coordenação de ações. Geralmente, esta coordenação é realizada por um membro do grupo formal (Capítulo) de segurança de aplicações.

A guilda de *Security Champions* ajuda a segurança de software ágil quando dissemina o pensamento preventivo, o conhecimento sobre ameaças cibernéticas e a segurança desde o início do desenvolvimento, promovendo a segurança contínua e proativa versus segurança reativa e improvisada, estabelecendo melhores práticas e ferramentas comuns para todos os esquadrões, visando garantir que a segurança seja uma responsabilidade compartilhada e tratada por ada esquadrão. Uma cultura ágil forte influencia como o capítulo de segurança de aplicações está organizado e também como ele se relaciona com as outras estruturas sociais. Por exemplo, a guilda de *Security Champions* pode ser liderada e coordenada por um membro do capítulo de segurança de aplicações.

5.2.3. O grupo de segurança de aplicações

O capítulo de segurança de aplicações é um grupo organizado de especialistas em áreas da segurança de aplicações e tem sua própria organização interna. Hoje em dia, adota-se uma nomenclatura baseada em cores para identificar as especialidades e áreas de atuação deste grupo, que geralmente é composta pelos times vermelho, azul e amarelo. Opcionalmente, também há os times roxo, laranja e branco.

O time vermelho, ou *Red Team*, é o grupo de segurança ofensiva formado pelos *hackers* éticos. Este grupo possui visão adversarial e realiza ataques simulados furtivos (ocultos, sem

aviso) contra os controles de segurança, salvaguardas e contramedidas. As operações do *Red Team* servem para mostrar a insuficiência dos procedimentos de resposta à incidentes ou dos controles de segurança por meio dos testes não anunciados e furtivos (às escondidas). Antes de iniciar uma campanha de testes de intrusão, o *Red Team* deve definir a meta específica e as regras do jogo. Por exemplo, observando os seguintes pontos: ambiente de teste, janela de manutenção, não destruir a produção, etc. O resultado da operação do *Red Team* tem que ser útil para o *Blue Team* e para os desenvolvedores (*Yellow Team*). O *Blue Team* responde aos testes como se fossem incidentes reais.

O time azul, ou *Blue Team*, é o grupo de segurança defensiva. Grupo de defensores que tem o trabalho mais difícil: proteger os sistemas de software e os dados confidenciais tanto do *Red Team* quanto dos adversários reais. Atuando como *threat hunters*, procuram ativamente por ameaças e podem fazer as correções necessárias diretamente ou elaborar recomendações de melhoria, acompanhando a realização das correções. *Threat hunting* é a tarefa de identificação de ataques atuais ou passados e a contenção de ataques em andamento. Em resumo, o *Blue Team* se concentra na detecção de atividades maliciosas, proteção dos ativos de valor e manutenção da segurança. O *Blue Team* detecta, responde e se recupera dos incidentes de segurança, refinando e praticando a capacidade de responder aos incidentes. A capacidade de resposta à incidentes é obrigatória para o *Blue Team* funcionar como *Purple Team*.

O time roxo, ou *Purple Team*, é a combinação dos times vermelho e azul. O *Purple Team* combina ataques do *Red Team* e respostas do *Blue Team* em um esforço colaborativo e com ciclos de melhoria contínua. Os *Purple Teams* são úteis para equipes de segurança de aplicações com pouca capacidade de resposta à incidentes. Já os *Red Teams* são mais úteis para quem faz resposta aos incidentes de forma organizada (eficaz?). O *Purple Team* age como *Red Team* e *Blue Team* integrados e com comunicação próxima e interação fácil.

O *Purple Team* não é um *White Team*, que geralmente facilita a comunicação entre os times *Red* e *Blue* para supervisão e orientação. O *White Team* consiste de patrocinadores, facilitadores e outros *stakeholders*. O *White Team* não é técnico, não ataca e nem defende. Mas somente o *White Team* pode saber quando *Red Team* ataca.

Finalmente, o *Yellow Team*, ou time amarelo, é a denominação usada para se referir ao esquadrão de desenvolvimento ágil que vai implementar os controles de segurança e corrigir as vulnerabilidades; isto é, os desenvolvedores de software. A interface entre o *Yellow Team* e os times *Red* e *Blue* pode ser feita pelo time *Orange* (Laranja), que é responsável pelas ações de treinamento, capacitação e conscientização dos desenvolvedores sobre os métodos, práticas, ferramentas e tecnologias de segurança de software, além de ser o responsável pela guilda de *Security Champions*.

5.2.4. Desenvolvimento ágil de software seguro

Para Gary McGraw [McGraw 2004], segurança não é uma característica que pode ser simplesmente adicionada ao software. Em vez disso, ela seria uma propriedade emergente a partir de como o software é construído e usado. Assim, o software seguro é aquele que contempla com um grau alto de confiança justificada, mas não com certeza absoluta, um conjunto substancial de propriedades explícitas de segurança e de serviços de segurança [McGraw 2004]. Por isso, o software seguro resulta de um ciclo de desenvolvimento com segurança e de uma arquitetura de segurança, cujas propriedades emergentes se expressam durante o seu funcionamento.

Existem métodos, metodologias ou processos de desenvolvimento de software seguro, que são maneiras de organizar e conduzir as atividades de segurança durante o desenvolvimento de software, desde a concepção até a implantação da aplicação, incorporando tarefas, papeis e fluxos específicos para segurança. Nos últimos trinta anos, diversas destas metodologias foram propostas e experimentadas tanto pela academia quanto pela indústria. As mais conhecidas eram voltadas para os ambientes onde o desenvolvimento do software ocorre linearmente, em fases bem definidas e separadas no tempo, desde uma fase inicial de definição de objetivos e requisitos, passando pela elaboração de arquitetura, análise e projeto, implementação e testes, até a entrega e implantação em produção.

Um ponto chave de todas as metodologias é evitar que vulnerabilidades sejam incorporadas ao software, em vez de simplesmente descobrir e corrigir vulnerabilidades. Esta abordagem considera que o custo da descoberta tardia (em produção) de defeitos exploráveis como vulnerabilidades em ataques reais é muito alto. Por outro lado, o custo de correção de vulnerabilidades é menor quanto mais cedo ela ocorre no ciclo de vida da aplicação. Em se tratando de ataques cibernéticos, alguns custos são incalculáveis e irreparáveis, como por exemplo, a perda de vidas, a destruição de reputação e a exposição pública.

Uma vez que é mais barato descobrir as vulnerabilidades cedo, surge a ideia de *push left* ou *shift left*, que, ao olhar para um desenho de processo de software com atividades iniciais a esquerda e atividades finais a direita, adota a postura de antecipar as atividades de segurança (trazendo-as para o lado esquerdo do desenho sempre que possível). Exemplos de atividades de segurança que poderiam ser antecipadas são as seguintes: análise de segurança de arquitetura, especificação de requisitos de segurança, revisão de código (com ferramentas), avaliações de segurança e testes de intrusão.

Os métodos ágeis mudam a rigidez linear e sequencial do processo de desenvolvimento de software, fazendo com que em um período de tempo curto (por exemplo, uma *sprint* de 2 semanas), todas as fases sejam visitadas, por vezes de maneira concorrente e paralela, resultando em um software funcional (porém incompleto) a cada ciclo curto de desenvolvimento, evoluindo de maneira iterativa e incremental.

Uma vez que a cada ciclo curto há uma nova versão do software posta em execução, não é mais suficiente descobrir as vulnerabilidades cedo, mas também é importantíssimo corrigi-las rápido. No desenvolvimento ágil e seguro [Fisher 2022], a ideia *push left* evolui para o conceito *pushing left while accelerating right*, pregando que as atividades de segurança (todas elas) devem ser realizadas dentro de um ciclo curto (uma *sprint*). Isto significa que o rigor e complexidade das tarefas devem ser dosados a fim de torná-las mais rápidas, deixando refinamentos e detalhamentos para outros ciclos curtos. Por exemplo, em uma situação ideal, haveria um ciclo completo de verificações e testes de segurança, correções de vulnerabilidades e retestes a cada *sprint* de desenvolvimento [Fisher 2022].

Não há surpresa em perceber que o desenvolvimento ágil de software seguro somente é viável com grande uso de automação e o apoio dos *Security Champions*. Isto não significa que, por exemplo, testes de intrusão manuais, detalhados e demorados, realizados por *hackers* éticos não ocorram mais. Eles ainda acontecem, porém, fora do ciclo curto de desenvolvimento. Já as verificações e testes de segurança automatizados complementam os testes manuais e são usados para ganhar escala na testagem com detecção rápida dos problemas simples.

5.2.5. A esteira de desenvolvimento de software seguro e *DevSecOps*

DevSecOps [dev 2021] pode ser entendido como a integração contínua de princípios, processos e tecnologias de segurança (*Sec*) às práticas e processo da cultura *DevOps*, entendida como um movimento cultural formado por um conjunto de práticas e uma abordagem colaborativa que visa integrar o desenvolvimento de software (*Dev*) com as operações de infraestrutura e suporte (*Ops*). O objetivo principal do *DevOps* é aumentar a eficiência e a agilidade no desenvolvimento, entrega e operação de software, permitindo que as equipes de desenvolvimento e operações trabalhem de forma mais próxima e coordenada. A adoção plena da cultura *DevSecOps* é uma jornada com quatro momentos ou estágios de maturidade bem definidos. Em cada momento, a automação das atividades de segurança é adotada conforme a maturidade do time de desenvolvimento nas práticas *DevSecOps*.

No primeiro momento, o esquadrão ainda inexperiente adota um repositório de software e boas práticas de programação reforçadas na IDE e na geração de *build* (compilação e empacotamento), mas ainda não tem um pipeline ou esteira bem definida. A frequência de geração de *builds* pode ainda ser baixa e descontínua. Esta equipe deseja adotar as práticas de programação segura e faz isso com ferramentas SAST e SCA integrados à IDE, ao repositório de código fonte e possivelmente à geração da *build*.

No segundo momento, o esquadrão de software adota as práticas fundamentais de construção ou geração, integração e teste contínuos. Neste momento, já pode haver um *pipeline* automatizado que culmina na implantação do software em um ambiente de testes de aceitação ou homologação. Tudo que era feito por *scripts* isolados (incluindo as atividades de segurança) passa a ser realizado de maneira ordenada em um *pipeline*, que pode até ser interrompido se vulnerabilidades inaceitáveis forem detectadas. A existência de um ambiente de teste torna possível a realização de varreduras de vulnerabilidades automáticas com o auxílio de ferramentas DAST.

No terceiro momento, o esquadrão de software experiente realiza com desenvoltura e agilidade suas tarefas, atingindo o estágio de entrega contínua e implantação contínua. A automação das tarefas corriqueiras torna possível a sua realização com frequência alta. Neste momento, a construção e instalação automatizadas são apoiadas por processos automáticos de *failover* e *rollback*. Além disso, varreduras de vulnerabilidades e testes com ferramentas DAST são complementados, no ambiente de implantação (e produção) por teste *fuzzing* de injeção de falhas, testes de segurança interativos (*Interactive Application Security Tests* - IAST) e testes de intrusão (manuais).

Finalmente, no último estágio de maturidade *DevSecOps*, o esquadrão de software realiza todas as tarefas do estágio anterior com frequência alta, mais as ações de monitoramento contínuo e proteção em tempo de execução da aplicação no ambiente de produção. Neste momento, usam-se as ferramentas de autoproteção automática da aplicação (*Runtime Application Self-Protect* – RASP), como por exemplo, um WAF adaptativo integrado ao ambiente de execução da aplicação [da Silva et al. 2019] e capaz de fazer *patches* virtuais na aplicação em tempo real.

5.3. Avaliação e testes de segurança com auxílio de ferramentas

Esta seção está organizada como três tutoriais complementares de ferramentas usadas em avaliações e testes de segurança de software e que representam as categorias principais de ferramentas utilizadas em uma esteira *DevSecOps*: SAST, SCA e DAST. Ao aplicar ferramentas como parte

do processo de desenvolvimento, é possível fortalecer a segurança do software, identificando e corrigindo potenciais vulnerabilidades bem antes da implantação, garantindo assim um produto final mais confiável e protegido contra ameaças de segurança.

Avaliações e testes de segurança auxiliados por ferramentas aceleram a geração de resultados que podem ser tratados rapidamente. Porém, ferramentas automáticas não substituem o testador experiente. As ferramentas são softwares e podem ser incompletas e defeituosas, conforme estudos recentes [Braga et al. 2017, Braga et al. 2019]. Por isso, é necessário ter cautela na triagem de falsos positivos, evitando alarmes falsos que possam prejudicar a eficiência do processo. Além disso, as técnicas SAST, SCA e DAST são complementares entre si e, devem ser usadas em conjunto para mitigar falsos negativos (omissões), garantindo a identificação abrangente de vulnerabilidades.

A revisão de código com *Static Application Security Test* (SAST) é uma técnica essencial, apoiada por ferramentas, para a análise de vulnerabilidades no código-fonte, permitindo a detecção antecipada de possíveis falhas de segurança. Essa abordagem pode ser aplicada durante a fase de codificação, por meio da integração com IDEs, na compilação ou na geração de *builds*. Além disso, as ferramentas SAST podem ser integradas a pipelines CI/CD ou usadas de forma autônoma.

A avaliação de segurança com *Software Composition Analysis* (SCA) é uma técnica de verificação amplamente utilizada e apoiada por ferramentas para análise de vulnerabilidades em dependências de software, como componentes e bibliotecas de terceiros. Essa técnica é aplicada tanto na detecção antecipada de vulnerabilidades, durante as fases iniciais do desenvolvimento, quanto na detecção tardia, mais próxima à produção ou implantação do software. Por exemplo, é comum utilizar o SCA para realizar varreduras de vulnerabilidades em contêineres e imagens de implantação. Além disso, as ferramentas SCA também podem ser usadas para análise de licenciamento de software de código aberto (*Open Source Software* - OSS) e podem ser integradas a *pipelines* de CI/CD ou utilizadas de forma autônoma.

O teste de segurança com *Dynamic Application Security Test* (DAST) é uma técnica apoiada por ferramentas utilizada para analisar vulnerabilidades em tempo de execução de uma aplicação de software. Essa técnica é empregada para identificar vulnerabilidades próximas do momento de implantação ou entrega, ou seja, em fases tardias do ciclo de desenvolvimento. Para realizar a análise, é necessário que a aplicação em análise esteja em funcionamento em um ambiente de testes. A ferramenta DAST pode ser incorporada a *pipelines* CI/CD ou usada de forma autônoma como uma ferramenta independente.

5.3.1. Aplicações vulneráveis de aprendizado

Neste curso, as ferramentas SAST, SCA e DAST são exercitadas sobre aplicações propositalmente vulneráveis, livres e gratuitas, implementadas em uma variedade de tecnologias diferentes e utilizadas no aprendizado de técnicas de segurança, a saber: WebGoat em Java, bWAPP em PHP, Juice Shop com microsserviços em JavaScript e Gruyere em Python. Nem todas elas são utilizadas em todos os casos, algumas apenas nas aulas práticas em laboratório.

O OWASP Juice Shop é uma aplicação moderna construída em node.js com uma arquitetura de microsserviços. A página principal do projeto no OWASP está em `https://owasp.org/www-project-juice-shop/`. O código fonte está em `https://`

github.com/bkimminich/juice-shop. Um guia prático das tarefas de testes de segurança pode ser encontrado em https://pwning.owasp-juice.shop/. Há diversas opções de instalação ou implantação disponíveis em https://github.com/bkimminich/juice-shop#setup. Porém, a opção mais fácil e direta utiliza um contêiner Docker, com instruções em https://hub.docker.com/r/bkimminich/juice-shop.

A listagem 5.1 mostra os comandos para instalação por contêiner Docker. Quando a instalação é bem sucedida, a aplicação fica disponível em http://localhost:3000/.

Listagem 5.1. Juice Shop em Docker.

```
1 #Juice Shop em Docker
2 docker pull bkimminich/juice-shop
3 docker run -p 3000:3000 bkimminich/juice-shop
```

O OWASP WebGoat é a aplicação vulnerável tradicional do OWASP para o aprendizado de vulnerabilidades e foi desenvolvido em JavaEE. A página do WebGoat está em https://owasp.org/www-project-webgoat/. Este tutorial usa a versão 7.1 do WebGoat (para Java 8) que pode ser obtida em https://github.com/WebGoat/WebGoat. Um repositório com todo o código fonte do WebGoat 7.1, incluindo as lições e o programa principal, está em https://github.com/whoissqr/WebGoat-v71-all-in-one.

Também neste caso, a opção mais fácil e direta de instalação é por meio de um contêiner Docker. A listagem 5.2 mostra os comandos para instalação do WebGoat por contêiner Docker. Quando a instalação é bem sucedida, a aplicação fica disponível em http://localhost:8080/WebGoat.

Listagem 5.2. WebGoat em Docker.

```
1 #WebGoat em Docker
2 docker pull webgoat/webgoat-7.1
3 docker run -p 8080:8080 -t webgoat/webgoat-7.1
```

O bWAPP (*buggy Web APPlication*) é uma aplicação PHP sobre MySQL bastante utilizada em exercícios de análise de vulnerabilidades. O tutorial oficial do bWAPP está em https://www.mmebvba.com/sites/default/files/downloads/bWAPP_intro.pdf.

Outro tutorial interessante está em https://wooly6bear.files.wordpress.com/2016/01/bwapp-tutorial.pdf. Uma máquina virtual *standalone* do bWAPP está em https://sourceforge.net/projects/bwapp/files/bee-box/. Porém, a opção mais fácil e direta utiliza um contêiner Docker, com instruções em https://hub.docker.com/r/raesene/bwapp.

A listagem 5.3 mostra os comandos para instalação por contêiner Docker. Quando a instalação é bem sucedida, a aplicação fica disponível em http://localhost:8090/. Em seguida, deve-se ativar a aplicação pela url http://localhost:8090/install.php, seguir as instruções da página e entrar na aplicação com login "bee"e senha "bug".

Listagem 5.3. bWAPP em Docker.

```
1 docker pull raesene/bwapp
2 docker run -d -p 8090:80 raesene/bwapp
```

Gruyere é uma aplicação web vulnerável para aprendizado desenvolvida em Python que foi originalmente criada pelo Google e hoje é mantida por terceiros. O site oficial da

Gruyere (`https://google-gruyere.appspot.com/`) oferece um tutorial de testes de segurança apoiado por um laboratório, uma máquina virtual para implantação local da aplicação e o código fonte. A opção mais fácil é direta utiliza um contêiner Docker em `https://hub.docker.com/r/karthequian/gruyere`.

A listagem 5.4 mostra os comandos para instalação por contêiner Docker. Quando a instalação é bem sucedida, a aplicação fica disponível em `http://localhost:8008/`.

Listagem 5.4. Gruyere em Docker.

```
1 #Google Gruyere em Docker
2 docker pull karthequian/gruyere
3 docker run -d -p 8008:8008 karthequian/gruyere
```

5.3.2. Revisão de código auxiliada por ferramentas SAST

Esta seção trata revisão de código apoiado por ferramentas SAST livres e de código aberto como, por exemplo, Horusec (`https://horusec.io/`) e o Sonarqube (`https://www.sonarqube.org/`). A seção utiliza as ferramentas como comandos de linha e compara os resultados em nível macro; isto é, totalizações de vulnerabilidades gerais por criticidade (ou risco). Exemplos individualizados de vulnerabilidades são oferecidos apenas visando ilustração. A análise prática dos resultados é feita preferencialmente em sala de aula ou laboratório.

5.3.2.1. Análise estática de código-fonte com SonarQube *Community*

O SonarQube *community* (`https://www.sonarqube.org/`) é uma ferramenta para inspeção contínua da qualidade do código, incluindo análise estática de código para uma grande variedade de linguagens, detecção de defeitos e vulnerabilidades e métricas de qualidade. Já o SonarLint (`https://www.sonarlint.org`) é uma extensão para IDE de código aberto que realiza inspeção de código durante a codificação. Como um corretor ortográfico, o SonarLint marca falhas e fornece *feedback* em tempo real e orientações de correção. É possível sincronizar o SonarLint com o perfil definido no SonarQube. A versão *community* tem mostrado bons resultados para detecção de violação de boas práticas de codificação, duplicação de código, complexidade excessiva, exibir métricas de código e cobertura.

No entanto, para a identificação de vulnerabilidades, há ressalvas. A edição *community* não possui o mesmo conjunto de regras das edições comerciais (*Developer*, *Cloud* e *Enterprise*). Por isso, observam-se mais falsos positivos (alarmes falsos) e falsos negativos (omissões) na edição comunitária que em outras edições, além de questões de integração de *scanners* modernos e atualização das regras para tecnologias de software mais novas. Esta situação diminui a confiança e prejudica a adoção da ferramenta como SAST. Por outro lado, existe uma comunidade de usuários bastante ativa e que pode ser consultada na busca por soluções para as lacunas da edição gratuita em situações específicas.

A distribuição gratuita para a comunidade de desenvolvedores pode ser obtida em `https://www.sonarsource.com/open-source-editions/`. Há três opções de instalação complementares: o SonarLint, um *plug-in* para integração em IDEs; o *SonarQube Server*, uma aplicação administrativa de gestão de varreduras e o `sonar-scanner`, o *scanner* de vulnerabilidade propriamente dito. O aluno interessado deve procurar a documentação de cada opção. As listagens a seguir mostram a utilização do `sonar-scanner`.

A listagem 5.5 mostra o comando para realização de uma varredura de vulnerabilidades com a ferramenta SAST SonarQube *Community* sobre o código-fonte da aplicação Juice Shop. Primeiramente o comando `pwd` mostra que o *prompt* de comando está na pasta `sast/-sources`. Em seguida, o comando `ls -l` lista o conteúdo da pasta `/sast/sources`, revelando uma pasta de código fonte para cada uma das aplicações vulneráveis. Já o comando `cd juice-shop-master/` posiciona o *prompt* na pasta do Juice Shop, enquanto o comando `sonar-scanner` realiza a varredura de vulnerabilidades com os seguintes parâmetros específicos:

- `-Dsonar.projectKey=juice-shop` é o identificador do projeto na aplicação administrativa do sonar e é criado junto com o projeto de análise de código fonte na ferramenta.
- `-Dsonar.sources=.` é a pasta corrente onde está o código-fonte.
- `-Dsonar.host.url=http://localhost:9000` é a URL da aplicação administrativa do sonar que vai receber o resultado da avaliação de segurança.
- `-Dsonar.login=sqa_d0d5432 ... 7a4f2b73b` é o *token* de autenticação usado pelo *scanner* no acesso autorizado à *console* administrativa. Este *token* é obrigatório e pode ser gerado durante a criação do projeto de *scan* ou apenas uma vez.

Se a instalação do SonarQube *out-of-the-box* NÃO mostra alertas para o Juice Shop, isso pode ser um defeito de configuração do *scanner*, que não é capaz de detectar problemas de segurança em *Typescrypt* apenas com a configuração *default*. Sabe-se que as vulnerabilidades existem, então as ausências são omissões (falsos negativos) do SonarQube. Por outro lado, o SonarQube mostra *Security Hotspots* para o Juice Shop nas seguintes categorias OWASP Top 10 2017: A1 (*Injection*), A2 (*Broken Authentication*), A3 (*Sensitive Data Exposure*), A6 (*Security Misconfiguration*). O aluno interessando deve procurar na comunidade pelas configurações de regras de segurança para *Typescrypt*.

Listagem 5.5. Varredura de código-fonte do Juice Shop com o SonarQube.

```
1 # scan do juiceshop
2 $ pwd
3 ./sast/sources
4 $ ls -l
5 total 20
6 drwxr-xr-x  6 ******* domain users 4096 mar 13 15:20 bwapp-code-master
7 drwxr-xr-x  4 ******* domain users 4096 mar 13 14:59 gruyere-app
8 drwxr-xr-x 24 ******* domain users 4096 abr 26 10:02 juice-shop-master
9 drwxr-xr-x  7 ******* domain users 4096 mar 13 16:02 secDevLabs-master
10 drwxr-xr-x 73 ******* domain users 4096 jul 24 14:54 WebGoat-v71-all-in-one-master
11 $ cd juice-shop-master/
12 $ sonar-scanner \
13   -Dsonar.projectKey=juice-shop \
14   -Dsonar.sources=. \
15   -Dsonar.host.url=http://localhost:9000 \
16   -Dsonar.login=sqa_d0d5432 ... e1c7a4f2b73b
```

A listagem 5.6 mostra o comando para uma varredura de vulnerabilidades com a ferramenta SAST SonarQube *Community* sobre o código-fonte da aplicação WebGoat 7.1. Primeiramente o comando `pwd` mostra que o *prompt* de comando está na pasta `/sast/sources`. Em seguida, comando `cd WebGoat-v71-all-in-one-master` posiciona o *prompt*

na pasta do WebGoat, enquanto o comando `sonar-scanner`, com parâmetros específicos, realiza a varredura de vulnerabilidades. O comando `sonar-scanner` utiliza os seguintes parâmetros específicos (diferentes da execução anterior):

- `-Dsonar.projectKey=WebGoat` é o identificador do projeto de *scan* na aplicação administrativa do sonar.
- `-Dsonar.sources=.` é a pasta corrente onde está o código-fonte.

No geral, o SonarQube mostra muitos alertas de vulnerabilidades do OWASP Top 10 de 2017 para o WebGoat. Entre eles, estão os seguintes: A3 *Sensitive Data Exposure*, A1 *Injection*, A6 *Security Misconfiguration*, A4 *XML External Entities* (XXE) e A5 *Broken Access Control*. O SonarQube também mostra muitos *Security Hotspots* do OWASP Top 10 de 2021 para o WebGoat, a saber: A1 (*Broken Access Control*), A2 (*Cryptographic Failures*), A3 (*Injection*), A4 (*Insecure Design*), A5 (*Security Misconfiguration*), A7 (*Identification and Authentication Failures*). O resultado da análise depende da versão do *scanner*, do acesso ao código fonte e da configuração da aplicação. Além disso, muitos alertas podem ser falsos positivos.

Listagem 5.6. Varredura de código-fonte do WebGoat 7.1 com o SonarQube.

```
1 # scan do webgoat
2 $ pwd
3 ./sast/sources
4 $ cd WebGoat-v71-all-in-one-master
5 $ sonar-scanner \
6   -Dsonar.projectKey=WebGoat \
7   -Dsonar.sources=. \
8   -Dsonar.java.binaries=../sast/binaries/webgoat-container-7.1-exec.jar\
9   -Dsonar.host.url=http://localhost:9000 \
10  -Dsonar.login=sqa_d0d5432 ... 7a4f2b73b
```

A listagem 5.7 mostra o comando para realização de uma varredura de vulnerabilidades com a ferramenta SAST SonarQube *Community* sobre o código-fonte da aplicação bWAPP. Primeiramente o comando `pwd` mostra que o *prompt* de comando está na pasta `/sast/-sources`. Em seguida, o comando `cd bwapp-code-master` posiciona o *prompt* na pasta do bWAPP, enquanto o comando `sonar-scanner` realiza a análise com parâmetros específicos, a saber:

- `-Dsonar.projectKey=bWAPP` é o identificador do projeto de *scan* na aplicação administrativa do sonar.
- `-Dsonar.sources=.` é a pasta corrente onde está o código-fonte.

O SonarQube na configuração padrão pode não mostrar alertas para o bWAPP, mas associa vários *Security Hotspots* ao OWASP Top 10 2017, a saber: A1 (*Injection*), A2 (*Broken Authentication*), A3 (*Sensitive Data Exposure*) (criptografia fraca), A6 (*Security Misconfiguration*), A7 *Cross-Site Scripting (XSS)*), A9 (*Components with vulnerabilities*) e A10 (*Insufficient Logging and Monitoring*). O aluno interessando deve procurar na comunidade SonarQube pelas configurações de regras de segurança para PHP.

Listagem 5.7. Varredura de código-fonte do bWAPP com o SonarQube.

```
1 # scan do bWAPP
2 $ pwd
3 ./sast/sources
4 $ cd bwapp-code-master
5 sonar-scanner \
6   -Dsonar.projectKey=bWAPP \
7   -Dsonar.sources=. \
8   -Dsonar.host.url=http://localhost:9000 \
9   -Dsonar.login=sqa_d0d5432 ... 7a4f2b73b
```

A listagem 5.8 mostra o comando para realização de uma varredura de vulnerabilidades com a ferramenta SAST SonarQube *Community* sobre o código-fonte da aplicação Gruyere. Com o *prompt* na pasta `sast/sources/gruyere-app`, o comando `sonar-scanner` é executado com parâmetros específicos:

- `-Dsonar.projectKey=Gruyere` é o identificador do projeto de *scan* na aplicação administrativa do sonar.
- `-Dsonar.sources=.` é a pasta corrente onde está o código-fonte.

O *scanner* padrão do SonarQube pode não mostrar alertas nem *Security Hotspots* para o *Gruyere*. O resultado da análise depende da versão do *scanner*, do acesso ao código fonte e da configuração da aplicação. Há nitidamente uma deficiência do *scanner* padrão para a linguagem Phyton, resultando em falsos negativos (omissões). Assim como das vezes anteriores, para obter resultados mais detalhados, aluno interessando deve procurar na comunidade pelas configurações de regras de segurança específicas de Python.

Listagem 5.8. Varredura de código-fonte do Gruyere com o SonarQube.

```
1 #scan do Gruyere
2 $ pwd
3 ./sast/sources
4 $ cd gruyere-app
5 $ sonar-scanner \
6   -Dsonar.projectKey=Gruyere \
7   -Dsonar.sources=. \
8   -Dsonar.host.url=http://localhost:9000 \
9   -Dsonar.login=sqa_d0d5432 ... 7a4f2b73b
```

5.3.2.2. Análise estática de código-fonte com HoruSec

O Horusec (`https://horusec.io/`) é uma ferramenta de análise estática de código aberto, projetada para identificar vulnerabilidades de segurança durante a fase de desenvolvimento. Com suporte para as principais linguagens de programação, o Horusec auxilia os desenvolvedores na detecção e correção proativa de vulnerabilidades em seu código. Além disso, o Horusec funciona como um integrador de vários *scanners* de vulnerabilidades, permitindo a consolidação de resultados e simplificando o processo de análise de segurança.

Uma vez instalado, a utilização do HoruSec *standalone* por linha de comando é bastante simples. As listagens 5.9, 5.10, 5.11 e 5.12 mostram trechos dos resultados das análises do HoruSec sobre os códigos-fonte das aplicações vulneráveis. Em todos os casos, a ferramenta é ativada a partir da pasta de código fonte da aplicação com o comando `horusec start -p . --disable-docker="true"`, com a opção Docker desligada.

A varredura com mais vulnerabilidades de risco alto (críticas e altas) foi a do Juice Shop (364 + 101 = 465), seguida pela do WebGoat em segundo (62 + 193 = 255) e pela do bWAPP em terceiro (147 + 11 = 158). A Gruyere ficou em último com apenas uma vulnerabilidade crítica descoberta. No geral, muitos alertas podem ser falsos positivos, mas todos os exemplos de vulnerabilidades mostrados nas listagens são positivos verdadeiros. Uma vez que sabemos que Gruyere possui várias vulnerabilidades exploráveis, pode-se concluir que, neste caso, a varredura com o HorusSec possui vários falsos negativos (omissões).

Listagem 5.9. Varredura de código-fonte do Juice Shop com o HoruSec.

```
# scan do juiceshop
$ pwd
./sast/sources
$ cd juice-shop-master/
$ horusec start -p . --disable-docker="true"

. . .
# Exemplo de vulnerabilidade encontrada
================================================================================

Language: Leaks
Severity: CRITICAL
Line: 80
Column: 41
SecurityTool: HorusecEngine
Confidence: MEDIUM
File: ./frontend/src/app/Services/two-factor-auth-service.spec.ts
Code: expect(req.request.body).toEqual({ password: 's3cr3t!' })
RuleID: HS-LEAKS-26
Type: Vulnerability
ReferenceHash: edeff4a1dc1234ed7ea5cb2c5e8b6c7065f802776470fe12985c209296162382
Details: (1/1) * Possible vulnerability detected: Hard-coded password
The software contains hard-coded credentials, such as a password or cryptographic
key, which it uses for its own inbound authentication, outbound communication to
external components, or encryption of internal data. For more information checkout
the CWE-798 (https://cwe.mitre.org/data/definitions/798.html) advisory.

================================================================================

In this analysis, a total of 474 possible vulnerabilities were found and we
classified them into:
Total of Vulnerability CRITICAL is: 364
Total of Vulnerability HIGH is: 101
Total of Vulnerability MEDIUM is: 5
Total of Vulnerability LOW is: 4
```

Listagem 5.10. Varredura de código-fonte do WebGoat 7.1 com o HoruSec.

```
# scan do webgoat
$ pwd
./sast/sources
$ cd WebGoat-v71-all-in-one-master
$ horusec start -p . --disable-docker="true"

. . .
# Exemplo de vulnerabilidade encontrada
================================================================================

Language: JavaScript
Severity: CRITICAL
Line: 22
Column: 25
SecurityTool: HorusecEngine
Confidence: MEDIUM
File: ./xxe/src/main/resources/plugin/XXE/js/xxe.js
Code: var result = eval('(' + req.responseText + ')');
```

```
RuleID: HS-JAVASCRIPT-2
Type: Vulnerability
ReferenceHash: 2dc6ab35cd1e763bb67697b71c824079e8672604b83e78ccd74cef3938eae043
Details: (1/1) * Possible vulnerability detected: No use eval
The eval function is extremely dangerous. Because if any user input is not handled
correctly and passed to it, it will be possible to execute code remotely in the
context of your application (RCE - Remote Code Executuion). For more information
checkout the CWE-94 (https://cwe.mitre.org/data/definitions/94.html) advisory.

================================================================================

In this analysis
     , a total of 299 possible vulnerabilities were found and we classified them into:
Total of Vulnerability LOW is: 5
Total of Vulnerability CRITICAL is: 62
Total of Vulnerability HIGH is: 193
Total of Vulnerability MEDIUM is: 39
```

Listagem 5.11. Varredura de código-fonte do bWAPP com o HoruSec.

```
# scan do bWAPP
$ pwd
./sast/sources
$ cd bwapp-code-master
$ horusec start -p . --disable-docker="true"

...
# Exemplo de vulnerabilidade encontrada
================================================================================

Language: JavaScript
Severity: CRITICAL
Line: 465
Column: 20
SecurityTool: HorusecEngine
Confidence: MEDIUM
File: ./bWAPP/js/json2.js
Code: j = eval('(' + text + ')');
RuleID: HS-JAVASCRIPT-2
Type: Vulnerability
ReferenceHash: 928b1f21a15339da9b0296fe25efa13a8908b97667280a0d6f4bed00b24ad1bf
Details: (1/1) * Possible vulnerability detected: No use eval
The eval function is extremely dangerous. Because if any user input is not handled
correctly and passed to it, it will be possible to execute code remotely in the
context of your application (RCE - Remote Code Executuion). For more information
checkout the CWE-94 (https://cwe.mitre.org/data/definitions/94.html) advisory.

================================================================================

In this analysis, a total of 158 possible vulnerabilities were found and we
classified them into:
Total of Vulnerability CRITICAL is: 147
Total of Vulnerability HIGH is: 11
```

Listagem 5.12. Varredura de código-fonte do Gruyere com o HoruSec.

```
#scan do Gruyere
$ pwd
./sast/sources
$ cd gruyere-app
$ horusec start -p . --disable-docker="true"
. . .

# Exemplo de vulnerabilidade encontrada
================================================================================

Language: JavaScript
Severity: CRITICAL
Line: 25
```

```
13 Column: 6
14 SecurityTool: HorusecEngine
15 Confidence: MEDIUM
16 File: ./resources/lib.js
17 Code: eval('(' + httpRequest.responseText + ')');
18 RuleID: HS-JAVASCRIPT-2
19 Type: Vulnerability
20 ReferenceHash: 985b9e6639f077e0c29136190bbc01be3a187bf6da24f635a90efa0f73d018a1
21 Details: (1/1) * Possible vulnerability detected: No use eval
22 The eval function is extremely dangerous. Because if any user input is not handled
23 correctly and passed to it, it will be possible to execute code remotely in the
24 context of your application (RCE - Remote Code Execution). For more information
25 checkout the CWE-94 (https://cwe.mitre.org/data/definitions/94.html) advisory.
26
27 ================================================================================
28
29 In this analysis, a total of 1 possible vulnerabilities were found and we
30 classified them into:
31 Total of Vulnerability CRITICAL is: 1
```

5.3.3. Avaliação de segurança auxiliada por ferramentas SCA

Esta seção trata ferramentas SCA livres e gratuitas, tais como OWASP Dependency-Check (`https://owasp.org/www-project-dependency-check`) e o Trivy (`https://trivy.dev/`) (especialmente quando aplicado a imagens Docker). A seção utiliza as ferramentas como comandos de linha e compara os resultados em nível macro; isto é, totalizações de vulnerabilidades gerais por criticidade. Exemplos individualizados de vulnerabilidades são oferecidos apenas visando ilustração. A análise prática dos resultados é feita preferencialmente em sala de aula ou laboratório.

5.3.3.1. Avaliação de segurança com o OWASP Dependency-Check

O Dependency-Check (`https://owasp.org/www-project-dependency-check`) é uma ferramenta livre e de código aberto do OWASP e faz a varredura de vulnerabilidades antecipada de aplicações para encontrar componentes e dependências vulneráveis nas fases iniciais do desenvolvimento. Ela atua sobre código binário da aplicação é complementar a outras ferramentas similares, como o Trivy, que é indicado para etapas tardias do pipeline. Trata o item A06 (*Vulnerable and Outdated Components*) da lista Top 10 2021 do OWASP e suporta várias linguagens de programação, podendo ser integrado a *pipelines* de CI/CD.

A avaliação de segurança com o Dependency-Check é feita com o comando de linha `dependency-check.sh --scan <<pasta com bibliotecas>>`, que produz como saída um relatório em `html`. As avaliações das quatro aplicações vulneráveis pelo Dependency-Check apresentaram resultados esclarecedores. A análise foi bem sucedida apenas em relação ao Juice Shop e ao WebGoat, são aplicações web com arquitetura moderna e fortemente baseada em componentes. Isto faz sentido, uma vez que a preocupação com dependências inseguras é relativamente nova e relacionada ao uso crescente de componentes de terceiros no software atual.

A Tabela 5.1 mostra as vulnerabilidades por dependência, indicando também a quantidade de CVEs associados à dependência, o grau de confiança na análise e a quantidade de evidências encontradas (que justificam a confiança). No Juice Shop, foram encontradas 5 vulnerabilidade críticas e 15 altas, de um total de 30 vulnerabilidades, em componentes variados, todas

Tabela 5.1. Vulnerabilidades encontradas com o Dependency-Check.

Aplicação	Dependência	Severidade	CVEs	Confiança	Evidências
Juice Shop	bench.js	HIGH	1		3
Juice Shop	blaze.jar	MEDIUM	1	Highest	86
Juice Shop	dottie:2.0.3	HIGH	1	Highest	9
Juice Shop	express-jwt:0.1.3	CRITICAL	1	Highest	10
Juice Shop	hbs:4.2.0	MEDIUM	1	Highest	10
Juice Shop	jquery.js	MEDIUM	6		3
Juice Shop	jsonwebtoken:0.4.0	CRITICAL	4	Highest	8
Juice Shop	moment.js	HIGH	4		3
Juice Shop	moment.min.js	HIGH	4		3
Juice Shop	notevil:1.3.3	MEDIUM	1	Highest	8
Juice Shop	request:2.88.2	MEDIUM	1	Highest	9
Juice Shop	sanitize-html:1.4.2	HIGH	4	Highest	8
Juice Shop	semver:7.3.8	HIGH	1		8
WebGoat	bootstrap.min.js	MEDIUM	7		3
WebGoat	jquery-1.10.2.min.js	MEDIUM	6		3
WebGoat	jquery-ui-1.10.4.custom.min.js	MEDIUM	5		5
WebGoat	underscore-min.js	HIGH	1		3
WebGoat	underscore.js	HIGH	1		3

elas com CVEs identificados. Já no WebGoat, duas (2) vulnerabilidades altas foram encontradas, de um total de 20 vulnerabilidades. Não houve relato de vulnerabilidades nas outras aplicações.

Vale observar que nestes casos, em se tratando de componentes externos de terceiros e fora do controle do time de desenvolvimento das aplicações analisadas, a estratégia de correção recomendada é a atualização das dependências vulneráveis por versões mais seguras, observando-se a manutenção do código fonte que utiliza a dependência atualizada. Além disso, a ferramenta Dependency-Track (`https://dependencytrack.org/`) pode ser usada quando o objetivo final é a gestão de segurança das dependências, no conceito de *Software Bill of Materials* (SBOM), indo além da análise de segurança de componentes de SCA.

5.3.3.2. Avaliação de segurança com o Trivy

O Trivy (`https://trivy.dev/`) é uma ferramenta gratuita para fazer a varredura de vulnerabilidades em imagens Docker, sistema de arquivos e arquivos de Infraestrutura como Serviço (*infrastructure as a Service* – IaC) e geração e verificação de *Software Bill of Materials* (SBOM). Ela pode ser adicionada a esteira para verificação tardia de vulnerabilidades em imagens Docker e rodar localmente durante o desenvolvimento das imagens de contêiner. A ferramenta deve ser usada em conjunto com outras para aumentar a cobertura da segurança das aplicações.

Na listagem 5.2, o comando `docker images` lista as imagens instaladas das quatro aplicações vulneráveis. As saídas dos comandos foram editados para melhorar a formatação e o uso do espaço. A varredura de segurança com Trivy sobre imagens Docker é feita com a linha de comando `trivy image NOME-IMAGEM-DOCKER`. A listagem mostra os comandos Trivy para cada uma das imagens, nas linhas 8, 19, 38 e 49. Em cada caso, o resultado da varredura é direcionada para um arquivo correspondente à aplicação analisada. Todas as varreduras fazem análises de vulnerabilidades e de segredos dos componentes da aplicação e também do

Tabela 5.2. Vulnerabilidades críticas e altas encontradas com o Trivy.

Aplicação	Componentes	Total geral	Total Críticas	Total Altas
Juice Shop	bkimminich/juice-shop	21	0	4
Juice Shop	Node.js (node-pkg)	66	7	14
Juice Shop	/juice-shop/lib/insecurity.ts (secrets)	1	0	1
Juice Shop	**Totais**	88	7	19
WebGoat	webgoat/webgoat-7.1 (debian 8.6)	513	76	169
WebGoat	Java (jar)	118	36	44
WebGoat	**Totais**	631	112	213
bWAPP	raesene/bwapp (ubuntu 14.04)	1778	0	28
bWAPP	/private/ssl-cert-snakeoil.key (secrets)	1	0	1
bWAPP	**Totais**	1779	0	29
Gruyere	karthequian/gruyere (alpine 3.3.3)	1	1	0
Gruyere	**Totais**	1	1	0

sistema operacional da imagem.

A tabela 5.2 resume as vulnerabilidades identificadas pelo Trivy nestas avaliações. No Juice Shop, 88 CVEs foram identificadas na imagem e no node.js, sendo 7 críticas e 19 altas. No WebGoat, foram 631 CVEs na imagem e arquivos Jar, sendo 112 críticas e 213 altas. No bWAPP, foram 1779 CVEs na imagem, com apenas 28 altas. Gruyere apresentou o resultado modesto, com apenas uma CVE crítica. De modo geral, Trivy gerou mais alertas de vulnerabilidades que Dependency-Check.

Listagem 5.13. Varredura de imagens Docker com Trivy.

```
$ docker images
REPOSITORY                    TAG       IMAGE ID       CREATED         SIZE
bkimminich/juice-shop         latest    8493311c32c9   5 months ago    580MB
webgoat/webgoat-7.1           latest    20fef7540300   6 years ago     460MB
raesene/bwapp                 latest    8be28fba48ec   7 years ago     441MB
karthequian/gruyere           latest    4ed76f33b77f   3 years ago     210MB
$
$ sudo trivy image bkimminich/juice-shop > ./trivy-juice.txt
11:56:05  INFO  Vulnerability scanning is enabled
11:56:05  INFO  Secret scanning is enabled
11:56:05  INFO
       If your scanning is slow, please try '--scanners vuln' to disable secret scanning
11:56:05  INFO  Please see also https://aquasecurity.github
     .io/trivy/v0.43/docs/scanner/secret/#recommendation for faster secret detection
11:56:07  INFO  Detected OS: debian
11:56:07  INFO  Detecting Debian vulnerabilities...
11:56:07  INFO  Number of language-specific files: 1
11:56:07  INFO  Detecting node-pkg vulnerabilities...
11:56:07  INFO  Table result includes only package
      filenames. Use '--format json' option to get the full path to the package file.
$
$ sudo trivy image webgoat/webgoat-7.1    > ./trivy-webgoat.txt
11:57:190 INFO  Vulnerability scanning is enabled
11:57:19  INFO  Secret scanning is enabled
11:57:19  INFO
       If your scanning is slow, please try '--scanners vuln' to disable secret scanning
11:57:19  INFO  Please see also https://aquasecurity.github
     .io/trivy/v0.43/docs/scanner/secret/#recommendation for faster secret detection
11:57:34  INFO  JAR files found
11:57:34  INFO  Java DB Repository: ghcr.io/aquasecurity/trivy-java-db:1
11:57:34  INFO  Downloading the Java DB...
446.69 MiB / 446.69 MiB
      [-------------------------------------------------] 100.00% 7.27 MiB p/s 1m2s
11:58:37  INFO  The Java DB is cached for 3 days. If you
     want to update the database more frequently, the '--reset' flag clears the DB cache.
```

```
29 11:58:37  INFO  Analyzing JAR files takes a while...
30 11:58:38  INFO  Detected OS: debian
31 11:58:38  INFO  Detecting Debian vulnerabilities...
32 11:58:38  INFO  Number of language-specific files: 1
33 11:58:38  INFO  Detecting jar vulnerabilities...
34 11:58:38  WARN  This OS version is no longer supported by the distribution: debian 8.6
35 11:58:38  WARN  The vulnerability
      detection may be insufficient because security updates are not provided
36 11:58:38  INFO  Table result includes only package
      filenames. Use '--format json' option to get the full path to the package file.
37 $
38 $ sudo trivy image raesene/bwapp         > ./trivy-bwapp.txt
39 12:01:50  INFO  Vulnerability scanning is enabled
40 12:01:50  INFO  Secret scanning is enabled
41 12:01:50  INFO
       If your scanning is slow, please try '--scanners vuln' to disable secret scanning
42 12:01:50  INFO  Please see also https://aquasecurity.github
     .io/trivy/v0.43/docs/scanner/secret/#recommendation for faster secret detection
43 12:02:15  INFO  Detected OS: ubuntu
44 12:02:15  INFO  Detecting Ubuntu vulnerabilities...
45 12:02:15  INFO  Number of language-specific files: 0
46 12:02:15  WARN  This OS version is no longer supported by the distribution: ubuntu 14.04
47 12:02:15  WARN  The vulnerability
      detection may be insufficient because security updates are not provided
48 $
49 $ sudo trivy image karthequian/gruyere   > ./trivy-gruyere.txt
50 12:02:49  INFO  Vulnerability scanning is enabled
51 12:02:49  INFO  Secret scanning is enabled
52 12:02:49  INFO
       If your scanning is slow, please try '--scanners vuln' to disable secret scanning
53 12:02:49  INFO  Please see also https://aquasecurity.github
     .io/trivy/v0.43/docs/scanner/secret/#recommendation for faster secret detection
54 12:02:56  INFO  Detected OS: alpine
55 12:02:56  INFO  Detecting Alpine vulnerabilities...
56 12:02:56  INFO  Number of language-specific files: 1
57 12:02:56  INFO  Detecting python-pkg vulnerabilities...
58 12:02:56  WARN  This OS version is no longer supported by the distribution: alpine 3.3.3
59 12:02:56  WARN  The vulnerability
      detection may be insufficient because security updates are not provided
```

Ferramentas SCA fazem análise estática sobre pacotes binários do software. Nesse sentido, outro tipo de análise estática não explorado neste texto é a análise de binários de aplicativos móveis com auxílio de ferramentas. O MobSF (`https://github.com/MobSF/Mobile-Security-Framework-MobSF`) é uma ferramenta de análise de vulnerabilidades em aplicativos móveis capaz de fazer análises estáticas e dinâmicas sobre os binários dos aplicativos em vários formatos (por exemplo, APK, XAPK, IPA e APPX). Testes de segurança em aplicativos móveis já foram tratados em outro minicurso [Braga et al. 2012].

5.3.4. Testes de segurança auxiliados por ferramentas DAST

Esta seção trata ferramentas DAST livres e gratuitas, tais como, por exemplo, OWASP ZAP `https://www.zaproxy.org/`) e Burp Community (`https://portswigger.net/burp`). O texto consiste em testes de segurança manuais apoiados pelas ferramentas DAST, com base em catálogos de vulnerabilidades conhecidas, como OWASP Top 10 [top 2021] e CVEs conhecidos (`https://cve.mitre.org/`). O OWASP ZAP também é usado em varreduras de vulnerabilidades.

5.3.4.1. Testes de segurança com OWASP ZAP

O OWASP ZAP `https://www.zaproxy.org/`) é uma ferramenta livre e de código aberto, para a realização de análise dinâmica e projetada para identificar vulnerabilidades em aplicações web. Ele oferece recursos abrangentes para avaliar a segurança de um aplicativo, incluindo a detecção de falhas de segurança comuns, como injeção de SQL, *cross-site scripting* (XSS) e configurações incorretas. Com suporte para testes automatizados e manuais, o OWASP ZAP é amplamente utilizado por desenvolvedores e profissionais de segurança para análises dinâmicas em aplicações e testes de segurança em APIs.

Os comandos de linha mostrados na listagem 5.14 fazem as varreduras de vulnerabilidades com OWASP ZAP nas quatro aplicações web vulneráveis usadas neste texto. A linha de comando o ZAP contém os seguintes elementos:

- `zap.sh` é o *script* de ativação do ZAP.
- `-cmd` indica execução por comando de linha.
- `-quickurl http://<<EXEMPLO.COM>>` informa a URL alvo da varredura.
- `-quickprogress` mostra uma barra de progresso em texto.
- `-quickout <<arquivo>>` informa arquivo de saída. A extensão indica o tipo do arquivo (.html, .json, .md, .xml).

A tabela 5.3 mostra resumidamente os alertas emitidos pelo ZAP para cada uma das aplicações analisadas. A varredura detectou problemas em 4 níveis de alertas: alto, médio, baixo e informacional (vazamento de informação). Apenas as aplicações Gruyere e Juice Shop possuem alertas altos. Em quantidade de alertas, a distribuição fica em Gruyere com 123 alertas, bWAPP com 409 alertas, WebGoat com 171 alertas e Juice Shop com 370 alertas, fazendo do bWAPP a aplicação mais vulnerável desta análise com o ZAP. Claro, exceto pelos possíveis falsos positivos, os alertas são verdadeiros.

Listagem 5.14. Varreduras de vulnerabilidades com OWASP ZAP.

```
1
2 $ zap.sh -cmd -quickurl http://localhost:3000 -quickprogress -quickout ./zap-juice.html
3
4 ...
5
6 $ zap.sh -cmd
      -quickurl http://localhost:8080/WebGoat -quickprogress -quickout ./zap-webgoat.html
7 ...
8
9 $ zap.sh -cmd -quickurl http://localhost:8090 -quickprogress -quickout ./zap-bwapp.html
10
11 ...
12
13 $ zap.sh -cmd -quickurl ttp://localhost:8008 -quickprogress -quickout ./zap-gruyere.html
```

O OWASP ZAP foi utilizado até este ponto como uma ferramenta de linha de comando para varredura de vulnerabilidades. Porém, ele é muito mais que isso. O ZAP também possui uma interface gráfica (GUI) e um dos seus usos mais importantes é como *proxy* de aplicação (ou

Tabela 5.3. Alertas emitidos pelo ZAP durante as varreduras das aplicações vulneráveis.

Aplicação	Alerta	Nível de Risco	Quant.
Gruyere	Cross Site Scripting (Refletido)	Alto	2
Gruyere	Metadados de nuvem potencialmente expostos	Alto	1
Gruyere	Ausência de tokens Anti-CSRF	Médio	3
Gruyere	Content Security Policy (CSP) Header Not Set	Médio	10
Gruyere	Hidden File Found	Médio	4
Gruyere	Injeção CRLF	Médio	1
Gruyere	Missing Anti-clickjacking Header	Médio	10
Gruyere	Cookie No HttpOnly Flag	Baixo	1
Gruyere	Cookie without SameSite Attribute	Baixo	1
Gruyere	Server Leaks Version Information via HTTP Response Header	Baixo	11
Gruyere	X-Content-Type-Options Header Missing	Baixo	11
Gruyere	Cookie Poisoning	Informativo	2
Gruyere	Cookie com Escopo Fraco	Informativo	1
Gruyere	Divulgação de Informações - Comentários Suspeitos	Informativo	1
Gruyere	Modern Web Application	Informativo	4
Gruyere	User Agent Fuzzer	Informativo	60
bWAPP	Application Error Disclosure	Médio	27
bWAPP	Ausência de tokens Anti-CSRF	Médio	5
bWAPP	Content Security Policy (CSP) Header Not Set	Médio	36
bWAPP	Hidden File Found	Médio	1
bWAPP	Missing Anti-clickjacking Header	Médio	35
bWAPP	Navegação no Diretório	Médio	32
bWAPP	Cookie No HttpOnly Flag	Baixo	2
bWAPP	Cookie without SameSite Attribute	Baixo	2
bWAPP	Vaza informações via HTTP Response Header X-Powered-By	Baixo	10
bWAPP	Server Leaks Version Information via HTTP Response Header	Baixo	81
bWAPP	X-Content-Type-Options Header Missing	Baixo	75
bWAPP	Cookie com Escopo Fraco	Informativo	2
bWAPP	Divulgação de Informações - Comentários Suspeitos	Informativo	1
bWAPP	GET for POST	Informativo	1
bWAPP	User Agent Fuzzer	Informativo	96
bWAPP	User Controllable HTML Element Attribute (Potential XSS)	Informativo	3
WebGoat	Ausência de tokens Anti-CSRF	Médio	3
WebGoat	Content Security Policy (CSP) Header Not Set	Médio	4
WebGoat	Missing Anti-clickjacking Header	Médio	3
WebGoat	Session ID in URL Rewrite	Médio	1
WebGoat	Cookie without SameSite Attribute	Baixo	2
WebGoat	Server Leaks Version Information via HTTP Response Header	Baixo	14
WebGoat	X-Content-Type-Options Header Missing	Baixo	7
WebGoat	Cookie com Escopo Fraco	Informativo	2
WebGoat	Divulgação de Informações - Comentários Suspeitos	Informativo	3
WebGoat	User Agent Fuzzer	Informativo	132
Juice Shop	Metadados de nuvem potencialmente expostos	Alto	1
Juice Shop	Configuração Incorreta Entre Domínios	Médio	85
Juice Shop	Content Security Policy (CSP) Header Not Set	Médio	70
Juice Shop	Cross-Domain JavaScript Source File Inclusion	Baixo	124
Juice Shop	Divulgação de Data e Hora - Unix	Baixo	1
Juice Shop	Divulgação de Informações - Comentários Suspeitos	Informativo	2
Juice Shop	Modern Web Application	Informativo	63
Juice Shop	User Agent Fuzzer	Informativo	24

proxy de navegador web). Os próximos parágrafos mostram resumidamente como configurar o OWASP ZAP e em seguida usá-lo para testar manualmente a segurança de aplicações web.

Após instalado com sucesso, o ZAP pode ser iniciado pelo ícone na área de trabalho ou pelo *script* de linha de comando (zap.sh ou zap.bat). A GUI do ZAP inicia com uma janela de abertura, progresso e anúncios (até a versão 2.12.0).

- Em seguida, o ZAP solicita que o usuário escolha como deseja persistir a sessão.
- Neste treinamento, prefere-se a opção Não.
- Clicar no botão de Início.

A tela inicial (início rápido) do ZAP oferece três maneiras de iniciar:

- Varredura automatizada (*Automated Scan*).
- Exploração manual (*Manual Explore*).
- Documentação (*Learn more*).

A seguir, as funcionalidades de varredura automática e exploração manual são demonstradas sobre as aplicações vulneráveis para aprendizado Juice Shop e WebGoat 7.1 (já apresentadas anteriormente). As aplicações vulneráveis devem ser iniciadas antes de prosseguir com os exercícios. A função *Automated Scan* equivale à varredura em comando de linha e pode ser ativada do seguinte modo:

- Clicar no botão de *Automated Scan*.
- Na Janela *Automated Scan*.
  - Fornecer a URL a ser varrida, p.ex. `http://localhost:8080/WebGoat`.
  - Usar as opções *Spider*, *Ajax Spider* com *Firefox headless*.
  - Clicar no botão *Attack*/Ataque.
- O progresso da varredura é mostrado na barra de progresso do *Spider/AjaxSpider*.
- Os ataques automáticos são mostrados na aba Varredura Ativa.
- O resultado do ataque é mostrado na aba Alertas.
- A varredura pode alertar sobre 4 níveis de problemas: alto, médio, baixo e informacional (vazamento de informação).

O teste automático de outra aplicação mostra resultado diferente? Para verificar esta hipótese, basta repetir o procedimento anterior com uma URL diferente. Por exemplo, a URL do Juice Shop (`http://localhost:3000`).

A exploração manual (*Manual Explore*) segue o seguinte procedimento inicial:

- Na Janela Início Rápido, clique no botão de Manual Explore.
- Na janela Manual Explore.
  - Fornecer a URL a ser explorada, p.ex. `http://localhost:8080/WebGoat`.
  - Desmarcar a caixa *Enable HUD*.
  - Escolher o *browser* a ser aberto: Firefox ou Chrome.
  - Ativar o botão *Launch Browser*.

A exploração manual é a atividade de teste de segurança manual propriamente dita e utiliza-se, conforme a necessidade do testador, de diversos procedimentos, tais como criação de *breakpoints* e filtros, intercepção e modificação de requisições, edição e reenvio de requisições e o uso de opções de *proxy*. Estes procedimentos são detalhados da seguir.

A criação de *breakpoints* e filtros é útil para a interceptação apenas das requisições úteis para o trabalho de teste. Por exemplo, utiliza-se o seguinte procedimento para criar um *breakpoint* para as requisições HTTP do tipo POST:

- Clicar no botão X vermelho, *breakpoint*, na barra de botões de controle de *breakpoint* do ZAP (a barra com botões de ativa/desativa, avançar, *play next*, e *breakpoint*).
- Na janela aberta de criação de *breakpoints*, preencher os campos:
  - *String*: POST.
  - Localização: Cabeçalho da Solicitação (*Request Header*).
  - *Match*: Contém / *Contain*.
  - Clicar no botão Salvar.

Os *breakpoints* criados aparecem na aba ou guia de *breakpoints* e podem ser ativados/-desativados pelo *checkbox*. Assim o controle manual de *breakpoint* (botões verde/vermelho) não é mais necessário.

Já o filtro de requisições para o WebGoat pode ser configurado do seguinte modo. Na guia de histórico de requisições, clicar no botão de Filtro, inserir no campo URL *Inc Regex* a *string* `.*.WebGoat.*.` (incluindo os pontos) para mostrar apenas as requisições relacionadas ao WebGoat.

A intercepção e modificação de requisições pode ser experimentada com o seguinte procedimento (com o *breakpoint* ativado):

- No WebGoat 7.1, ir para o menu General → Http *basics*.
- No campo de formulário *Enter your name*, digite a *string* “teste” e ative o botão *Go!*.
- O ZAP indicará que interceptou a requisição.
- No corpo da requisição, modificar o valor do parâmetro como quiser.

- Por exemplo, adicionando "123" ao final da *string* "teste".
- Clicar em um dos botões *Play Next* ou Avançar para dar andamento à requisição.
- (Lembrar de desativar o *breakpoint*!)

A edição e reenvio de requisições podem ser experimentadas com facilidade. No Histórico de requisições do ZAP, selecionar a requisição mais recente e seguir o procedimento:

- Clicar com botão direito do mouse, escolher a opção Reenviar.
- O ZAP abre a janela editor manual de requisições.
- Na janela de edição de requisição:
  - Na guia Requisição modificar o parâmetro no corpo.
  - Na guia Resposta, que está vazia ainda, clicar no botão Enviar.
  - A resposta é enviada.
  - Lembrar de fechar a janela do editor manual de requisições.

As opções de *proxy* podem ser experimentadas com o seguinte procedimento. O ZAP pode servir de intermediário entre o navegador e a aplicação web. Há três maneiras de fazer isto:

- A maneira antiga, configurar manualmente o ZAP como *proxy* de aplicação no navegador. Esta opção não será usada neste treinamento.
- Por meio da opção de início rápido Manual Explore, como visto anteriormente.
- Com o botão de ativação do navegador (ícone do navegador), abrir um navegador intermediado e controlado remotamente pelo ZAP *Proxy*.

Os procedimentos explicados até aqui são usados em dois exemplos de exploração de vulnerabilidades. O exemplo a seguir mostra a exploração de uma vulnerabilidade de tratamento incorreto de erro no WebGoat com OWASP ZAP:

- No WebGoat 7.1 via ZAP Proxy.
  - Menu *Improper Error Handling* → *Fail Open Authentication Scheme*.
  - No formulário, digitar user/password administrativos (`webgoat/webgoat`).
  - clicar no botão Login.
  - Logou como admin? Fazer logout.
- No ZAP, ativar o *breakpoint* de interceptação de requisições.
- No WebGoat, fazer novamente o login *webgoat/webgoat* no mesmo formulário.
- O ZAP indica que interceptou a requisição, remover o parâmetro senha do corpo da requisição.

  - Era assim: `Username=webgoat&Password=webgoat&SUBMIT=Login`.
  - Fica assim: `Username=webgoat&SUBMIT=Login`.
- Enviar a requisição (Botão *Play* dos controles de *breakpoint*).
- Terá logado como admin sem fornecer a senha!
- Erro de projeto de programa! WebGoat falha em aberto em caso de erro na autenticação.

O Exemplo de exploração a seguir contorna ou burla a validação de dados de entrada feira pelo lado cliente (*frontend*) da aplicação.

- Entrar no seu deploy do WebGoat `http://localhost:8080/WebGoat/`.
- Logar na aplicação como usuário/senha: guest/guest
- Selecionar a opção de menu *Parameter Tampering* → *Exploit Hidden Fields*.
- Testar a funcionalidade de *Shopping Cart* (Carrinho de Compras).
  - Editar quantidade, campo *Quantity*;
  - Atualizar o carrinho, botão *UpdateCart*;
  - Realizar a compra, botão *Purchase*.
- Qual o comportamento do valor total da compra? Será possível editá-lo?
- Abrir as ferramentas de desenvolvedor do seu navegador.
- Com a ferramenta de seleção (ponteiro), selecionar o FORM do carrinho de compras.
- Encontrar no FORM a campo *"Price"* do tipo HIDDEN.
- No código do FORM para o campo *"Price"*, editar o valor 2999.99 para 0.99.
- Na aplicação, ativar o botão *UpdateCart*.
- O que aconteceu? Qual o valor final da compra?
- Isto é uma vulnerabilidade de código ou uma falha de projeto?

5.3.4.2. Testes de segurança com *Burp Community*

Esta subseção mostra o funcionamento da ferramenta de testes de intrusão *Burp Suite Community* (`https://portswigger.net/burp`) para auxiliar a descoberta e a exploração de vulnerabilidades presentes nas aplicações avaliadas, em particular, o bWAPP. O *Burb Suite Community Edition* pode ser baixado de `https://portswigger.net/burp/communitydownload`. Baixar a versão apropriada para o sistema operacional e que seja a mais recente e estável. Seguir o processo de instalação conforme instruções do programa. O procedimento a seguir realizar as configurações iniciais do Burp:

- Ativar a ferramenta Burp como desejar (ícone ou comando de linha).
- *Burp Community* só aceita projetos temporários, na janela inicial, clicar no botão *Next*.
- Usar as opções *default*. Clicar no botão *Start Burp*.
- Observar que nada acontece na aba (ou guia) *Target*.
- Ir para a aba *Proxy*.
- Burp possui um *browser* embarcado e configurado como *proxy*, botão *Open browser*.
- Na guia Proxy → Intercept, desativar o controle *Intercept is on* (de on para off)
- Na janela do *browser*, digitar a URL ou IP do bWAPP (que deve estar ativado).
  - Selecionar a aplicação bWAPP, logar na aplicação com usuário/senha `bee/bug`.
- No Burp, guia *Target* → *site map*, abrir a pasta bWAPP e a engrenagem (login.php).
  - Observar que as informações de login e senha estão expostas!
  - Clicar com botão direito em bWAPP e selecionar a opção *Add to scope* e apertar o botão *Yes*.
- Clicar na barra *Filter* e selecionar a opção *Show only in-scope items*.

As configurações do Burp foram realizadas com sucesso. Como o ZAP, o Burp também possui funcionalidades que auxiliam os testes manuais: opções de *proxy*, *Repeater* e *Intruder*. A opções de *proxy* do Burp podem ser configuradas do seguinte modo:

- Acessar a aba Proxy → Options.
  - Na seção *Intercept Client Requests*, desativar a regra *File extension Does not match*.
  - Na seção *Intercept Server Responses*, ativar as regras de interceptação e ativar a regra *Content type header Matches text*.
  - Na guia *Proxy* → *Intercept*, ativar o controle *Intercept is on* (de *off* para *on*).
- Testar o comportamento destas modificações da interceptação da bWAPP.
- Em *Proxy* → *Options*, na seção *Response Modification*, ativar a opção *Unhide hidden form fields* e a opção *highlight unhidden fields*.
- Na guia *Proxy* → *Intercept*, desativar o controle *Intercept is on* (de *on* para *off*).
- Testar o comportamento destas modificações da interceptação da bWAPP.
  - Selecionar o *bug Insecure DOR (Change Secret)* e clicar no botão *Hack*.
  - Um campo *hidden* aparecerá na janela do *browser*.

Repeater é uma ferramenta muito útil em testes manuais. Ele permite a manipulação (alteração) de parâmetros das requisições individuais para a repetição de uma requisição com parâmetros personalizados pelo testador. Para usar o *Repeater*:

- Na aba *Proxy* → HTTP *History*, achar a requisição de login da aplicação bWAPP.
- Clicar na requisição com o botão direito do mouse e selecionar a opção *Send to Repeater*.
- O rótulo da aba *Repeater* vai mudar de cor para vermelho, indicando que foi ativada.
- A aba *Repeater* mostra a requisição que será repetida.
- Ativar o botão *Send* para enviar a requisição como está.
- Visualizar a resposta obtida (o login foi bem-sucedido).
- Editar o campo `password=bug` para `password=bag` e ativar o botão *Send*.
- Visualizar a resposta, o botão *Render* para ver em HTML (o login foi mal-sucedido).

Intruder é uma ferramenta muito útil em testes de força bruta. Ele permite a manipulação (alteração) de parâmetros das requisições e o envio de diversas requisições personalizadas em massa. Para usar o *Intruder*:

- Na aba *Proxy* → HTTP *History*, achar a requisição de login da aplicação bWAPP.
- Clicar na requisição com o botão direito do mouse e selecionar a opção *Send to Intruder*.
- O rótulo da aba *Intruder* vai mudar de cor para vermelho, indicando que foi ativado.
- Na aba *Intruder* → *Positions*, manter *attack type* em *Sniper*.
  - Observar todos os campos que podem ser manipulados. Clicar no botão *Clear §*.
  - Selecionar o parâmetro *bug*, clicar no botão *ADD §*, o parâmetro fica assim: *§bug§*.
- Na aba *Intruder* → *Payloads*, manter o *Payload type* em *Simple list*.
  - Na caixa de *Payload options*, adicionar (ADD) os itens da lista manualmente.
  - Digitar cada item da lista bag, beg, big, bog, bug seguido de enter.
  - `bag <ENTER>beg<ENTER>big<ENTER>bog<ENTER>bug`.
  - Clicar no botão *Start Attack* no canto superior direito da janela do Burp.
  - O progresso do ataque é mostrado em outra janela.

Os procedimentos explicados até aqui são usados em exemplos de exploração de vulnerabilidades. O procedimento a seguir mostra a exploração de HTML *Injection* no bWAPP com Burp. Ainda na tela do *bWAPP* em *proxy* pelo *Burp*.

- Selecionar o bug *HTML Injection - Reflected (POST)*.

- Preencher os campos de nomes e sobrenome e clicar em *Go!*,
  - p.ex., preencher com "Tony"e "Stark".
  - O conteúdo fornecido pelo usuário é mostrado no navegador.
- No *Burp*, *Proxy* → *Intercept*, ativar a Interceptação (*Intercept is on*).
- No *browser*, preencher novamente com "Tony"e "Stark". Clicar em *Go*.
- No *Burp*, verificar que a requisição interceptada.
  - Observar os parâmetros `firstname=Tony&lastname=Stark&form=submit`.
  - Substituir a linha de parâmetros por `firstname=<h1><a href="http://www.google.com">Clique</a></h1> &lastname=<h2>Hahah!</h2>&form=submit`.
  - Clicar no botão *Forward* enquanto houver requisições para tratar.
- No *Browser*, observa-se que o HTML foi inserido na página e apresentado ao usuário.

Outra injeção de HTML via *GET* no bWAPP:

- Selecionar o bug *HTML Injection - Reflected (GET)*.
- Preencher os campos de nomes e sobrenome e clicar em *Go!*,
  - p.ex., preencher com "Tony"e "Stark".
  - O conteúdo fornecido pelo usuário é mostrado no navegador e na URL também!
- No *Burp*, *Proxy* → *Intercept*, ativar a Interceptação (*Intercept is on*).
- No *browser*, preencher novamente com "Tony"e "Stark". Clicar em *Go*.
- No *Burp*, verificar que a requisição interceptada
  - Observar os parâmetros `firstname=Tony&lastname=Stark&form=submit`.
  - Substituir a linha de parâmetros por `firstname=<h1><a href="http://www.google.com">Clique</a></h1> &lastname=<h2>Hahah!</h2>&form=submit`.
  - Clicar no botão *Forward* enquanto houver requisições para tratar.
- No *Browser*, observa-se que o HTML foi inserido na página e apresentado ao usuário.

Mais uma injeção de HTML via *GET* no bWAPP:

- Selecionar o bug *HTML Injection - Reflected (URL)*.
  - Não esquecer de clicar em *Hack!*
  - A página mostra a URL atual (*Current URL*)

- No *Burp*, *Proxy* → *Intercept*, ativar a Interceptação (*Intercept is on*).
- No *browser*, recarregar a página (F5 ou botão de recarregar no navegador).
- No *Burp*, verificar que a requisição interceptada
  - Substituir o endereço IP do *header Host* por algum outro endereço.
    - P.ex., *www.google.com*
  - Clicar no botão *Forward* enquanto houver requisições para tratar.
- No *Browser*, observa-se que a URL atual (*Current URL*) foi adulterada.

O próximo exemplo explora um XSS JSON no *bWAPP* em *proxy* pelo *Burp*.

- Selecionar o bug *XSS - Reflected (JSON)*.
- Testar a busca com qualquer nome de filme, p.ex., *"Matrix"*.
  - Observar que o conteúdo do campo de busca volta para o usuário;
    - *"Matrix"*?
- No *Burp*, em *Proxy* → *HTTP History*, achar a última requisição com a busca *"Matrix"*.
  - Observar o conteúdo da resposta (*Response*). Onde está a palavra *"Matrix"*?
    - Dica: Usar a função de busca na parte inferior da baixa da *Response*.
  - A palavra buscada (*"Matrix"*) está inserida entre *tags* `<script> ... </script>`
    - O JSON de resposta é construído no trecho.
    - Um *Eval* vulnerável também está exposto.
- O ataque consiste em customizar um JSON de resposta com um *script* embutido.
  - Finalizar o JSON, fechar o *script* e inserir o *script* malicioso.
  - Observar de onde o JSON é finalizado e copiar os caracteres finais
    - Dica: a linha do JSON termina com `"}]}';`
  - Fechar o *script* do JSON e inserir o *script* malicioso
    - `"}]}'; </script><script>alert(1);</script>`
  - Outro exemplo de *script* malicioso:
    - `"}]}';</script><script>alert(document.cookie);</script>`

Ainda no *bWAPP* em *proxy* pelo *Burp*, o próximo exemplo explora uma *SQL Injection*.

- No *bWAPP*
  - Selecionar o bug *SQL Injection (POST/Select)*.
  - Escolher um filme da lista de seleção e clicar em *Go*.

- No *Burp*, *Proxy* → *Proxy History*, selecionar a requisição mais recente.
  - Observar como o parâmetro de busca *movie* é usado.
  - Copiar a requisição para o *Repeater*.
- No *Burp Repeater*
  - Modificar *movie* da requisição para `movie=1 or 1=1\#&action=go`.
  - Clicar em *Send*. O que acontece? Dica: o comportamento não foi modificado.
  - Modificar *movie* da requisição para *movie=200 union select 1,2,3,4,5,6,7#*
  - Clicar em *Send*. O que aconteceu dessa vez?

5.4. Falhas de projeto de software e segurança de APIs

Esta seção relaciona vulnerabilidades de API às falhas de projeto de software, discutindo as falhas de projeto de software mais comuns (*Top 10 Software Design Flaws*) e seu impacto na segurança das APIs. A seção contempla os testes de segurança de APIs REST auxiliados por ferramentas DAST (p.ex., OWASP ZAP) e SAST (p.ex., SonarQube, HorusSec). A seção também aborda duas classificações bastante conhecidas de ameaças e ataques mais comuns contra APIs: *OWASP Top 10 API Risks* (edição de 2023) e maus usos de criptografia relacionados a *JSON Web tokens* (JWT).

5.4.1. Falhas de projeto de software

De acordo com [iee 2014], existem 10 falhas de design de software mais comuns que devem ser evitadas quando do projeto de uma arquitetura. As próximas seções detalham cada uma delas.

5.4.1.1. Ganhar ou criar confiança, nunca supor que ela já existe

Em uma aplicação, cliente-servidor (*frontend - backend*), implementar no cliente as funções de segurança do servidor expõe estas funções ao ambiente menos confiável do cliente ou *frontend*. Designs que colocam essas funções de segurança no cliente são inerentemente fracos e inseguros. Por exemplo, dados confidenciais de autorização, controle de acesso, verificação de regras de segurança e lógica de negócios ficam vulneráveis. Existem vários exemplos de clientes pouco confiáveis, tais como: navegadores de internet, clientes ricos (*Thick/Fat/Rich clients*), dispositivos móveis, software embarcado e chamadas de APIs por parceiros externos. Dados enviados por clientes não confiáveis são considerados comprometidos até prova do contrário e, por isso, devem ser validados antes de usados.

Essa falha de design geralmente acontece quando projetistas fazem o seguinte (incorretamente): assumem que APIs do servidor sempre serão chamadas na mesma ordem, acreditam que a interface do usuário é sempre capaz de restringir o que o usuário envia para o servidor, tentam construir a lógica de negócios ou validações exclusivamente no lado do cliente ou tentam armazenar segredos no cliente, mesmo usando criptografia forte.

5.4.1.2. O mecanismo de autenticação não pode ser desviado nem adulterado

Autenticação é o ato de validar a identidade de algo ou alguém. Um software seguro deve impedir que um usuário não autenticado tenha acesso aos sistemas e serviços. Depois que o usuário é autenticado, o sistema seguro também deve impedir que o usuário altere sua identidade sem nova autenticação. Em geral, um sistema deve considerar a força da autenticação que um usuário forneceu antes de agir, a qual depende do fator de autenticação usado: senha, *token*, biometria, ou uma combinação de dois deles. Por exemplo, autenticação por *token* de sessão mantido em *cookie* poder ser suficiente em alguns casos, mas não em todas as ocasiões.

Um sistema que possui um mecanismo de autenticação é vulnerável ao desvio de autenticação quando permite que um usuário acesse o serviço / API diretamente por um *token* ou URL "escondida", sem exigir uma credencial de autenticação obtida por meio do processo de autenticação normal. Em geral, é preferível ter um único método, componente ou sistema responsável pela autenticação de usuários porque isto evita inconsistências entre réplicas das bases de usuários. Uma vez que o mecanismo de autenticação tenha sido escolhido, deve ser utilizado em todas ações de autenticações. Por outro lado, tal mecanismo único pode funcionar como um gargalo lógico que não pode ser contornado, sendo um ponto de falha único e, por isso, sujeito a ataques de negação de serviço.

5.4.1.3. Autorizar somente após autenticar (Autorização != Autenticação)

A autorização sempre deve ser feita por verificação explícita de uma permissão, mesmo depois que uma autenticação acabou de ser realizada. A autorização depende dos privilégios do usuário autenticado e também do contexto da solicitação (por exemplo, horário, lugar, equipamento, etc.). Há casos em que a autorização (de um usuário para um sistema ou serviço) precisa ser revogada. Se a infraestrutura de autorização não permitir a revogação, o sistema é vulnerável aos abusos de usuários com autorizações desatualizadas. Para operações confidenciais, sensíveis ou críticas, a autorização pode exigir ainda a reautenticação do usuário.

Autenticação não é binária, porque os usuários podem ser obrigados a apresentar (novas) evidências (mais fortes) de sua identidade. Além disso, a autenticação geralmente não é contínua porque acontece a intervalos periódicos. Por exemplo, entre os momentos de (re)autenticação, um usuário pode estar autenticado, mas se afastar do dispositivo ou entregá-lo para outra pessoa. A autorização de uma operação sensível pode exigir reautenticação ou autenticação mais forte e políticas de segurança podem exigir duas ou mais autorizações em ações críticas (obedecendo ao princípio de separação de responsabilidades). Assim como ocorre com a autenticação, uma infraestrutura comum (um único mecanismo) deve ser usada para as verificações de autorização, a fim de evitar inconsistências, mas com a desvantagem do ponto único de falha.

5.4.1.4. Separar controle e dados, não executar comandos não confiáveis

Misturar dados e instruções de controle em uma única estrutura (por exemplo, strings em requisições HTTP), pode levar a vulnerabilidades de injeção. Falta de separação estrita entre dados e comandos/controles resulta em dados não confiáveis. O atacante manipula os dados para controlar o fluxo de execução de uma aplicação. Em software escrito em linguagens de mais baixo

nível, como C, C++, *assembly*, ou software embarcado, a mistura entre dados e controle pode causar corrupção de memória e pode levar a ataques de negação de serviço, vazamentos de dados e execução de código malicioso. Em linguagens de alto nível, a mistura de controle e dados pode ocorrer na interpretação de comandos em tempo de execução das linguagens de programação.

Software que ignora o princípio da separação estrita entre dados e código, sem separação explícita entre dados e instruções, são inerentemente inseguros. Esta falha de projeto geralmente ocorre quando software monta strings concatenando dados não confiáveis e instruções de controle confiáveis. Vulnerabilidades de injeção surgem quando os dados não confiáveis não são validados nem jogados fora. APIs propensas a injeção de comandos possuem risco alto de exploração da vulnerabilidade de injeção. Exemplos de tais vulnerabilidades incluem injeção de SQL, injeção de JavaScript (em ataques XSS), injeção de XML e injeção de comando do sistema operacional.

5.4.1.5. Garantir que todos os dados são validados explicitamente

É importante garantir que todos os dados são validados explicitamente. Uma boa pática, neste caso é usar validadores centralizados para validar todos os dados de entrada do mesmo jeito. O validador é um filtro ou interceptador na arquitetura da aplicação. Ele converte dados externos para um formato interno, antes de validações sintáticas e semânticas.

Protocolos de comunicação devem validar todos os campos de todas as mensagens recebidas (antes de processá-las). Arquivos de dados em formatos complexos (XML, imagens, textos ricos, etc.) devem passar por validações sintáticas (de formato e estrutura) e semânticas (da lógica de negócios). Na validação sintática devem ser usadas bibliotecas de validação que reconhecem formatos estruturados, como, por exemplo, e-mail, URLs, CPFs. Além disto, tipos de dados estruturados devem ser usados para capturar suposições sobre validade de dados. Por exemplo, *string* que representa data/hora deve ser validada se contém uma data/hora bem formada (p.ex., `DD/MM/YYYY`). Já na validação semântica, a validação de entrada geralmente depende do estado da aplicação (da lógica de negócios) e consiste em reavaliar suposições, premissas e pré-condições sobre os dados, nos trechos de código próximos ao uso dos dados.

Nas APIs, os *endpoints* (as funções da API) que consomem strings com controle e dados ao mesmo tempo devem ser evitados. Neste casos, é preferível expor *endpoints* que consumam tipos estruturados (com segregação estrita entre dados e comandos/controle), como por exemplo, os dados com tipos fortes: inteiros, booleanos e alfabéticos. Nos aplicativos, APIs que misturam dados e *flags* de controle em parâmetros de strings também devem ser evitados. Se um aplicativo deve usar uma API legada (propensa a injeção), esta API deve ser acessada por meio de uma API interna (um *proxy* ou *wrapper* de API) com segregação de dados e controle. Nos aplicativos que transformam dados em códigos, os dados devem ser validados com rigor ou cuidado maior e as ações executadas com os comandos gerados devem ser limitadas e âmbito restrito.

Comandos como `eval` ou `exec` são comuns nas linguagens interpretadas, como Java, Python, Ruby e JavaScript. Estes comandos consomem uma cadeia de caracteres (*string*) e invocam o interpretador da linguagem ou executam um comando do sistema operacional. Se o uso dos comandos `exec` ou `eval` são inevitáveis, uma API intermediária (*wrapper* de API) deve ser colocada entre o código da aplicação e as interfaces de uso do `eval` ou `exec`, enquanto só a API mais segura é exposta para a aplicação. Em programação orientada a objetos,

reflexão computacional é um recurso poderoso de introspecção de programas e alteração do código ou do comportamento de uma aplicação em tempo de execução. O uso de reflexão computacional é escolha de design arriscada porque defeitos podem levar a execução de código malicioso, como por exemplo, inclusão ou modificação de métodos/funções em um objeto.

5.4.1.6. Usar criptografia corretamente

É um fato conhecido que a maioria das vulnerabilidades de criptografia está no uso incorreto de bibliotecas criptográficas e não em vulnerabilidades do código criptográfico em si [Braga and Dahab 2016, Braga and Dahab 2017]. Os maus usos ou usos incorretos de criptografia por desenvolvedores de software estão fora do escopo deste texto e já foram tratados detalhadamente em minicursos anteriores [Braga and Dahab 2015b, Braga and Dahab 2018, Braga and Dahab 2019].

Uma vez que implementações criptográficas de qualidade e boa reputação estão disponíveis para todos, a fonte mais comum de problemas com criptografia é o software em torno da criptografia, escrito por desenvolvedores não especialistas em criptografia nem em segurança. Os falhas de projeto associadas ao uso incorreto de criptografia são as seguintes: não prever adaptação ou evolução da criptografia, usar ou inventar sua própria criptografia, usar incorretamente bibliotecas ou APIs criptográficas, gestão (incluindo geração, distribuição, armazenamento) inadequada de chaves criptográficas, gerar aleatoriedade insuficiente ou falsa e usar diversas implementações criptográficas diferentes e descentralizadas.

5.4.1.7. Identificar os dados sensíveis e como eles devem ser mantidos

Só é possível proteger o que se conhece, por isso é importante identificar os dados sensíveis para saber como protegê-los melhor. Projetistas devem identificar dados confidenciais ou privados das suas aplicações e determinar como protegê-los. Neste casos, uma avaliação com base na Lei Geral de Proteção de Dados (LGPD) pode ser utilizada. O software deve proteger os dados sensíveis nas diversas situações, sejam dados em repouso (armazenados), em trânsito e em uso. A sensibilidade dos dados e as proteções decorrentes dela podem mudar com o tempo porque negócios evoluem, regulamentações mudam, sistemas são interconectados e novas fontes de dados são incorporadas à aplicação.

A sensibilidade dos dados tratados pela aplicação depende do contexto de uso específico (legislação, política, contratos, vontade do usuário, etc.). Uma política de classificação em níveis orienta o manuseio seguro dos dados. Um exemplos bastante comum de níveis de classificação é a escala de acesso a informação em público, restrito, privado, secreto e ultra secreto. As aplicações em software geralmente tratam muitos tipos de dados sensíveis, que podem ser fornecidos pelo usuário, derivados pela aplicação, coletados de sensores, material criptográfico, informações bancárias, de compras, de cartões e informações privadas (p. ex., PII), entre outros.

5.4.1.8. Sempre considerar a criatividade do usuário final

Muitos usuários são capaz de vislumbrar jeitos diferentes de usar a aplicação. A segurança da aplicação depende do que os usuários fazem com ela, por isso é importante sempre considerar como os usuários usam a aplicação. Os usuários legítimos de um software variam desde o pessoal da implantação, operação, configuração e manutenção até os usuários finais. Há tanto usuários legítimos sem interesse em segurança quanto usuários legítimos e mal-intencionados. Além do usuário final que usa a aplicação por meio de uma interface, os programadores que usam APIs também podem ser atacantes.

Usabilidade e experiência do usuário são geralmente importantes para a operação segura do software porque o usuário não se sente compelido a burlar controles de segurança difíceis de usar. De modo geral, o software deve ser configurado e usado com segurança por meio de interfaces intuitivas e fáceis de usar e controles de segurança expressivos e sem excesso. Quando a segurança é muito difícil de configurar para uma grande população de usuários, ela nunca será configurada para todos ou não será configurada corretamente, permanecendo incompleta.

Ataques de escalada de privilégios podem resultar de falhas ou falta de autorização explícita. Uma falha comum ocorre quando a autorização não está vinculada ao usuário autenticado. Isto é, o software supõe erroneamente que o usuário não tem acesso ou tem acesso, sem uma verificação explícita. A falta de compartimentação e isolamento entre usuários é outro problema comum em que um usuário acessa dados de outro usuário. Além disso, configurações "abertas"favorecem navegação “livre” do usuário legítimo e mal intencionado.

Muitas vezes existem decisões de projeto que exigem um compromisso ou equilíbrio entre segurança e usabilidade. “Danos colaterais” a usabilidade podem ocorrer na implementação de segurança na interface do usuário e vice-versa. Por exemplo, dificultar o acesso para evitar vazamentos de dados em interfaces acessíveis, ou facilitar o *shoulder surfing* em aplicativos móveis usados pelos passageiros do transporte público.

Muitas vezes, vulnerabilidades aparecem de situações raras e exceções. Por exemplo, ao não considerar o apagamento e a revogação do usuário ao final do seu ciclo de vida, por exemplo em uma rescisão contratual ou troca de função. Outro exemplo seria não considerar requisitos de segurança específicos para classes diferentes de usuários com necessidades específicas de usabilidade (tais como PCDs, proficiência em idiomas, crianças, idosos, pessoas com diferentes capacidades cognitivas) em softwares utilizados pelo público em geral.

5.4.1.9. Os componentes externos mudam a superfície de ataque

Atualmente, o uso de *frameworks* de software, bibliotecas externas ou outros componentes de terceiros é prática comum no desenvolvimento de software. Porém, a integração ou dependência de componentes externos inseguros aumenta a superfície de ataque. Uma boa prática é incluir no processo de desenvolvimento seguro ou pipeline *DevSecOps* a avaliação de segurança do componente externo novo. Há diversos exemplos de possíveis problemas de segurança com componentes de terceiros: carregar uma biblioteca com vulnerabilidades conhecidas (CWE, CVE, etc.), incluir uma biblioteca com recursos extras que envolvem riscos de segurança, reutilizar uma biblioteca que não atenda aos padrões atuais de segurança, utilizar serviço de terceiros e passar a responsabilidade da segurança para ele, errar na configuração da segurança

de uma biblioteca (adotar padrões seguros), incluir biblioteca que envia requisições para o site do criador ou algum parceiro, incluir biblioteca que recebe solicitações de alguma fonte externa, usar componente externo com muitos níveis de inclusão ou dependência "recursiva" e, por último, incluir componentes com interfaces desconhecidas, tais como, por exemplo, comandos de linha, interface web, autenticação própria, interface de depuração, modo desenvolvedor, *backdoor* de acesso extraordinário).

5.4.1.10. Prever alterações futuras em componentes de segurança

Assim como todo software, os componentes de segurança também sofrem mudanças corretivas e evolutivas. A arquitetura de segurança deve ser flexível ao considerar alterações futuras para evolução dos componentes de segurança. As aplicações e os componentes de segurança devem ser projetados com uma visão das mudanças futuras, tais como: atualizações seguras de partes ou do todo, propriedades de segurança que mudam com o tempo, código é atualizado para versões novas, isolamento total ou parcial de funcionalidades comprometidas em ataques, alterações em objetos mantidos em segredo, alterações nas propriedades de segurança de componentes fora de controle (externos) e alterações nos tipos de permissões e de usuários.

5.4.2. Testes de segurança e vulnerabilidades comuns em APIs REST

Vulnerabilidades de APIs [top 2023] são vulnerabilidades de projeto, semânticas ou da lógicas da aplicação e seguem um racional simples [tes 2020]: em geral, a lógica das aplicações é defeituosa de alguma forma e as falhas lógicas são bastante variadas. As falhas lógicas vão desde defeitos simples em trechos de código até vulnerabilidades complexas na interoperação de componentes. Às vezes, elas são óbvias e fáceis de detectar; outras vezes, são muito sutis e não são percebidas em revisões de código ou testes de intrusão. As falhas lógicas dificilmente são eliminadas por meio de programação segura simples, nem detectadas por análise estática de código ou testes de intrusão comuns.

Detectar e prevenir falhas lógicas muitas vezes requer pensamento lateral (fora da caixa). Por outro lado, há recomendações simples para evitar as vulnerabilidades semânticas, em particular aquelas relacionadas a APIs inseguras [Stuttard and Pinto 2008]: validação (sintática e semântica) dos dados de entrada da lógica de negócios, proteção contra falsificação de transações, requisições, solicitações, pedidos, etc., autotestes (automáticos) de integridade do processamento, proteções contra variações de tempo e canais laterais de tempo, imposição de limites ao número de vezes que funções críticas são chamadas, controle do "passo a passo" dos fluxos de trabalho (*workflows*), controles internos contra "maus usos" das aplicações e APIs, controles contra *upload* de arquivos de tipos errados, corrompidos ou maliciosos, projetar aplicações com as boas práticas de projeto seguro.

As vulnerabilidades semânticas em APIs estão relacionadas ao mau uso ou ao abuso de funcionalidades legítimas das aplicações e podem ser causadas por falhas de projeto e não por bugs/defeitos de codificação insegura. Ataques a estas vulnerabilidades geralmente envolvem fraudes na lógica de negócios da aplicação e e podem ser automatizados para abuso em larga escala [Watson and Zaw 2018]. As próximas subseções mostram como explorar algumas das vulnerabilidades de API mais comuns [top 2023] encontradas no OWASP Juice Shop.

5.4.2.1. Descobrindo a API do Juice Shop

O procedimento a seguir descreve como descobrir a API de uma aplicação a partir da análise do código fonte do *frontend* web com o auxílio das ferramentas do desenvolvedor.

1. Abrir a aplicação Juice Shop no navegador com `http://localhost:3000`.
2. Uma vez na tela/página inicial da aplicação Juice Shop, acessar o código fonte do *frontend* da aplicação a partir das ferramentas do desenvolvedor → Sources.
3. Visualizar o arquivo `main.js` (ou algo parecido) e usar o botão *Pretty printing* ou botão **{}** para deixar o código fonte legível.
4. Buscar (menu três pontos verticais → buscar) pela palavra *"path"*.
5. A API da aplicação pode ser facilmente identificada no resultado da busca.
6. Outras buscas possíveis são "API"e "REST". Diversos *endpoints* de API são facilmente identificáveis nesta busca.
7. Identificar a API do *dashboard* de desafios e acessá-la. *Spoiler alert!* O *dashboard* está em `http://localhost:3000/#/score-board`.

5.4.2.2. Quebra de autorização em nível de objeto

De acordo com OWASP API01:2023 (*Broken Object Level Authorization*) [top 2023], APIs tendem a expor *endpoints* para manipulação de objetos internos, criando uma ampla camada de ataque ao controle de nível de acesso. Para evitar problemas, o controle de autorização deve ser verificado em toda função que tenha acesso à fontes de dados que utilizem dados enviados pelo usuário. O procedimento a seguir descreve como um usuário autenticado consegue visualizar sem autorização a cesta de compras de um outro usuário. O objetivo o ataque é é visualizar a cesta de compras de um usuário, qualquer um, sem estar logado como ele.

1. Criar dois usuários com login/senha, `user1@juice/user1` e `2@juice/user2`.
2. Encher a cesta do `user2` com diversos produtos do JuiceShop.
3. Descobrir e explorar a API da cesta de compras.
4. Exploração 1 (exploração do armazenamento local):
   (a) abrir as ferramentas de desenvolvedor.
   (b) Entrar o App JuiceShop logado como `user1`.
   (c) acessar a cesta de compras do `user1`.
   (d) em *DevTools* → *Network*, observar que a API REST da cesta é simples. Por exemplo, `http://localhost:3000/rest/basket/7`. O que é o número 7?
   (e) em *DevTools*, *Application* → *Session Storage* aparecem dois elementos `bid = 7` e `ItemTotal = 0`. será o mesmo número 7?

(f) Conclusão deduzida: bid quer dizer *BasketID*.

(g) Alterar bid para um valor qualquer e atualizar a página. Por exemplo, `bid = 6`!

(h) O que acontece? *Spoiler Alert*! Aparece a cesta de compras do `user2`.

5. Exploração 2 (interceptação da requisição):

   (a) no *OWASP Zap Proxy* do Juice Shop, interceptar e reenviar a requisição do `basket`.

   - Botão direito na requisição do histórico;
   - Na tela de reenvio, substituir o valor de `bid` na requisição por outro valor qualquer até achar um valor válido.

   (b) Esta exploração pode ser automatizada para baixar as cestas de compras de todos os usuários.

   (c) Como poderia ser automatizado e por que este ataque acontece?

5.4.2.3. Autenticação de usuário quebrada

A acordo com OWASP API02:2023 (*Broken User Authentication*) [top 2023], mecanismos de autorização não raramente são implementados incorretamente, permitindo que atacantes comprometam *tokens* de autenticação ou explorem falhas na implementação para assumir identidades temporária ou permanentemente. Isto compromete a habilidade dos sistemas em identificar o usuário/cliente comprometendo a segurança da API. O procedimento a seguir descreve quatro jeitos diferentes de violar o mecanismo de autenticação da aplicação Juice Shop visando obter acesso às credenciais do usuário e logar na aplicação de maneira indevida.

- Exploração 1 (Vazamento das credenciais dos usuários pela API):

  1. Logar na aplicação como `user1@juice`.
  2. no OWASP *ZAP Proxy* do Juice Shop, interceptar e reenviar a requisição de *products/search*.
  3. `GET http://localhost:3000/rest/products/search?q=`.
     - Botão direito na requisição do histórico.
     - Na tela de reenvio, substituir *q=* por *q='))–*. Por que não *q='–* ?
  4. *Spoiler alert*! A requisição *products/search* sofre de SQLi!
  5. Na tela de reenvio, substituir `q=` pelo seguinte *')) UNION SELECT id, email, password, '4', '5', '6', '7', '8', '9' FROM Users–*.
  6. Reenviar a requisição injetada!
  7. Pergunta 1: Como sei que o nome da tabela é *Users*? Deduzindo que na API REST, serviços são iguais a tabelas tabelas.
  8. Pergunta 2: Como sei a quantidade de campos? Por tentativa e erro.

De posse do *dump* da base de dados do usuário, o atacante pode fazer o seguinte descobrir a senha dos usuários.

- Na resposta da requisição, procurar pelo `user1@juice`.
- O *hash* da senha desse usuário está no campo *description*.
- Por exemplo, em `24c9e15e52afc47c225b757e7bee1f9d`.
- No website `https://crackstation.net/`:
  - Digitar o *hash* e apertar o botão *CrackHashes*.
  - Deu um *match* perfeito com md5 e user1 e assim descobre-se o algoritmo de *hash* usado na criptografia da senha!
- Repetindo o processo para o usuário admin:
  - O *hash* é `0192023a7bbd73250516f069df18b500`.
  - O *website* `https://crackstation.net/` diz que a senha é `admin123`.
- Tarefa: descobrir a senha dos outros administradores!

As próximas explorações são mais diretas sobre a interface de login e a API de autenticação. Elas exploram uma vulnerabilidade de injeção de SQL sobre a API ou interface de login.

- Exploração 2 (SQLi sem conhecer usuário algum)
  1. Logar na aplicação com *login* `' or 1=1-` e qualquer senha.
  2. A aplicação vai logar como a primeira entrada na tabela *Users*.
  3. Por que é o Admin?
- Exploração 3 (SQLi com e-mail de administrador).
  1. Logar na aplicação com *login* `admin@juice-sh.op'-` e qualquer senha.
  2. 2a. parte da consulta SQL é comentada e a senha não é verificada.

A próxima exploração faz um ataque de dicionário sobre a senha fraca do administrador.

- Exploração 4 (Ataque de dicionário de senhas fracas)
  1. Logar na aplicação com *login* `admin@juice-sh.op` e senha `admin123`.
  2. Senha *default* descoberta por tentativa e erro ou busca exaustiva com dicionário de senhas fracas.
  3. Faze a busca exaustiva.

5.4.2.4. Autorização quebrada em nível de propriedade de objeto

De acordo com OWASP API3:2023 (*Broken Object Property Level Authorization*) [top 2023], ao permitir que um usuário acesse um objeto por meio de um *endpoint* de API, é importante validar se o usuário tem acesso às propriedades específicas do objeto que estão sendo acessadas. Um *endpoint* de API está vulnerável se: o *endpoint* de API expõe propriedades de um objeto que são consideradas sensíveis e não devem ser lidas pelo usuário (anteriormente denominado: "Exposição Excessiva de Dados") ou o *endpoint* de API permite que um usuário altere, adicione ou exclua o valor de uma propriedade sensível do objeto, à qual o usuário não deveria ter acesso (anteriormente denominado: "Atribuição em Massa"). O procedimento a seguir descreve como usar uma API para acessar mais informação do que é mostrada pela interface gráfica da aplicação.

- Abrir a aplicação Juice Shop com OWASP Zap Proxy:
  - Logar como administrador (usar qualquer dos métodos anteriores).
  - Acessar `http://localhost:3000/#/administration`, aparece uma lista de usuários.
  - clicar em qualquer um dos usuários da lista (ícone de olho).
- No OWASP ZAP:
  - Verificar no histórico de requisições que a API Users foi ativada.
  - Por exemplo, `http://localhost:3000/api/Users/2`, o que significa o 2?
  - Reenviar esta requisição com outros parâmetros.
  - Por exemplo, 3, 5, 8 devolve o json completo do usuário correspondente.
  - Sem parâmetros, devolve o json de todos os usuários.

5.4.2.5. Consumo irrestrito de recursos

De acordo com OWASP API4:2023 (*Unrestricted Resource Consumption*) [top 2023], atender a solicitações da API requer recursos como largura de banda de rede, CPU, memória e armazenamento, podendo até ser pagos por solicitação, como no caso de envio de e-mails/SMS/ligações telefônicas, validação biométrica, por exemplo. Uma API está vulnerável ao uso irrestrito de recursos se pelo menos um dos seguintes limites estiver ausente ou configurado de forma inadequada (por exemplo, muito baixo/alto): tempos limite de execução, memória máxima alocável, número máximo de descritores de arquivo, número máximo de processos, tamanho máximo de arquivo para *upload*, número de operações a serem executadas em uma única solicitação de cliente da API (por exemplo, agrupamento de GraphQL), número de registros por página para retornar em uma única solicitação-resposta, limite de gastos de provedores de serviços de terceiros.

Por exemplo, no Juice Shop, a falta de limites para a quantidade de requisições de login torna possível um ataque de força bruta visando testar em massa muitas opções de senha para um usuário. Este ataque pode ser realizado com o auxílio da ferramenta de *fuzzing* (chamada de *fuzzer*) do OWASP ZAP. O passo a passo é o seguinte: gera uma requisição de login incorreto

para alimentar o *fuzzer*, ativa o *fuzzer* nessa requisição no histórico, adiciona um Fuzzer de Strings com as senhas de teste (para uma quantidade maior de senhas de teste, um *fuzzer* de arquivos pode ser usado) e iniciar o *fuzzer*. O *fuzzer* contorna o *frontend* da aplicação, que fica no mesmo ponto, enquanto o resultado do *brute force* é mostra na janela do *fuzzer*, incluindo qual teste foi bem-sucedido (acertou a senha do usuário).

5.4.2.6. Acesso irrestrito a fluxos de negócios sensíveis

De acordo com OWASP API6:2023 (*Unrestricted Access to Sensitive Business Flows*) [top 2023], ao criar um *endpoint* de API, é importante entender qual fluxo de negócio ele expõe. Alguns fluxos de negócio são mais sensíveis do que outros, no sentido de que um acesso excessivo a eles pode prejudicar o negócio. Exemplos comuns de fluxos de negócio sensíveis e o risco de acesso excessivo associado a eles: fluxo de compra de um produto, fluxo de criação de comentário/postagem, fluxo de fazer uma reserva, fluxo de criação de usuários. Um *endpoint* de API está vulnerável se expõe um fluxo de negócio sensível sem restringir adequadamente o acesso a ele. Por exemplo, no Juice Shop, é possível criar vários usuários e massa porque a API de criação de usuário tem acesso sem autenticação. Com uma ferramenta de desenvolvimento de APIs, como o Postman, é possível contornar o *frontend* e criar vários usuários.

5.4.2.7. Configurações de segurança mal feitas

De acordo com OWASP API8:2023 (*Security Misconfiguration* [top 2023], existem diversas situações que fazem uma API vulnerável: falta de fortalecimento de segurança adequado em qualquer parte da pilha da API; permissões configuradas de forma inadequada nos serviços de nuvem; sistemas estão desatualizados; recursos desnecessários estão habilitados (por exemplo, verbos HTTP, recursos de registro); falta de *Transport Layer Security* (TLS); diretrizes de segurança ou controle de cache não são enviadas aos clientes; política de *Cross-Origin Resource Sharing* (CORS) está ausente ou configurada de forma incorreta; mensagens de erro incluem rastreamentos de pilha ou expõem outras informações sensíveis.

Por exemplo, no Juice Shop, uma configuração insegura do servidor web pode favorecer um ataque de *path traversal*. Na página *About Us*, procurar um link para a URL `http://localhost:3000/ftp/legal.md`. A pasta ftp está aberta e pode ser acessa diretamente pelo caminho `http://localhost:3000/ftp`. Serviços gratuitos de varredura de vulnerabilidades HTTP/HTTPS podem ser usados para detectar falhas de configuração ou configurações inseguras em cabeçalhos HTTP/ HTTPS. Três dos mais conhecidos são o *Mozilla Observatory* (`https://observatory.mozilla.org`), o *Security Headers* (`https://securityheaders.com/`) e o *Qualys SSL Labs – SSL Server Test* (`https://www.ssllabs.com/ssltest/`).

5.4.2.8. Consumo inseguro de APIs

De acordo com OWASP API10:2023 (*Unsafe Consumption of APIs*) [top 2023], desenvolvedores tendem a confiar mais nos dados recebidos de APIs de terceiros ou de *frontends* inseguros do que na entrada do usuário e, portanto, tendem a adotar padrões de segurança mais fracos. Para

comprometer APIs, os atacantes focam em serviços ou códigos de terceiros integrados em vez de tentar comprometer diretamente a API alvo. Na versão, os problemas de injeção tinha sua própria categoria API08:2019 - *Injection*. O procedimento a seguir descreve como explorar uma vulnerabilidade *Cross-Site Scripting* (XSS) refletido no campo de busca da aplicação Juice Shop. É curioso observar que o ataque XSS até poderia ser realizado sobre a API, mas o resultado somente é visível na interface web da aplicação.

- Entrar na aplicação Juice Shop
- Logar na aplicação com usuário e senha (criar usuário, se necessário)
- Fazer uma busca por um nome de fruta (em inglês), O que acontece?
- O campo de busca exibe de volta para o usuário as palavras buscadas!
- O campo de busca aceita tags HTML?
- Testar com: `<h1> orange`.
- O campo de busca aceita javascript?
- Testar com: `<script>alert(1)<\scritp>`.
- A busca aceita DOM?
- Testar com: `<iframe src="javascript:alert('xss')">`.
- O resultado do XSS é visível na interface!!!

5.4.3. Segurança em *JSON Web Tokens*

JSON Web Token (JWT) é um padrão aberto (RFC 7519) que define uma forma compacta e autocontida de transmitir informações de forma segura entre as partes comunicantes como um objeto JSON. Essas informações podem ser verificadas e consideradas confiáveis porque são assinadas digitalmente. JWTs podem ser assinados usando um segredo (com o algoritmo HMAC) ou um par de chaves pública/privada usando RSA ou ECDSA.

Existem diversas maneiras incorretas de usar um JWT que levam a vulnerabilidades exploráveis em ataques reais. A primeira e mais comum é gerar um *token* JWT assinado e não verificá-lo ao recebê-lo. Por exemplo, ao gerar um *token* sem assinatura (`"alg":"none"`) e, por isso, sem capacidade de preservação de integridade nem de autenticidade.

Outra vulnerabilidade é a revelação de segredos (senhas e chaves) embutidos no JWT. Um *token* JWT comum não garante sigilo e não pode ser utilizado para transporte de segredos. Por exemplo, no caso de uso incorreto em que a chave secreta do HMAC é embutida no próprio *token* JWT. Também existe a possibilidade de confusão entre tecnologias criptográficas, quando o desenvolvedor, incorretamente, trata uma chave pública RSA ou ECDSA embutida no *token* como se ela fosse uma chave secreta HMAC e usada como *tag* de autenticação.

O Juice Shop possui um exemplo didático de *token* JWT inseguro. Ao aceitar o login de um usuário legítimo, a aplicação devolve várias informações úteis, entre elas o *token* de

autenticação da sessão do usuário. Esse *token* pode ser copiado (via acesso a resposta da requisição de login) e analisado no website `https://jwt.io`, em que podem ser observadas algumas das vulnerabilidades citadas nesta seção.

5.5. Proteção de aplicações web com WAF

A seção mostra o funcionamento do ModSecurity, um *firewall* de aplicações web (WAF de HTTP), e seu conjunto de regras padrão (*Core Rule Set* – CRS) para detectar, alertar e bloquear ataques baseados na exploração de vulnerabilidades conhecidas. A seção também mostra como um WAF pode falhar ao não ser capaz de detectar vulnerabilidades semânticas relacionadas à lógica de negócios das aplicações.

Existem duas maneiras de incluir um WAF na arquitetura de uma aplicação web. A primeira é embutindo o WAF no mesmo servidor web em que a aplicação está executando. Esta opção tem a vantagem de simplificar a arquitetura geral e a desvantagem de onerar o servidor web da aplicação. Por outro lado, o fortalecimento de segurança pode ser feito de modo específico em cada aplicação, como um *patch* virtual. A segunda opção é separar o WAF em outro servidor web diferente daqueles usados pelas aplicações, em uma arquitetura conhecida como *proxy* reverso. Além da função de segurança, este servidor web pode incluir funções de alta disponibilidade e balanceamento de carga. Neste caso, o fortalecimento de regras é geral a toda a arquitetura, mas ainda assim pode receber o nome de *patch* virtual.

Neste texto, o WAF está implantado em um *proxy* reverso, aquele que, em uma arquitetura cliente-servidor, fica próximo ao servidor e serve de intermediário na comunicação das aplicações web com os clientes (navegadores) na Internet. No exemplo de WAF que se segue, os seguintes elementos de arquitetura são identificáveis:

- Aplicação web: alguma das aplicações vulneráveis (Juice Shop, bWAPP, WebGoat).
- *Proxy* reverso: implantação do servidor web apache2 com o modproxy instalado.
- *Firewall* de aplicação: modsecurity com *Core Rule Set* (CRS) implantado no apache2.
- (Opcional) *Proxy* do navegador: OWASP ZAP ou o Burp *Community* em modo *proxy*.
- Cliente ou *frontend*: navegador web ou cliente de API, como o Postman.

5.5.1. Configuração do WAF ModSecurity no Servidor Web Apache

A listagem 5.15 mostra os comandos instalação do servidor web Apache e o ModSecurity neste servidor em uma máquina Ubuntu. Primeiro, ocorre a instalação do servidor web e o teste das funções básicas do servidor. Em seguida, habilitam-se os módulos de *proxy* do Apache. Finalmente, ocorre a instalação do ModSecurity para Apache. Como boa prática, o Apache é reiniciado depois de cada tarefa e, se a página padrão do Apache aparecer em `http://localhost` após cada tarefa, a implantação está funcionando corretamente. As configurações de segurança *default* do WAF (isto é, as configurações que determinam a rigidez com que as regras são aplicadas) não foram modificadas.

Listagem 5.15. Instalação do ModSecurity no Apache.

```
1 $ sudo apt-get update
```

```
2 $ apt-get install apache2
3 # Para testar, abrir o navegador e acessar http://localhost
4
5 # Para parar|inciar|reiniciar|recarrgar configs|habilitar|desabiliatr o servico
6 $ sudo systemctl stop | start | restart | reload | disable | enable apache2
7
8 # Instalacao e/ou habilitacao dos modulos proxy do apache
9 $ sudo a2enmod proxy proxy_http proxy_balancer lbmethod_byrequests
10
11 # Restart do Apache
12 $ sudo systemctl restart apache2
13
14 #Download e instalacao do modulo ModSecurity Apache
15 $ sudo apt install libapache2-mod-security2
16
17 # Restart do Apache
18 $ sudo systemctl restart apache2
19
20 #verificando se a versao instalada eh maior ou igual a 2.9
21 $apt-cache show libapache2-mod-security2
22
23 $ systemctl restart apache2
```

As configurações tanto do Apache quanto do ModSecurity prosseguem com a edição dos arquivos de configuração. Nas distribuições Ubuntu recentes, o Apache é instalado em `/etc/apache2` e o ModSecurity em `/etc/modsecurity`.

- No servidor web, as configurações são as seguintes:
  - No arquivo `/etc/apache2/envvars`, configurar o local dos logs para `/etc/apache2/log` na variável `APACHE_LOG_DIR=/etc/apache2/log`. Isto vai facilitar a inspeção dos arquivos de log.
  - No arquivo `/etc/apache2/mods-enable/security2.conf`, configurar a carga das configurações e das regras do WAF se o módulo mod_security estiver carregado (as regras estão em `/etc/modsecurity/rules`).
    - `IncludeOptional /etc/modsecurity/*.conf`
    - `Include /etc/modsecurity/rules/*.conf`
  - No arquivo `/etc/apache2/sites-enabled/000-default.conf`, configurar os *hosts* virtuais das aplicações como, por exemplo, ilustrado para o Juice Shop na listagem 5.16.
- No ModSecurity, as configurações são as seguintes:
  - Implantar o *Core Rule Set* (CRS) do OWASP (`https://coreruleset.org/`).
    - Baixar o arquivo de regras de `https://github.com/coreruleset/coreruleset/releases`. Aqui, usa-se a versão 3.2.1 do CRS para ModSecurity 2.9.3.
    - Copiar a pasta de regras (`rules`) para a pasta `/etc/modsecurity/rules`.
    - Copiar o arquivo `crs-setup.conf` para a pasta `/etc/modsecurity`.
    - Copiar o arquivo de configuração *default* ModSecurity para um novo `cp modsecurity.conf-recommended modsecurity.conf`.

- Fazer as seguintes configurações no arquivo `modsecurity.conf`:
  - `SecAuditLog /etc/apache2/log/modsec_audit.log`.
  - Função de segurança do WAF em detecção apenas `SecRuleEngine DetectionOnly` ou bloqueio `SecRuleEngine On`.

Listagem 5.16. Exemplo de arquivo *000-default.conf*.

```
1 ## Juice Shop
2 <VirtualHost *:80>
3     ProxyPreserveHost On
4     ProxyPass         / http://127.0.0.1:3000/
5     ProxyPassReverse / http://127.0.0.1:3000/
6 </VirtualHost>
```

Cada uma das aplicações vulneráveis, na nova configuração com WAF, foi novamente varrida pelo *scanner* DAST. Ao confrontar o *scanner* DAST contra o WAF sobre as aplicações vulneráveis, percebeu-se uma redução na quantidade total de alertas de vulnerabilidades das aplicações. Por outro lado, o *scanner* DAST alertou sobre possíveis vulnerabilidades no WAF ou no servidor web Apache usado como *proxy*, que podem ser falsos positivos. Apesar deste ser um resultado positivo, este não é o maior benefício do WAF. As próximas seções mostram que o maior benefício de um WAF está na proteção contra os ataques específicos de aplicações web, como por exemplo, aqueles do OWASP TOP 10 [top 2021].

5.5.2. Bloqueio de XSS, SQLi e RCE com ModSecurity

Todas as ações de detecção ou bloqueio realizadas pelo ModSecurity ficam registradas no arquivo `/etc/apache2/log/modsec_audit.log`. A inspeção deste arquivo é bastante útil para entender como funciona a detecção e bloqueio dos ataques. O ModSecurity é capaz de detectar e bloquear diversos ataques comuns que exploram vulnerabilidades simples as aplicações web, tais como os ataques de injeção de *script* (XSS), injeção de SQL (SQLi) e de execução remota de comandos do sistema operacional (RCE). A implantação padrão do ModSecurity usa as configurações de segurança mais leves, de rigor menor. Diversos testes de vulnerabilidades realizados ao longo deste texto podem ser facilmente detectados (ou bloqueados) pelo WAF com a configuração padrão, a saber (a lista a seguir não é exaustiva):

- Na aplicação Juice Shop, seguintes vulnerabilidades:
  - Injeção de SQL (`' or 1 = 1 #`) no campo de login.
  - XSS simples (`<script>alert(1)</script>`) no campo de busca.
- Na aplicação WebGoat, as seguintes lições:
  - *Injection Flaws* → *String SQL Injection*: `' or 'a'='a' -`.
  - *Injection Flaws* → *Numeric SQL Injection*: `101 or 1=1`.
  - *Cross-Site Scripting (XSS)* → *Reflected XSS Attacks*: `<script>alert(1)</script>`.
  - *Cross-Site Scripting (XSS)* → *Stored XSS Attacks*: `<script>alert(1)</script>`.

  - *Cross-Site Scripting (XSS) → Reflected XSS Attacks*: `<a onmouseover="alert(1)" href="#">read this!</a>`
  - *Injection Flaws → Command Injection*: `" & ls -l ."`.
  - *Ajax Security → Dangerous Use of Eval*: `<script>alert(1)</script>`.

- Na aplicação bWAPP, os seguintes *bugs*:
  - *SQL Injection (Search/GET)*: `' or 1=1 #`.
  - *SQL Injection (Login Form)*: `' or 1=1 #`.
  - *XSS - Reflected (GET)*: `<script>alert(1)</script>` separado em duas partes.
  - *XSS - Reflected (JSON)*: `"}]}'; </script><script>alert(1);</script>`.
  - *PHP Eval function*: `http://localhost/php_eval.php?eval=echo shell_exec ("ls -l");`.
  - *Directory Traversal - Files*: `http://localhost/directory_traversal_1.php?page=../../../../../../../../../etc/passwd`.

5.5.3. Falsos negativos do ModSecurity

O ModSecurity, assim como qualquer outra ferramenta de segurança que tenta classificar um *payload* recebido externamente, sem executá-lo, em benigno ou malicioso, pode cometer enganos e não alertar sobre problemas reais. Essas omissões são chamadas de falsos negativos. De modo geral, quanto mais rica a semântica do ataque, mais difícil a sua detecção pelas ferramentas automáticas.

Diversos testes de vulnerabilidades realizados ao longo teste texto NÃO são detectados nem bloqueados pelo ModSecurity na configuração padrão de instalação, a saber (a lista a seguir não é exaustiva):

- Na aplicação Juice Shop, seguintes vulnerabilidades:
  - Todas as vulnerabilidades de APIs derivadas de vulnerabilidades semânticas ou de falhas de *design*.
  - DOM XSS (`<iframe src="javascript:alert('xss')">`) no campo de busca.
  - *Directory Traversal - Directories*: `http://localhost/ftp`.
- Na aplicação WebGoat, as seguintes lições:
  - *Injection Flaws → Command Injection*: `" & dir ."`.
  - *Ajax Security → Dangerous Use of Eval*: `123'); alert(1);('`.
  - *Parameter Tampering → Bypass Client Side JavaScript Validation*.

  - *Parameter Tampering → Exploit Hidden Fields*

- Na aplicação bWAPP, os seguintes *bugs*:
  - *HTML Injection - Reflected (POST)*: `<h1><a href="http://www.google.com">Link!</a></h1>`.
  - *HTML Injection - Reflected (GET)*: `<h1><a href="http://www.google.com">Link!</a></h1>`.
  - *HTML Injection - Stored (blog)*: `<h1><a href="http://www.google.com">Link!</a></h1>`.
  - *Directory Traversal - Directories*: `http://localhost/document`.

É possível avaliar se estes falsos positivos ainda ocorrem com outros níveis de rigor (paranoia) do ModSecurity. Em uma instalação nativa do *Core Rule Set*, o nível de paranoia é definido pela variável `tx.paranoia_level` no arquivo `crs-setup.conf`. Existem 4 níveis de paranoia. O nível 1 oferece segurança básica com a mínima necessidade de ajustar falsos positivos. O nível 2 é adequado quando dados de usuários reais estão envolvidos e pode apresentar falsos positivos. O nível 3 oferece segurança de nível bancário online e apresenta muitos falsos positivos, que devem ser tratados com regras de exceção. Finalmente, o nível 4 possui as regras mais restritivas e possivelmente com mais falsos positivos que os outros níveis.

5.6. Comparação de ferramentas, falsos positivos e falsos negativos

Esta seção compara as ferramentas SAST, SCA, DAST e WAF, ilustrando as falhas das ferramentas SAST e SCA em relação às DAST e WAF e vice-versa. Considera-se nesta comparação a implantação *out-of-the-box* das ferramentas.

No geral, ferramentas de análise estática de código (seja código fonte ou binário), visando segurança, tendem a ser conservadoras e pessimistas, emitindo alertas mesmo quando não há muita confiança nem certeza na conclusão da análise, visando prevenir o máximo possível de ameaças. Este comportamento produz muitos falsos positivos (alarmes falsos). As ferramentas SAST usadas pelos desenvolvedores em esquadrões ágeis devem ser configuradas com a sensibilidade mínima a fim de que emitam alertas apenas quando houver bastante confiança (certeza?) de que existe de fato uma vulnerabilidade explorável. Este decisão de configuração diminui a quantidade de falsos positivos, poupando tempo do *Security Champion* que iria para analises desnecessárias. Por outro lado,a ferramenta de cobertura minimalista tem menor capacidade de detecção de problemas, possivelmente, aumentando os falsos negativos.

Geralmente, a cobertura das ferramentas de análise de código (fonte ou binário) é limitada pela incapacidade destas ferramentas de reconhecem todas as instâncias de vulnerabilidades documentadas e suas variações em situações específicas. Além disso, podem haver limitações de ferramentas em percorrer todos os caminhos e fluxos de controle em códigos fonte, tendo bastante dificuldade em identificar vulnerabilidades em trechos de código ou estruturas de dados complexas, resultando em pontos cegos onde as ferramentas não são capazes de enxergar vulnerabilidades, que são omitidas dos alertas. A omissão de um alerta sobre uma vulnerabilidade verdadeira em um falso negativo da ferramenta.

O desempenho das ferramentas SAST pode ser analisado com segue. A ferramenta SonarQube obteve um desempenho relativamente satisfatório em relação ao WebGoat (poucos

falsos positivos e muitos falsos negativos) e modesto em relação ao Juice Shop (poucos positivos verdadeiros e muitos falsos negativos), tendo decepcionado para o bWAPP e a Gruyere (em ambos, majoritariamente falsos negativos). Isto se deve ao fato de haver uma diversidade grande de tecnologias de software novas e, em muitos casos, a versão do *scanner* disponível gratuitamente está bastante desatualizada em relação a essas novas tecnologias.

A ferramenta HoruSec apresentou desempenho bastante interessante, com relatórios longos recheados de alertas para Juice Shop, WebGoat e bWAPP. Mesmo assim, muitos alertas são falsos positivos. Uma vez que HoruSec possui *scanners* mais atualizados que o SonarQube, os alertas obtidos cobrem tecnologias de software mais modernas, como, por exemplo, TypeScript e JWT, além de uma capacidade melhor de detecção de segredos embutidos no código fonte.

O desempenho das ferramentas SCA foi o seguinte. A ferramenta Dependency-Check apresentou uma quantidade de alertas bastante menor que a ferramenta Trivy porque esta última foi utilizada sobre imagens Docker e, por isso, teve acesso a mais dados para análise, não apenas as bibliotecas de terceiros, como aconteceu com o Dependency-Check. Mais artefatos analisados levam a uma superfície de ataque maior e a mais descobertas de vulnerabildiades. As duas ferramentas se complementam quando uma é usada em testes de integração e construção iniciais e a outra é usada mais a frente na esteira de desenvolvimento, quando já existe uma imagem Docker para análise. Uma vez a análise com ferramenta SCA cobre dependências externas e componentes de terceiros, as vulnerabilidades encontradas são diferentes daquelas encontradas pelas ferramentas SAST, que analisam o código fonte. Por isso, a interseção entre os resultados das análises SAST e SCA é mínima. As SCAs apresentam menos falsos positivos que as SASTs porque seus resultados tendem a ser associados a CVEs, considerando-se as mesmas aplicações em análise.

O desempenho das ferramentas DAST pode ser analisa do seguinte modo. As DASTs, quando utilizadas como *scanners* dinâmicos de vulnerabilidades (por exemplo, o ZAP em linha de comando), apresentam menos falsos positivos que SASTs e SCAs porque seus resultados são, no geral, explorações testadas sobre a aplicação em funcionamento. O ZAP em modo *scanner* apresentou bons resultados mesmo para as aplicações ignoradas pelas outras ferramentas (bWAPP e Gruyere). Isto se deve ao fato de todas as aplicações vulneráveis possuírem configurações inseguras de HTTP e não usarem HTTPS. Já, quando usadas em testes manuais, elas auxiliam as explorações de vulnerabilidades reais. Porém, nestes casos, a ferramenta DAST pode possuir uma cobertura menor da aplicação devido a dificuldade em percorrer todos os caminhos de dados e fluxos de execução e controle da aplicação em análise.

Finalmente, como último desafio de análise, o desempenho da ferramenta WAF foi o seguinte. Ao confrontar o *scanner* DAST contra o WAF ModSecurity de configuração padrão sobre as aplicações vulneráveis, percebeu-se uma pequena redução na quantidade total de alertas de vulnerabilidades das aplicações. Por outro lado, o *scanner* DAST alertou sobre uma possível vulnerabilidade no WAF, que pode ser um falso positivo.

Na análise do WAF contra testes de segurança manuais, como os realizados ao longo deste texto, o *firewall* de aplicação web ModSecurity de configuração padrão se mostrou eficiente em detectar vulnerabilidades técnicas simples, como aquelas do OWASP Top 10, mas não foi capaz de detectar aquelas vulnerabilidades de semântica complexa, como as vulnerabilidades de API, tipicamente descobertas em testes de intrusão. Por outro lado, o WAF pode ser personalizado pela adição de regras exclusivas para o contexto de uso, assim como também o aumento de rigor no *Paranoia Level*, o que deve permitir resultados melhores que estes

obtidos com as configurações *out-of-the-box* do ModSecurity e do *Core rule Set* do OWASP.

5.7. Considerações finais

Este texto apresentou o conceitos de desenvolvimento de software seguro ágil e aspectos de cultura *DevSecOps* relevantes para a segurança de aplicações quando promovida por *Security Champions* dentro de esquadrões de desenvolvimento de software.

Quatro tipos de ferramentas de segurança de aplicações (SAST, SCA, DAST e WAF) foram analisadas e comparadas entre si visando esclarecer sobre a melhor maneira de usá-las em conjunto, de forma complementar, em uma esteira de desenvolvimento de software seguro. O desempenho das ferramentas foi analisado por meio de testes e avaliações práticas de segurança contemplando suas capacidades de detecção de vulnerabilidades em aplicações sabidamente vulneráveis.

A análise comparativa de ferramentas de segurança ao longo de um processo de desenvolvimento de software e avaliação sistemática do desempenho de ferramentas SAST não são novidades e já podem até ser encontradas sobre nichos específicos [Braga and Dahab 2015a, Braga et al. 2017, Braga et al. 2019]. Por outro lado, a cultura de segurança ágil e as esteiras *DevSecOps* ainda não são comuns no desenvolvimento de software e sua disseminação ampla é um desafio para a comunidade de software seguro.

A diminuição de falsos positivos e falsos negativos pode contribuir significativamente para o aumento da confiança e, por conseguinte, maior utilização destas ferramentas. Estudos recentes [Rodrigues et al. 2020a, Rodrigues et al. 2020b, Rodrigues et al. 2023] mostram que a utilização de técnicas de aprendizado de máquina podem melhorar muito o desempenho das ferramentas SAST e que a próxima geração destas ferramentas, ao incorporar as técnicas de análise avançadas, será muito superior às ferramentas disponíveis de forma gratuita atualmente. Esta visão otimista pode ser compartilhada com as ferramentas SCA, DAST e WAF.

Agradecimentos

Este trabalho foi realizado pelo grupo de segurança de aplicações do CPQD, dentro do programa ***Security Champions* de Desenvolvimento de Software Seguro** apoiado pela instituição.

Referências

[iee 2014] (2014). Avoiding the top ten security flaws. *URL: https://ieeecs-media.computer.org/media/technical-activities/CYBSI/docs/Top-10-Flaws.pdf*.

[saf 2019] (2019). Software security takes a champion – a short guide on building and sustaining a successful security champions program. *URL: http://safecode.org/wp-content/uploads/2019/02/Security-Champions-2019-.pdf*.

[tes 2020] (2020). Owasp web security testing guide v4.2. *URL: https://owasp.org/www-project-web-security-testing-guide*.

[top 2021] (2021). Owasp top 10 – 2021. *URL: https://owasp.org/Top10/*.

[dev 2021] (2021). Six pillars of devsecops series. *URL: https://cloudsecurityalliance.org/blog/2021/09/09/six-pillars-of-devsecops-series/*.

[top 2023] (2023). Owasp api security top 10. *URL: https://owasp.org/API-Security/*.

[Braga and Dahab 2015a] Braga, A. and Dahab, R. (2015a). A Survey on Tools and Techniques for the Programming and Verification of Secure Cryptographic Software. In *XV Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais — SBSeg 2015*, pages 30–43, Florianópolis, SC, Brazil.

[Braga and Dahab 2015b] Braga, A. and Dahab, R. (2015b). Introdução à Criptografia para Programadores: Evitando Maus Usos da Criptografia em Sistemas de Software. In *Caderno de minicursos do XV Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais — SBSeg 2015*, pages 1–50. Sociedade Brasileira de Computação.

[Braga and Dahab 2016] Braga, A. and Dahab, R. (2016). Mining Cryptography Misuse in Online Forums. In *2016 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, pages 143–150.

[Braga and Dahab 2017] Braga, A. and Dahab, R. (2017). A Longitudinal and Retrospective Study on How Developers Misuse Cryptography in Online Communities. In *XVII Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais (SBSeg'17)*, Brasília, DF, Brazil.

[Braga and Dahab 2018] Braga, A. and Dahab, R. (2018). Criptografia assimétrica para programadores - evitando outros maus usos da criptografia em sistemas de software. In *Caderno de minicursos do XVIII Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais — SBSeg 2018*, pages 1–50. Sociedade Brasileira de Computação.

[Braga and Dahab 2019] Braga, A. and Dahab, R. (2019). Introdução à criptografia para administradores de sistemas com tls, openssl e apache mod_ssl. In *Minicursos do XXXVII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC 2019)*. Sociedade Brasileira de Computação.

[Braga et al. 2017] Braga, A., Dahab, R., Antunes, N., Laranjeiro, N., and Vieira, M. (2017). Practical Evaluation of Static Code Analysis Tools for Cryptography: Benchmarking Method and Case Study. In *The 28th IEEE International Symposium on Software Reliability Engineering (ISSRE)*.

[Braga et al. 2019] Braga, A., Dahab, R., Antunes, N., Laranjeiro, N., and Vieira, M. (2019). Understanding how to use static analysis tools for detecting cryptography misuse in software. *IEEE Transactions on Reliability*, 68(4):1384–1403.

[Braga et al. 2012] Braga, A., do Nascimento, E. N., da Palma, L. R., and Rosa, R. P. (2012). Introdução à segurança de dispositivos móveis modernos–um estudo de caso em android. *Sociedade Brasileira de Computação*.

[da Silva et al. 2019] da Silva, J., Braga, A., Rubira, C., and Dahab, R. (2019). An approach for adaptive security of cloud applications within the atmosphere platform. In *Anais do XIX Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais*, pages 397–402. SBC.

[Fisher 2022] Fisher, D. (2022). *Application Security Program Handbook—A Guide for Software Engineers and Team Leaders*. Manning Publications Co.

[Institute 2023] Institute, P. M. (2023). *Agile Practice Guide*. Project Management Institute, Newton Square, PA.

[McGraw 2004] McGraw, G. (2004). Software security. *IEEE Security and Privacy*, 2(02):80–83.

[Miller 2022] Miller, N. (2022). Security culture 1.0. *URL: https://owasp.org/www-project-security-culture*.

[Pressman 2018] Pressman, R. S. (c2018.). *Software engineering :*. McGraw-Hill,, Chennai :, 7th ed. edition. Includes index.

[Rodrigues et al. 2020a] Rodrigues, G., Braga, A., and Dahab, R. (2020a). A machine learning approach to detect misuse of cryptographic apis in source code. In *Anais do XX Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais*, pages 1–14, Porto Alegre, RS, Brasil. SBC.

[Rodrigues et al. 2020b] Rodrigues, G. E. d. P., Braga, A. M., and Dahab, R. (2020b). Using graph embeddings and machine learning to detect cryptography misuse in source code. In *2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 1059–1066.

[Rodrigues et al. 2023] Rodrigues, G. E. d. P., Braga, A. M., and Dahab, R. (2023). Detecting cryptography misuses with machine learning: Graph embeddings, transfer learning and data augmentation in source code related tasks. *IEEE Transactions on Reliability*, pages 1–12.

[Stuttard and Pinto 2008] Stuttard, D. and Pinto, M. (2008). *The Web Application Hacker's Handbook: Discovering and Exploiting Security Flaws*. Wiley.

[Watson and Zaw 2018] Watson, C. and Zaw, T. (2018). *OWASP Automated Threat Handbook Web Applications*. OWASP Foundation.

Capítulo 6

Proteção de Sistemas Biométricos

Marco Antonio Torrez Rojas (IFC), Charles Christian Miers (UDESC), Marcos Antonio Simplício Jr (POLI-USP), Luis Henrique de Almeida Fernandes (POLI-USP), Rafael Yamada de Oliveira (POLI-USP), Gabriela Guilherme de Andrade (IFC), Isaak Gomes de Araújo (IFC), Sara de Almeida Sehnem (IFC), Vinicius Dacio da Silva (IFC)

Abstract

The protection of biometric information is a growing concern due to the pivotal role of biometrics in today's identification and authentication mechanisms. This chapter contextualizes the subject, covering key concepts, models, procedures and technologies related to the construction of reliable biometric systems (and, hence, required for ensuring the security of systems that rely on biometric data). We address the fundamentals of biometrics, including the characteristics used for personal identification and the techniques for capturing and analyzing biometric data. Furthermore, we discuss the importance of protecting biometric information along its lifecycle, considering privacy, security, revocability, and potential vulnerability concerns. This review establishes a solid foundation for understanding the challenges and requirements in this constantly evolving field.

Resumo

A proteção de informações biométricas é uma preocupação crescente, em especial devido ao papel central da biometria nos mecanismos atuais de identificação e autenticação. Este capítulo contextualiza o assunto, abordando os principais conceitos, modelos, procedimentos e tecnologias relacionados à construção de sistemas biométricos robustos (e, portanto, necessárias para garantir a segurança dos sistemas que dependem de dados biométricos). O documento aborda conceitos fundamentais sobre biometria, incluindo as características utilizadas para identificação pessoal e as técnicas de captura e análise de dados biométricos. Além disso, discute-se a importância de proteger informações biométricas ao longo de todo o seu ciclo de vida, considerando questões de privacidade, segurança, revogabilidade e possíveis vulnerabilidades. Esta revisão estabelece uma base sólida para a compreensão dos desafios e requisitos neste campo em constante evolução.

6.1. Introdução

A biometria desempenha um papel fundamental em várias áreas da nossa sociedade moderna, desde o acesso a sistemas de controle de segurança até a autenticação em aplicações *online*. Sistemas biométricos se fundamentam na utilização de características físicas ou comportamentais aproximadamente exclusivas de um indivíduo para identificá-lo ou autenticá-lo. No primeiro caso, da identificação, as características biométricas do indivíduo são comparadas com todos os registros em um banco de dados em busca de uma correspondência próxima o suficiente, permitindo ao sistema dizer a qual usuário a leitura biométrica pertence (e, por exemplo, autorizá-lo a acessar um local protegido). Já no segundo caso, da autenticação, é comum que o usuário primeiro se identifique, por exemplo escolhendo seu nome de usuário em uma lista, ou apresentando um identificador em um teclado numérico; em seguida, a leitura biométrica capturada é comparada apenas com o indivíduo associado ao identificador apresentado, de modo que a autenticação tem sucesso apenas em caso de elevada correspondência. Em ambos os tipos de aplicação, a biometria surge como uma abordagem promissora para solucionar a crescente necessidade de soluções com maior usabilidade para os processos de identificação e autenticação, sem degradar sua segurança e confiabilidade. A biometria engloba uma ampla gama de aplicações práticas. Por exemplo, no controle de acesso físico, sistemas biométricos substituem ou complementam métodos tradicionais, como chaves ou cartões de identificação, oferecendo uma autenticação bastante segura e confiável [Li and Jain, 2015]. Ao dispensar uma boa memória ou o cuidado com dispositivos físicos nesses cenários, a boa usabilidade trazida por sistemas biométricos pode trazer ganhos de produtividade ao simplificar e agilizar processos administrativos, economizando tempo e recursos [Jain et al., 1996]. Já em dispositivos como *smartphones* e caixas eletrônicos, a autenticação biométrica, como reconhecimento facial ou leitura de impressões digitais, proporciona uma forma rápida e conveniente de desbloqueio e acesso, em particular para usuários que têm dificuldade em lembrar de senhas complexas. Além disso, a autenticação biométrica é comumente aplicada como fator adicional em transações financeiras, fornecendo maior segurança e reduzindo riscos de fraudes [Matos, 2000].

Este documento tem por objetivo contextualizar o uso de biometria em sistemas modernos, abordando os principais tipos conceitos, modelos, procedimentos e tecnologias relacionados à construção de sistemas biométricos robustos. Para isso, a Seção 6.2 apresenta uma breve revisão dos principais conceitos sobre biometria, os principais tipos e características, e exemplos típicos de aplicação em diversos setores. Também uma introdução a norma ISO/IEC 24745:2022 que aborda aspectos relacionados à proteção de dados biométricos. Na Seção 6.3 é introduzido o conceito de sistemas biométricos e os seus principais subsistemas, e respectivas funcionalidades no processamento dos dados biométricos. Também é apresentado o ciclo de vida de tratamento dos dados biométricos, bem como exemplos de aplicação dos sistemas biométricos. Na Seção 6.4 são apresentado os requisitos de segurança recomendados nos sistemas biométricos, em especial os requisitos de renovação e revogação de *templates* biométricos. Também são apresentadas e analisadas as principais ameaças relacionadas ao ciclo de vida de tratamento dos dados biométricos. Finalmente, são apresentados os mecanismos que possibilitam prover proteção dos dados biométricos. Na Seção 6.5 é apresentado a classificação dos sistemas biométricos com base no local em que os dados de referência e identidade biométricos

dos indivíduos são armazenados e comparados. Também são apresentados os modelos de aplicação dos dados biométricos e os modelos de segurança relacionados. Na Seção 6.6 são apresentadas as considerações finais sobre os principais pontos abordados no minicurso. Também são discutidos demandas e oportunidades futuras de pesquisa com relação a segurança e privacidade de dados biométricos.

6.2. Características Biométricas

A base da identificação de usuários em sistemas biométricos são as chamadas *características biométricas*. Essencialmente, elas são formadas por atributos singulares e distintos de cada indivíduo, podendo então ser usadas como padrão para validação de cada usuário do sistema. Essas características podem ser divididas em duas categorias principais: físicas, ou seja, mensuráveis diretamente dos usuários e pouco variáveis (e.g., impressão digital e íris); e comportamentais, baseadas na análise de padrões apresentados pelos usuários (e.g., modo de caminhar, ou forma de assinar manualmente um documento) [Li and Jain, 2015]. Cabe destacar que diferentes características biométricas possuem benefícios e limitações distintas. Por exemplo, características físicas como a impressão digital e a íris apresentam alta exclusividade; porém, elas podem ser mais difíceis de capturar em alguns contextos (e.g., em ambientes nos quais é obrigatório o uso de equipamentos de segurança para proteger mãos e olhos), ou mais fáceis de burlar em outros (e.g., captura de digitais de alvo a partir de superfícies por ele tocadas). Por outro lado, características comportamentais como a assinatura manual e a voz podem ser menos exclusivas, mas mais fáceis de serem coletadas e aceitas pelos usuários.

Essa avaliação comparativa é essencial para a seleção da biometria mais adequada a cada aplicação específica, garantindo segurança, usabilidade e eficiência nos sistemas biométricos utilizados em diferentes áreas, como segurança de dispositivos, controle de acesso e autenticação de transações. É importante observar que, além das características biométricas físicas (anatômicas) e comportamentais (baseadas em ações ou padrões de comportamento), existem algumas características biométricas adicionais que podem ser utilizadas para identificação ou autenticação, embora sejam menos comuns. Algumas destas incluem: características químicas (envolvem a análise de características químicas únicas no corpo humano, como o uso de análise de composição química da pele, saliva ou suor); características do cérebro (reconhecimento biométrico baseado em padrões cerebrais, incluindo a análise de padrões de atividade cerebral ou conectividade funcional); entre outras. A Figura 6.1 ilustra uma classificação de algumas das principais tecnologias e características biométricas utilizadas atualmente.

6.2.1. Características Físicas

As características físicas são aquelas que podem ser medidas diretamente do corpo humano, constituindo atributos aproximadamente únicos e mensuráveis associados a cada indivíduo. Alguns exemplos de características físicas utilizadas na biometria são:

- **Impressões Digitais:** As impressões digitais são padrões distintivos formados por cristas e vales que constituem a superfície dos dedos. Através da análise dessas características, é possível criar modelos únicos para cada indivíduo, permitindo sua identificação precisa [Maltoni et al., 2009]. O processo envolve a aquisição de imagens de alta reso-

Figura 6.1: Classificação das principais tecnologias biométricas [Oliveira Filho, 2014].

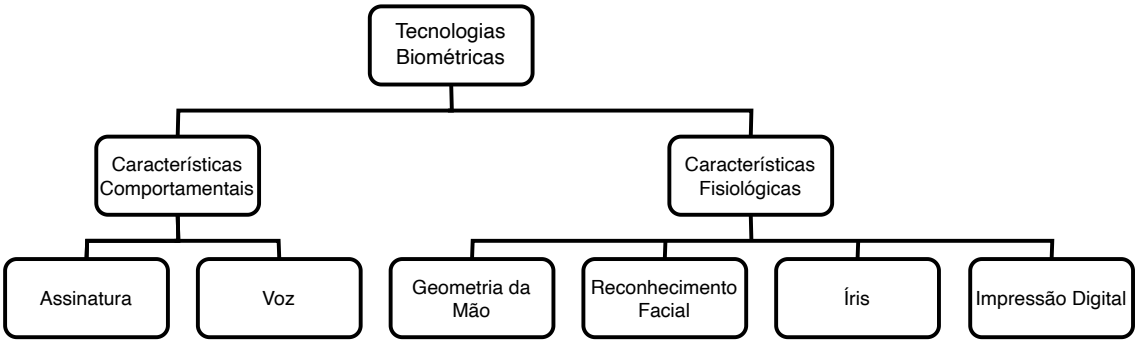
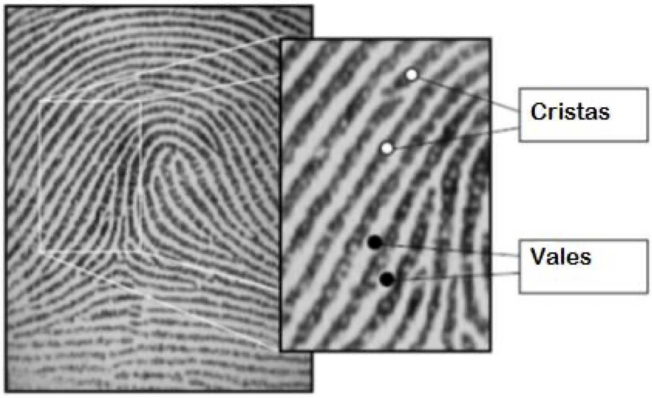


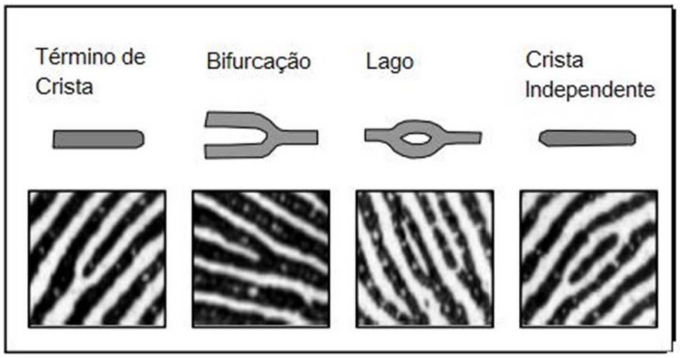
Figura 6.2: Cristas e vales em uma impressão digital [Faria, 2014].



lução das impressões digitais, seguido de pré-processamento para melhorar a qualidade e realçar as características essenciais (Direção das Cristas e Padrões de Arranjo). A extração de características das impressões digitais envolve a identificação de pontos de interesse, como bifurcações e terminações, que são utilizados para criar um modelo ou *template* da impressão. Esses modelos são então armazenados e comparados com outras impressões digitais para efetuar a autenticação ou identificação de usuários [Maltoni et al., 2009]. A Figura 6.2 ilustra os padrões de cristas e vales que se encontram na identificação de impressões digitais. A Figura 6.3 ilustra os detalhes identificados em impressões digitais que são empregados nas capturas para a análise comparativa de impressões digitais.

- **Reconhecimento Facial:** O reconhecimento facial envolve a análise das características faciais de uma pessoa, como a forma do rosto, posição dos olhos, nariz e boca. Com base nas características extraídas, um *template* ou representação matemática do rosto é criado, capturando suas informações essenciais. Essa característica é comumente utilizada em sistemas de segurança e controle de acesso. A verificação da biometria é feita então a partir da semelhança desse *template* e da imagem do rosto capturada diretamente do indivíduo interagindo com o sistema, levando à liberação ou negação de acesso [Ratha et al., 2001a].
- **Geometria da Mão:** A geometria da mão analisa a estrutura e as proporções da mão de um indivíduo, incluindo o tamanho dos dedos, formato da palma e posição das

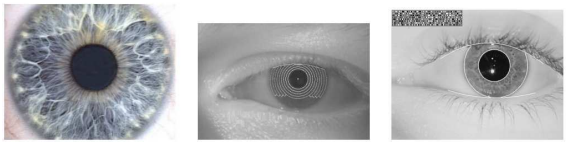
Figura 6.3: Tipos de minúcias encontradas em uma impressão digital [Faria, 2014].



articulações.

- **Reconhecimento de Íris e Retina:** O reconhecimento de íris e retina analisa as características únicas presentes nos olhos de um indivíduo. A íris é a parte colorida do olho, enquanto a retina é a camada interna sensível à luz. Essas características são altamente distintivas de cada indivíduo. A Figura 6.4 ilustra a imagem da íris adquirida sob condições ideais (esquerda). É mostrada a fase de aplicação do algoritmo de extração de características (centro), e a íris com seu código de íris associado (direita).

Figura 6.4: Processo de captura e processamento do reconhecimento íris/retina. Fonte: [Costa et al., 2006].



6.2.2. Características Comportamentais

As características comportamentais referem-se aos padrões de comportamento de um indivíduo. Essas características são mais difíceis de serem copiadas ou forjadas, pois estão relacionadas à forma como uma pessoa age ou se comporta. Alguns exemplos de características comportamentais utilizadas na biometria são [Li and Jain, 2015]:

- **Voz:** A análise da voz envolve a identificação de características distintivas, como a frequência, o tom e o padrão de fala. Existem também sistemas baseados em desafio-resposta, em que o usuário deve ler uma frase específica escolhida dinamicamente como parte do processo de verificação biométrica.
- **Assinatura:** A análise da assinatura envolve a identificação dos traços únicos presentes na maneira como uma pessoa assina seu nome. Essa característica é amplamente utilizada em autenticação em documentos e transações.
- **Forma de Digitação:** A forma de digitação analisa os padrões de pressão aplicada às teclas, velocidade entre teclas distintas, e duração do pressionamento durante a

digitação [Araújo et al., 2005]. Essa característica pode ser utilizada em sistemas de autenticação em computadores e dispositivos móveis.

- **Termograma:** A biometria termográfica, também conhecida como termograma ou termografia, é uma técnica não invasiva e sem contato para medir e registrar a temperatura do corpo humano ou de outros objetos. Essa tecnologia é baseada na captura e análise da radiação infravermelha emitida por corpos e objetos que emitem calor. A termografia é amplamente utilizada em diversas aplicações, incluindo medicina, indústria, segurança física e vigilância, e até mesmo na conservação do meio ambiente, encontrando também alguma aplicação em sistemas biométricos [Cross and Smith, 1995].

Existem diferentes critérios que podem ser considerados relevantes para a adoção de um tipo específico de característica biométrica em sistemas de segurança. Critérios comumente encontrados na literatura são [Jain et al., 1996]: universalidade, que indica o quão comum é a presença dessa característica na população alvo do sistema (e.g., assinaturas manuais podem ser poucos difundidas entre usuários analfabetos); exclusividade, ou a capacidade de ser única para cada indivíduo; permanência, que mede a estabilidade da característica biométrica ao longo do tempo, considerando fatores ambientais adversos (e.g., o uso de produtos de limpeza abrasivos pode apagar impressões digitais) ou naturais (e.g., envelhecimento); coletabilidade, que se refere à facilidade e conveniência na coleta dos dados biométricos (e.g., a necessidade de treinamento específico para uso do coletor biométrico afeta negativamente esta métrica); desempenho, em termos de precisão e confiabilidade na identificação; aceitabilidade, que tem relação com grau de aceitação e conforto dos usuários quanto à sua utilização (e.g., tecnologias que usam leitores próximos aos olhos costumam ser mais invasivas do que as baseadas em câmeras comuns, distantes do indivíduo); e facilidade de circunvenção, considerando tentativas de falsificar informações biométricas ou burlar o sistema sem a conivência do usuário.

Embora não haja um consenso na literatura, a evolução em termos de tecnológica de *hardware* e *software* pode afetar de forma distinta a aquisição de cada característica biométrica, a Tabela 6.1 apresenta um comparativo entre tecnologias contemporâneas [Balakrishnan et al., 2021] e tecnologias clássicas [Jain et al., 1996] das principais características biométricas aqui discutidas, com base nesses critérios. Cumpre notar, entretanto, que cada aplicação específica costuma necessitar de uma análise mais detalhada do que essa avaliação comparativa ampla, considerando as particularidades do cenário alvo. Por exemplo, embora a tabela considere que voz seja de coleta mais difícil do que face, essa facilidade pode facilmente se inverter em ambientes médicos em que são usados equipamentos de proteção facial e máscara. Como outro exemplo, atualmente a facilidade de circunvenção de autenticação por face em sistemas remotos tem sido fortemente influenciada por técnicas de inteligência artificial (o chamado "deep fake" [Wojewidka, 2020]). Entretanto, em sistemas em que a biometria é coletada por sensores localmente, técnicas de circunvenção como o uso de máscaras 3D teriam que passar por diversos sensores voltados a verificar a presença de um usuário humano em frente à câmera (conceito conhecido como *liveness*, ou "prova de vida") [Hernandez-Ortega et al., 2023]; exemplos incluem o uso de múltiplas câmeras, para captura do rosto de diferentes ângulos, e sensores de calor.

É relevante também ressaltar que a utilização de características comportamentais, em conjunto com características físicas, costuma ser interessante para aprimorar ainda mais a segurança e a precisão dos sistemas biométricos. A combinação inteligente dessas abordagens permite o desenvolvimento de sistemas multimodais, que utilizam múltiplas características para verificar a identidade de um indivíduo. Em particular, o uso de características comportamentais é útil em cenários nos quais a captura de características físicas pode ser desafiadora ou indesejada [Li and Jain, 2015]. Por exemplo, em sistemas de autenticação remota, o reconhecimento de voz ou a análise da forma de digitação podem ser preferíveis em comparação com a coleta de impressões digitais ou outros dados biométricos físicos. À medida que a biometria se torna mais integrada no cotidiano dos usuários, é fundamental considerar a privacidade e a proteção dos dados biométricos, garantindo o uso responsável e ético dessas tecnologias para o benefício de todos.

Tabela 6.1: Comparativo entre as abordagens clássica e contemporânea de biometria

Biometrias	**Universalidade**	**Exclusividade**	**Permanência**	**Coletabilidade**	**Desempenho**	**Aceitabilidade**	**Circunvenção**
Face	alta	baixo	média	alta	baixo	alta	alta/baixo
Impressão Digital	média	alta	alta	média	alta	média	alta/média
Geometria da Mão	média	média	média	alta	média	média	média
Veias das mãos	média	média	média	média	média	média	alta
Iris	alta/média	alta	alta	média	alta	baixo	alta/baixo
Retina	alta	alta	média	baixo	alta	baixo	alta
Assinatura	baixo	baixo	baixo	alta	baixo	alta	alta/baixo
Voz	média	baixo	baixo	média	baixo	alta	alta/baixo
Termograma	alta	alta	baixo	alta	média	alta	alta

Na Tabela 6.1 podemos notar que existem colunas com um único valor e colunas com dois valores. As colunas que possuem um único valor informam que a avaliação efetuada por [Balakrishnan et al., 2021] e [Jain et al., 1996] são a mesma. Nas colunas que possuem dois valores, o primeiro valor é referente a avaliação de [Jain et al., 1996], e o segundo valor é referente a avaliação de [Balakrishnan et al., 2021].

6.2.3. Métodos de Captura e Análise de Dados Biométricos

A captura de dados biométricos é um processo fundamental para realizar a identificação e autenticação biométrica. Os métodos, técnicas e dispositivos de captura podem variar dependendo da característica biométrica.

Os sensores ópticos são comumente utilizados na captura de características como impressões digitais, reconhecimento facial, íris e retina. Esses sensores capturam imagens detalhadas das características biométricas e as convertem em dados digitais [Oliveira Filho, 2014]. Processo similar acontece com câmeras de alta resolução, que costumam ser amplamente utilizadas para capturar imagens faciais, geometria da mão e outros traços físicos. Por meio de algoritmos específicos, é possível extrair as informações relevantes e criar modelos biométricos aproximadamente exclusivos para cada indivíduo.

No caso da captura de voz, são utilizados microfones. As amostras de som capturadas são analisadas por meio de algoritmos de reconhecimento de voz, que identificam padrões e características distintivas para autenticação biométrica.

Já para assinaturas manuais de usuários, é comum o uso de canetas digitais. Os sensores embutidos na caneta registram os movimentos e a pressão exercida durante a escrita, gerando uma representação digital precisa da assinatura, semelhante a técnica

já citada de formas de digitação. Outra alternativa consiste no uso de mesas digitalizadoras. Combinadas ou não com canetas digitais, esses dispositivos permitem que os usuários escrevam suas assinaturas diretamente na superfície digital, capturando características biométricas relevantes (e.g., velocidade e trajetória do movimento) em tempo real [Oliveira Filho, 2014].

6.2.4. Aplicações da biometria

A biometria tem encontrado uma amplitude de aplicações em diferentes setores, em particular devido à conveniência de seu uso em tarefas de identificação e autenticação de indivíduos, com base em características únicas do corpo humano. Algumas das principais áreas em que a biometria é amplamente utilizada incluem [Jain et al., 1999]

- **Identificação pessoal:** A aplicação mais comum da biometria é na identificação pessoal. As impressões digitais, o reconhecimento facial, a íris, a voz e a geometria da mão são alguns dos atributos biométricos utilizados para estabelecer a identidade de uma pessoa de maneira única e confiável. Isso tem sido aplicado em *smartphones*, *tablets* e *laptops* para autenticação de usuário, substituindo senhas e PINs.
- **Controle de acesso:** A biometria é frequentemente utilizada em sistemas de controle de acesso para garantir a segurança em locais restritos. Impressões digitais e reconhecimento facial são os métodos mais comuns nesse contexto. Estes são usados em aeroportos, prédios corporativos, laboratórios de pesquisa e outras instalações nas quais a identificação precisa é crucial para a entrada ser autorizada.
- **Aplicações forenses:** A biometria desempenha um papel importante na investigação criminal e na aplicação da lei. As impressões digitais são usadas para comparar e identificar suspeitos, enquanto a análise de voz pode ser útil para fins de reconhecimento de voz e identificação de locutor. Além disso, o reconhecimento facial é utilizado para confrontar imagens de vigilância e auxiliar na identificação de suspeitos.
- **Saúde e cuidados médicos:** A biometria tem sido aplicada na área da saúde para garantir a segurança do paciente, acessar registros médicos eletrônicos e controlar o acesso a áreas restritas, como salas de cirurgia e laboratórios. Além disso, sistemas biométricos são utilizados para monitorar sinais vitais, como frequência cardíaca e padrões de respiração, fornecendo informações precisas para diagnósticos e tratamentos.
- **Serviços financeiros:** A biometria está sendo cada vez mais utilizada nos serviços financeiros para aumentar a segurança das transações. As impressões digitais, reconhecimento facial e voz são usados para autenticação de identidade em caixas eletrônicos, pagamentos móveis e autenticação de transações *online*, substituindo senhas e códigos de acesso que podem ser comprometidos.
- **Educação:** A biometria também possui aplicações na área educacional, principalmente em instituições de ensino e exames. Os sistemas biométricos podem ser utilizados para registro de presença de alunos, controle de acesso a áreas restritas e garantir a integridade dos exames, evitando fraudes e substituições de identidade.

Essas são apenas algumas das muitas aplicações da biometria, revelando o seu amplo potencial em garantir a segurança, facilitar processos de identificação e melhorar a eficiência em várias áreas. Com o contínuo desenvolvimento de tecnologias biométricas

e aprimoramento dos algoritmos, se espera que a biometria desempenhe um papel ainda mais significativo no futuro, transformando a forma como se autentica e identifica.

6.2.5. ISO/IEC 24745:2022

A ISO/IEC 24745:2022 [ISO/IEC 24745, 2022] é uma versão atualizada e aprimorada da ISO/IEC 24745:2011 e aborda a necessidade de mecanismos de autenticação seguros para aplicações fornecidas pela Internet, como serviços bancários *online* e atendimento médico remoto por exemplo. Com o aumento da dependência da Internet, a autenticação adequada entre os usuários e os serviços se torna cada vez mais crítica. A norma aborda especificamente o uso de técnicas biométricas como um mecanismo de autenticação confiável, servindo como base para o levantamentos realizados durante o processo de revisão sistemática sobre biometria, segurança e sistemas.

A ISO/IEC 24745:2022 possui diversos aspectos relevantes, mas em função da limitação de páginas do minicursos, destacam-se [ISO/IEC 24745, 2022]:

- **Autenticação biométrica:** A norma reconhece a autenticação biométrica como uma abordagem confiável para verificar a identidade de um indivíduo. Isso envolve o uso de características comportamentais e fisiológicas, como impressões digitais, padrões de voz, imagens da íris e imagens faciais para reconhecimento automatizado.
- **Compromisso entre privacidade e segurança:** A autenticação biométrica levanta preocupações sobre a privacidade dos dados biométricos dos indivíduos. Embora a vinculação precisa entre o indivíduo e a credencial biométrica forneça uma forte garantia de autenticação, também apresenta desafios em relação à proteção dos dados biométricos e à prevenção de seu uso indevido.
- **Proteção de dados biométricos:** A norma aborda a proteção dos dados biométricos em relação aos requisitos de confidencialidade, integridade e renovabilidade/revogabilidade durante o armazenamento e a transferência. Isso inclui a vinculação segura entre uma referência biométrica (BR) e uma referência de identidade (IR), além de fornecer diretrizes para o gerenciamento e o processamento seguros e em conformidade com a privacidade dos dados biométricos.
- **Modelos de aplicação de sistemas biométricos:** São descritos diferentes cenários de aplicação de sistemas biométricos, incluindo o armazenamento e a comparação de referências biométricas. Além disso, inclui uma análise de ameaças e as contramedidas inerentes aos modelos de aplicação de sistemas biométricos.
- **Privacidade do indivíduo:** Fornece orientações sobre a proteção da privacidade do indivíduo durante o processamento de informações biométricas. Isso inclui garantir que as informações biométricas sejam processadas de maneira segura e em conformidade com as regulamentações de privacidade aplicáveis.

Em síntese, este capítulo proporciona uma visão abrangente das características biométricas, suas aplicações e o processo de captura e análise de dados. Explora-se a diversidade de usos da biometria, desde a identificação pessoal até aplicações forenses, saúde, serviços financeiros e educação. Com isso compreende-se a importância da avaliação criteriosa das características biométricas em termos de suas propriedades exclusivas, confiabilidade e aceitabilidade pelos usuários. Além disso, enfatiza-se a necessidade de considerar a privacidade dos dados biométricos em todas as etapas. A Seção 6.3 aprofunda as abordagens sobre Sistemas Biométricos, na qual são examinadas as diferentes

abordagens, tecnologias e desafios associados a esses sistemas, ampliando o entendimento sobre como a biometria é implementada para garantir identificação e autenticação precisas e seguras.

6.3. Sistemas biométricos

A presente seção aborda os principais aspectos dos sistemas biométricos e seu funcionamento, bem como seus subsistemas e como estes se relacionam de acordo com a ISO/IEC 24745 [ISO/IEC 24745, 2022].

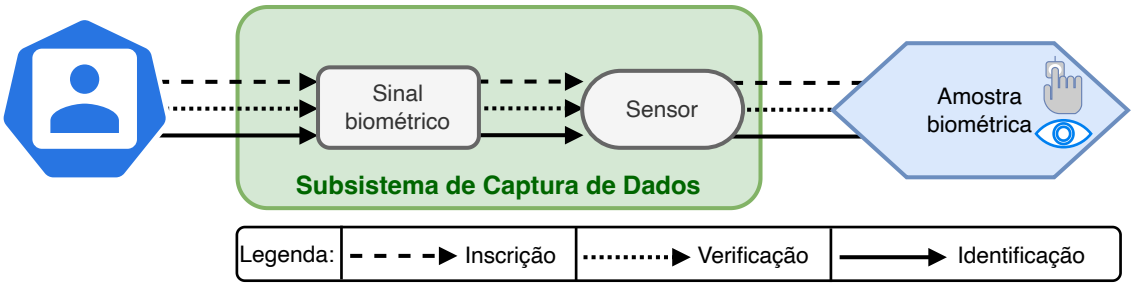
6.3.1. Sistemas e subsistemas biométricos

A biometria associada às tecnologias computacionais é conhecida como sistemas biométricos. Os sistema biométricos são empregados para realizar o reconhecimento por meio de características físicas ou comportamentais, que sucintamente se referem a um conjunto de diversas etapas de um processo com a finalidade de identificação e autenticação de indivíduos [Marcondes, 2019]. De um modo geral, um sistema biométrico tem como principal função vincular, atendendo requisitos de segurança, uma referência de identidade (IR) com uma referência biométrica (BR) de acordo com a aplicação em que está inserido. Os conceitos de IR e BR podem ser definidos, respectivamente, como um atributo não biométrico capaz de identificar exclusivamente um indivíduo dentro de um domínio específico e uma ou mais amostras ou modelos biométricos pertencentes a um titular dentro do sistema biométrico [ISO/IEC 24745, 2022]. A escolha de uma IR dependerá do contexto de uso da aplicação, por exemplos: número do Registro Civil (RG), número de matrícula/inscrição associada ao domínio (instituição de ensino, companhia, clube, etc.) e número de passaporte. Já a BR é a representação de uma característica biométrica, e.g., uma imagem facial armazenada digitalmente em um passaporte ou um modelo de minúcias de impressão digital em uma carteira de identidade. Além disso, ao processar amostras biométricas por algoritmos, as características são convertidas para representações matemáticas que também são consideradas referências biométricas.

Existe uma sequência de processos para vinculação de uma IR a uma BR e sua utilização dentro do sistema biométrico. Cada um destes processos é realizado por um subsistema, que são partes menores e independentes dentro do sistema biométrico principal. Estes subsistemas tem funções específicas que se relacionam sequencialmente para atingir o objetivo principal de cadastrar, identificar e verificar a identidade de um individuo, sendo estas a captura de dados, processamento de sinal, armazenamento, comparação e decisão. Vários subsistemas podem ser incluídos adequando à necessidade da aplicação, mas, conforme referencia a norma, os cinco principais subsistemas são:

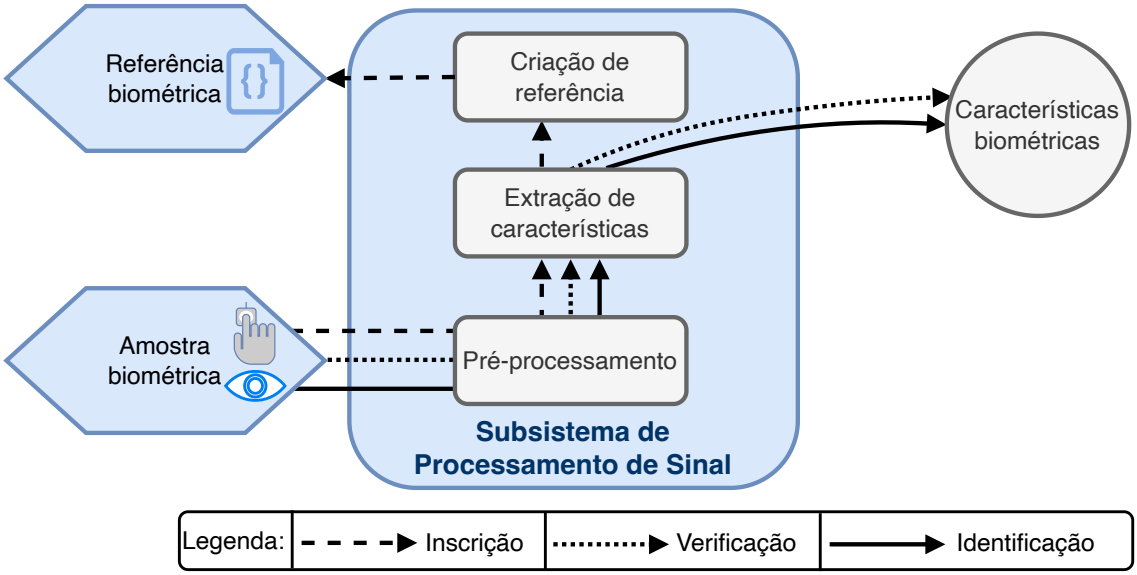
- **Captura de Dados:** É o conjunto de elementos que contém dispositivos ou sensores para coletar sinais de uma característica biométrica e convertê-los em um formato adequado para ser utilizado. A Figura 6.5 ilustra quando o usuário insere a sua biometria e o sinal biométrico é capturado por um sensor que converte para uma amostra biométrica que é enviada ao próximo subsistema. O Subsistema de Captura de Dados depende do tipo de biometria utilizada e afeta todo o desempenho do sistema a partir da qualidade da amostra obtida e do sensor ou dispositivo de coleta.

Figura 6.5: Subsistema de Captura de Dados.



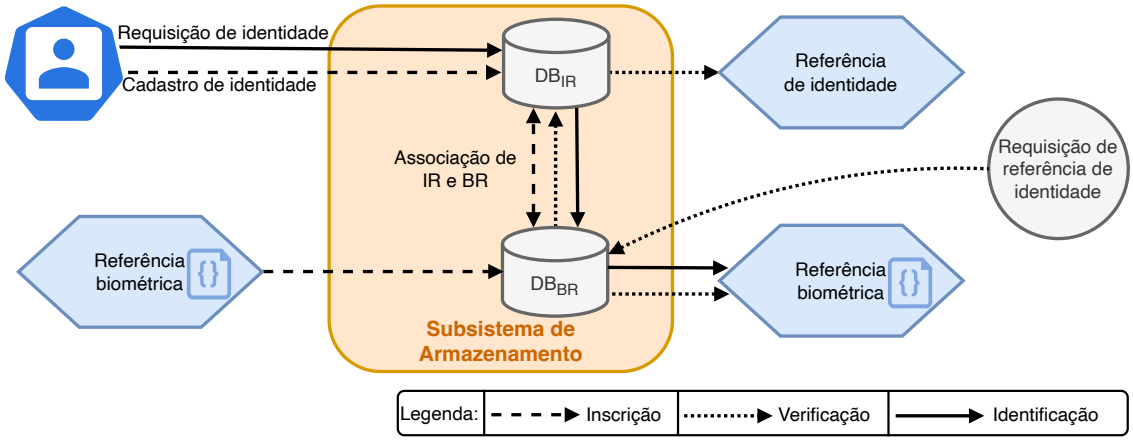
- **Processamento de Sinal:** Esta etapa é responsável por aprimorar a biometria bruta extraindo um conjunto de recursos possíveis de serem comparados com outras amostras biométricas previamente registradas em um Subsistema de Armazenamento. A Figura 6.6 indica que primeiramente é feito um pré-processamento no qual o sistema realiza correções, ajustes e remoção de ruídos para melhorar a qualidade da amostra obtida. Após isso, o sistema extrai características distintivas dependendo do tipo de biometria utilizada e, baseado nas propriedades extraídas, pode enviar diretamente ao sistema de comparação (processo de identificação ou verificação) ou criar uma referência biométrica para ser armazenada (processo de inscrição).

Figura 6.6: Subsistema de Processamento de Sinal.



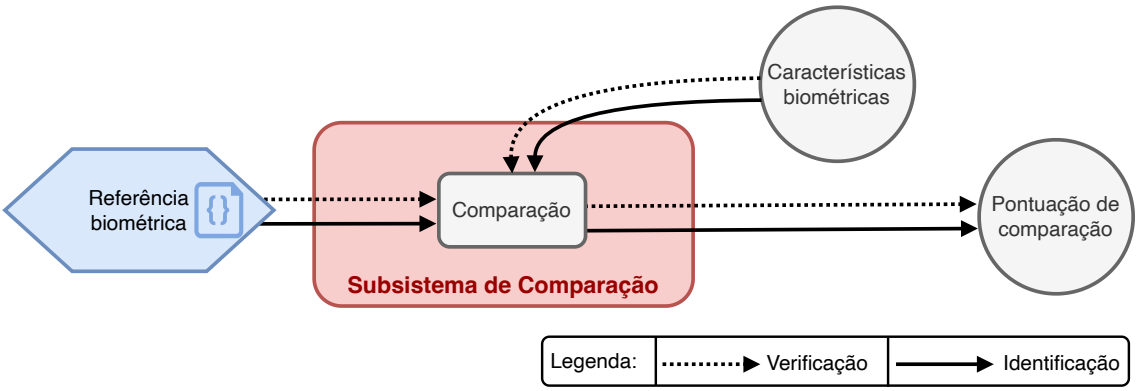
- **Armazenamento de Dados:** São bancos de dados que armazenam e vinculam as BRs e IR. A Figura 6.7 ilustra o processo em que um banco de dados recebe a referência de identidade do usuário e a referência biométrica processada pelo Subsistema de Processamento de Sinal. Assim, pode atender requisições externas, como por exemplo fornecer uma BR armazenada para comparação ou uma IR para o processo de identificação, e.g., os bancos de dados geralmente são separados por questões de segurança e privacidade.

Figura 6.7: Subsistema de Armazenamento.



- **Comparação:** É nesse subsistema que é determinada a similaridade entre as características biométricas capturadas e as referências biométricas armazenadas previamente no Subsistema de Armazenamento conforme a Figura 6.8. Em um processo de verificação, é utilizada a comparação 1:1, na qual a característica biométrica é comparada com uma BR armazenada pertencente a um titular único e produz uma pontuação para comparação. Já em em um processo de identificação, a pontuação é baseada em uma comparação 1:N entre a característica obtida e um conjunto de BRs armazenadas pertencentes a mais de um titular.

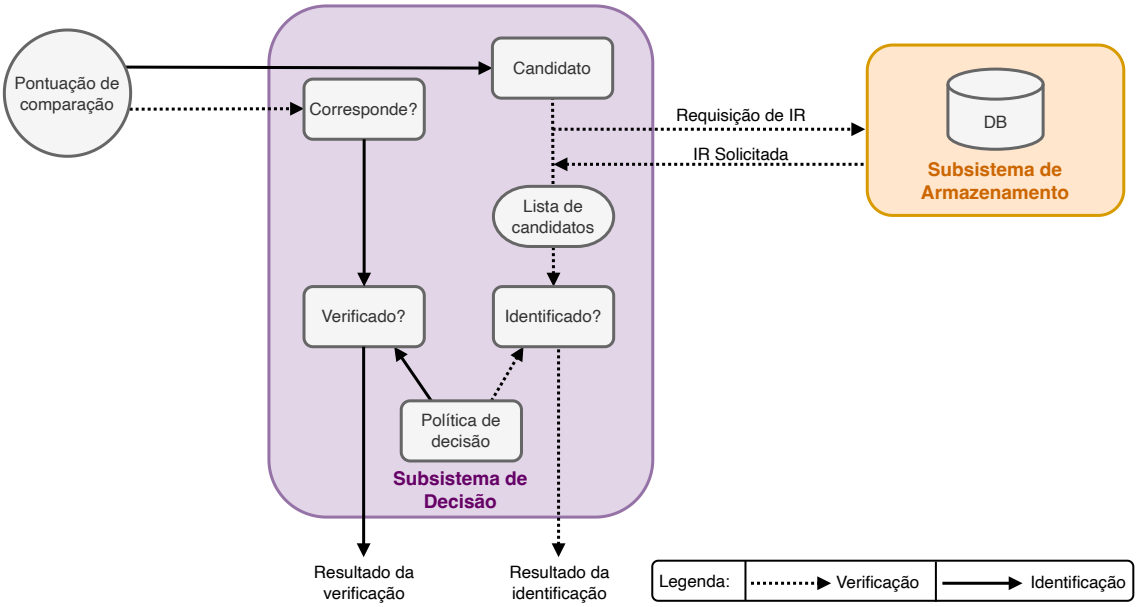
Figura 6.8: Subsistema de Comparação.



- **Decisão:** É o momento em que o subsistema determina se a amostra capturada é similar o suficiente com a referência armazenada para que ambas tenham a mesma fonte. O subsistema tem como entrada a pontuação obtida no Subsistema de Comparação e a saída é decisão tomada, que se difere nos dois processos de verificação e identificação exemplificados na Figura 6.9. Na verificação, é averiguado se a pontuação obtida é suficiente para corresponder ao usuário cadastrado e, baseado na política de decisão, o sistema retorna a aceitação ou rejeição do titular dos dados biométricos. Na identificação, o sistema determina, pela pontuação obtida, se o usuário é um candidato ou não.

Para isso, solicita ao banco de dados a(s) IR(s) associada(s) à(s) BR(s) candidata(s) e pode retornar uma identidade ou uma lista de candidatos, também baseado na política de decisão aplicada.

Figura 6.9: Subsistema de Decisão.



Além dos cinco principais subsistemas, outros podem ser incluídos conforme a necessidade. Estes podem ser relacionados às medidas de segurança, privacidade, formatação de dados, adaptação de referência (quando se deseja minimizar fatores externos que podem afetar a taxa de reconhecimento ou para atualizar referências que podem ser alteradas por efeitos de envelhecimento, por exemplo), entre outros.

6.3.2. Ciclo de vida das informações em sistemas biométricos

O ciclo de vida de informações biométricas se refere ao conjunto de etapas que um dado é submetido desde a sua coleta, uso e descarte adequados. O gerenciamento da privacidade durante todo esse processo é relevante e deve estar em conformidade com as regulamentações e leis de proteção de dados. Esta subseção aborda requisitos e diretrizes para garantir a privacidade das informações biométricas bem como as etapas do ciclo de vida em que esses dados são submetidos.

6.3.2.1. Requisitos e diretrizes

Dados biométricos são informações pessoalmente identificáveis (PII), i.e., que estes podem estar direta ou indiretamente vinculados ao titular a quem se referem e podem permitir identificá-lo. Sendo assim, para tratar as questões relacionadas à privacidade e proteção de dados pessoais em sistemas de tecnologia da informação, incluindo os sistemas que utilizam as tecnologias biométricas, recomenda-se que a norma internacional [ISO/IEC

29100, 2011] seja aplicada. Para garantir a proteção contra ameaças à privacidade na utilização de dados biométricos, são estabelecidos três requisitos e diretrizes de privacidade:

- **Irreversibilidade:** Um dos principais objetivos da [ISO/IEC 29100, 2011] é garantir a transparência em relação às práticas de coleta e uso de dados pessoais. Isso significa que os dados biométricos não devem ser utilizados para qualquer finalidade diferente da originalmente pretendida. Para isso, estes devem ser processados por transformações irreversíveis antes do armazenamento, evitando que seja possível a recuperação dos traços biométricos que originaram a BR. Uma das maneiras de alcançar esse objetivo é através da aplicação de uma função de transformação inversível, ou unidirecional, como a cartesiana e polar.
- **Desvinculabilidade:** Ao capturar dados biométricos, o sistema deve evitar que informações biométricas sejam correlacionadas entre bancos de dados de uma mesma aplicação ou de aplicações distintas. No caso de uma mesma aplicação, se o modelo biométrico do usuário for revogado, a mesma biometria deve ser capaz de gerar um novo modelo diferente do antigo. Já em aplicações distintas, a desvinculabilidade deve garantir que não haja relação entre dois modelos da mesma biometria, evitando a comparação cruzada entre bancos de dados.
- **Confidencialidade:** A segurança de dados também deve ser assegurada segundo a [ISO/IEC 29100, 2011]. Sendo assim, um sistema biométrico deve ter confidencialidade para proteger BRs contra o acesso não autorizado de entidades. Uma das principais medidas para evitar esse risco à privacidade é separar o armazenamento e utilizar servidores protegidos com controle de acesso, e utilizar criptografias específicas durante a transmissão e o armazenamento dos dados biométricos.

6.3.3. Privacidade do ciclo de vida

O ciclo de vida engloba todas as etapas em que um dado biométrico entra no sistema até o seu desuso, sendo estes: coleta, transferência, uso, armazenamento, retenção, arquivamento e *backup* de dados e descarte. De acordo com a [ISO/IEC 24745, 2022], os requisitos de privacidade que devem ser cumpridos em cada uma das etapas são:

- **Coleta:** O consentimento do sujeito ao tomar posse de seus dados biométricos é de extrema importância nessa etapa. Não obstante, mesmo que a organização tenha a permissão, esta deve apenas extrair a quantidade mínima de informações biométricas necessárias para a finalidade pretendida. O usuário deve ser informado sobre a finalidade da coleta e por quanto tempo seus dados vão ser retidos, quais as alternativas caso o mesmo não queira ou não possa ser cadastrado, o tipo e a quantidade de informações biométricas capturadas, como estas serão processadas no sistema biométrico e informações de identificação sobre o responsável, e a organização que vão gerir seus dados.
- **Transferência:** Esta etapa diz respeito a transferências de informação biométrica entre a organização e um terceiro. Para garantir a privacidade nesse processo, além do consentimento explícito ou implícito no serviço solicitado pelo sujeito, deve-se fornecer a ele informações como: quem irá recebê-las, qual o conteúdo e a quantidade que será transferida, qual entidade irá efetuar a transmissão, a finalidade e o período de retenção dos dados.

- **Uso:** Ao entrar em um sistema biométrico, os dados podem ser acessados, processados ou modificados para a finalidade específica do sistema. Sendo assim, o titular deve estar ciente e dar permissão para esta utilização. Na condição da organização ter a intenção de utilizar as informações para outros fins diferente dos especificados, novamente deve informar a descrição da nova finalidade e o período da retenção de dados, tendo a aceitação do usuário e evitando o desvio de função ou a utilização para obter dados relacionados à saúde ou genética do mesmo.
- **Armazenamento:** Como um Subsistema de Armazenamento pode ser distribuído, para garantir a confidencialidade e a integridade dos dados, pode ser necessário que as informações sejam identificadas como PII confidenciais. As organizações devem manter as informações física ou logicamente separadas de outras PIIs do titular para reduzir o impacto na privacidade do titular.
- **Retenção:** Se refere ao período em que os dados biométricos serão mantidos no sistema até o descarte. Para evitar que gere riscos tanto à organização quanto à informação biométrica, é necessário retê-las apenas pelo tempo necessário. Além disso, para não haver problemas durante auditorias, é importante garantir justificativas concretas para manter esses dados.
- **Arquivamento e *backup* de dados:** Esta etapa é responsável por contribuir na garantia de disponibilidade e integridade dos dados biométricos ao longo do tempo. O arquivamento serve para armazenar informações que não estão mais em uso ativo para preservação permanente ou a longo prazo. A criação de cópias de segurança dos dados biométricos em intervalos regulares pelo *backup*, assim como o arquivamento, deve atender políticas de segurança e privacidade mantendo controle de acesso aos dados.
- **Descarte:** Após fazer uma coleta de dados biométricos, caso o objetivo tenha sido atingido, o período de retenção dos dados tenha sido expirado ou o titular conteste o consentimento na coleta ou uso das informações biométricas, a organização ou terceiros que a possuem devem descartá-las de forma segura. Isso serve até mesmo para armazenamento distribuído e dados de *backup*.

6.3.4. Principais processos dos sistemas biométricos

Para efetivar o reconhecimento de uma pessoa, um sistema biométrico é composto por três processos principais:

- **Inscrição:** É realizado o cadastro de dados por meio dos sensores do Subsistema de Captura de Dados que envia a amostra biométrica para o Subsistema de Processamento de Sinal. É nesse momento em que os dados são processados e convertidos para um formato que o sistema entenda e finalmente são extraídas as referências biométricas para serem armazenadas e utilizadas para comparação futuramente [T. R. Jacqueline, 2012]. Além disso, durante a inscrição também é necessário que o usuário apresente sua IR para a associação com a BR cadastrada. Esse processo é finalizado no Subsistema de Armazenamento, onde o tipo de aplicação vai definir como e onde serão guardadas as informações.
- **Verificação:** A verificação se inicia da mesma forma que a inscrição, capturando os dados biométricos e processando-os para comparação com modelos armazenados. Porém, em vez de seguir para o banco de dados, as informações vão para o Subsistema de

Comparação, e, com base na pontuação de similaridade entre estas e a referência cadastrada no processo de inscrição, o Subsistema de Decisão determinará se pertencem ao mesmo indivíduo.
- **Identificação:** O processo de identificação é muito similar ao de verificação. Entretanto, ao chegar no Subsistema de Comparação, o sistema calcula a similaridade entre a amostra obtida e outras referências armazenadas. Assim, o Subsistema de Decisão retorna se existe pelo menos um ou mais correspondentes as informações biométricas fornecidas.

A verificação e a identificação realizam a autenticação e dependem dos dados armazenados na inscrição. A principal diferença entre estas é a forma de comparação e decisão. Na identificação, o usuário apresenta sua biometria e o Subsistema de Comparação determina a similaridade entre esta e um conjunto de BRs de mais de um titular já cadastrados, retornando as pontuações de comparação entre estas. Logo, o objetivo da identificação é obter uma identidade ou uma lista de candidatos em uma busca 1:N de acordo com a pontuação de comparação e com a política de decisão. Já na verificação, além da biometria, o indivíduo deve informar sua IR para que o sistema faça uma busca específica (1:1) e verifique se os dados informados pertencem a fonte que reivindicou. A decisão desse processo também depende da pontuação de comparação e da política de decisão do sistema.

6.3.5. Aplicações dos sistemas biométricos

Existem diversos sistemas que utilizam a biometria para trazer mais segurança e conveniência para o usuário. Esta seção elenca alguns exemplos de uso de biometria nos quais os *templates* biométricos necessitam ser tratados adequadamente.

6.3.5.1. Urnas eletrônicas

O sistema de votação eletrônico brasileiro tem por objetivo garantir segurança e transparência no processo eleitoral. Com a evolução desse sistema, foram realizados aprimoramentos para resolver problemas relacionados à segurança e privacidade do eleitor, entre estes, em 2006, a incorporação da identificação biométrica [Schauren, 2016].

O cadastro da biometria é um processo que envolve a coleta das informações biométricas dos eleitores, como suas impressões digitais, para criar um registro único e seguro. Ao apresentar um documento oficial com foto, é feita a coleta de IR baseada na carteira de identidade, carteira de motorista ou passaporte. A coleta da amostra biométrica é feita por meio de um *scanner* biométrico (normalmente dos dedos indicador e polegar das duas mãos). Para evitar duplicidade, as impressões digitais e fotos coletadas dos eleitores são comparadas, uma a uma, com as outras armazenadas no Cadastro Eleitoral por meio do Sistema Automatizado de Identificação Biométrica (ABIS) [Tribunal Superior Eleitoral, 2023]. Em seguida, os dados são processados e armazenados em um banco de dados da Justiça Eleitoral de forma segura e cifrada.

No dia da eleição, acontece a identificação do eleitor. Ao se apresentar à Mesa Receptora de Votos, o indivíduo apresenta novamente o documento com foto e insere

os dados biométricos para liberação da urna. Ao inserir a digital no leitor disposto no Terminal do Mesário, o sistema faz até quatro tentativas de reconhecimento das digitais. Durante o período de votação, a urna eletrônica não tem contato com nenhuma rede de computadores. Portanto, todas e somente as informações dos eleitores da seção em específico são inseridas previamente e contem um lacre, que quando removido a torna inutilizável [Tribunal Reginal Eleitoral, 2023]. Por se tratar de um processo de identificação, o sistema busca uma correspondência única da impressão digital apresentada pelo eleitor em relação a todas as impressões digitais cadastradas no banco de dados eleitoral, garantindo assim que apenas os eleitores autorizados votem e que não haja duplicação de votos.

6.3.5.2. Controle de acesso

O controle de acesso pode ser aplicado em diversos cenários e utilizando diversos tipos de biometria. Recentemente, divulgou-se o caso de uma torcedora de um time brasileiro que foi fatalmente atingida por estilhaços de uma garrafa de vidro antes de uma partida de futebol. O clube mandante da partida havia instalado um sistema de reconhecimento facial que auxiliou na identificação do responsável por atirar a garrafa.

Nessa aplicação, o cadastro é feito pelo site do clube no qual o usuário informa seus dados pessoais e, caso esteja acessando de um computador, recebe um *QR Code* para acesso a câmera do celular para cadastro da face. Caso já esteja no celular, a câmera apenas abre para o registro da biometria facial [Palmeiras, 2023].

A política de privacidade disponibilizada no site oficial do clube dispõe dos tratamentos realizados nos dados pessoais, como por exemplo garantir a criptografia, acesso somente a pessoas autorizadas, incluindo o torcedor titular, e a não divulgação dos mesmos exceto por determinação judicial [Palmeiras, 2021]. No dia do evento esportivo, os torcedores devem passar por catracas com câmeras que devem identificá-los. Uma luz verde é emitida caso a imagem facial captada seja a mesma do torcedor que adquiriu o ingresso para a respectiva seção. O site não disponibiliza detalhes do processamento de dados e do armazenamento realizado pela aplicação.

6.3.5.3. Autenticação por voz

A detecção por alto-falante ou *speaker*, pode ser utilizada em diversos serviços e aplicativos. A Instituição de Engenharia e Tecnologia e Wiley [Mtibaa et al., 2021] publicou uma abordagem em que essa tecnologia é aplicada de forma segura.

Na fase de inscrição dessa aplicação não é definido qual dispositivo realiza a captura, mas cita sua utilização em *smartphones*, que utilizam seu próprio microfone para essa função. No processamento de dados são utilizados recursos *Mel-frequency cepstral coefficients*s (MFCCs) para converter o sinal de áudio de fala em uma representação mais compacta e adequada para análise. Ainda nessa fase, um *token* é gerado e utilizado para embaralhar a amostra obtida antes do armazenamento. Na aplicação, o armazenamento é distribuído de acordo com o Modelo G da [ISO/IEC 24745, 2022], no qual os dados são

registrados parcialmente em um *token* (informado pelo usuário durante a fase de verificação) e em um servidor.

Durante a fase de verificação, a captura de dados é feita da mesma forma da inscrição e são aplicadas transformações para proteger a amostra obtida. O sistema faz a comparação baseado em cálculos da distância de Hamming e a decisão é tomada com base em um limite predefinido, ambas acontecem no lado do servidor, que nunca tem acesso a voz e a referência biométrica do titular. Assim, são garantidos os requisitos de segurança do ciclo de vida das informações no sistema. A irreversibilidade é alcançada por meio das transformações aplicadas que tornam as referências computacionalmente inviáveis de obterem a original. A pontuação obtida em um arcabouço que avalia a desvinculabilidade em sistemas de proteção de modelo biométrico aponta que a abordagem é totalmente desvinculada, pois não foi possível identificar se dois vetores protegidos inscritos em aplicações diferentes pertencem a mesma pessoa.

Existem muitos outros tipos de aplicações de sistemas biométricos que se adéquam conforme a necessidade de autenticar indivíduos. Cada uma destas aplicações têm processos e subprocessos bem definidos e comuns umas as outras, bem como requisitos para manter a privacidade em cada etapa do ciclo de vida das informações utilizadas.

6.4. Segurança de sistemas biométricos

A rápida evolução da tecnologia biométrica trouxe consigo um amplo espectro de aplicações, desde autenticação pessoal até controles de acesso. No entanto, essa crescente adoção também apresenta desafios cruciais, sendo a segurança um destes. À medida que sistemas biométricos utilizam dados brutos, como a captura de impressões digitais, a questão da proteção dos modelos biométricos torna-se crucial. Essa evolução não está isenta de desafios, especialmente quando se trata da segurança dos dados sensíveis envolvidos [Patel et al., 2015a].

Um dos riscos emergentes nos sistemas biométricos é o comprometimento dos modelos biométricos, no qual um modelo comprometido de um indivíduo pode ser usado indevidamente para acessar as informações de outro. Esse cenário é conhecido como correspondência cruzada entre bancos de dados. Nesse contexto, surge a necessidade premente de implementar requisitos rigorosos para garantir a confidencialidade, integridade e disponibilidade dos dados biométricos.

A fim de enfrentar esses desafios, a ISO/IEC 24745:2022 estabelece uma estrutura sólida de requisitos para sistemas biométricos. Esses requisitos são cruciais para mitigar os riscos associados ao comprometimento de modelos biométricos e garantir a operação segura e eficaz dos sistemas. No entanto, além das normas, a discussão sobre requisitos adicionais que vão além da segurança pura e simples surge como um tópico relevante, buscando uma proteção abrangente contra uma gama de ameaças potenciais e proporcionar uma proteção abrangente aos dados sensíveis.

Este trabalho também aborda a segurança durante a transmissão de dados biométricos, explorando estratégias para proteger a confidencialidade e a integridade dos dados à medida que são transferidos entre diferentes subsistemas. Este processo é particularmente crucial devido à crescente complexidade e variação geográfica desses sistemas.

Nesse contexto, busca-se explorar a importância dos requisitos em sistemas biométricos, considerando tanto as diretrizes estabelecidas por normas quanto os requisitos adicionais propostos por especialistas na área. Além disso, é examinado como esses requisitos se traduzem em medidas práticas para garantir a integridade dos modelos biométricos, prevenir o uso indevido e assegurar a proteção das informações dos indivíduos. Ao reunir as preocupações de segurança e as orientações normativas, visa-se fornecer uma compreensão abrangente dos elementos essenciais para construir sistemas biométricos robustos e seguros.

6.4.1. Requisitos de segurança dos sistemas biométricos

Para garantir a segurança adequada de sistemas biométricos a [ISO/IEC 24745, 2022] é bem direta e sugere o cumprimento de quatro requisitos básicos:

- Confidencialidade: refere-se à propriedade que protege as informações contra acesso ou divulgação não autorizados. Nos sistemas biométricos, as BRs são geralmente armazenadas em bancos de dados específicos. Durante as etapas de inscrição e identificação, por exemplo, os dados biométricos são transmitidos entre as entidades envolvidas para comparação. Caso haja uma interceptação, acesso por entidades não autorizadas ou mesmo vazamento do banco de dados, a identidade dos indivíduos não será revelada. Portanto, a confidencialidade deve assegurar que uma referência biométrica obtida fora do sistema, como resultado de um vazamento, não possa ser associada a nenhum indivíduo. Para garantir a confidencialidade dos dados biométricos armazenados, é necessário utilizar algoritmos de criptografia.
- Integridade: a integridade garante que um dado dentro do sistema biométrico é confiável. Esta propriedade é fundamental para o devido funcionamento do sistema. Está presente principalmente nas etapas de coleta dos dados. No processo de autenticação a integridade do processo está diretamente relacionada a confiabilidade da amostra, se uma BR não for confiável a autenticação também não será. As BRs não confiáveis podem ocorrer devido aos seguintes motivos:
  - Corrupção acidental devido a mau funcionamento de hardware ou software;
  - Modificação acidental ou intencional de um BR de "boa-fé"por uma entidade autorizada (i.e., inscrito não autorizado ou proprietário do sistema), sem intervenção de um invasor; e
  - Modificação (incluindo substituição) de um BR de um inscrito autorizado por um invasor. A integridade pode ser garantida utilizando técnicas de criptografia combinadas com outras técnicas, como o uso de *smart cards*, técnicas de código de autenticação de mensagem (MAC), marcação de tempo, etc.
- Renovabilidade e revogabilidade: As BRs desempenham um papel fundamental nos sistemas biométricos. Sua segurança e privacidade são de extrema preocupação, uma vez que uma variedade de ameaças pode comprometer uma amostra contendo uma BR. Em caso de vazamento de dados, as perdas podem ser catastróficas, uma vez que a biometria de um indivíduo ficará exposta. É importante ressaltar que a biometria é algo que não pode ser alterado, o que intensifica a necessidade de proteção contra acesso não autorizado e comprometimento de dados pessoais. Para enfrentar esse desafio, a revogação das referências biométricas comprometidas se mostra indispensável.

A renovabilidade é proporcionada pelo processo de diversificação, o qual consiste na criação de múltiplas referências biométricas independentes entre si, a partir de uma ou mais amostras biométricas transformadas de um mesmo indivíduo. Essa abordagem não apenas preserva a privacidade do indivíduo, mas também garante a segurança das informações do proprietário, ao criar representações únicas e não diretamente relacionadas entre si. Por outro lado, a revogação de uma BR envolve o cancelamento do modelo comprometido. Nesse processo, todas as permissões são revogadas e uma nova BR é emitida pelo sistema. O novo modelo apresenta uma representação biométrica totalmente distinta e sem qualquer relação com a anterior. É importante ressaltar que esse novo modelo não deve corresponder a nenhuma outra referência comprometida. A revogação pode ser acionada por diversas razões, como a ocorrência de violações de segurança ou a validade por um período de tempo específico. Em todos esses casos, a revogação preserva a segurança do sistema e a privacidade do indivíduo, evitando potenciais usos indevidos de modelos biométricos comprometidos.

- Disponibilidade: é a garantia de acesso contínuo a informações por pessoas autorizadas. As medidas de segurança, como defesa contra ataques *Distributed Denial-of-Service* / negação de serviço distribuída (DDoS) e *backups* regulares, são essenciais. Redundância de hardware também mantém a disponibilidade, permitindo troca imediata em caso de falha, assegurando acesso seguro e confiável às informações.

Esses requisitos desempenham um papel fundamental na gestão de dados sensíveis, incluindo os biométricos. No entanto, é importante notar que os elementos de confidencialidade, integridade e disponibilidade não se limitam apenas aos sistemas biométricos. Na verdade, esses são comuns em diversos tipos de sistemas que lidam com informações sensíveis. Portanto, a presença desses três requisitos não é exclusiva dos sistemas biométricos, mas sim uma característica compartilhada por muitos sistemas que tratam de dados delicados [Li and Jain, 2009]. Apesar da norma [ISO/IEC 24745, 2022] sugerir estes requisitos, outros autores como Nafea [Nafea et al., 2016] sugerem que um sistema biométrico com alto nível de proteção deve atender aos seguintes requisitos:

- Segurança: A privacidade e a integridade dos dados biométricos devem ser extremamente difíceis de serem violadas através de meios computacionais. Isso envolve a implementação de medidas de segurança robustas para proteger os dados biométricos contra acesso não autorizado e uso indevido.
- Diversidade: A diversidade implica que a probabilidade de correspondência cruzada entre diferentes bancos de dados biométricos seja minimizada. Ou seja, os dados de um indivíduo em um sistema biométrico não devem ser relacionados com os dados em outro sistema.
- Revogabilidade: A capacidade de revogar um modelo biométrico comprometido é essencial. Se um modelo biométrico for comprometido, deve ser possível cancelá-lo e gerar um novo modelo para substituí-lo.
- Desempenho: A implementação de medidas de proteção não deve prejudicar o desempenho do sistema biométrico como um todo. É importante que as medidas de segurança adicionais não tenham um impacto negativo nas operações do sistema, garantindo um funcionamento eficiente e eficaz.

6.4.2. Ameaças e contramedidas de segurança dos sistemas biométricos

A segurança em sistemas biométricos é de extrema importância, uma vez que lida com informações altamente sensíveis, ou seja, as características biométricas únicas dos indivíduos. Como em qualquer sistema, existem ameaças que podem comprometer o seu funcionamento, principalmente nos subsistemas responsáveis pela captura e transmissão dos dados biométricos. Além dos requisitos de privacidade no ciclo de vida (Subseção 6.3.3) também existem ameaças potenciais que devem ser evitadas entre os subsistemas. Embora seja impossível evitar completamente todas as ameaças, é fundamental adotar medidas rigorosas para mitigá-las e proteger a integridade e confidencialidade dessas informações sensíveis [ISO/IEC 24745, 2022].

Algumas das ameaças potenciais enfrentadas pelos sistemas biométricos incluem: (i) ataques de falsificação, no qual uma pessoa malintencionada tenta imitar a biometria de outro indivíduo para obter acesso indevido; (ii) ataques de *Denial-of-Service* / negação de serviço (DoS) , que visam sobrecarregar o sistema, impedindo-o de responder às solicitações legítimas de autenticação; e (iii) ataques de interceptação, nos quais os dados biométricos são capturados durante a transmissão e utilizados indevidamente. Analisando o ciclo de vida (Subseção 6.3.3) estas ameaças podem ocorrer nas etapas de coleta e transferência.

Para lidar com essas ameaças, é essencial implementar protocolos de segurança no *template* ou modelo biométrico [Jain and Kant, 2015]. Isso inclui o uso de algoritmos de criptografia para proteger os dados biométricos armazenados e transmitidos. Além disso, é fundamental ter mecanismos de detecção de ataques e tentativas de falsificação, como análise de padrões anômalos ou uso de características biométricas multifatoriais, que envolvem a utilização de mais de uma característica biométrica, como face e impressão digital ou impressão digital e escrita, por exemplo.

A necessidade de proteger os modelos biométricos tem se tornado cada vez mais evidente, o que implica evitar o armazenamento desses dados exatamente como são obtidos dos usuários. Uma estratégia recomendada para proteger o modelo biométrico é a utilização de dados auxiliares para transformar os dados biométricos de referência em um novo formato. Nesse sentido, [Kaur and Khanna, 2016] apontam alguns pontos cruciais a serem considerados:

- Falsificação de identidade/roubo de identidade: caso os dados biométricos de alguém sejam perdidos, estes correm o risco de serem explorados por indivíduos mal-intencionados para obter acesso não autorizado a contas e serviços, e.g., um invasor poderia extrair secretamente impressões digitais latentes de um usuário, possibilitando a reconstrução de sua identidade física ou digital.
- Sensibilidade: Além de serem utilizados para identificação única, os dados biométricos contêm informações pessoais e confidenciais, como histórico médico e condições físicas.
- Vinculabilidade (*Linkability*): O compartilhamento crescente de dados biométricos pode levar ao rastreamento e localização de usuários em diferentes bancos de dados, permitindo a correlação de seus perfis. Esse cruzamento de informações para determinar a relação entre os modelos de referência armazenados deve ser evitado.

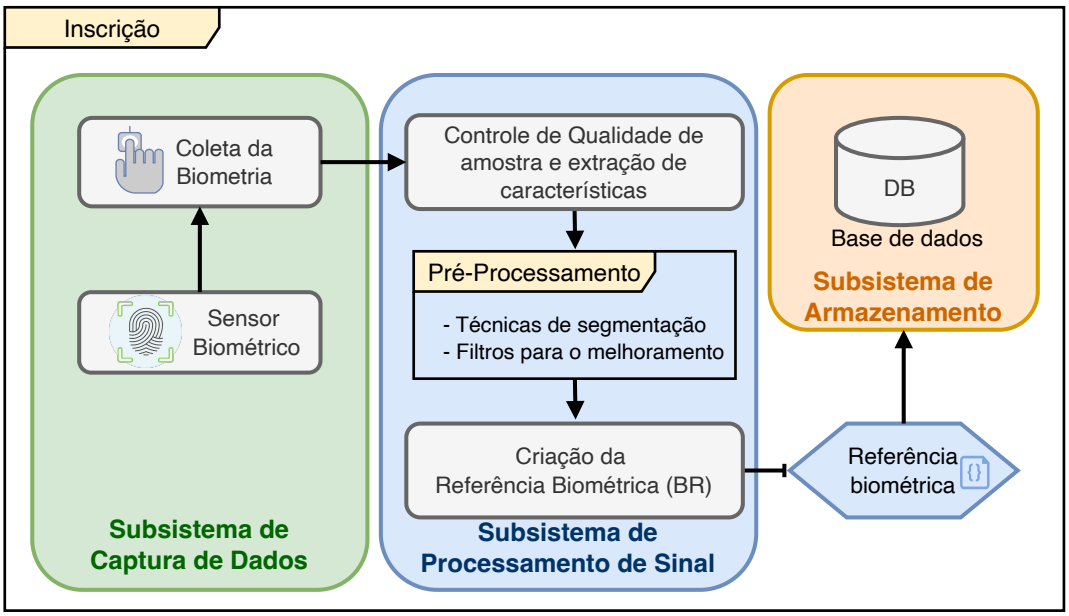
6.4.3. Funcionamento dos sistemas biométricos

Segundo [Jain and Kant, 2015] um sistema biométrico possui ao menos quatro partes: (i) sensor biométrico para a captura de dados; (ii) extrator de características; (iii) armazenamento em uma base de dados; e (iv) comparação para os processos de verificação e identificação. Já para o processo de inscrição envolve a captura de dados, extração de características e armazenamento.

6.4.3.1. Processo de Inscrição

O processo de inscrição de um novo usuário é a etapa inicial, no qual os dados biométricos do individuo são coletados e armazenados para possibilitar identificação futura no sistema. Este processo engloba o uso de três subsistemas: Captura de dados, Processamento de sinal e Armazenamento. O modelo representa o ciclo desta etapa pode ser visualizado na Figura 6.10.

Figura 6.10: Processo de inscrição de um sistema biométrico.



Para dar início a coleta dos dados biométricos pelo Subsistema de Captura de Dados (Figura 6.10), utiliza-se um sensor ou leitor biométrico para capturar as características únicas do indivíduo e após a coleta é enviado para o subsistema de processamento de sinal. Esse primeiro passo é essencial para obter os dados necessários para a identificação biométrica. Em seguida, os dados coletados passam por um processo de pré-processamento no subsistema de processamento de sinal, no qual a qualidade da amostra é avaliada. Essa qualidade está diretamente relacionada à eficiência e precisão do leitor biométrico utilizado e à forma como o dado biométrico foi fornecido pelo indivíduo. Caso a qualidade da amostra seja baixa, existem técnicas disponíveis para melhorá-la, e.g., podem ser aplicadas técnicas de segmentação ou filtros para aprimorar os pontos com ruídos, garantindo que a informação biométrica seja mais clara e precisa [Faria, 2014, Jain et al., 1999].

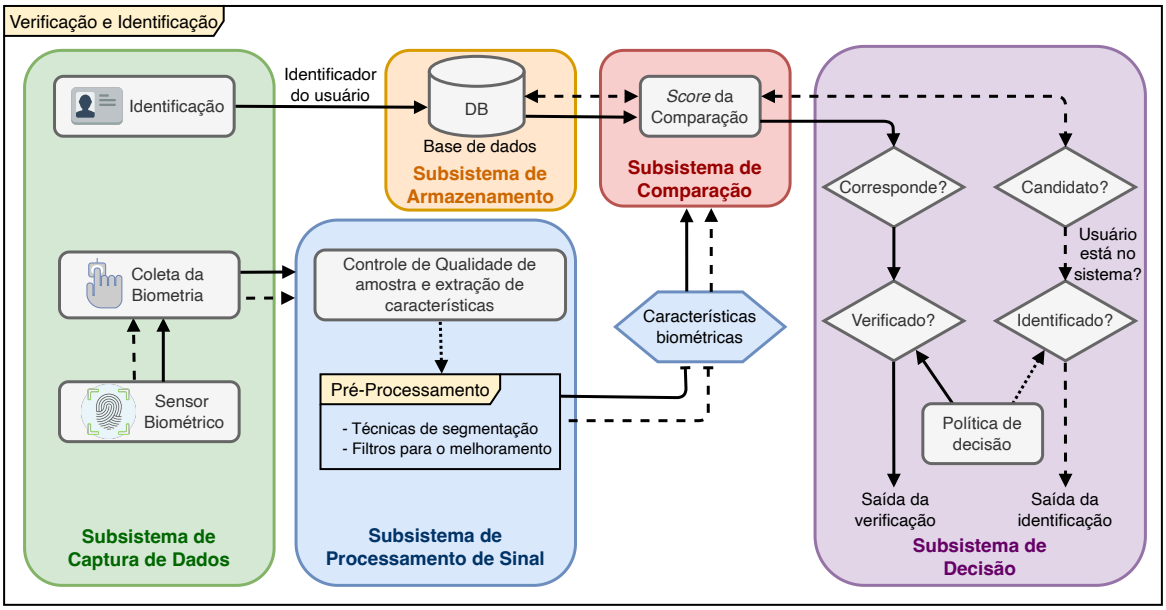
Após o pré-processamento, é gerada uma representação digital dos dados biométricos capturados, a referência biométrica BR. Esta referência é uma forma compacta

de armazenar as informações biométricas, e será usada para comparação e correspondência durante os processos de autenticação e identificação. Como pode-se verificar na Figura 6.10 no Subsistema de Processamento de Sinal nos módulos de controle de qualidade de amostra, pré-processamento e criação de referência biométrica (BR). A última etapa do processo de inscrição ocorre no Subsistema de Armazenamento como visto na Figura 6.10. Após a coleta e o tratamento da amostra biométrica, é crucial proceder ao armazenamento seguro das informações em um banco de dados que é realizado no Subsistema de Armazenamento. Nessa etapa, são aplicadas técnicas avançadas de proteção dos dados, como *fuzzy vault* [Juels and Sudan, 2006] que permite armazenar informações biométricas em um "cofre"seguro, o algoritmo transforma a informação biométrica em um conjunto de pontos em um espaço tridimensional que durante a comparação busca os pontos que estão mais próximos da chave criptográfica, permitindo que o usuário seja autenticado, isto garante que as características únicas dos indivíduos sejam preservadas com total segurança [Faria, 2014]. Essas medidas visam proteger os dados biométricos dos usuários contra acessos não autorizados, garantindo a confidencialidade e a integridade das informações [Faria, 2014].

6.4.3.2. Processo de Verificação e Identificação

Os sistemas biométricos são empregados na autenticação de indivíduos, sendo as operações principais são executadas de verificação e identificação [Bolle et al., 2013]. O processo de identificação e verificação são duas etapas cruciais em sistemas biométricos. A identificação busca encontrar uma pessoa em um banco de dados comparando suas características únicas com várias entradas. Já a verificação compara a amostra de uma pessoa com um modelo previamente fornecido para confirmar sua identidade. A Figura 6.11 ilustra a interação dos módulos envolvidos.

Figura 6.11: Funcionamento da verificação e identificação de um sistema biométrico.



A verificação é um processo em que o usuário apresenta uma credencial, como um código de identificação ou um cartão de acesso, juntamente com uma amostra biométrica, como uma impressão digital ou uma varredura facial. Essa etapa ocorre no Subsistema de Captura de Dados. A amostra coletada é então direcionada ao Subsistema de Processamento de Sinal para tratamento e controle de qualidade. A Figura 6.11 ilustra a interação nos Subsistemas de Captura de Dados e Processamento de Sinal. Após o tratamento, a amostra é encaminhada ao Subsistema de Comparação, o qual confronta a amostra com os dados biométricos armazenados no banco de dados, gerando uma pontuação de similaridade. Essa pontuação representa o grau de semelhança entre a amostra fornecida e a amostra armazenada. O processo de comparação segue o modelo 1:1, também conhecido como busca fechada, uma vez que o sistema procura um único registro específico para verificar a autenticidade do usuário que forneceu a amostra biométrica. O Subsistema de Decisão avalia se o usuário está presente na base de dados e se corresponde a um candidato válido. Com base nas políticas de decisão, o sistema produz a saída correspondente. Por outro lado, a identificação é um processo mais abrangente, em que o usuário fornece apenas uma amostra biométrica sem a necessidade de apresentar uma credencial. Após ser processada pelo Subsistema de Processamento de Sinal, a amostra é utilizada para realizar uma busca no banco de dados contendo múltiplos registros biométricos. Essa busca segue o modelo 1:N, ou busca aberta, no qual o sistema explora várias entradas para identificar o usuário desconhecido [Costa et al., 2006, ISO/IEC 24745, 2022]. O sistema compara a amostra com os registros biométricos existentes, gerando uma pontuação de comparação. Se uma correspondência suficientemente próxima for encontrada, o sistema identifica a presença do usuário no sistema. Com base nas políticas de decisão, o sistema retorna o resultado da verificação [Costa et al., 2006, ISO/IEC 24745, 2022]. A Figura 6.11 indica essa constante troca de informações entre os Subsistemas de Armazenamento e Comparação para executar a Identificação.

6.4.4. Principais Ameaças e Contramedidas

A análise e compreensão das principais ameaças e a implementação de contramedidas eficazes são elementos essenciais para a garantia da segurança de sistemas biométricos que lidam com informações sensíveis. Neste contexto, explorar as principais ameaças e estratégias para mitigá-las desempenha um papel crítico na sua proteção à privacidade. Os ataques podem ocorrer em várias áreas do sistema. A [ISO/IEC 24745, 2022] classifica esses ataques de acordo com os diferentes subsistemas ou etapas do sistema nos quais estão suscetíveis a ocorrer. Nesse sentido, é importante destacar os principais ataques identificados em nossa pesquisa. A análise da literatura pesquisada revelou que os subsistemas mais impactados por ataques foram a Captura de Dados e o Armazenamento. Isso evidencia uma preocupação mais acentuada dos pesquisadores com esses módulos específicos.

A segurança dos sistemas biométricos é fundamental na era da tecnologia digital, mas enfrenta desafios devido a uma variedade de ataques. Esses ataques buscam explorar vulnerabilidades nos processos de autenticação e identificação, comprometendo a integridade e a privacidade dos dados biométricos. Desde ataques de força bruta até manipulação de dados durante o processamento, a compreensão dessas ameaças é vital para desenvolver estratégias eficazes de proteção. Nesta perspectiva, são descritos os principais ataques

que os sistemas biométricos enfrentam e como funcionam:

- Ataque de Força Bruta: é um ataque no qual o invasor tenta entrar no sistema com diversas entradas de dados possíveis. Em sistemas biométricos pode ser aplicado inserindo sequências de características biométricas até que alguma consiga adentrar no sistema.
- Pessoa no Meio (MitM): ocorre quando um invasor intercepta os dados transmitidos em um meio de comunicação, podendo visualizar, modificar ou roubar os dados.
- Correspondência Cruzada (CrossRef): o ataque de correspondência cruzada ocorre em sistemas biométricos quando um adversário tenta identificar se um mesmo indivíduo está inscrito em dois ou mais sistemas independentes. Nesse tipo de ataque, o adversário tem acesso a modelos protegidos (*templates*) de diferentes sistemas e tenta verificar se esses modelos pertencem ao mesmo indivíduo, comprometendo assim a privacidade e segurança dos dados biométricos [Kelkboom et al., 2011].
- Ataque de Apresentação (Falsificação de Sensor): também conhecidos como ataques de *spoofing* ou apresentação falsa. Nesse tipo de ataque, um elemento biométrico falso, como um dedo sintético ou uma representação facial artificial, é exibido ao sensor por um indivíduo não autorizado, visando contornar os sistemas de identificação. Além disso, o agente mal-intencionado pode causar danos físicos ao sistema de reconhecimento ou inundá-lo com solicitações de acesso fraudulentas. É muito vulnerável já que o atacante não necessita de um conhecimento prévio para executar [Jain and Kant, 2015].
- Captura/reprodução de Sinais: Ao coletar dados biométricos em estado bruto, o sensor os encaminha por um canal de comunicação ao módulo de extração de recursos para pré-processamento. Este canal, localizado entre o sensor e o módulo de processamento de sinal, corre o risco de ser interceptado, permitindo o roubo e armazenamento dos dados biométricos. Posteriormente, esses dados armazenados podem ser reproduzidos no módulo de extração de recursos para burlar o sensor [Jain and Kant, 2015].
- Manipulação não autorizada de dados durante o processamento: Após coletar os dados biométricos brutos, o sensor os encaminha ao módulo de extração de características. No entanto, um invasor pode manipular esse processo, forçando o módulo de extração a gerar valores de características selecionados pelo intruso, em vez de utilizar os valores de características originados dos dados autênticos coletados pelo sensor [Jain and Kant, 2015].
- Manipulação das pontuações de comparação: O sensor é alvo de um ataque visando gerar a pontuação mais elevada escolhida pelo impostor, a fim de contornar o sistema de autenticação biométrica, independentemente dos valores provenientes do conjunto original de recursos de entrada [ISO/IEC 24745, 2022].
- Banco de dados comprometido: Isso acontece quando um impostor compromete a segurança do banco de dados ao inserir novos modelos, alterar modelos existentes ou remover modelos já presentes.
- Ataque de escalada: Os ataques de escalada envolvem um aplicativo que encaminha modelos de minúcias criados de forma sintética para o mecanismo de comparação. Com base na pontuação resultante da comparação, o aplicativo altera os modelos de maneira aleatória repetidamente até que o limite de decisão seja ultrapassado [Martinez-Diaz et al., 2006].

- DDoS: Um atacante emprega múltiplas máquinas para conduzir o ataque. Essas máquinas, conhecidas como *bots*, são selecionadas devido à sua vulnerabilidade. O atacante instala software nas máquinas *bots*, visando executar um ataque DDoS contra o alvo, resultando na indisponibilidade desejada desse alvo [Conrads, 2019].
- Manipulação de *Threshold*: Um intruso pode manipular o resultado informado pelo módulo de comparação. Nesse tipo de ataque, o impostor tem a capacidade de modificar a pontuação de correspondência que é enviada pelo canal de comunicação entre o módulo de comparação e o dispositivo de aplicação. Isso é feito com o objetivo de alterar a pontuação original e, por consequência, a decisão original (aceitar ou rejeitar) do módulo de comparação [Jain and Kant, 2015].

A Tabela 6.2 apresenta as principais ameaças que estão sujeitos os subsistemas, seguido das contramedidas para mitigá-las. Os conceitos de referência biométrica (BR) e referência de identidade (IR) estão na seção 6.3.1, os conceitos de referências biométricas renováveis (RBR) estão na seção 6.4.6 e podem ser consultados para esclarecer o conteúdo da tabela.

Tabela 6.2: Diferentes subsistemas biométricos seguidos das principais ameaças que podem sofrer e contramedidas que devem ser aplicadas.

Ameaças nos Subsistemas Biométricos		
Subsistemas	Ameaças	Contramedidas
Captura de Dados	Falsificação de sensor Captura/reprodução de sinais do sensor	Detecção de ataque de apresentação. Biometria multimodal Desafio/Resposta Dispositivo de captura de criptografia de hardware
Processamento de Sinal	Manipulação não autorizada de dados durante o processamento	Uso de algoritmo confiável
Comparação	Manipulação das pontuações de comparação	Servidor e/ou cliente seguro OCC confiável
Armazenamento	Banco de dados comprometido Substituição não autorizada de BR/IR Modificação não autorizada de BR/IR Exclusão não autorizada de BR/IR Ataque DDoS	Referências biométricas revogáveis e renováveis Separação dos Dados Controle de acesso ao banco de dados Sinal BR/RBR/IR Cifrar BR/RBR/IR Planejamento de contingência adequado e procedimentos de recuperação
Decisão	Ataque de escalada DDoS Manipulação de *Threshold*	Canal seguro Ocultar score de comparação Rede/Canal seguro Controle de acesso a configuração de *Threshold* Proteção do valor de *Threshold*

De acordo com que recomenda a [ISO/IEC 24745, 2022], a principal medida para

prevenir ataques é empregar um algoritmo confiável para cifrar os dados biométricos. Os recursos de modelos biométricos renováveis reforçam ainda mais a segurança. Além disso, existem outras contramedidas que podem ser adotadas:

- Detecção de Ataques de Apresentação: Uma contra medida para este ataque é a detecção dos sinais vitais, como pressão sanguínea e temperatura.
- Biometria Multimodal: envolve o uso de mais de um tipo de traço biométrico, como impressão digital e reconhecimento facial, para autenticar a identidade de um indivíduo. Isso aumenta a segurança e a confiabilidade do processo de autenticação.
- Desafio-resposta: é um método de autenticação que envolve o sistema emissor apresentando um desafio ao usuário, que deve responder corretamente para comprovar sua identidade. A resposta é geralmente baseada em um conhecimento prévio compartilhado entre o sistema e o usuário.
- Uso de algoritmo confiável: é fundamental para assegurar a segurança, pois esses algoritmos passam por testes rigorosos e suas vulnerabilidades são identificadas e abordadas. Isso contribui para uma proteção mais eficaz contra ameaças.
- Servidor e/ou cliente seguro: é essencial para proteger os dados e as comunicações. Isso envolve medidas como autenticação forte, criptografia e prevenção contra acessos não autorizados, garantindo a integridade e confidencialidade dos dados.
- Separação dos Dados: consiste em isolar informações em compartimentos diferentes para garantir a segurança. Isso ajuda a evitar vazamentos acidentais ou intencionais, minimizando o risco de que um acesso não autorizado a um conjunto de dados comprometa outros.
- Ocultar score de comparação: envolve não revelar a pontuação resultante da comparação biométrica para proteger a segurança e privacidade, dificultando ataques baseados na análise dessas pontuações.
- Controle de acesso a configuração de *threshold*: refere-se a gerenciar com rigor o acesso e ajuste do limiar de comparação utilizado em sistemas biométricos. O *threshold* determina a aceitação ou rejeição de uma correspondência biométrica com base na pontuação resultante da comparação de características.

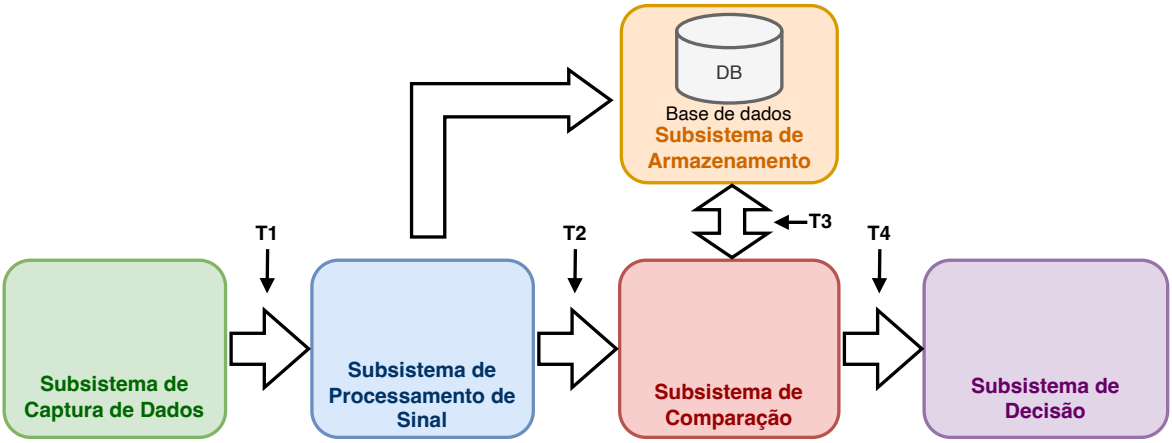
6.4.5. Ataques Durante a Transmissão

Nas fases que englobam a transferência de dados biométricos entre diferentes subsistemas, existe um considerável risco para a integridade desses dados, especialmente em sistemas distribuídos. Isso ocorre porque os dados são transferidos entre várias partes do sistema, as quais estão localizadas geograficamente em diferentes locais devido à natureza distribuída do sistema.

Como descreve [Faria, 2014], um sistema biométrico é constituído de uma infraestrutura composta de vários módulos de software e hardware interligados, envolvendo captura, extração de características, banco de dados de amostras, combinador e módulo de decisão, é entendível que possa haver brechas em pontos desta infraestrutura que comprometem a sua segurança. Estas brechas podem ser citadas como sabotagem e sobrecarga, na qual a sabotagem se refere a danos físicos cometidos contra os componentes da infraestrutura.

A Figura 6.12 exemplifica os principais pontos da Tabela 6.2 e indica os dados transmitidos, principais ameaças / contramedidas, bem como a transmissão dos dados entre os subsistemas.

Figura 6.12: Principais pontos e momentos de ataques ao subsistemas.



Diversos ataques podem ocorrer nos meios de transmissão entre os subsistemas. Seus principais alvos são identificados na Figura 6.12, sendo representadas pelos itens: T1 - Subsistema de Captura de Dados e Processamento de Sinal; T2 - Subsistemas de Processamento de Sinal e Comparação; T3 - Subsistema de Comparação e Armazenamento; e T4 - Subsistema de Comparação e o de Decisão. A Tabela 6.3 descreve os dados transmitidos, principais ameaças e contramedidas.

Tabela 6.3: Ameças e contramedidas entre os subsistemas.

Ameaças entre os subsistemas			
Canais de Comunicação	Dados	Ameaças	Contramedidas
Captura de Dados - Processamento de sinal (T1) Processamento de Sinal e Comparação (T2)	Amostra biométrica e traço	Espionagem *Replay* Força Bruta	Canal seguro/Cifrado. Desafio/Resposta. Política de tempo Limite.
Armazenamento - Comparação (T3)	Referência Biométrica	Espionagem *Replay* Pessoa no meio/MitM Escalada	Canal seguro/Cifrado. Desafio/Resposta. Canal seguro/Cifrado. Verificação de integridade dos dados biométricos com assinatura digital ou MAC. Canal seguro.
Comparação - Decisão (T4)	*Score* de comparação	Manipulação de pontuação de comparação	Canal seguro.

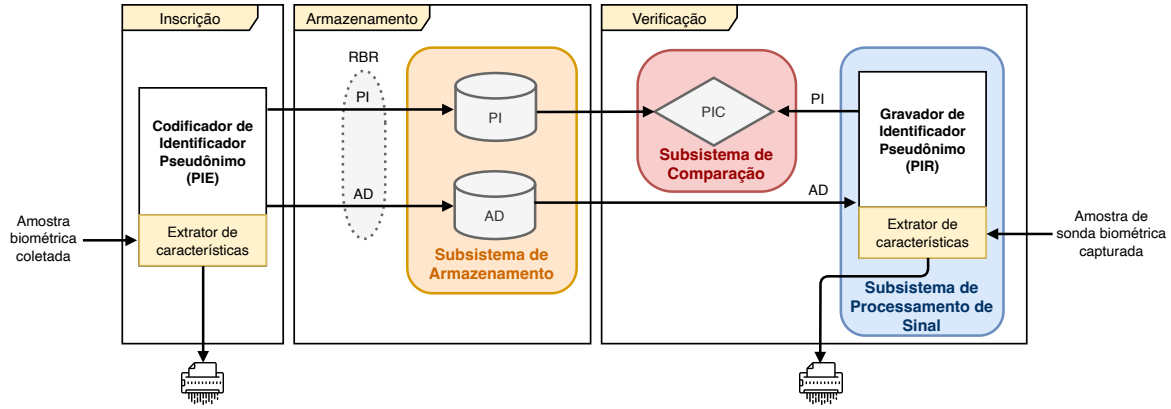
É inegável que todos os subsistemas possuem ameaças potenciais. No entanto, o essencial na segurança reside na capacidade de antecipar essas ameaças e concentrar

esforços na proteção dos pontos mais suscetíveis aos ataques. Deste modo, a Tabela 6.3 informa os canais de comunicação que podem apresentar riscos, bem como os dados que podem comprometidos, seguido das possíveis ameaças e contramedidas que podem ser tomadas para mitigá-las.

6.4.6. Tipos de referências biométricas renováveis e revogáveis

A biometria renovável ocorre com a aplicação dos conceitos de renovabilidade e revogabilidade nas referências biométricas, este conceito é chamado de referências biométricas renováveis (RBR). O processo de criação de RBRs envolve o processo de diversificação que geram múltiplas referências irreversíveis e não vinculáveis a partir das mesmas características biométricas que podem ser usadas para renovar um RBR ou fornecer referências independentes em diferentes aplicações. Seu uso é uma importante contramedida contra ataques que possam comprometer a identidade de um indivíduo no sistema. Essa prática pode ser vista na aplicação de técnicas como a biometria cancelável [ISO/IEC 24745, 2022]. A arquitetura para a criação de uma RBR é ilustrada na Figura 6.13.

Figura 6.13: Tipos de referências biométricas renováveis e revogáveis.



A biometria renovável ocorre basicamente com a aplicação dos conceitos de renovabilidade e revogabilidade nas referências biométricas. Sua utilização é uma importante contramedida a ataques que possam comprometer a identidade de um indivíduo dentro do sistema. As RBRs consistem em dois elementos de dados essenciais: um Identificador Pseudônimo (PI) e os Dados de Recursos Biométricos Correspondentes (AD). Ambos são gerados durante o processo de inscrição e são armazenados porque são necessários para realizar processos de verificação ou identificação biométrica. Caso seja necessária a renovação de uma RBR para um determinado indivíduo, um novo PI correspondente será gerado, acompanhado por um novo conjunto de AD e uma amostra biométrica recentemente capturada, usando o mesmo processo de criação de RBR. As novas informações de PI e AD são então entregues ao sujeito ou prestador de serviços responsável, enquanto as informações de PI e AD anteriores são revogadas e descartadas. Durante a inscrição, um estágio de extração de recursos biométricos é conduzido para gerar dados a partir da amostra biométrica coletada. Em seguida, um codificador de identificador pseudônimo (PIE) é empregado para criar a RBR, composta pelo PI e AD correspondentes [ISO/IEC 24745, 2022].

Uma vez que a RBR é criada, a amostra biométrica original e os dados de recursos

extraídos podem ser devidamente descartados, proporcionando maior segurança e privacidade. O RBR resultante é armazenado em um meio adequado de armazenamento, como um cartão inteligente ou um banco de dados eletrônico. Vale ressaltar que o PI e AD podem ser separados fisicamente ou logicamente, reforçando ainda mais a proteção dos dados biométricos e a confidencialidade do processo [ISO/IEC 24745, 2022].

De acordo com [Kaur and Khanna, 2016] um esquema biométrico cancelável deve cumprir quatro objetivos:

1. Diversidade: muitos modelos canceláveis podem ser gerados a partir da mesma biometria para aplicações diferentes.

2. Renovabilidade/ Revogabilidade: revogação direta e reemissão caso o modelo esteja comprometido.

3. Não inversibilidade: Para evitar a falsificação, não deve ser possível obter informações sobre recurso biométrico original do modelo transformado.

4. Desempenho: O reconhecimento realizado com o modelo transformado não deve prejudicar o desempenho do sistema. Este também deve possuir compatibilidade com versões anteriores.

6.4.7. Biometria Cancelável

A biometria cancelável visa garantir a segurança e a privacidade dos modelos biométricos. Essa técnica consiste em aplicar distorções intencionais e repetíveis nos sinais biométricos, com base em transformações específicas que possibilitam a comparação dos modelos biométricos no domínio transformado. Essa transformação torna os modelos biométricos permanentemente protegidos, tornando difícil ou impossível a recuperação dos dados biométricos brutos a partir dos modelos. Em vez de armazenar os dados biométricos originais, a biometria cancelável mantém os dados do modelo transformado, garantindo assim a confidencialidade dos usuários. A obtenção dos modelos transformados é realizada através da aplicação de uma função de transformação não reversível especialmente projetada para esse propósito.

Essa transformação pode ser aplicada tanto no domínio original quanto no domínio do recurso biométrico. Um dos principais benefícios é a sua capacidade de fornecer revogabilidade. Isso significa que, caso a biometria de um indivíduo seja comprometida, esta pode ser facilmente recriada utilizando uma nova transformação. Dessa forma, a segurança do sistema é mantida mesmo em casos de comprometimento. Outro benefício importante é a preservação da privacidade. A transformação torna computacionalmente difícil recuperar a biometria original a partir dos dados transformados, garantindo a confidencialidade dos dados biométricos dos usuários [ISO/IEC 24745, 2022] [Gomez-Barrero et al., 2016].

Essa abordagem proporciona uma camada adicional de segurança, pois mesmo que os modelos biométricos sejam comprometidos, não será possível reconstruir as informações biométricas originais, protegendo assim a identidade dos usuários e preservando sua privacidade [Yang et al., 2022].

Deste modo, a biometria cancelável evita problemas de correspondência cruzada entre diferentes bancos de dados. Cada aplicativo utiliza uma transformação diferente, impedindo a comparação direta entre as informações armazenadas em diferentes sistemas. Além disso, essa abordagem não degrada a precisão dos algoritmos de correspondência. As características estatísticas dos recursos biométricos são aproximadamente mantidas após a transformação, o que possibilita o uso de algoritmos de correspondência existentes. Comparada à criptografia biométrica, na qual uma chave criptográfica é combinada com o modelo biométrico, a biometria cancelável oferece uma camada adicional de segurança. Enquanto a criptografia biométrica depende totalmente do sigilo da chave secreta, a biometria cancelável garante a desvinculação e a capacidade de revogar ou renovar os modelos transformados [Kaur and Khanna, 2016] [Patel et al., 2015b].

6.4.8. Segurança dos registros de dados biométricos

Os sistemas biométricos precisam armazenar dois tipos de referências: referências biométricas ou biométricas renováveis [ISO/IEC 24745, 2022] (BR ou RBR, geradas pelos sistemas de captura de biometria) e referências de identidade (IR, nomes de usuários, números únicos ou qualquer dado que identifique unicamente um indivíduo dentro do sistema). Com estas identidades, o sistema pode associar uma ou mais biometria (BR) a um indivíduo. Além disso, pode-se armazenar estas referências (IR e BR) na mesma base de dados ou em base de dados separadas.

6.4.8.1. Base de dados única

A forma como as referências BR e IR são armazenadas permite determinar quais requisitos de segurança o sistema atende. É possível armazenar uma referência de quatro maneiras:

- Inalterada: não oferece nenhum tipo de segurança, nem integridade, nem confidencialidade são satisfeitos.
- Cifrada: pode-se cifrar uma única referência (BR ou IR) ou ambas. Desta forma, a referência cifrada satisfaz confidencialidade e, dependendo da cifra utilizada, integridade fraca.
- Autenticada: autenticando uma ou ambas as referências, é garantido que o sistema satisfaz o requisito de integridade para a referência autenticada.
- Autenticada-cifrada: autenticando e cifrando alguma das referências, garante a integridade e confidencialidade para a referência em questão.

É possível também empregar esses métodos de armazenamento em RBR, fazendo isso, são satisfeitos tanto os requisitos supralistados quanto o requisito de *renewability/-revocability*. Com estas possibilidades, são possíveis dezenove tipos de cenários, que atendem total ou parcialmente os requisitos de segurança descritos (Subseção 6.4.1). A Tabela 6.4 delinea os cenários possíveis para proteção das BRs ou IRs. Cada um destes cenários garante diferentes requisitos de segurança para determinadas referências.

Tabela 6.4: Possíveis cenários de proteção das identidades e os requisitos de segurança satisfeitos.

Cenário	Confidencialidade		Integridade		*Renewability*	Irreversibilidade	Confidencialidade	Armazenamento da IR	Armazenamento da BR
	IR	BR	IR	BR					
1								IR	nda
2		X		0		0		IR	enc
3				X				IR	aut
4		X		X		0		IR	aut-enc
5	X		0					enc IR	nda
6			X					aut IR	nda BR
7	X		X					aut-enc IR	nda BR
8	X	X	0	0		0	0	enc IR	enc BR
9			X	X				aut IR	aut BR
10	X	X	X	X		0	0	aut-enc IR	aut-enc BR
11	X	X	0	X		0	0	enc IR	aut-enc BR
12		X	X	X		0		aut IR	aut-enc BR
13	X	X	X	0		0	0	aut-enc IR	enc BR
14	X		X	X				aut-enc IR	aut BR
15					X	X		IR	RBR
16		0	X	X	X	X		aut IR	aut div. RBR
17	X	X	X	X	X	X	0	aut-enc IR	aut-enc div. RBR
18	X	X	0	0	X	X	0	enc IR	enc RBR
19		X	X	X	X	X		aut IR	aut-enc, div. RBR

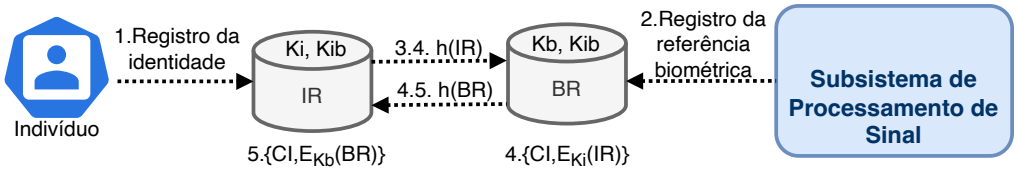
enc - cifrado
aut - autenticado
aut-enc - autenticado e cifrado
X - satisfeito
0 - parcialmente satisfeito

6.4.8.2. Caso de bases de dados separadas

Utilizar bases de dados separadas para o armazenamento da BR e IR é, em geral, considerado uma boa prática de segurança [ISO/IEC 24745, 2022]. Separando estes dados, o vazamento dos dados de um servidor não necessariamente compromete a identidade e acesso à conta de um usuário. No entanto, para que esta separação seja o mais efetiva possível, é ideal que a separação seja tanto física (servidores diferentes em localizações diferentes) e de operadores (as bases devem ser administradas por operadores diferentes com chaves criptográficas diferentes para qualquer proteção que esteja sendo utilizada). Com isto, para se utilizar bases separadas para o armazenamento das referências, surgem alguns problemas relacionados à segurança da comunicação entre estas bases. É preciso definir como conectar a BR e IR (ou PI e AD em caso de uma RBR) para autenticação, e também, como fazer isso de forma segura. Em geral, esta funcionalidade é provida por um *identificador comum* (CI).

O CI deve ser formado a partir das duas (ou mais) referências sendo conectadas para que todas as bases sejam capazes de identificar e autenticar o usuário. Uma forma de implementar um CI é utilizando algum tipo de MAC formado por ambas as referências para formar um único identificador CI. Um diagrama de uma implementação similar ([ISO/IEC 24745, 2022]) é apresentado na Figura 6.14.

Figura 6.14: Exemplo de possível implementação de esquema de proteção utilizando bases de dados separadas para IR e BR.



O diagrama na Figura 6.14 mostra o fluxo das referências entre bases, especificamente para o registro de uma BR. Este tráfego garante que tanto a base com IR quanto com a BR tenham o CI e possam autenticar o usuário caso necessário. De forma similar ao uso de somente uma base de dados, a forma de armazenamento do das referências determinam quais requisitos de segurança serão satisfeitos pelo sistema. A Tabela 6.5[1] A seguir, assim como na Tabela 6.4.

Tabela 6.5: Possíveis cenários de proteção das identidades e os requisitos de segurança satisfeitos por cada um em um cenário de armazenamento das referências em diferentes bases de dados. Fonte: Adaptado de [ISO/IEC 24745, 2022].

Cenário	Confidencialidade		Integridade		*Renewability BR*	Irreversibilidade	Confidencialidade	Armazenamento da IR	Armazenamento da BR
	IR	BR	IR	BR					
1								IR	nda
2		X		0		0		IR	enc
3				X				IR	aut
4		X		X		0		IR	aut-enc
5	X		0					enc IR	nda
6			X					aut IR	nda BR
7	X		X					aut-enc IR	nda BR
8	X	X	0	0		0	0	enc IR	enc BR
9			X	X				aut IR	aut BR
10	X	X	X	X		0	0	aut-enc IR	aut-enc BR
11	X	X	0	X		0	0	enc IR	aut-enc BR
12		X	X	X		0		aut IR	aut-enc BR
13	X	X	X	0		0	0	aut-enc IR	enc BR
14	X		X	X				aut-enc IR	aut BR
15		0			X	X		IR, PI	AD
16		0	X	X	X	X		aut IR, aut PI	aut AD
17	X	X	X	X	X	X	0	aut-enc IR, aut-enc PI	aut-enc AD
18	X	X	0	0	X	X	0	enc IR, enc PI	enc AD
19	X	X	X	X	X	X		aut IR, aut-enc PI	aut-enc AD

enc - cifrado
aut - autenticado
aut-enc - autenticado e cifrado
X - satisfeito
0 - parcialmente satisfeito

6.4.9. Mecanismos para proteção dos dados biométricos

Nesta seção são apresentados os principais mecanismos que buscam efetuar a proteção dos dados biométricos. Estes mecanismos apresentam diversas abordagens que buscam esta proteção, as duas principais técnicas são os criptossistemas biométricos e a biometria cancelável.

6.4.9.1. Criptossistemas biométricos

Uma das principais maneiras de proteger *templates* biométricos é com técnicas denominadas Criptossistemas Biométricos [Rathgeb and Uhl, 2011]. Estes sistemas funcionam de forma similar a sistemas de criptografia tradicionais, mas precisam acomodar algumas características específicas a dados biométricos, e.g., a imprecisão que a leitura de dados biométricos apresenta, pela própria natureza não-fixa dos dados.

Fuzzy Commitment Esta técnica se baseia na ideia de *"fuzzy logic"* [Zadeh, 1965]. Em oposição a sistemas de criptografia comuns, que requerem uma chave precisa para de-

[1] Em todos os casos, o CI é armazenado em ambas as bases, juntamente com o IR (ou PI) e BR (ou AD).

cifrar os dados cifrados, os sistemas "*fuzzy*"aceitam chaves suficientemente similares à original [Juels and Wattenberg, 1999]. Por exemplo, cifrando algum dado D com a chave "SBSEG", em um sistema tradicional só se consegue decifrar este dado com a chave exata. Porém, um sistema criptográfico *fuzzy* teórico pode aceitar até uma letra errada, como na chave "SBSIG". Esta tolerância, em primeira análise, parece introduzir uma falha de segurança. Contudo, como sistemas biométricos sempre apresentam ruído na leitura da biometria, a tolerância do *fuzzy commitment* permite consistentemente identificar um usuário, mesmo que o leitor não apresente a biometria idêntica à apresentada no cadastro. Para se obter este comportamento, [Juels and Wattenberg, 1999] utilizam códigos de correção de erros (ECC). Ao salvar uma BR na fase de cadastro, é criado também uma chave K que será combinada com BR (com, por exemplo, uma função de *hash*) criando *helper data* (HD)). Com HD, ao tentar se autenticar, tem-se uma BR' "corrompida" pelo ruído introduzido na coleta. Agora, usando um ECC, é tentado recuperar a chave K original utilizando HD e BR'. Se o ruído não for superior à capacidade de correção do ECC, será obtida a chave original K e autenticando o usuário com esta. Caso fosse apresentado uma biometria completamente diferente ou extremamente distorcida, não se tem a chave exata K ao final. Dessa forma, é recusada a autenticação do usuário.

Fuzzy Vault Com o *Fuzzy Vault* [Juels and Sudan, 2006], se pretende obter a mesma "*fuzzynes*"que o *fuzzy commitment*. Para isso, é criado a ideia de um *vault* ou "cofre"que armazena as minúcias da biometria armazenada. É construído o cofre primeiro, projetando as minúcias coletadas em um polinômio escolhido aleatoriamente a adicionando pontos "falsos"ao cofre para que as minúcias não possam ser recuperadas, somente com acesso ao cofre. Com estes dados armazenados, ao realizar autenticação, é disposto um conjunto de minúcias que, caso se sobreponham o suficiente com as minúcias cadastradas, serão o suficiente para reconstruir o polinômio original e confirmar a identidade do usuário. Biometrias não compatíveis ou muito corrompidas não se sobrepõe o suficiente às minúcias originais, e geram um polinômio diferente do cadastrado.

Bio Hashing Esta técnica consiste em aplicar diversas transformações à BR ([Jin et al., 2004] usa transformadas em ondaletas e Fourier-Mill juntamente com outros tratamentos) para gerar um vetor de recursos. Pode-se então combinar este vetor com um número aleatório para gerar o *hash* biométrico. Na autenticação, pode-se aplicar o *hash* na biometria proposta e compará-lo com o *hash* da biometria original cadastrada, permitindo ou não a autenticação. O benefício de utilizar o *bio-hash* em oposição à biometria original, é a segurança que se ganha caso o *hash* seja vazado. Sem o número aleatório usado no cadastro, é praticamente impossível obter a BR original a partir do *hash*, mantendo a identidade do usuário em caso de vazamento.

Criptografia homomórfica Pode-se dizer que um algoritmo criptográfico é homomórfico se um ou mais tipo de operações, ao serem aplicados nos dados cifrados, são também aplicados nos dados originais [Armknecht et al., 2015]. Isso permite trabalhar com os dados cifrados, mantendo privacidade, sem ter a necessidade de decifrar os dados para realizar cálculos com eles. Com esta capacidade, em sistemas biométricos, é possível

trabalhar com uma BR cifrada, e calcular métricas de similaridade entre *templates* (como similaridade de cossenos [Gomez-Barrero et al., 2017]). Assim, atua-se somente no domínio dos dados cifrados, minimiza-se a necessidade de armazenar os *templates* as claras.

6.4.9.2. Técnicas de biometria cancelável

As técnicas de Biometria Cancelável, assim como Criptosistemas Biométricos, tem o propósito de proteger os *templates* e referências biométricas. Porém, este tipo de técnica, tenta mitigar principalmente o perigo de um vazamento do dado biométrico. As Características biométricas são, em geral, permanentes para um indivíduo. Por isso, caso uma BR desprotegida seja vazada, a biometria dos usuários afetados pode ser considerada comprometida para o resto de sua vida [Ratha et al., 2001b]. Para características como digitais, pode-se simplesmente utilizar outros dedos, mas, por exemplo, rostos não podem serem facilmente alterados. Essa situação acontece em contraste a sistemas de autenticação baseados em senhas (*keywords*), que podem, a qualquer momento, serem revogadas e alteradas caso sejam comprometidas. Por isso, é necessário este tipo de revogabilidade ou "cancelabilidade"para os *templates*. Assim, caso as BRs sejam comprometidas, é possível, do mesmo modo que as senhas, simplesmente revogá-las, e o usuário continua utilizando a mesma biometria sem medo de o *template* vazado ser utilizado por terceiros.

Bio Convolving Para se utilizar esta técnica, é preciso que a BRs possa ser representada por um conjunto de sequências [Maiorana et al., 2010, Patel et al., 2015a]. Para transformar a biometria original, esta é dividida em n sequências de tamanho d, sendo d é o um número aleatório chamado de chave. Com estas n sequências, se obtém o *template* transformado e realizando a convolução entre cada uma das sequências. Este *template* ainda representa a biometria, entretanto é possível gerar vários *templates* diferentes alterando somente d. Desta forma, então tem-se a possibilidade de revogar um *template* comprometido e criar um novo a partir da biometria original.

Random projections Este método [Khan et al., 2015] consiste em projetar a BR em um "espaço aleatório". Neste espaço, cada valor é escolhido aleatoriamente e, ainda mais, deve preservar a distância entre dois pontos no *template* original. Caso o segundo requisito não seja satisfeito, a projeção não poderá ser utilizada como uma "*proxy*"da biometria original.

Transformações não inversíveis As transformações deste tipo [Pabitha and Latha, 2013] [Rathgeb and Uhl, 2011] são um dos métodos mais simples para criação de *templates* biométricos canceláveis. O objetivo desta técnica é aplicar uma ou mais transformações não inversíveis na BR original, no ato do cadastro. Como a transformação é não inversível, um atacante, ou até mesmo o agente armazenando a BR transformada, não consegue obter a biometria original com base no *template* transformado. Para realizar autenticação, basta aplicar as mesmas transformações à BR sendo testada e compará-la com o *template* transformado original. Dessa forma, no caso de um vazamento de dados, o *template* transformado pode ser revogado sem dano ao usuário. Para criar uma nova BR, basta realizar

a transformação novamente utilizando outras transformações ou parâmetros diferentes da primeira transformação.

Filtros Bloom Um Filtro Bloom [Bloom, 1970] é uma estrutura de dados que pretende armazenar a possível existência ou não de um elemento nesta estrutura. Em outras palavras, dado um elemento *n*, deseja-se saber se este está presente na base de dados. O Filtro Bloom pode dizer que *n* possivelmente está na base ou que não há possibilidade de *n* estar na base. Durante este processo, a base nunca guarda *n* em si, mas simplesmente se *n* está, ou não, cadastrado na base. Em geral, Filtros Bloom funcionam utilizando várias funções de *hash* diferentes [Rathgeb et al., 2014]. Ao cadastrar uma BR na base, é calculado todos os *hashes* da BR. O resultado deste cálculo resulta em uma série de posições em um *bit-array* que serão definidos como 1. Na autenticação, é feito o mesmo processo para a BR sob teste. Se nenhuma das posições no *array* é 1, sabe-se com certeza que a BR testada não está cadastrada na base. Caso um ou mais bit seja 1, sabe-se somente que "é possível"que a BR tenha sido cadastrada, mas não se pode ter certeza. Para minimizar esta incerteza, utiliza-se um número maior de funções de *hash* para evitar que uma BR não cadastrada tenha bits selecionados.

6.5. Modelos de aplicação e segurança de sistemas biométricos

Ao projetar um sistema biométrico deve-se analisar as condições e o meio em que o sistema será implementado, nesse sentido, existem diferentes tipos de modelos que servem para as mais diversas aplicações de segurança. Nessa seção serão discutidos os diferentes tipos de modelos de sistemas biométricos e suas aplicações tendo como base a norma [ISO/IEC 24745, 2022].

6.5.1. Modelos de aplicação de sistemas biométricos

De acordo com a norma [ISO/IEC 24745, 2022], de forma geral, um sistema biométrico pode ser classificado considerando o local no qual as BR e as IR são armazenadas e comparadas. No que diz a respeito de segurança, cada modelo possui seus benefícios e problemas em termos de transmissão, processamento de dados e armazenamento. Conceitualmente, vários diferentes tipos de modelos são possíveis, mas neste minicurso são abordados apenas aqueles com aplicações reais que são discutidas na norma [ISO/IEC 24745, 2022]. Assim, onze diferentes tipos de modelos que vão de A a K como expostos na Tabela 6.6.

Tabela 6.6: Modelos de aplicação de sistemas biométricos.

		Armazenamento			
		Servidor	Cliente	*Token*	Distribuída
Comparação	Servidor	A		B	G
	Cliente	C	D	E	H
	Token			F	
	Distribuída	I		J	K

As plataformas em que os dados podem ser armazenados e comparados são:

- Servidor: É basicamente um computador conectado remotamente ao cliente por meio de uma rede.
- Cliente: A característica principal de um cliente é a de garantir os serviços de *front-end* a um sistema biométrico, além de ser responsável de se comunicar com o servidor e com o *token*. Logo, o cliente é a plataforma que faz a comunicação do usuário com o servidor e com o *token*. Nesse sentido, o cliente pode ser os computadores pessoais, celulares e seus equivalentes. Observa-se que muitas vezes os sensores biométricos são conectados ou integrados ao cliente.
- *Token*: *Token* nada mais é do que um dispositivo portátil capaz de armazenar as BR e em alguns casos é capaz de executar comparações. Dessa forma, *tokens* podem ser USB *memory sticks*, *e-passaports* e *smart cards* entre outros.
- Distribuída: Em relação ao armazenamento, o termo faz referência ao armazenamento das BR e RBR serem em pelo menos duas entidades (*token*, servidor, e cliente) e as referências biométricas não estão disponíveis com apenas uma das entidades. No que diz a respeito da comparação, para obter o resultado da verificação a execução da comparação deve envolver pelo menos duas entidades (*token*, servidor, e cliente).

Os diversos modelos expostos de A a K descrevem diferentes topologias de localização para os diversos subsistemas. Nesse sentido, os modelos de A a F, podem ser usados tanto referências biométricas normais quanto RBR, o que irá determinar o tipo de BR serão as características de segurança e privacidade desejadas para o projeto. Em contrapartida, os modelos G e H são aplicados apenas para RBRs. Os modelos I, J, K além de serem possíveis projetos com BR ou RBR estes descrevem o caso em que a comparação é distribuída em vários locais o que evita que a informação seja coletada em apenas um local para ser comparada. A norma [ISO/IEC 24745, 2022] recomenda que ao projetar um sistema biométrico deve-se preferencialmente seguir algum dos modelos descritos na Subseção 6.5.1 ou uma combinação desses. Isso, levando em consideração sua aplicação bem como as características de privacidade e de segurança de cada um dos modelos. Além disso, a norma [ISO/IEC 24745, 2022] ressalta que ao projetar um sistema biométrico ou implementar qualquer um dos modelos descritos, seu desempenho, privacidade, segurança devem ser avaliados seguindo a norma [ISO/IEC 30136, 2018], norma especializada em avaliar a precisão, sigilo e privacidade dos *templates* biométricos e sistemas de proteção.

6.5.2. Segurança de modelos de aplicação biométrica

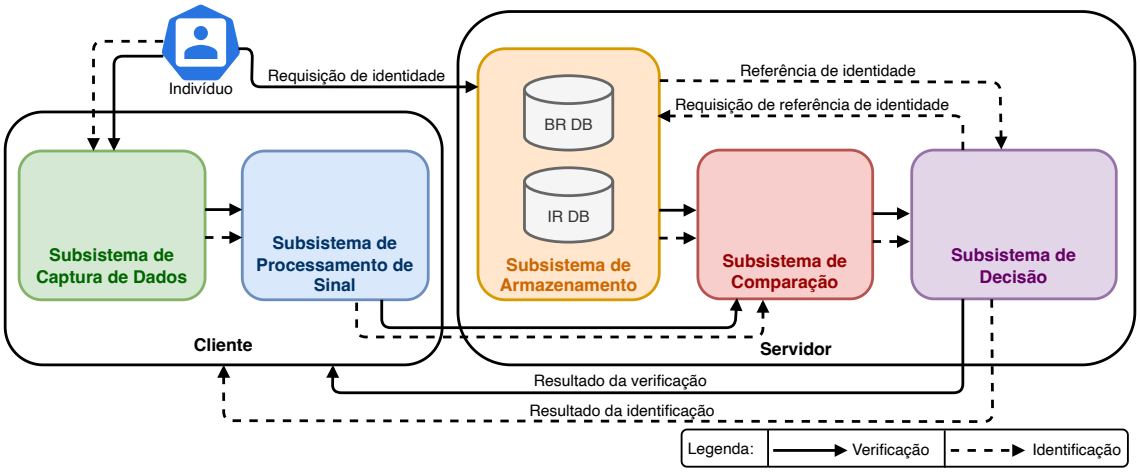
Nessa seção, são abordados cada um dos modelos, destacando suas características e aplicações.

6.5.2.1. Modelo A - comparado no servidor e armazenado no servidor.

No Modelo A, o processo de inscrição dos dados biométricos, as BRs e as respectivas IRs são associadas e armazenadas no servidor. Assim, na identificação, os dados biométricos são capturados e processados no cliente e então as BRs extraídas são transmitidas para o

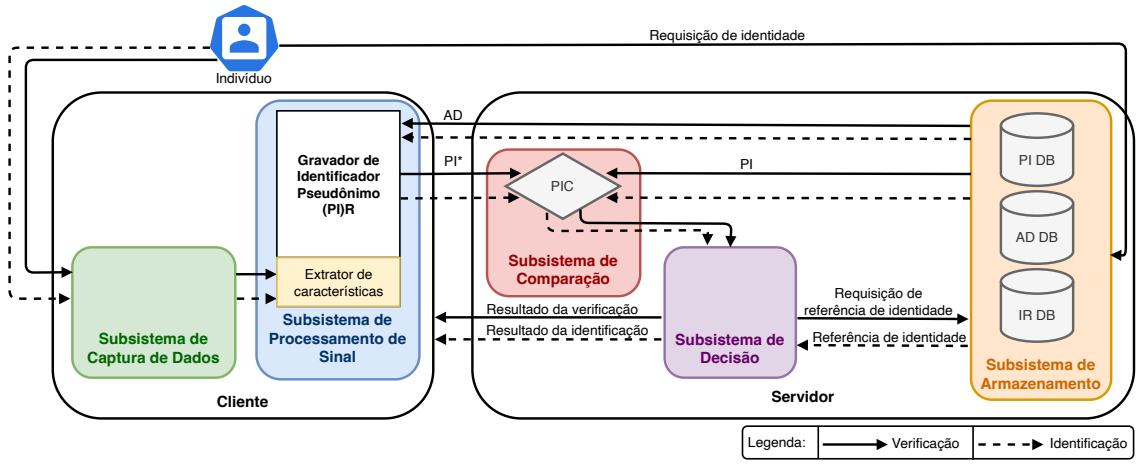
servidor no qual ocorre comparação com as BRs armazenadas no banco de dados e assim a decisão é tomada pelo servidor, como identificado no fluxograma pela linha tracejada – Figura 6.15.

Figura 6.15: Modelo A - Armazenado no servidor e comparado no servidor usando BRs.



No caso do processo de verificação, além da BR, é enviado ao servidor uma requisição de identidade. Assim, a BR presente no banco de dados que está associada a IR reivindicada é comparada com a BR enviada para o servidor e então a decisão é tomada pelo servidor como ilustrado no fluxograma pela linha contínua na Figura 6.15. A Figura 6.16 ilustra os processos de identificação e verificação do Modelo A para o uso de RBR.

Figura 6.16: Modelo A - Armazenado no servidor e comparado no servidor usando RBRs.

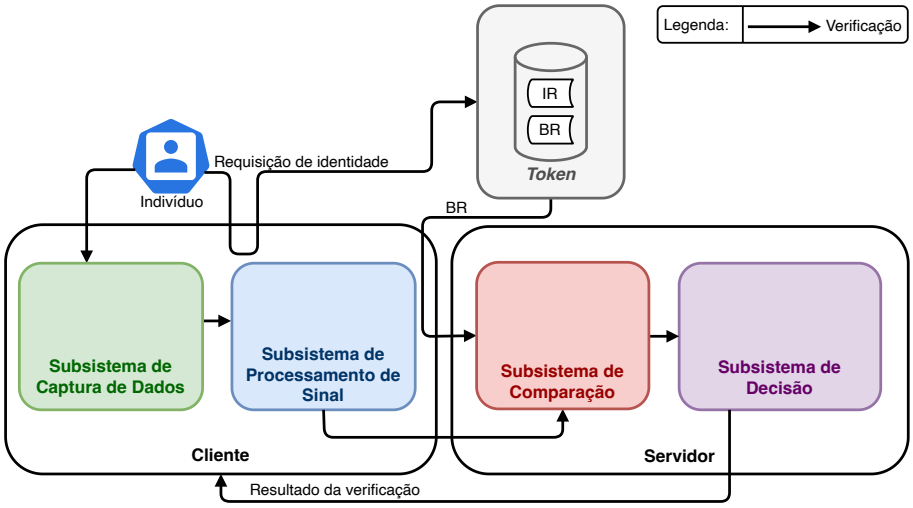


De um ponto de vista de privacidade, nesse tipo de modelo é recomendado o uso de referências biométricas renováveis RBR. Nesse tipo modelo, na inscrição são armazenados os Identificador Pseudônimo (PI) ao invés das BRs. As PIs armazenadas no servidor são associadas às IRs e aos dados auxiliares (AD), que são dados que fazem parte das RBRs e podem ser necessários para reconstruir as PIs. O processo de identificação e verificação é semelhante ao descrito anteriormente para o caso do uso de BR, mas ao invés das BRs serem transmitidas e comparadas, são utilizadas as PIs. As PIs dos dados biométricos extraídos, representados por PI* no fluxograma da Figura 6.16, são geradas no cliente pelo processo de Gravador de Identificador Pseudônimo (*Pseudonymous identifier recorder* - PIR) que utiliza os AD enviadas pelo servidor e os dados biométricos extraídos por meio de sensores para construir as PIs que serão enviadas ao servidor para a comparação. Tanto o processo de verificação quanto o de identificação para RBR podem ser observados no fluxogramas da Figura 6.16 e ocorrem de forma análoga ao caso BR.

6.5.2.2. Modelo B - Comparado no servidor e armazenado no *token*.

A Figura 6.17 ilustra o processo verificação do Modelo B para o uso de BR. Nesse modelo, no processo de inscrição dos dados biométricos, as BRs e as respectivas IRs são associadas e armazenadas em um *token*. Assim, na verificação, os dados biométricos são capturados e processados no cliente e as BRs extraídas são enviadas para o servidor, ocorrendo a comparação com os dados presentes no *token* e assim a decisão.

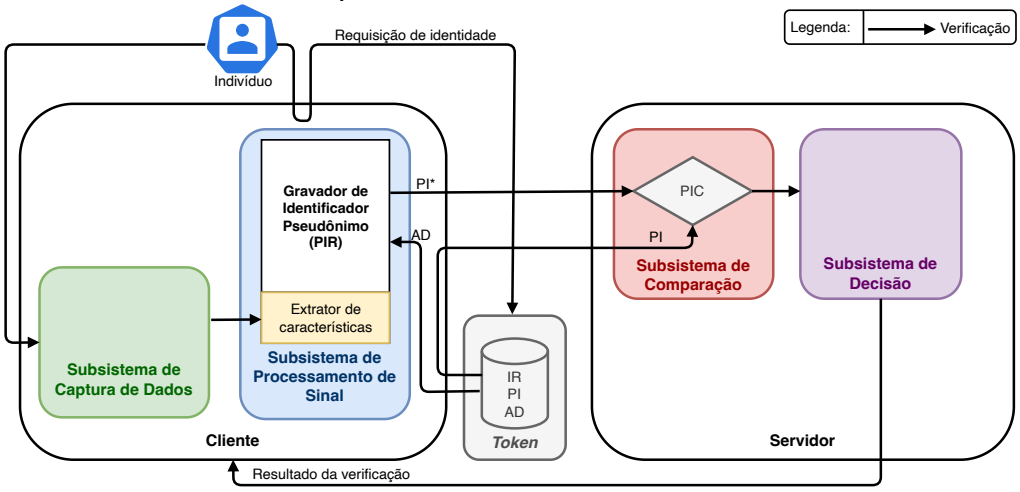
Figura 6.17: Modelo B - Comparado no servidor e armazenado no *token* usando BRs.



Na Figura 6.17, caso um usuário queira verificar sua identidade deve conectar fisicamente um *token* no cliente e enviar seus dados biométricos coletados por meio de um sensor. Dessa forma, o cliente reivindica a identidade do usuário ao *token* e envia tanto as BRs extraídas quanto as BRs do *token* para o servidor no qual ocorre a comparação e então a decisão é tomada pelo próprio servidor. Esse modelo geralmente é usado apenas para verificação, pois não há nenhum dado biométrico no servidor para comparação a não ser aquele que o usuário envia por meio do *token*. Observa-se que como esse modelo faz uso de um *token* portátil não necessita de medidas de segurança no banco de dados uma vez que o *token* está seguro com o usuário. Por outro lado, necessita de segurança na

rede para que os dados enviados para o servidor tanto do *token* quanto do cliente sejam seguros. A Figura 6.18 ilustra o processo de verificação do modelo B para o uso de RBR.

Figura 6.18: Modelo B - Comparado no servidor e armazenado no *token* usando RBRs.

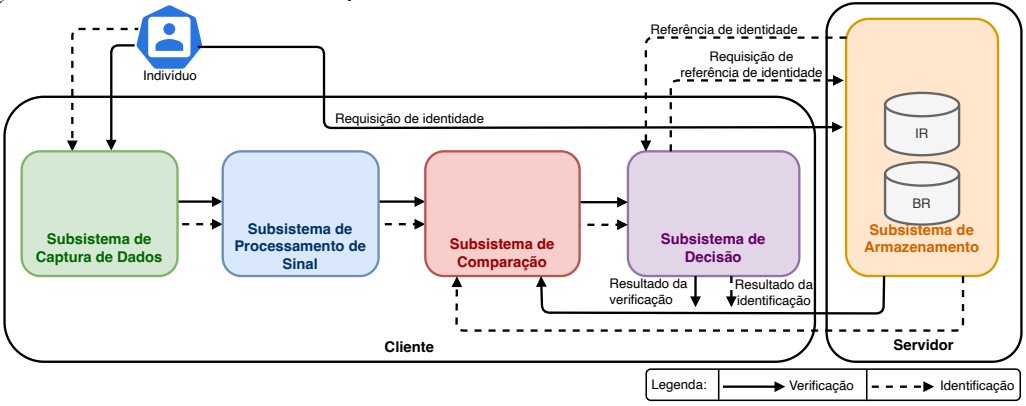


Já no caso com RBR, na inscrição dos dados, são armazenados os PIs, IRs e os ADs no *token* ao invés das BRs e IRs. Ademais, no processo de verificação, a PI que será transmitida ao servidor, representados por PI* no fluxograma da Figura 6.18, é gerada pelo processo de gravador de identificador pseudônimo (PIR) que utilizada os AD do *token* e os dados biométricos extraído por meio de sensores para construir a PI que serão enviadas para o servidor juntamente com a PI presente no *token* e assim ocorre a comparação e a decisão de forma semelhante ao caso com BR.

6.5.2.3. Modelo C - Comparado no cliente e armazenado no servidor.

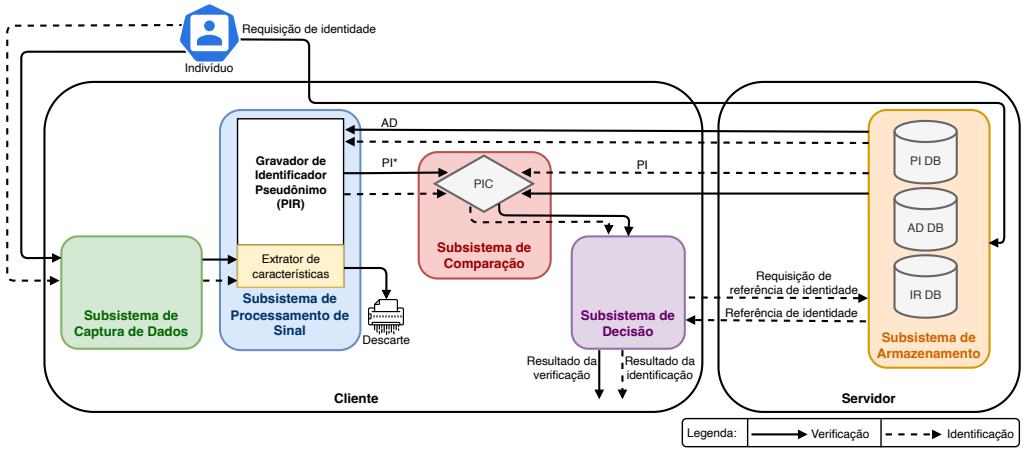
No Modelo C, no processo de inscrição dos dados biométricos, as BRs e as respectivas IRs são associadas e armazenadas no servidor. Assim, no processo de identificação, os dados biométricos são coletados e processados no cliente e as BRs extraídas, no lado do cliente, são comparadas com as BRs armazenadas no banco de dados do servidor que são enviadas ao cliente, e assim, a decisão é tomada pelo próprio cliente, como mostra o fluxograma de linha tracejada da Figura 6.19.

Figura 6.19: Modelo C - Comparado no cliente e armazenado no servidor usando BRs.



Já no processo de verificação, é enviada ao servidor uma requisição de identidade. Assim, a BR presente no banco de dados do servidor que está associada a IR reivindicada é enviada ao cliente e comparada com a BR extraída e então a decisão é tomada pelo servidor como ilustrado pelo fluxograma de linha contínua da Figura 6.19. Observa-se que o cliente deve haver um sensor e um algoritmo de decisão embutido. A Figura˜6.20 ilustra os processos de identificação e verificação do Modelo C para o uso de RBR.

Figura 6.20: Modelo C - Comparado no cliente e armazenado no servidor usando RBRs.

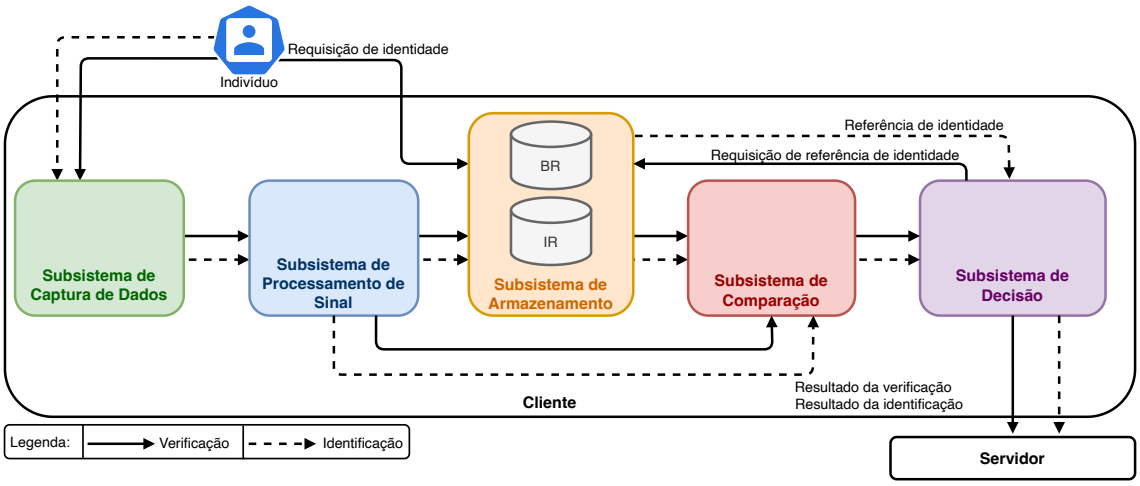


Para o uso de RBR, na inscrição são armazenadas as PI ao invés das BRs. As PI armazenadas no servidor são associadas às IR e aos AD. O processo de identificação e verificação é semelhante ao descrito anteriormente para o caso do uso de BR, mas ao invés das BRs serem transmitidas e comparadas, são utilizadas as PIs. As PIs dos dados biométricos extraídos, representados por PI* no fluxograma da Figura 6.20, são geradas no cliente pelo processo de PIR que utiliza os AD enviadas pelo servidor e os dados biométricos extraídos por meio de sensores para construir as PIs (PI*) que serão comparadas no cliente com a PIs enviadas pelo servidor. Tanto o processo de verificação quanto o de identificação para RBR podem ser observados nos fluxogramas da Figura 6.20 e ocorrem de forma análoga ao caso BR.

6.5.2.4. Modelo D - Comparado no cliente e armazenado no cliente.

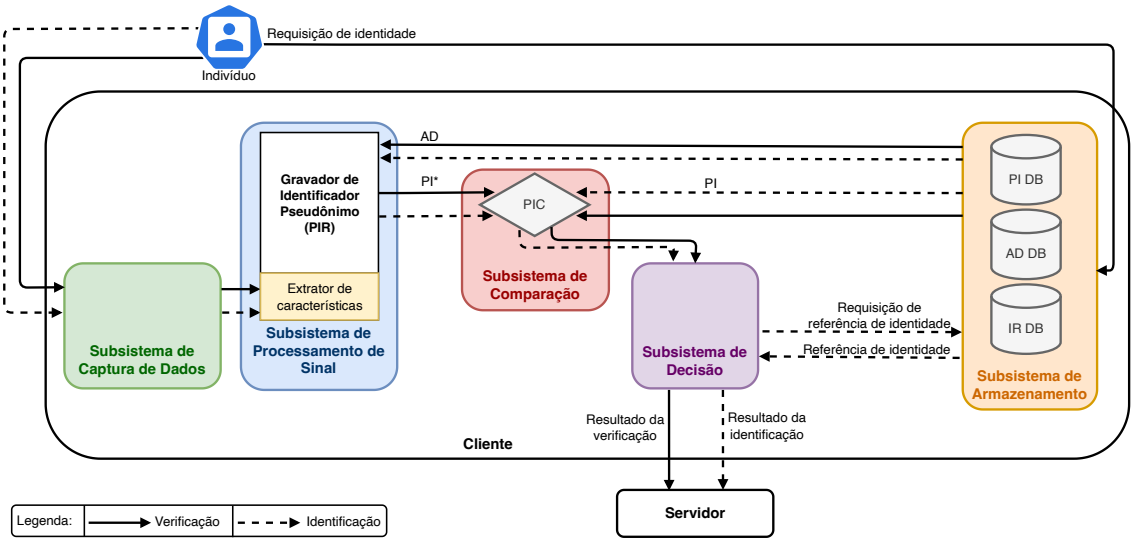
No Modelo D, no processo de inscrição dos dados biométricos, as BRs e as respectivas IRs são associadas e armazenadas no cliente. Assim, na identificação, os dados biométricos são coletados e processados no cliente e as BRs são extraídas. Essa BR extraída é comparada com a BRs presentes no banco de dados do próprio cliente e a decisão é tomada por um algoritmo de decisão presente no mesmo, como ilustrado no fluxograma pela linha tracejada (Figura 6.21).

Figura 6.21: Modelo D - Comparado no cliente e armazenado no cliente usando BRs.



No processo de verificação, é enviada ao cliente uma requisição de identidade. Assim, a BR presente no banco de dados do cliente que está associada a IR reivindicada é comparada com a BR extraída e então a decisão é tomada pelo próprio cliente como exposto pelo fluxograma de linha contínua (Figura 6.21). Nesse sentido, assim como no Modelo C, o cliente deve conter um sensor e um algoritmo de decisão embutido. Esse tipo de modelo é amplamente utilizado para autenticação, principalmente para *laptops*, telefones celulares e fechaduras eletrônicas, pois esse modelo não exige conexão com a rede. Ressalta-se que em alguns casos a autenticação final pode ser feita por um servidor que confirma a verificação feita pelo cliente. A Figura 6.22 ilustra os processos de identificação e verificação do Modelo D para o uso de RBR.

Figura 6.22: Modelo D - Comparado no cliente e armazenado no cliente usando RBRs.

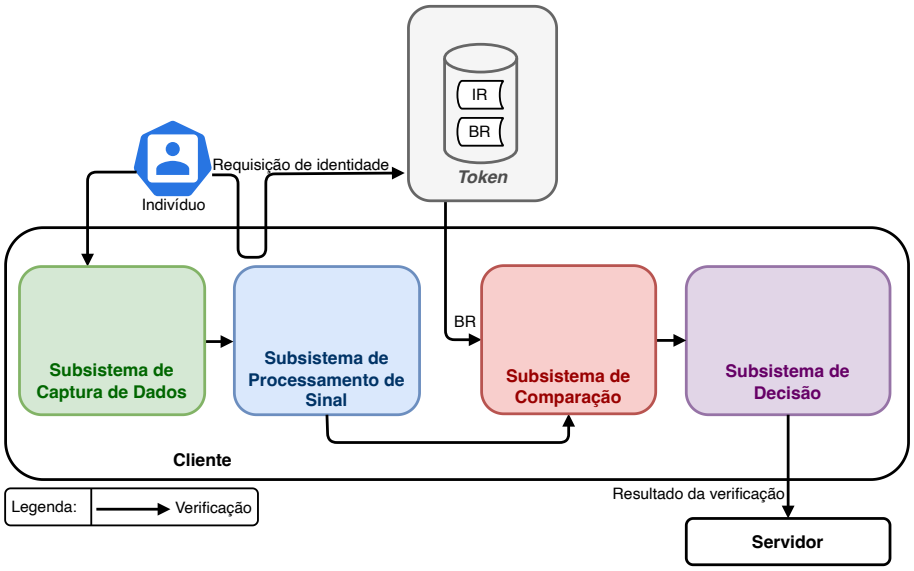


Assim, para o uso de RBR, na inscrição são armazenadas as PI ao invés das BRs. As PIs armazenadas no cliente são associadas às IRs e aos ADs. O processo de identificação e verificação é semelhante ao descrito anteriormente para o caso do uso de BR, mas ao invés das BRs serem comparadas, são utilizadas as PIs. As PIs dos dados biométricos extraídos, representados por PI* no fluxograma da Figura 6.22, são geradas no cliente pelo PIR que utiliza os AD enviadas pelo banco de dados do cliente e os dados biométricos extraídos por meio de sensores para construir as PIs (PI*) que serão comparadas no próprio cliente com a PIs do banco de dados. Tanto o processo de verificação quanto o de identificação para RBR podem ser observados nos fluxogramas da Figura 6.22 e ocorrem de forma análoga ao caso do uso de BR.

6.5.2.5. Modelo E - Comparado no cliente e armazenado no *token*.

No Modelo E, no processo de inscrição dos dados biométricos, as BRs e as respectivas IRs são associadas e armazenadas em um *token*. Assim, na verificação, os dados biométricos são capturados e processados no cliente e as BR extraídas são comparadas com os dados presentes no *token* e toma a decisão, como mostra no fluxograma da Figura 6.23. Então, caso um usuário queira verificar sua identidade deve conectar fisicamente um *token* no cliente e enviar seus dados biométricos ao cliente por meio de um sensor. Dessa forma, o cliente reivindica a identidade do usuário ao *token* e a BR do *token* é comparada com a BR extraída e então a decisão é tomada pelo próprio cliente.

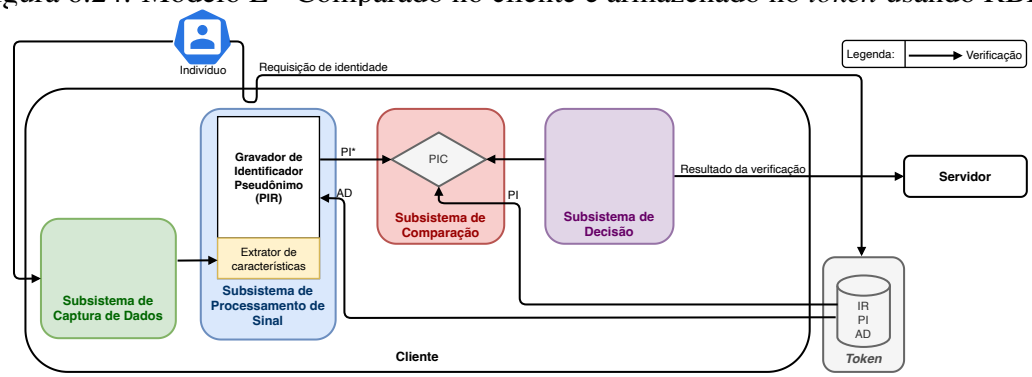
Figura 6.23: Modelo E - Comparado no cliente e armazenado no *token* usando BRs



Para implementar o Modelo E, assim como no Modelo C, o cliente deve conter tanto o sensor quanto um algoritmo capaz de fazer a comparação e a decisão. Da mesma maneira que Modelo D, em alguns casos a autenticação final pode ser feita por um servidor que confirma a verificação feita pelo cliente. Além disso, observa-se que esse modelo é utilizado apenas para verificação e frequentemente usado na forma em que o cliente

funciona como do tipo quiosque e é instalado em lugares públicos como aeroportos e prédios comerciais para autenticação pessoal. Um exemplo de uso é no controle de fronteira em aeroportos internacionais em que o e-passaporte funciona como *token*. A Figura 6.24 ilustra os processo de verificação do Modelo E para o uso de RBR.

Figura 6.24: Modelo E - Comparado no cliente e armazenado no *token* usando RBRs.

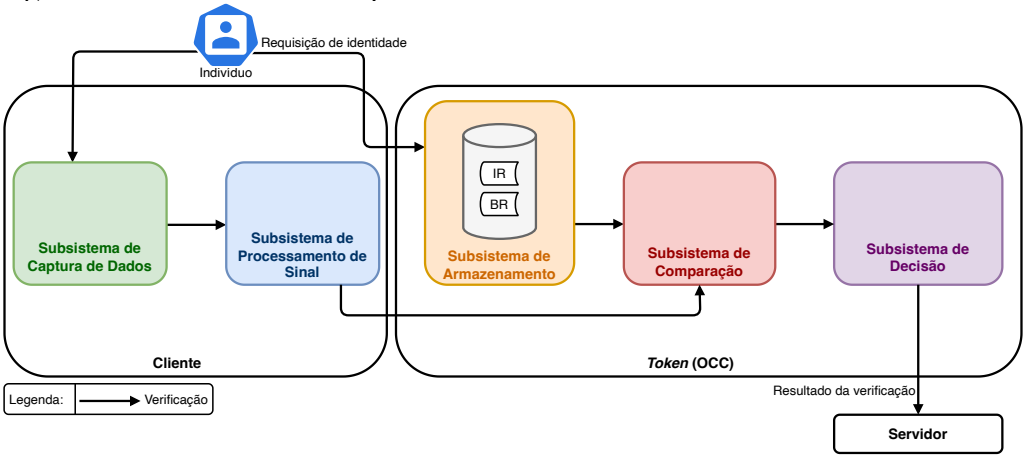


Já no caso com RBR, na inscrição dos dados, são armazenados os PIs, IR e o ADs no *token* ao invés das BRs e IRs. Adicionalmente, no processo de verificação, a PI que será transmitida ao servidor, representados por PI* no fluxograma (Figura 6.24), é gerada pelo processo de PIR que utilizada os AD do *token* e os dados biométricos extraído por meio de sensores para construir a PI (PI*) no cliente que serão comparadas com a PI presente no *token* e a decisão ocorre de forma semelhante ao caso do uso de BR.

6.5.2.6. Modelo F - comparado no *token* e armazenado no *token*.

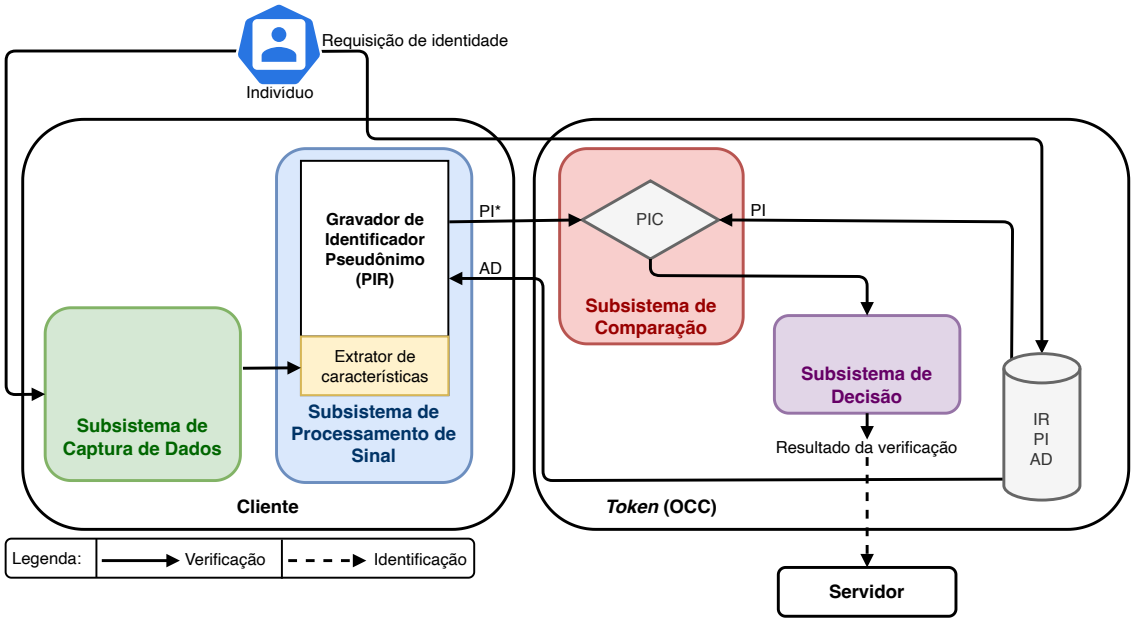
No Modelo F, no processo de inscrição dos dados biométricos, as BRs e as respectivas IRs são associadas e armazenadas em um *token*. Assim, na verificação, os dados biométricos são capturados e processados no cliente e as BR extraídas são enviadas para o *token* e então são comparadas com os dados presentes no próprio. Assim, a decisão é tomada como descrito no fluxograma da Figura 6.25.

Figura 6.25: Modelo F - comparado no *token* e armazenado no *token* usando BRs



Caso um usuário queira verificar sua identidade deve enviar seus dados biométricos para o cliente com o *token* do tipo *on-card comparison* (OCC). Assim, o cliente extrai as BRs e as IRs do usuário e as envia para o *token* para o processo de comparação. O resultado da comparação é enviado ao servidor. Dessa forma, os dados biométricos limitam-se ao *token* e ao cliente, e não é passado para o servidor. Nesse caso, o cliente pode ser um caixa eletrônico e o *token* ou cartão do tipo OCC do usuário. Observa-se que esse tipo de tecnologia é amplamente utilizado em sistemas bancários por ser consideravelmente segura, uma vez que assume-se que o *token* é capaz de fornecer um ambiente de execução seguro e isolado. A Figura 6.26 ilustra os processo de verificação do Modelo F para o uso de RBR.

Figura 6.26: Modelo F - comparado no *token* e armazenado no *token* usando RBRs.

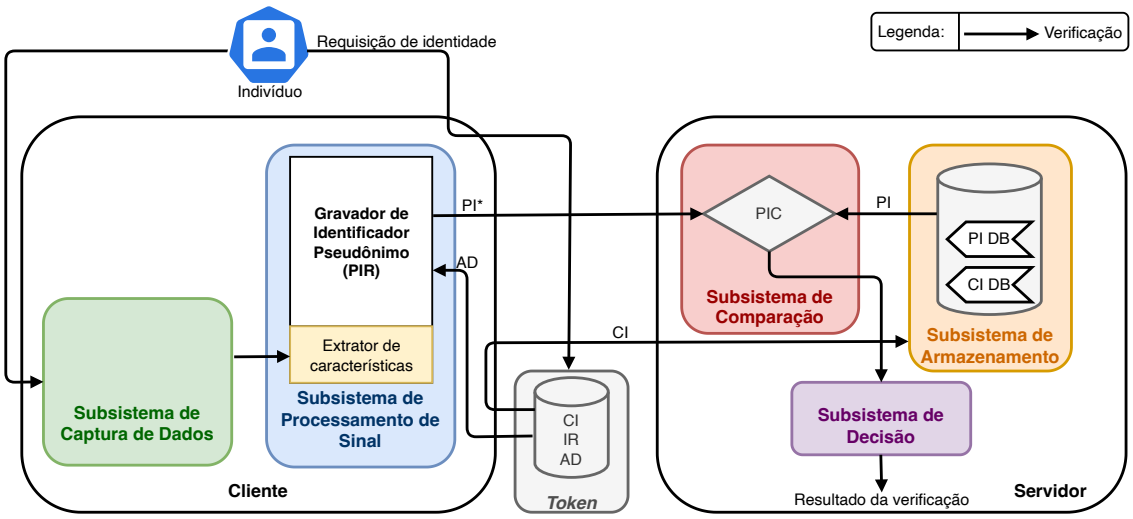


Já no caso com RBR, na inscrição dos dados, são armazenados os PIs, IRs e o ADs no *token* ao invés das BRs e IRs. Além disso, no processo de verificação, a PI que será usada na comparação, representados por PI* no fluxograma da Figura 6.26, esta é gerada no cliente pelo processo de PIR que utilizada os AD do *token* e os dados biométricos extraídos por meio de sensores para construir a PI (PI*) que serão comparadas com a PI presente no *token*. A decisão ocorre de forma semelhante ao caso com o uso de BR como mostra o fluxograma da Figura 6.26. Observa-se que também é possível que a PIR ocorra no próprio *token* o que implica com que os AD permanecessem dentro do *token* e minimizando as chances de ter a privacidade comprometida.

6.5.2.7. Modelo G - Comparado no servidor e armazenado de forma distribuída.

O Modelo G é aplicável apenas para RBR. Logo, no processo de inscrição dos dados biométricos, a PI é criada e armazenada no servidor com um *identificador comum* (CI) já a IR, o AD e a CI correspondente é armazenada em um *token*. Durante a verificação, o *token* envia para o cliente os AD e o CI e assim como nos outros modelos, por meio do processo de PIR, o cliente transforma os dados biométricos em PI que é representado como PI* no fluxograma da Figura 6.27. Na sequência, o cliente envia a PI* e o CI para o servidor que compara com a PI correspondente a CI enviada e a decisão é tomada. O benefício desse modelo é a de que a verificação só é possível quando tanto o *token* quanto o servidor possuírem os dados corretos e, nesse sentido reduzir a chances de adulteração uma vez que tanto no servidor quanto no *token* os dados deverão estar comprometidos. Além disso, isso permite que as referências biométricas sejam revogadas no servidor sem necessitar do *token*.

Figura 6.27: Modelo G - comparado no servidor e armazenado de forma distribuída.

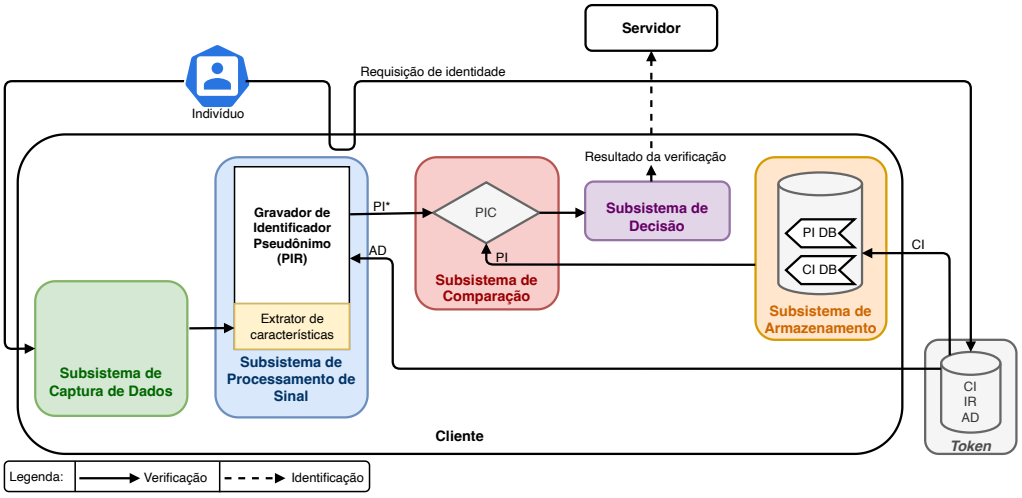


Outro benefício é o fato do usuário possuir o controle do processo de verificação, uma vez que ele é o detentor do *token*. Esse modelo é indicado para sistemas de autenticação para transação online como *e-banking*, transações com cartão de crédito *online* e como substitutos dos PIN ou como melhorias para caixas eletrônicos. Nesse sentido, esse modelo visa minimizar a quantidade de informações que são trocadas entre o cliente e o servidor, além de impedir a transmissão de partes dos dados RBR do servidor para o cliente. Ressalta-se que não é recomendável armazenar o PI em um *token* e os AD no servidor.

6.5.2.8. Modelo H - comparado no cliente e armazenado de forma distribuída.

O Modelo H também é aplicável apenas para RBR. Logo, no processo de inscrição dos dados biométricos, a PI é criada e armazenada no cliente com um CI. Já a IR, o AD e o CI correspondentes são armazenados em um *token*. Durante a verificação, o *token* envia para o cliente os AD e o CI e, assim como nos outros modelos, por meio do processo de PIRs, o cliente transforma os dados biométricos em PI que é representado como PI* no fluxograma da Figura 6.28. Na sequência, o subsistema do cliente compara a PI* com a PI correspondente ao CI e a decisão é tomada. Nesse modelo, o cliente pode ser do tipo quiosque, como os encontrados em locais públicos como aeroportos e em locais públicos edifícios, para autenticação pessoal. Este modelo também pode ser aplicado no controle de fronteira usando o passaporte eletrônico ou outro *token*.

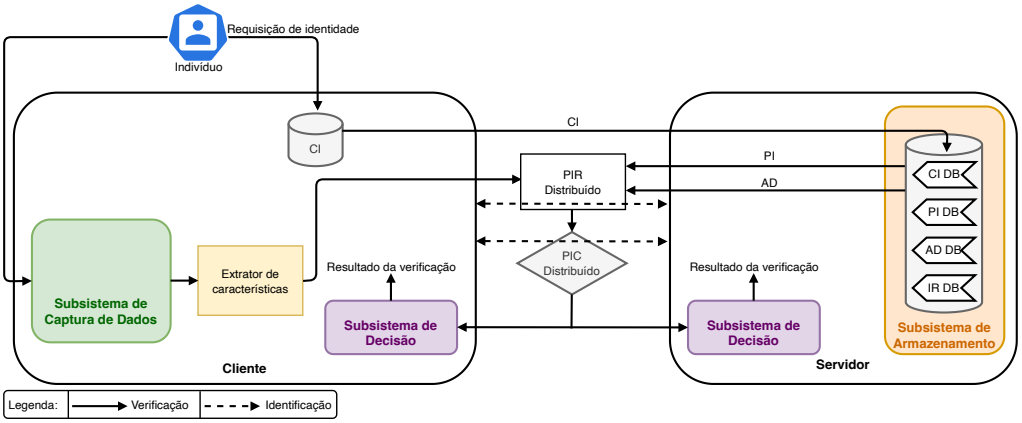
Figura 6.28: Modelo H - comparado no cliente e armazenado de forma distribuída.



6.5.2.9. Modelo I - Comparado de forma distribuída e armazenado no servidor.

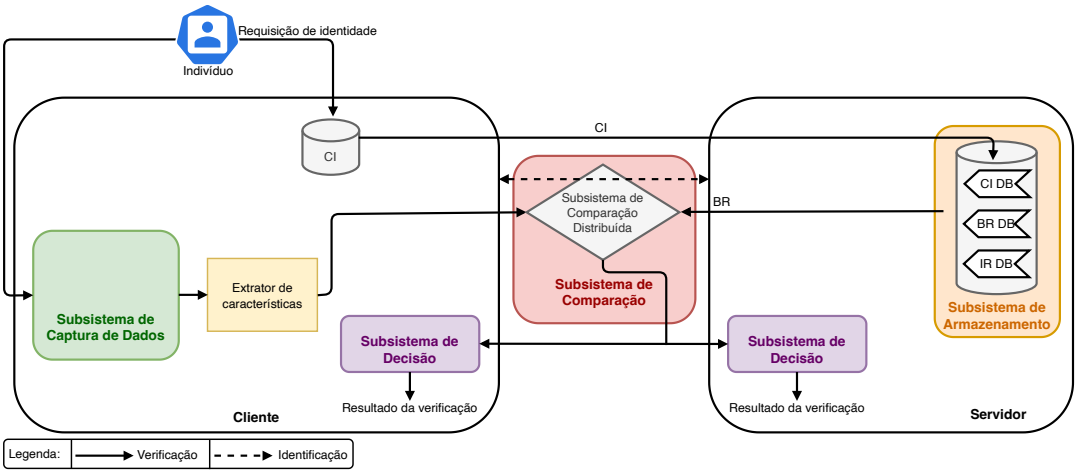
A Figura 6.29 ilustra os processo de verificação do Modelo I para o uso de BR.

Figura 6.29: Modelo I - Comparado de forma distribuída e armazenado no servidor usando BRs.



No Modelo I, no processo de inscrição dos dados biométricos, as AD, IR, PI e um CI são armazenados no servidor e o outro CI é armazenado no cliente. No processo de verificação, o servidor e o cliente nunca compartilham as AD, IR, PI mas apenas o CI. Nesse sentido, tanto o servidor quanto o cliente executam um protocolo interativo para realizar comumente os processos de comparação comparador de identificador pseudônimo / *pseudonymous identifier comparator* (PIC) e de PIR sem nunca compartilhar à outra parte os seus dados (AD, IR, PI no lado do servidor e os dados biométricos coletados no lado do cliente). Por fim, baseado nesse protocolo o interativo, o servidor e o cliente recebe apenas o resultado final da PIC. No fluxograma da Figura 6.29 é ilustrada uma das possíveis variações desse modelo usando RBRs. Esse modelo é indicado apenas aos casos em que a comunicação com a rede e a computação local são suficientemente boas, uma vez que a execução da verificação costuma ser mais pesada do que em comparação a um modelo que não há esse processo interativo. A Figura 6.30 ilustra os processo de verificação do Modelo I para o uso de RBR.

Figura 6.30: Modelo I - Comparado de forma distribuída e armazenado no servidor usando RBRs.

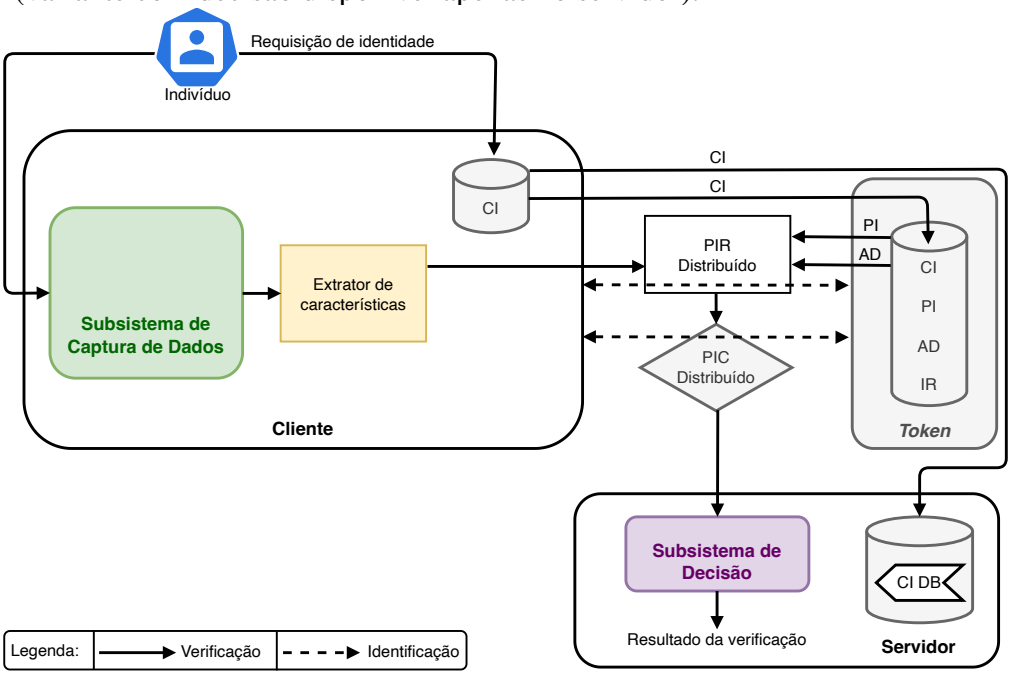


O Modelo I pode ser implementado usado RBRs (Figura 6.30), para que seja possível a implementação, o Subsistema de Comparação deve ser computado de forma distribuída por meio do uso de técnicas de comparação baseadas em métricas simples, como distância de *Hamming* ou distância euclidiana.

6.5.2.10. Modelo J - comparado de forma distribuída e armazenado no *token*.

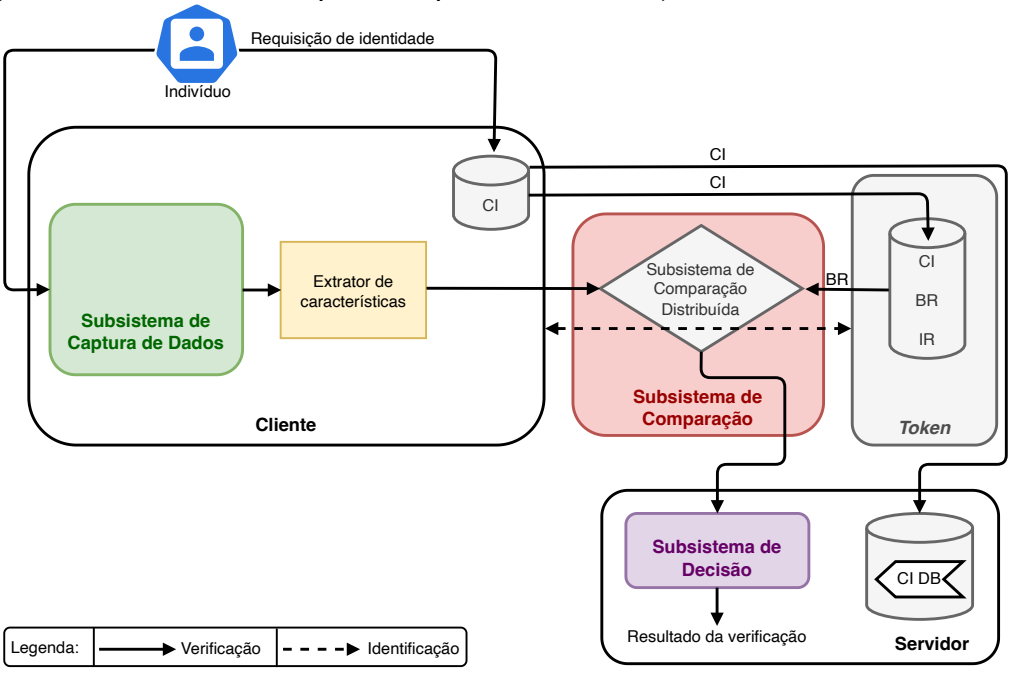
No Modelo J, no processo de inscrição dos dados biométricos, as AD, IR, PI e um CI são armazenados no *token* e o outro CI é armazenado no cliente. No processo de verificação, assim como no modelo anterior, o *token* e o cliente nunca compartilham as AD, IR, PI mas apenas o CI. Nesse sentido, tanto o servidor quanto o cliente executam um protocolo interativo para realizar comumente os processos de comparação (PIC) e de PIR sem nunca compartilhar à outra parte seus dados (AD, IR, PI no lado do *token* e os dados biométricos coletados no lado do cliente). Por fim, baseado nesse protocolo interativo, o *token* e o cliente recebem apenas o resultado final da PIC. A Figura 6.31 exemplifica uma das possíveis variações desse modelo usando RBRs. Esse modelo é indicado apenas aos casos em que a computação local for suficientemente boa.

Figura 6.31: Modelo J - Armazenado no *token*, comparado de forma distribuído usando RBR (variante com decisão disponível apenas no servidor).



A Figura 6.32 ilustra os processo de verificação do Modelo J para o uso de BR. Assim como no modelo anterior, o modelo pode ser implementado usado BRs como ilustrado no fluxograma da Figura 6.32 na condição de que o Subsistema de Comparação possa ser computado de forma distribuída.

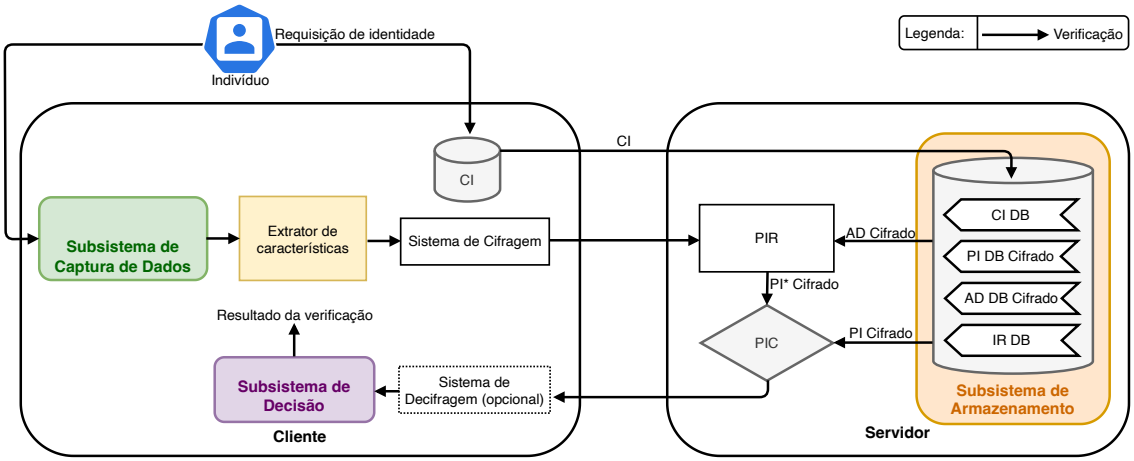
Figura 6.32: Modelo J - Armazenado no *token*, comparado de forma distribuído usando BR (variante com decisão disponível apenas no servidor).



6.5.2.11. Modelo K - comparado de forma distribuída e armazenado de forma distribuída.

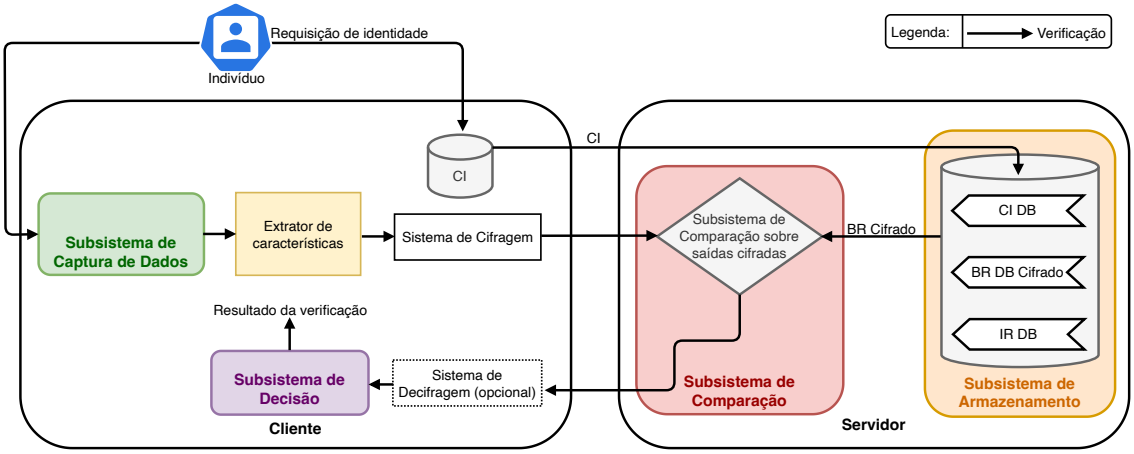
A Figura 6.33 ilustra os processo de verificação do modelo K para o uso de RBR. Nesse modelo, no processo de inscrição dos dados biométricos, as AD, IR, PI são armazenadas de forma distribuída no cliente, servidor e no *token* (caso exista) e um CI é armazenado em cada uma das plataformas. Nesse sentido, o AD e o PI são armazenados de forma cifrada, de modo que possam executar operações sem que seja decifrados previamente. Um exemplo de aplicação é o uso de criptografia homomórfica. Durante a verificação, os processos de PIR e PIC são realizados ou no lado do cliente, *token* ou servidor, e são executados diretamente nos dados cifrados. Na sequência, o resultado da comparação (PIC) é diretamente enviado para o Subsistema de Decisão, como ilustrado no fluxograma da Figura 6.34 para o caso da PIC e da PIR ocorrendo no servidor. Para garantir a confidencialidade dos dados cifrados, o proprietário da chave criptográfica deve ser a parte que não armazena esses dados cifrados. Esse modelo é indicado apenas aos casos em que a computação local for suficientemente boa, uma vez que a execução da verificação em modelos cifrados costuma exigir maior capacidade computacional.

Figura 6.33: Modelo K - Armazenada de forma distribuída, comparado distribuída (variante com PIR e PIC no lado do servidor) usando RBRs, sendo o Sistema de Decifragem opcional.



A Figura 6.33 ilustra os processo de verificação do Modelo K para o uso de BR. Assim como nos modelos anteriores, esse modelo pode ser implementado usado BRs na condição de que o Subsistema de Comparação possa operar de forma cifrada.

Figura 6.34: Modelo K - Armazenada de forma distribuída, comparado distribuidamente (variante com PIR e PIC no lado do servidor) usando BRs, sendo o Sistema de Decifragem é opcional.



6.6. Considerações finais

Os modelos servem como guias para os desenvolvedores de sistemas biométricos e devem ser escolhidos conforme as necessidades e o contexto do projeto. Com isso, para um sistema ser considerado seguro, é essencial que tenha como base algum desses modelos. Embora alguns dos modelos dessa descrito na Seção 6.5 sejam considerados mais teóricos e com poucas aplicações práticas, é importante que estes sejam levados em consideração em implementações e em um contexto adequado ou para servirem como inspiração para outras aplicações. A utilização das características biométricas dos seres humano continua em expressivo crescimento nas mais variadas aplicações, e em muitos cenários de forma indiscriminada. As características biométricas por se tratarem de dados sensíveis, necessitam de mecanismos de proteção adequados para garantir aspectos de segurança, privacidade e conformidade. No contexto brasileiro, a Lei Geral de Proteção de Dados (LGPD) regula as atividades de tratamento dos dados pessoais.

O presente capítulo apresenta as características de tratamento dos dados biométricos e suas especificidades em seu respectivo ciclo de vida, com base na norma ISO/IEC 24745:2022. Também os requisitos recomendados de segurança para o tratamento destes dados pelos sistemas, bem como as ameaças e as contramedidas recomendadas. Dada a característica ,por parte dos sistemas que utilizam dados biométricos para realizar a autenticação e autorização nos sistemas, as necessidades fundamentais de renovação e revogação destes dados biométricos em caso de comprometimento ou necessidade específica de alguma aplicação são de considerável relevância. Assim, os mecanismos para proteção de *templates* biométricos e as suas características foram introduzidas, apresentando os importantes conceitos de biometria revogável ou biometria cancelável. Além de modelos e cenários recomendados para tratamento destes dados sensíveis por parte dos sistemas em suas mais variadas aplicações, com base em requisitos de segurança apresentados.

Apesar dos requisitos de segurança e privacidade para o tratamento dos dados biométricos, e mecanismos de proteção dos *templates* biométricos, existem diversas de-

mandas que necessitam ser abordadas em trabalhos futuros, dada a complexidade que envolvem estes sistemas heterogêneos compostos por hardware e software, bem como aspectos legais. Por fim, existem oportunidades de pesquisa envolvendo o ciclo de vida de tratamento de dados com relação à segurança e privacidade, bem como tratamento das ameaças nas etapas (captura, extração de características, comparação, decisão e armazenamento) dos sistemas, em especial os sistemas de autenticação.

Agradecimentos

O presente trabalho foi em parte financiado pelo CNPq (Projeto 304643/20 20-3), CAPES (Código de Financiamento 001), FAPESP (Projeto 2020/09850-0), e Ripple's University Blockchain Research Initiative (UBRI).

Os autores agradecem o apoio do Laboratório de Arquitetura e Redes de Computadores (LARC) do Departamento de Engenharia de Produção e Sistemas Digitais (PCS) da Escola Politécnica da Universidade de São Paulo (USP).

Os autores agradecem o apoio do Laboratório de Processamento Paralelo e Distribuído (LabP2D) no Centro de Ciências tecnológicas (CCT) / Programa de Pós-Graduação em Computação Aplicada (PPGCAP) da Universidade do Estado de Santa Catarina (UDESC) e da Fundação de Amparo à Pesquisa e Inovação do Estado de Santa Catarina (FAPESC).

Referências

[Araújo et al., 2005] Araújo, L., Sucupira, L., Lizarraga, M., Ling, L., and Yabu-Uti, J. (2005). User authentication through typing biometrics features. *IEEE transactions on signal processing*, 53(2):851–855.

[Armknecht et al., 2015] Armknecht, F., Boyd, C., Carr, C., Gjøsteen, K., Jäschke, A., Reuter, C. A., and Strand, M. (2015). A guide to fully homomorphic encryption. Cryptology ePrint Archive, Paper 2015/1192. https://eprint.iacr.org/2015/1192.

[Balakrishnan et al., 2021] Balakrishnan, S., Venkatesan, V. K., and Syed Shahul Hameed, M. (2021). An embarking user friendly palmprint biometric recognition system with topnotch security. pages 1028–1032.

[Bloom, 1970] Bloom, B. H. (1970). Space/time trade-offs in hash coding with allowable errors. *Commun. ACM*, 13(7):422–426.

[Bolle et al., 2013] Bolle, R. M., Connell, J. H., Pankanti, S., Ratha, N. K., and Senior, A. W. (2013). *Guide to biometrics*. Springer Science & Business Media.

[Conrads, 2019] Conrads, J. (2019). Ddos attack fingerprint extraction tool : making a flow-based approach as precise as a packet-based. http://essay.utwente.nl/79567/.

[Costa et al., 2006] Costa, L. R., Obelheiro, R. R., and Fraga, J. S. (2006). Introdução à Biometria. In *Minicursos do VI Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais*, volume 1, page 49. SBC, Santos/SP.

[Cross and Smith, 1995] Cross, J. and Smith, C. (1995). Thermographic imaging of the subcutaneous vascular network of the back of the hand for biometric identification. In

Proceedings The Institute of Electrical and Electronics Engineers. 29th Annual 1995 International Carnahan Conference on Security Technology, pages 20–35. IEEE.

[Faria, 2014] Faria, B. G. (2014). Implementação e avaliação do abid (aplicativo biométrico de impressão digital) utilizando o método fuzzy vault e ferramentas open source. Master's thesis, Universidade Presbiteriana Mackenzie.

[Gomez-Barrero et al., 2017] Gomez-Barrero, M., Maiorana, E., Galbally, J., Campisi, P., and Fierrez, J. (2017). Multi-biometric template protection based on homomorphic encryption. *Pattern Recognition*, 67:149–163.

[Gomez-Barrero et al., 2016] Gomez-Barrero, M., Rathgeb, C., Galbally, J., Busch, C., and Fierrez, J. (2016). Unlinkable and irreversible biometric template protection based on bloom filters. *Information Sciences*, 370-371:18–32.

[Hernandez-Ortega et al., 2023] Hernandez-Ortega, J., Fierrez, J., Morales, A., and Galbally, J. (2023). Introduction to presentation attack detection in face biometrics and recent advances. *Handbook of Biometric Anti-Spoofing: Presentation Attack Detection and Vulnerability Assessment*, pages 203–230.

[ISO/IEC 24745, 2022] ISO/IEC 24745 (2022). Information security – cybersecurity and privacy protection – biometric information protection. Standard, International Organization for Standardization.

[ISO/IEC 29100, 2011] ISO/IEC 29100 (2011). Information technology — security techniques — privacy framework. Standard, International Organization for Standardization.

[ISO/IEC 30136, 2018] ISO/IEC 30136 (2018). Information technology — performance testing of biometric template protection schemes. Standard, International Organization for Standardization.

[Jain et al., 1996] Jain, A., Bolle, R., and Pankanti, S. (1996). Introduction to biometrics. In Jain, A. K., Bolle, R., and Pankanti, S., editors, *Biometrics*. Springer, Boston, MA.

[Jain et al., 1999] Jain, A., Bolle, R., and Pankanti, S. (1999). *Biometrics: personal identification in networked society*, volume 479. Springer Science & Business Media.

[Jain and Kant, 2015] Jain, R. and Kant, C. (2015). Attacks on biometric systems: an overview. *International Journal of Advances in Scientific Research*, 1(07):283–288.

[Jin et al., 2004] Jin, A. T. B., Ling, D. N. C., and Goh, A. (2004). Biohashing: two factor authentication featuring fingerprint data and tokenised random number. *Pattern Recognition*, 37(11):2245–2255.

[Juels and Sudan, 2006] Juels, A. and Sudan, M. (2006). A Fuzzy Vault Scheme. *Designs, Codes and Cryptography*, 38(2):237–257.

[Juels and Wattenberg, 1999] Juels, A. and Wattenberg, M. (1999). A fuzzy commitment scheme. In *Proceedings of the 6th ACM Conference on Computer and Communications Security*, CCS '99, page 28–36, New York, NY, USA. Association for Computing Machinery.

[Kaur and Khanna, 2016] Kaur, H. and Khanna, P. (2016). Biometric template protection using cancelable biometrics and visual cryptography techniques. *Multimedia Tools and Applications*, 75:16333–16361.

[Kelkboom et al., 2011] Kelkboom, E. J. C., Breebaart, J., Kevenaar, T. A. M., Buhan, I., and Veldhuis, R. N. J. (2011). Preventing the decodability attack based cross-matching in a fuzzy commitment scheme. *IEEE Transactions on Information Forensics and Security*, 6(1):107–121.

[Khan et al., 2015] Khan, S. H., Akbar, M. A., Shahzad, F., Farooq, M., and Khan, Z. (2015). Secure biometric template generation for multi-factor authentication. *Pattern Recognition*, 48(2):458–472.

[Li and Jain, 2009] Li, S. Z. and Jain, A. (2009). *Encyclopedia of Biometrics: I-Z.*, volume 1. Springer Science & Business Media.

[Li and Jain, 2015] Li, S. Z. and Jain, A. K., editors (2015). *Encyclopedia of Biometrics*. Springer, New York, NY, 2 edition.

[Maiorana et al., 2010] Maiorana, E., Campisi, P., Fierrez, J., Ortega-Garcia, J., and Neri, A. (2010). Cancelable templates for sequence-based biometrics with application to on-line signature recognition. *Trans. Sys. Man Cyber. Part A*, 40(3):525–538.

[Maltoni et al., 2009] Maltoni, D., Maio, D., Jain, A. K., and Prabhakar, S. (2009). *Handbook of Fingerprint Recognition*. Springer Professional Computing. Springer London, 2 edition.

[Marcondes, 2019] Marcondes, J. (2019). Biometria, sistema biométrico: O que é, como funciona?. Disponível em: https://gestaodesegurancaprivada.com.br/biometria-sistema-biometrico-o-que-e-como-funciona/. Acesso em: 18 jul 2023.

[Martinez-Diaz et al., 2006] Martinez-Diaz, M., Fierrez-Aguilar, J., Alonso-Fernandez, F., Ortega-Garcia, J., and Siguenza, J. (2006). Hill-climbing and brute-force attacks on biometric systems: A case study in match-on-card fingerprint verification. In *Proceedings 40th Annual 2006 International Carnahan Conference on Security Technology*, pages 151–159.

[Matos, 2000] Matos, R. M. d. (2000). Autenticação de usuários através da utilização de sistemas biométricos. Dissertação de mestrado, Universidade Federal do Rio Grande do Sul, Porto Alegre, RS, Brasil. Dissertação de Mestrado, UFRGS.

[Mtibaa et al., 2021] Mtibaa, A., Petrovska-Delacrétaz, D., Boudy, J., and Ben Hamida, A. (2021). Privacy-preserving speaker verification system based on binary i-vectors. *IET Biometrics*, 10(3):233–245.

[Nafea et al., 2016] Nafea, O., Ghouzali, S., Abdul, W., and Qazi, E.-u.-H. (2016). Hybrid multi-biometric template protection using watermarking. *The Computer Journal*, 59(9):1392–1407.

[Oliveira Filho, 2014] Oliveira Filho, I. d. L. (2014). *Algoritmo Papílio como Método de Proteção de Templates para Aumentar a Segurança em Sistemas de Identificação Biométricos*. Tese de doutorado, Universidade Federal do Rio Grande do Norte, Natal, RN, Brasil.

[Pabitha and Latha, 2013] Pabitha, M. and Latha, L. (2013). Efficient approach for retinal biometric template security and person authentication using noninvertible constructions. *International Journal of Computer Applications*, 69:28–34.

[Palmeiras, 2021] Palmeiras, S. E. (2021). Política de privacidade e proteção de dados. Disponível em: https://sep-bucket-prod.s3.amazonaws.com/wp-content/uploads/2021/08/politica-de-privacidade_12082021.pdf. Acesso em: 21 jul 2023.

[Palmeiras, 2023] Palmeiras, S. E. (2023). Comunicado: Cadastro de biometria facial dos lientes - passaporte. Disponível em: https://www.palmeiras.com.br/noticias/comunicado-cadastro-de-biometria-facial-dos-clientes-passaporte/. Acesso em: 21 jul 2023.

[Patel et al., 2015a] Patel, V. M., Ratha, N. K., and Chellappa, R. (2015a). Cancelable biometrics: A review. *IEEE Signal Processing Magazine*, 32(5):54–65.

[Patel et al., 2015b] Patel, V. M., Ratha, N. K., and Chellappa, R. (2015b). Cancelable biometrics: A review. *IEEE Signal Processing Magazine*, 32(5):54–65.

[Ratha et al., 2001a] Ratha, N. K., Connell, J. H., and Bolle, R. M. (2001a). Enhancing security and privacy in biometrics-based authentication systems. *IBM Systems Journal*, 40(3):614–634.

[Ratha et al., 2001b] Ratha, N. K., Connell, J. H., and Bolle, R. M. (2001b). Enhancing security and privacy in biometrics-based authentication systems. *IBM Systems Journal*, 40(3):614–634.

[Rathgeb et al., 2014] Rathgeb, C., Breitinger, F., Busch, C., and Baier, H. (2014). On application of bloom filters to iris biometrics. *IET Biometrics*, 3(4):207–218.

[Rathgeb and Uhl, 2011] Rathgeb, C. and Uhl, A. (2011). A survey on biometric cryptosystems and cancelable biometrics. *EURASIP Journal on Information Security*, 2011(1):3.

[Schauren, 2016] Schauren, L. F. (2016). Segurança no sistema brasileiro de votação eletrônica. Trabalho de Conclusão de Curso. Instituto de Informática. Universidade Federal do Rio Grande do Sul. Disponível em: https://lume.ufrgs.br/handle/10183/151030.

[T. R. Jacqueline, 2012] T. R. Jacqueline, Salem Nathálea, W. M. B. V. (2012). Modelo intencional genérico de sistemas biométricos. In *Anais do WER12 - Workshop em Engenharia de Requisitos, Buenos Aires, Argentina, Abril 24-27, 2012*.

[Tribunal Reginal Eleitoral, 2023] Tribunal Reginal Eleitoral (2023). Perguntas e respostas - parte 1. Disponível em: https://www.tre-sp.jus.br/eleicoes/eleicoes-anteriores/eleicoes-2018/perguntas-e-respostas-parte-1. Acesso em: 20 jul 2023.

[Tribunal Superior Eleitoral, 2023] Tribunal Superior Eleitoral (2023). Urna eletrônica. Disponível em: https://www.tse.jus.br/internet/temporarios/urna-seguranca/identificacao-biometrica.html. Acesso em: 20 jul 2023.

[Wojewidka, 2020] Wojewidka, J. (2020). The deepfake threat to face biometrics. *Biometric Technology Today*, 2020(2):5–7.

[Yang et al., 2022] Yang, W., Wang, S., Kang, J. J., Johnstone, M. N., and Bedari, A. (2022). A linear convolution-based cancelable fingerprint biometric authentication system. *Computers & Security*, 114:102583.

[Zadeh, 1965] Zadeh, L. (1965). Fuzzy sets. *Information and Control*, 8(3):338–353.

Sobre os organizadores

Alex Borges Vieira é professor associado do Departamento de Ciência da Computação da Universidade Federal de Juiz de Fora. Atuou como pesquisador visitante no Laboratório Nacional de Computação Científica onde, em 2019, realizou período de pós-doutoramento financiado pelo CNPq. Recebeu seu título de doutor em Ciência da Computação pela Universidade Federal de Minas Gerais em 2010, onde também se tornou mestre em 2005.

Edelberto Franco Silva é professor Adjunto do Departamento de Ciência da Computação da Universidade Federal de Juiz de Fora. Obteve os títulos de Mestre e Doutor em Computação pela Universidade Federal Fluminense (UFF), respectivamente em 2011 e 2016. Participou/Participa de diversos projetos de pesquisa financiados pela RNP, FINEP, FAPERJ, TBE, FAPEMIG, PTI, FAPESP, CAPES e CNPq. Participou da implantação da rede eduroam no Brasil e na América Latina. Foi coordenador do comitê assessor de pesquisa da área de Engenharias e Computação na UFJF junto à pró-reitoria de pesquisa e pós-graduação. Possui experiência na área de Ciência da Computação, com ênfase em Segurança em Redes de Computadores e Redes de Computadores.

Dianne Scherly Varela de Medeiros atualmente é professora adjunta da Universidade Federal Fluminense. Ela obteve seu título de D.Sc. em Engenharia Elétrica pela Universidade Federal do Rio de Janeiro, em 2017. Como parte de seu doutorado, ela esteve em intercâmbio de 2015 até o fim de 2016 na Université Pierre et Marie Curie (UPMC), Paris, França. Recebeu seu título de mestre em Engenharia de Telecomunicações pela UFF, em 2013 e seu B.Sc. em Engenharia de Telecomunicações pela, em 2011. Possui, também, certificado de especialista em Redes de Computadores, recebido pela Pontifícia Universidade Católica do Rio de Janeiro, Brasil, em 2013.

Roberto Samarone Dos Santos Araujo é professor associado da Universidade Federal do Pará (UFPA). Possui doutorado em ciência da computação pela Universidade Técnica de Darmstadt (Alemanha) e mestrado pela Universidade Federal de Santa Catarina (UFSC) ambos na área de segurança e criptografia aplicada.

Realização

SBC
Sociedade Brasileira de Computação

Organização

ufjf
Departamento de Ciência da Computação
Programa de Pós-Graduação em Ciência da Computação
www.pgcc.ufjf.br

Apoio

RNP
ORGANIZAÇÃO SOCIAL DO MCTI

Patrocinadores

nic.br cgi.br

Tempest

ESCOLA SUPERIOR DE REDES

C.E.S.A.R school

altasnet

HPE aruba networking

FACULDADE IBPTECH

FAPEMIG

CAPES