Universidade Estadual do Paraná
Campus Apucarana

LUKA ALVES CLARO

AVALIAÇÃO EXPERIMENTAL DE TÉCNICAS DE SEGURANÇA EM COMUNICAÇÃO DE BORDA

APUCARANA–PR

2025

LUKA ALVES CLARO

AVALIAÇÃO EXPERIMENTAL DE TÉCNICAS DE SEGURANÇA EM COMUNICAÇÃO DE BORDA

Trabalho de Conclusão de Curso apresentado ao curso de Bacharelado em Ciência da Computação da Universidade Estadual do Paraná para obtenção do título de Bacharel em Ciência da Computação.

Orientadora: Profa. Dra. Lailla Milainny Siqueira Bine

APUCARANA–PR

2025

Ficha catalográfica elaborada pelo Sistema de Bibliotecas da UNESPAR e
Núcleo de Tecnologia de Informação da UNESPAR, com Créditos para o ICMC/USP
e dados fornecidos pelo(a) autor(a).

Alves Claro, Luka
Avaliação Experimental de Técnicas de Segurança em Comunicação de Borda / Luka Alves Claro. -- Apucarana-PR,2025.
57 f.: il.

Orientador: Lailla Milainny Siqueira Bine.
Trabalho de Conclusão de Curso, Ciência da Computação - Universidade Estadual do Paraná, 2025.

1. Computação de Borda. 2. Segurança da Informação. 3. Internet das Coisas (IoT). 4. Criptografia Híbrida. 5. Man-in-the-Middle (MitM). I - Milainny Siqueira Bine, Lailla (orient). II - Título.

LUKA ALVES CLARO

AVALIAÇÃO EXPERIMENTAL DE TÉCNICAS DE SEGURANÇA EM COMUNICAÇÃO DE BORDA

Trabalho de Conclusão de Curso apresentado ao curso de Bacharelado em Ciência da Computação da Universidade Estadual do Paraná para obtenção do título de Bacharel em Ciência da Computação.

BANCA EXAMINADORA

Profa. Dra. Lailla Milainny Siqueira Bine
Universidade Estadual do Paraná
Orientadora

Prof. Dr. Nicollas Mocelin Sdroievski
Universidade Estadual do Paraná

Prof. Ms. Kleber Marcio de Souza
Universidade Estadual do Paraná

Apucarana–PR, 01 de dezembro de 2025

Este trabalho é dedicado às pessoas que exploram as fronteiras da Computação em Nuvem e de Borda transformando-as em soluções inovadoras.

AGRADECIMENTOS

Agradeço à minha orientadora, Professora Dra. Lailla Milainny Siqueira Bine, por ter me auxiliado em toda esta jornada no desenvolvimento deste trabalho. E ao Professor Dr. José Luis Seixas Junior, pelas dicas e sugestões de escolha de temas que fizeram com que eu conhecesse e pesquisasse mais a fundo sobre a Computação de Borda.

Agradeço também à minha família por todo o apoio que me deram, especialmente nestes últimos quatro anos longe de casa durante a graduação. Foram essenciais para que eu pudesse lidar com os altos e baixos desta jornada e concluir a graduação.

Aos meus amigos de turma que estiveram comigo desde o início da graduação e àqueles que saíram no meio do caminho, minha gratidão. Agradeço também aos meus companheiros de estágio no setor de TI da UNESPAR Apucarana, que acompanharam e me ajudaram desde o início da elaboração deste trabalho.

Não posso deixar de agradecer também às minhas amigas Paloma de Castro Leite e Camilla Chaves Queiroz, que me apoiaram muito, dedicaram várias horas para me ouvirem e debaterem comigo sobre as diversas ideias e epifanias que tive no decorrer deste trabalho.

"Três pessoas podem guardar um segredo, se duas delas estiverem mortas."
(Benjamin Franklin)

. **Avaliação Experimental de Técnicas de Segurança em Comunicação de Borda**. 55 p. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) – Universidade Estadual do Paraná, Apucarana–PR, 2025.

RESUMO

A Computação de Borda (*Edge Computing*) é um paradigma em ascensão para aplicações de Internet das Coisas (*Internet of Things* - IoT), permitindo o processamento de dados próximo à sua origem, como em sistemas de Reconhecimento de Placas Veiculares (*License Plate Recognition* - LPR). Contudo, essa arquitetura introduz novos desafios de segurança, especificamente no fluxo de dados entre o dispositivo de borda e o servidor central. Este trabalho apresenta uma pesquisa aplicada e experimental focada na segurança deste fluxo. O objetivo principal foi demonstrar a vulnerabilidade de dados sensíveis (placas de veículos) transmitidos em texto puro (*plain text*) e, em seguida, implementar e validar uma contramedida de segurança robusta. A metodologia envolveu a construção de um protótipo com um Raspberry Pi (borda) e um servidor (*backend*). Foi executado um ataque *Man-in-the-Middle* (MitM) utilizando *ARP spoofing* (*Address Resolution Protocol spoofing*), que comprovou a interceptação e visualização dos dados em texto puro. Como solução, foi implementado um esquema de criptografia híbrida, combinando o *Advanced Encryption Standard* (AES) para a cifragem dos dados e o algoritmo *Rivest-Shamir-Adleman* (RSA) para a troca segura de chaves. A repetição do ataque MitM no sistema protegido validou a eficácia da solução, garantindo a confidencialidade e a integridade dos dados, que se mostraram indecifráveis ao atacante. Adicionalmente, foi demonstrada uma vulnerabilidade de disponibilidade por meio de um ataque de Negação de Serviço (*Denial of Service* - DoS), que conseguiu saturar os recursos do dispositivo de borda. Conclui-se que a criptografia híbrida é uma solução eficaz para a proteção de dados em trânsito, mas que a segurança em dispositivos de borda requer uma abordagem multifacetada, considerando também a disponibilidade.

Palavras-chave: Computação de Borda, Segurança da Informação, Internet das Coisas (IoT), Criptografia Híbrida, Man-in-the-Middle (MitM).

. **Experimental Evaluation of Security Techniques in Edge Communication**. 55 p. Final Project (Bachelor of Science in Computer Science) – State University of Paraná, Apucarana–PR, 2025.

ABSTRACT

Edge Computing is an emerging paradigm for Internet of Things (IoT) applications, enabling data processing close to its source, such as License Plate Recognition (LPR) systems. However, this architecture introduces new security challenges, particularly in the data flow between the edge device and the central server. This work presents an applied and experimental study focused on the security of this data flow. The main objective was to demonstrate the vulnerability of sensitive data (vehicle license plates) transmitted in plain text and, subsequently, to implement and validate a robust security countermeasure. The methodology involved the development of a prototype using a Raspberry Pi (edge) and a server (backend). A Man-in-the-Middle (MitM) attack using Address Resolution Protocol (ARP) spoofing was executed, confirming the ability to intercept and read the plain-text data. As a solution, a hybrid cryptography scheme was implemented, combining Advanced Encryption Standard (AES) for data encryption and Rivest–Shamir–Adleman (RSA) algorithm for secure key exchange. Repeating the MitM attack on the secured system validated the solution's effectiveness, ensuring data confidentiality and integrity, as the data remained indecipherable to the attacker. Furthermore, an availability vulnerability was demonstrated via a Denial of Service (DoS) attack, which successfully saturated the edge device's resources. The results indicate that hybrid cryptography is an effective solution for protecting data-in-transit, but that security in edge devices requires a multifaceted approach that also addresses device availability.

Keywords: Edge Computing. Information Security. Internet of Things (IoT). Hybrid Cryptography. Man-in-the-Middle (MitM).

LISTA DE ILUSTRAÇÕES

Figura 1 – Hierarquia dos níveis de controle de cada categoria de computação em nuvem 26
Figura 2 – Casos de Uso da Computação de Borda 28
Figura 3 – Arquitetura IoT com *Mist, Fog e Cloud Computing* 29
Figura 4 – Etapas do processamento de imagem feito pelo OCR 30
Figura 5 – Comparação de uma arquitetura Tradicional com a de uma VM e de um Contêiner 31
Figura 6 – Topologia de rede montada para o experimento 37
Figura 7 – Fluxograma das etapas do desenvolvimento da pesquisa 38
Figura 8 – Arquivos utilizados no ambiente do Servidor 41
Figura 9 – Arquivos utilizados no ambiente do Raspberry Pi 42
Figura 10 – Tráfego inseguro entre a Borda e o Servidor 43
Figura 11 – Diagrama demonstrando o ataque MitM 43
Figura 12 – Diagrama demonstrando o Ataque DoS do tipo *SYN Flood* 44
Figura 13 – Placa analisada e processada pelo Raspberry Pi 45
Figura 14 – Placa Interceptada no Wireshark 46
Figura 15 – Ataque de *ARP spoofing* no Raspebrry Pi 46
Figura 16 – Ataque de *Arpspoofing* no Servidor 47
Figura 17 – Tráfego criptografado pela criptografia híbrida - *payload* do cliente . . . 48
Figura 18 – Tráfego criptografado pela criptografia híbrida - *payload* do servidor . . 48
Figura 19 – Raspberry Pi e a aplicação operando normalmente 49
Figura 20 – Terminal do atacante executando o ataque com o hping3 49
Figura 21 – Raspberry Pi durante o ataque de *SYN Flood* 50

LISTA DE ABREVIATURAS E SIGLAS

5G	Quinta Geração (de redes móveis)
AAM	Aborda Ataque Man-in-the-Middle
ACB	Aplica Computação de Borda
AES	Advanced Encryption Standard
API	Application Programming Interface
ARP	Address Resolution Protocol
BPF	Berkeley Packet Filter
CIA	Confidentiality, Integrity e Availability
CID	Confidencialidade, Integridade e Disponibilidade
CNNs	Convolutional Neural Network
CPU	Central Processing Unit
DoS	Denial of Service
DTLS	Datagram Transport Layer Security
ECC	Edge-Cloud Continuum
ECDH	Elliptic Curve Diffie-Hellman
ECDSA	Elliptic Curve Digital Signature Algorithm
FCI	Foco em Confidencialidade/Integridade
FDD	Foco em Disponibilidade/Desempenho
FPS	Foco Principal em Segurança
GCM	Galois/Counter Mode
Host OS	Host Operating System
HTTP	Hyper Text Transfer Protocol
IaaS	Infrastructure as a Service
ICH	Implementa Criptografia Híbrida

IoV	Internet of Vehicles
IoT	Internet of Things
IP	Internet Protocol
JSON	JavaScript Object Notation
JWT	JSON Web Token
LAN	Local Area Network
LPR	License Plate Recognition
MAC	Media Access Control
MitM	Man-in-the-Middle
OCR	Optical Character Recognition
PaaS	Platform as a Service
PC	Personal Computer
RAM	Random Access Memory
RSA	Rivest–Shamir–Adleman
SaaS	Software as a Service
SCP	Secure Copy Protocol
SHA-256	Secure Hash Algorithm 256-bit
SHA-512	Secure Hash Algorithm 512-bit
SMS	Short Message Service
SoCs	Systems-on-a-Chip
SSL	Secure Sockets Layer
SYN	Synchronization
TCC	Trabalho de Conclusão de Curso
TCP	Transmission Control Protocol
TI	Tecnologia da Informação
TLS	Transport Layer Security

UCF	Usa Criptografia no Fluxo
USB	Universal Serial Bus
UL CL	Uplink Classifier
UNESPAR	Universidade Estadual do Paraná
VM	Virtual Machine
WLAN	Wireless Local Area Network
XDP	Express Data Path

SUMÁRIO

1 INTRODUÇÃO 23
1.1 Objetivos 24
1.1.1 Objetivo Geral 24
1.1.2 Objetivos Específicos 24
1.2 Organização do Trabalho 24

2 FUNDAMENTAÇÃO TEÓRICA 26
2.1 Computação em Nuvem e Arquiteturas Distribuídas 26
2.2 Computação de Borda 27
2.3 Aplicações da Computação de Borda em Visão Computacional 29
2.4 Conteinerização 31
2.5 Segurança da Informação em Ambientes Distribuídos 32
2.6 Trabalhos Relacionados 33

3 MÉTODO DE PESQUISA 36
3.1 Arquitetura e Ambiente Experimental 36
3.2 Implementação da Mitigação de Segurança 38
3.3 Métricas e Critérios de Avaliação 39
3.4 Uso de Ferramentas de Inteligência Artificial 40

4 EXPERIMENTOS 41
4.1 Configuração do Ambiente Experimental 41
4.2 Cenário 1: Fluxo de Dados Básico (Inseguro) 42
4.3 Cenário 2: Demonstração da Vulnerabilidade (MitM em Tráfego Inseguro e Seguro) 43
4.4 Cenário 3: Teste de Disponibilidade (Ataque DoS) 44

5 RESULTADOS 45
5.1 Resultados do Cenário 1: Tráfego Inseguro (Man-in-theMiddle) 45
5.2 Resultados do Cenário 2: Tráfego Seguro (Validação da Criptografia Híbrida) 46
5.3 Resultados do Cenário 3: Ataque DoS 49

6 CONCLUSÃO 51

REFERÊNCIAS 52

1 INTRODUÇÃO

A tecnologia avança de forma acelerada, impulsionando a automação nos mais diversos setores. O controle de acesso veicular é uma das áreas que mais têm se beneficiado dessas inovações [1]. Com a crescente demanda por soluções inteligentes e eficientes em ambientes como condomínios, empresas e shoppings, a automação desse processo tornou-se não apenas um meio de aumentar a agilidade, mas uma necessidade para garantir a segurança e a integridade dos dados envolvidos.

Diante desse cenário, a Computação de Borda (*Edge Computing*) surge como uma solução eficiente. A sua principal vantagem reside no pré-processamento de informações sensíveis próximo à fonte dos dados, antes do envio para a nuvem, o que minimiza riscos e reduz significativamente a latência [2]. Estudos estimam que uma parcela expressiva dos dados criados será processada fora dos *data centers* centralizados [3], isto é, nas bordas da rede, oferecendo respostas mais rápidas e maior controle. No entanto, ao passo que soluciona questões de desempenho, a arquitetura distribuída da Computação de Borda introduz vetores de risco específicos que exigem estratégias de Segurança da Informação adaptadas a este contexto [4].

Com o aumento dos ciberataques que exploram vulnerabilidades em dispositivos de borda e meios de comunicação, a segurança torna-se um tema central. A tríade da segurança da informação — Confidencialidade, Integridade e Disponibilidade (CIA) — é o modelo fundamental para proteger dados em sistemas computacionais [5]. Isso é especialmente crítico ao lidar com dados sensíveis, como registros de entrada e saída e imagens capturadas por sistemas de monitoramento.

Nesse contexto, tecnologias como câmeras IP e sistemas de reconhecimento de placas (LPR - *License Plate Recognition*) [6] geram um grande volume de dados em tempo real. A abordagem tradicional, que depende do envio integral dessas informações para a nuvem, enfrenta problemas como congestionamento de rede e atrasos na tomada de decisão. Por outro lado, a Computação de Borda, embora resolva a latência, expande a superfície de ataque. Dispositivos de borda, frequentemente localizados em ambientes fisicamente acessíveis, tornam-se alvos de interceptações e manipulações, comprometendo a operação de sistemas críticos e a privacidade dos dados.

A discussão central que impulsiona este projeto é a necessidade de conciliar os benefícios operacionais da Computação de Borda com os requisitos rigorosos de segurança. O argumento principal é que a simples migração do processamento para a borda é insuficiente; é essencial que essa migração seja acompanhada pela implementação de uma arquitetura de segurança específica para este modelo híbrido (borda-nuvem) [7].

O problema central abordado neste trabalho é a vulnerabilidade do fluxo de dados entre dispositivos de borda e servidores centrais, especialmente quando trafegam por redes locais suscetíveis a interceptações. Essa questão torna-se crítica no caso de uso adotado por esta pesquisa: um sistema de controle de acesso veicular, onde a ausência de mecanismos de segurança robustos no trânsito das informações (placas de veículos) pode comprometer a confidencialidade e a integridade de todo o sistema.

Este trabalho defende que, por meio da aplicação de técnicas de criptografia no fluxo de dados, é possível mitigar significativamente os riscos associados, construindo um sistema confiável e resiliente, capaz de preservar a autenticidade dos dados veiculares mesmo sob ameaça de interceptação de tráfego.

1.1 Objetivos

As próximas seções descrevem os objetivos gerais e específicos desse trabalho.

1.1.1 Objetivo Geral

O objetivo geral deste trabalho é avaliar experimentalmente a segurança do fluxo de dados em um sistema de Computação de Borda, demonstrando vulnerabilidades práticas e validando a eficácia de uma solução de criptografia híbrida para mitigação de riscos.

1.1.2 Objetivos Específicos

Para alcançar o objetivo geral, foram definidos os seguintes objetivos específicos:

- Construir um protótipo funcional de controle de acesso veicular utilizando hardware de baixo custo (Raspberry Pi) como dispositivo de borda;
- Analisar a vulnerabilidade da comunicação em um cenário inseguro (HTTP em texto puro) por meio de interceptação passiva;
- Implementar uma camada de criptografia híbrida (AES e RSA) para garantir a confidencialidade e integridade dos dados em trânsito;
- Avaliar a resiliência do dispositivo de borda quanto à disponibilidade, submetendo-o a ataques de Negação de Serviço (DoS).

1.2 Organização do Trabalho

O restante deste trabalho está organizado da seguinte forma: O Capítulo 2 apresenta a Fundamentação Teórica, abordando os conceitos de Computação em Nuvem, Borda, Visão Computacional e Segurança. O Capítulo 3 detalha o Método de Pesquisa e

a arquitetura proposta. O Capítulo 4 descreve os Experimentos realizados e a configuração dos cenários. O Capítulo 5 apresenta e discute os Resultados obtidos nas análises de segurança. Por fim, o Capítulo 6 traz as Considerações Finais e sugestões para trabalhos futuros.

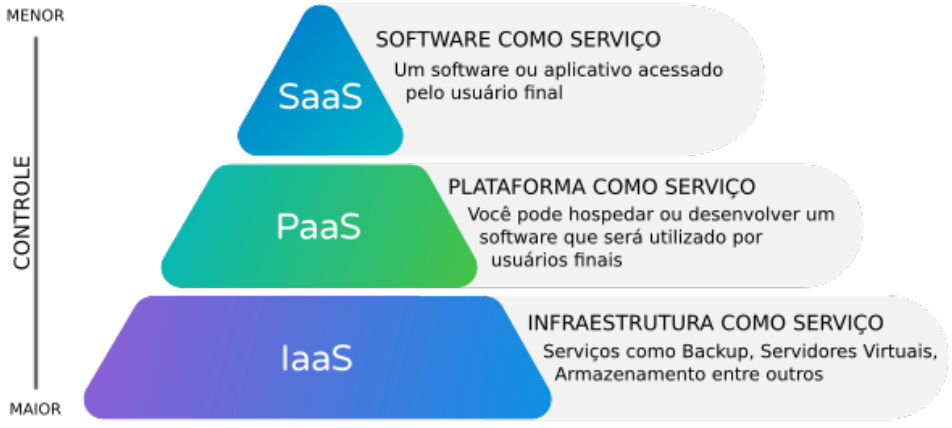
2 FUNDAMENTAÇÃO TEÓRICA

Esta seção aborda os pilares teóricos que fundamentam a presente pesquisa. Serão abordados os conceitos essenciais de Computação em Nuvem e sua evolução para a Computação de Borda, o contexto dos sistemas automatizados de controle de acesso veicular junto a visão computacional, e os princípios indispensáveis de Segurança da Informação e Criptografia que norteiam a proposta deste trabalho.

2.1 Computação em Nuvem e Arquiteturas Distribuídas

A Computação em Nuvem (*Cloud Computing*) representa um modelo de entrega de serviços computacionais pela internet, caracterizado pela elasticidade, escalabilidade e acesso sob demanda a um conjunto compartilhado de recursos configuráveis, como servidores, armazenamento, aplicações e redes. Esse paradigma transformou a infraestrutura de TI, permitindo que organizações substituam o alto investimento em *hardware* local por um modelo de custo operacional mais flexível [8]. Os serviços em nuvem são geralmente categorizados nos modelos Infraestrutura como Serviço (IaaS), Plataforma como Serviço (PaaS) e Software como Serviço (SaaS) [9]. A Figura 1 ilustra essa hierarquia.

Figura 1 – Hierarquia dos níveis de controle de cada categoria de computação em nuvem



Fonte: Adaptado de [10]

Apesar de sua predominância, o modelo de nuvem puramente centralizado encontra desafios em aplicações que geram um grande volume de dados e exigem processamento em tempo real [11]. A latência, resultante da distância física entre o local de geração dos dados e os data centers da nuvem, pode tornar inviável a operação de sistemas críticos que dependem de respostas em torno de milissegundos, como o controle de acesso veicular.

Portanto, o envio contínuo de dados brutos, como *streams* de vídeo, para a nuvem consome uma quantidade significativa de largura de banda [12] e pode gerar custos adicionais em alguns casos (quando se tem um armazenamento em uma nuvem pública, por exemplo), então se faz necessária uma curadoria e uma análise sobre o tipo e a quantidade de dados que serão enviados à nuvem, para não comprometer o fluxo de trabalho de um sistema.

Contudo, apesar da escalabilidade e flexibilidade oferecidas pela Computação em Nuvem, este modelo centralizado enfrenta limitações quando aplicado a cenários que exigem baixa latência e largura de banda otimizada. O envio massivo de dados brutos para processamento remoto pode gerar gargalos na rede e atrasos inaceitáveis para aplicações de tempo real. É para mitigar essas limitações e aproximar o processamento da fonte de dados que surge o paradigma da Computação de Borda, detalhado na seção a seguir.

2.2 Computação de Borda

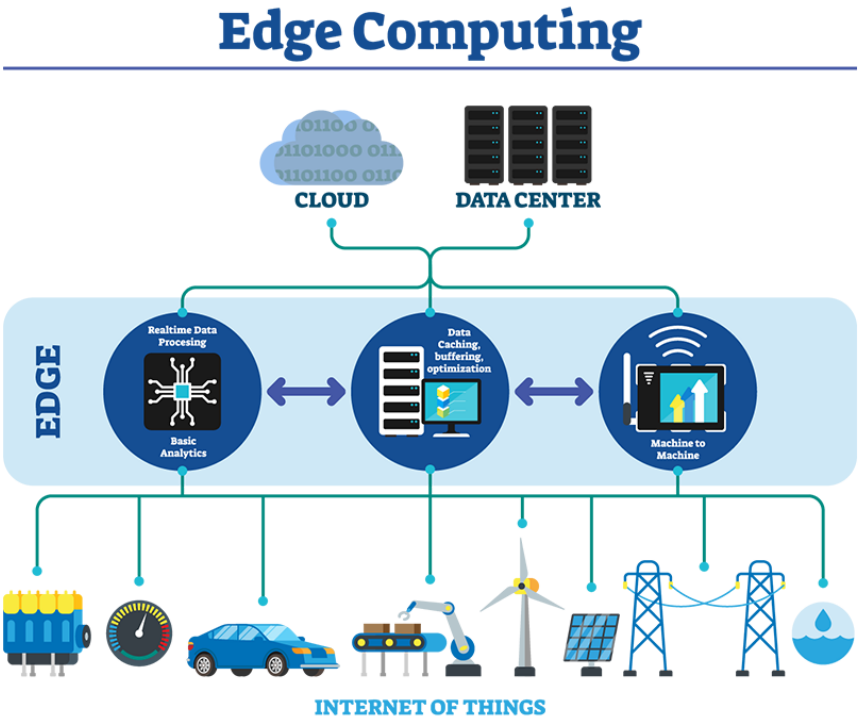
A Computação de Borda pode ser vista de duas formas: primeiro, como uma topologia de computação distribuída projetada especificamente para endereçar as limitações inerentes ao modelo centralizado da Computação em Nuvem[13], e segundo, como uma sub área da Computação em Nuvem[14].

Em um cenário onde dispositivos de IoT(*Internet of Things*), como câmeras, sensores e atuadores, geram volumes exponenciais de dados (o que chamamos de *Big Data*), a dependência exclusiva da nuvem para processamento traz desafios significativos de latência, consumo de largura de banda e custos operacionais [12]. Então, a Computação de Borda propõe uma arquitetura descentralizada onde o processamento de dados é deslocado da nuvem central para a "borda" da rede, ou seja, para um ponto mais próximo da fonte de onde os dados são gerados e coletados [15].

Esta subárea traz uma proposta de ideia que se baseia na otimização do fluxo de dados. Segundo Cristino e Caminha [16], ao transferir o processamento de dados para dispositivos de borda, como *gateways* ou computadores embarcados, é possível minimizar drasticamente a latência e possibilitar a tomada de decisões locais em milissegundos, sem necessidade de retorno completo à nuvem. Esta proximidade física é o que permite uma certa minimização da latência, pois as decisões críticas podem ser tomadas localmente, sem a necessidade de aguardar o ciclo completo de comunicação com a nuvem [16]. A arquitetura fundamental e os casos de uso deste modelo podem ser visualizados na Figura 2.

É importante frisar que a borda não visa substituir a nuvem, mas sim estendê-la, criando um "contínuo computacional", conceito chamado de ECC (*Edge-Cloud Continuum*) [18]. A nuvem permanece essencial para tarefas que exigem alto poder de pro-

Figura 2 – Casos de Uso da Computação de Borda

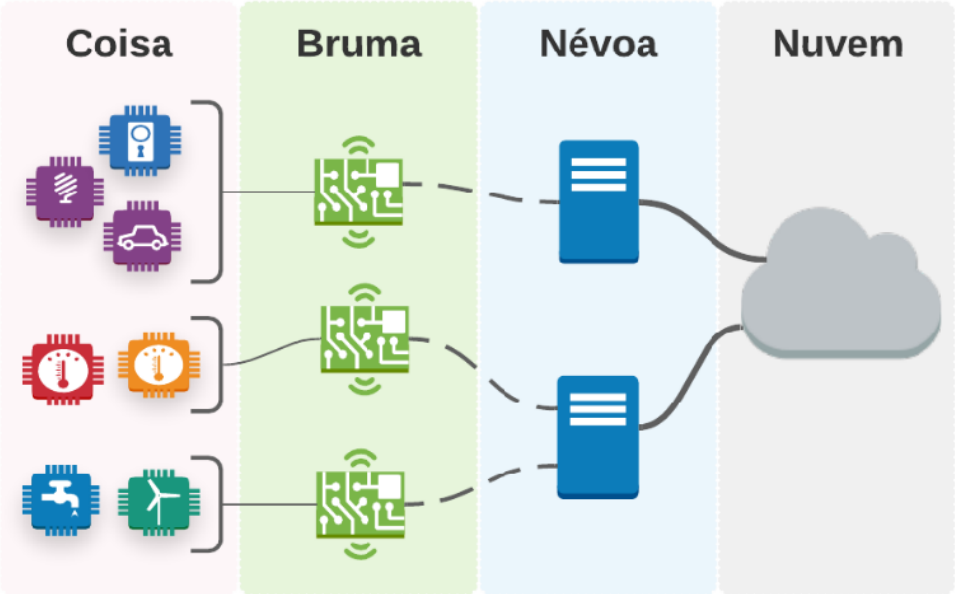


Fonte: Adaptado de [17]

cessamento e armazenamento de longo prazo, como o treinamento de modelos complexos de *Machine Learning* e a realização de análises de *Big Data*. Nesse modelo híbrido, os dispositivos de borda atuam como uma primeira camada inteligente, realizando tarefas de pré-processamento, filtragem e coleta de dados; essas ações acabam tornando-se insumos eficientes para as tarefas que serão feitas na nuvem, beneficiando também modelos de inteligência artificial [19].

Essa arquitetura resulta em benefícios operacionais claros. Primeiramente, há uma otimização massiva do uso da largura de banda, pois o tráfego de dados desnecessários ou redundantes pela rede é eliminado. Em segundo lugar, a eficiência geral do sistema aumenta, já que a nuvem é liberada de tarefas de processamento de baixo nível. Por fim, um dos benefícios mais vantajosos é a capacidade de operação autônoma. Em caso de falha na conexão com a internet ou instabilidade na comunicação com a nuvem, o sistema na borda pode continuar operando suas funções essenciais, garantindo a resiliência e a disponibilidade do serviço. Esta resiliência é válida para as outras áreas das arquitetu-ras distribuídas que funcionam também como uma extensão da *Cloud Computing*, tais como a *Fog Computing* (Computação de Névoa) e a *Mist Computing* (Computação de Bruma) [20]. A Figura 3 demonstra esta abstração das extensões da nuvem, ilustrando como as camadas de processamento se distribuem desde o dispositivo final até o servidor central.

Figura 3 – Arquitetura IoT com *Mist, Fog e Cloud Computing*



Fonte: Adaptado de [20]

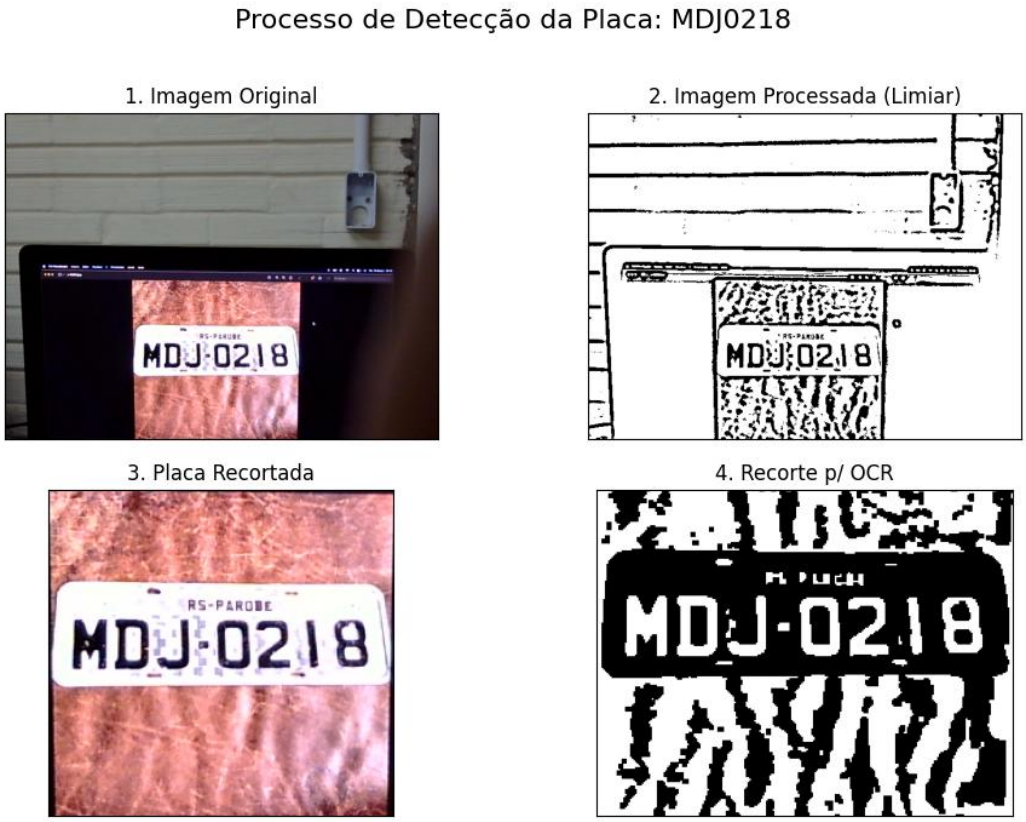
2.3 Aplicações da Computação de Borda em Visão Computacional

A Visão Computacional é uma área da ciência da computação e da inteligência artificial que visa capacitar os computadores a "ver", interpretar e compreender o mundo visual [21, 22]. De forma análoga à visão humana, ela utiliza dados de entrada, como imagens e vídeos, para extrair, processar, analisar e entender informações, permitindo que uma máquina tome decisões ou execute ações baseadas nessa compreensão. Esta área da computação não se limita a simplesmente replicar a captura de imagens, mas envolve uma série complexa de tarefas para extrair significado e contexto do conteúdo visual, como detecção de objetos, segmentação, rastreamento e reconhecimento de padrões [22].

O desenvolvimento desta área foi acelerado pelos avanços em *Machine Learning* (Aprendizado de Máquina) e, mais especificamente, em *Deep Learning* (Aprendizado Profundo) [23, 24]. Modelos como as Redes Neurais Convolucionais (CNNs) tornaram-se o estado da arte para tarefas de reconhecimento de imagem, superando métodos tradicionais em precisão e robustez [23, 6, 24, 21]. A aplicação específica de LPR (*License Plate Recognition*), que é fundamental para o domínio de aplicação deste trabalho, é um exemplo clássico de um pipeline de Visão Computacional [23, 6]. Esse processo geralmente envolve múltiplas etapas: 1) Aquisição da Imagem (captura pela câmera); 2) Detecção de Objetos (localização da placa do veículo dentro do enquadramento da imagem); 3) Segmentação (isolamento da placa do restante da cena); e 4) Reconhecimento, onde técnicas de Reconhecimento Óptico de Caracteres (OCR) são aplicadas para converter os pixels dos caracteres da placa em texto digital [6, 23]. A Figura 4 demonstra esse reconhecimento

de caracteres.

Figura 4 – Etapas do processamento de imagem feito pelo OCR



Fonte: Imagem do Autor

O sucesso e a alta precisão desses modelos de *Deep Learning*, no entanto, vêm acompanhados de uma demanda computacional significativa, especialmente durante o processo de inferência (a aplicação do modelo treinado em dados novos) [23, 6]. Essa alta carga computacional é um dos principais fatores que impulsionam a adoção da Computação de Borda [23, 25]. Enviar um fluxo de vídeo contínuo e de alta definição para a nuvem para realizar a análise LPR é, em muitos casos, inviável devido a três fatores principais: os custos de largura de banda, a latência inaceitável para uma aplicação em tempo real (como a abertura de uma cancela) [25, 23, 26] e os riscos de privacidade.

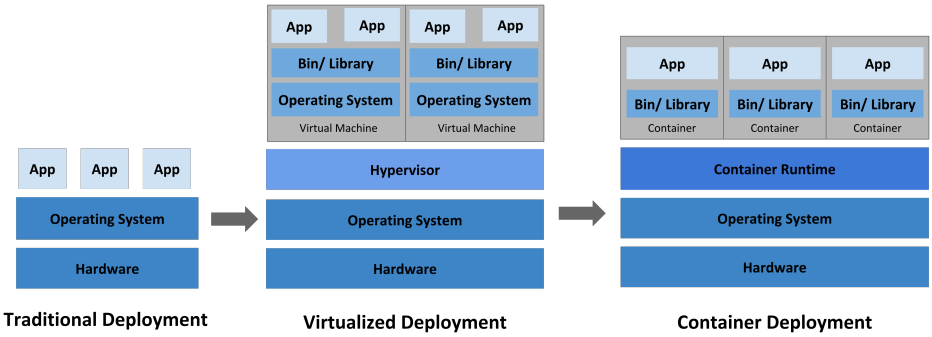
Ao processar o vídeo localmente na borda, evita-se a transmissão de dados brutos e sensíveis, como o fluxo de vídeo completo, que pode conter imagens dos ocupantes do veículo ou do entorno, minimizando a exposição desses dados a interceptações. Torna-se, portanto, muito mais eficiente e seguro executar a inferência do modelo de Visão Computacional diretamente no dispositivo de borda, como o Raspberry Pi, processando o vídeo localmente e em tempo real [23, 25, 26, 27].

2.4 Conteinerização

No desenvolvimento de sistemas distribuídos modernos, especialmente em arquiteturas de borda, a forma como as aplicações são empacotadas e implantadas é um fator importante a ser considerado, tanto para a sustentação quanto para a manutenção e segurança do projeto. A conteinerização, popularizada por tecnologias como o Docker [28], Podman [29] e Kubernetes [30], surge como uma solução de virtualização em nível de sistema operacional que oferece uma resposta eficaz a esses desafios.

O estudo de Casella [31] demonstra que, diferentemente das máquinas virtuais tradicionais (VMs), que dependem de um hipervisor — uma camada de *software* que emula o *hardware* físico (CPU, memória e disco) para executar um sistema operacional "convidado" (*Guest* OS) completo —, os contêineres são muito mais leves. Uma VM consome quantidades significativas de recursos e resulta em tempos de inicialização mais longos, pois precisa carregar um sistema operacional inteiro. Em contrapartida, um contêiner compartilha o kernel do sistema operacional do hospedeiro (Host OS), encapsulando apenas a aplicação e suas dependências (bibliotecas, bins e arquivos de configuração) em um ambiente isolado [28]. Para melhor entendimento, é comum comparar a arquitetura de contêineres com a de Máquinas Virtuais (VMs), como ilustrado na Figura 5.

Figura 5 – Comparação de uma arquitetura Tradicional com a de uma VM e de um Contêiner



Fonte: Adaptado de [30]

A adoção desta abordagem traz um conjunto de benefícios fundamentais para a implantação em dispositivos de borda. A portabilidade é um dos principais, pois a aplicação e suas dependências são empacotadas em uma imagem imutável [28]. Isso garante que o ambiente de execução seja idêntico e reprodutível, eliminando conflitos de dependência, seja em um servidor de desenvolvimento ou no dispositivo de borda (como o Raspberry Pi). Aliada a isso, a leveza [31] é uma característica essencial para *hardware* com recursos computacionais e memória limitados. Como os contêineres compartilham o *kernel* do hospedeiro e não carregam o *overhead* de um sistema operacional completo, eles iniciam em segundos e permitem uma maior densidade de serviços no mesmo dispositivo.

Por fim, a segurança é uma característica frequentemente discutida. Um contêiner executa o processo da aplicação em um *sandbox* (ambiente isolado), com seu próprio sistema de arquivos e pilha de rede. Segundo Alencar [28], o Docker possui recursos avançados de segurança. Porém, outros estudos, como o de Casella [31] e o de Miers *et al.* [32], demonstram que uma de suas principais fragilidades está na execução do *daemon* com privilégios administrativos, o que pode deixar brechas para modificações indevidas em arquivos restritos. Em suma, diante dos fatos apresentados, faz-se necessária uma análise cuidadosa das configurações dos arquivos e do ambiente em que os contêineres Docker forem implementados para garantir um ambiente seguro.

2.5 Segurança da Informação em Ambientes Distribuídos

Segundo Nascimento e Costa [33] e Ferreira [34], a segurança da informação visa proteger os dados e sistemas para garantir a continuidade do negócio, sustentando-se sobre a tríade de confidencialidade, integridade e disponibilidade (CIA) [5]. Neste trabalho, a Confidencialidade garante que os dados de placas e imagens dos veículos sejam inacessíveis a pessoas não autorizadas. A Integridade assegura que esses dados não possam ser alterados maliciosamente em trânsito, por exemplo, trocando uma placa autorizada por uma não autorizada. A Disponibilidade garante que o sistema de controle de acesso funcione ininterruptamente.

A arquitetura de borda, embora benéfica, introduz vetores de ataque específicos. O dispositivo de borda pode ser alvo de acesso físico não autorizado, ataques de negação de serviço (DoS) [35] na rede local e ataques Man-in-the-Middle (MitM) [36] para interceptar a comunicação entre a borda e a nuvem. Nesse caso, uma estratégia de segurança robusta, que vá além da proteção do perímetro da nuvem e inclua o fortalecimento dos dispositivos de borda, é indispensável.

Uma das principais ferramentas para garantir essa segurança é a criptografia, que é a ciência e a arte de escrever mensagens em forma codificada [26]. No contexto da Computação de Borda, a aplicação de técnicas criptográficas robustas é essencial para proteger o fluxo de dados em trânsito contra interceptações.

Historicamente, a criptografia pode ser classificada em duas categorias principais:

- **Criptografia Simétrica:** Utiliza uma única chave secreta compartilhada entre as partes. Algoritmos como o AES (Advanced Encryption Standard), citado por Queiroz [25], são o padrão da indústria devido à sua extrema velocidade e eficiência computacional, sendo ideais para cifrar grandes volumes de dados. Seu principal desafio é o problema da distribuição segura da chave, que pode apresentar problemas de escalabilidade [26].

- **Criptografia Assimétrica:** Utiliza um par de chaves (pública e privada). Dados criptografados com a chave pública só podem ser descriptografados com a chave privada correspondente. Algoritmos como o RSA (Rivest-Shamir-Adleman) e aqueles baseados em curvas elípticas (ECDH, ECDSA) são usados para estabelecer autenticidade e troca segura de informações [25]. Sua desvantagem é ser computacionalmente intensiva e mais lenta, o que a torna inviável para criptografar todo o volume de dados de uma comunicação [26].

Para mitigar as desvantagens de cada abordagem, a solução padrão da indústria é a criptografia híbrida, que combina a segurança da criptografia assimétrica para a troca de chaves com a velocidade da criptografia simétrica para a proteção dos dados. O modelo envolve o uso da criptografia assimétrica apenas no início para negociar uma chave de sessão temporária (simétrica). Uma vez estabelecida, a comunicação passa a usar a criptografia simétrica, muito mais rápida. Este é o princípio fundamental de protocolos como o TLS (Transport Layer Security) [26].

Além da confidencialidade, a criptografia fornece mecanismos para a integridade e autenticidade através de funções de *hash* criptográfico (como SHA-256), que criam uma "impressão digital" única dos dados. Qualquer alteração resultará, com altíssima probabilidade, em um *hash* diferente, permitindo a verificação da integridade [26]. Ao combinar *hashing* com criptografia assimétrica, criam-se assinaturas digitais, que provam inequivocamente que a mensagem veio do remetente esperado (autenticidade) e não foi alterada (integridade) [25]. Em uma arquitetura de borda, a aplicação correta desses conceitos é um requisito fundamental para proteger o fluxo de dados.

2.6 Trabalhos Relacionados

Nesta seção, são apresentados trabalhos que abordam a Computação de Borda em conjunto com aspectos de segurança, desempenho ou arquitetura para ambientes de IoT, contextualizando a pesquisa e destacando suas contribuições específicas. Na tabela 1 foram feitas as comparações entre os trabalhos relacionados, abordando as informações chaves de cada trabalho.

A crescente necessidade de processar dados mais perto de sua origem, impulsionada por requisitos de baixa latência, economia de banda e maior autonomia, consolidou a Computação de Borda como um paradigma relevante. No entanto, essa descentralização introduz novos desafios de segurança, especialmente no fluxo de dados entre a borda e outros sistemas, como servidores centralizados ou em nuvem.

Cesar [37] propõe uma arquitetura de controle de acesso para IoT utilizando Computação de Borda com o objetivo principal de melhorar a disponibilidade dos serviços. O trabalho utiliza a borda para descentralizar a arquitetura e otimizar a troca de dados

Tabela 1 – Tabela comparativa dos trabalhos relacionados

Ref.	ACB	FPS	FCI	FDD	UCF	ICH	AAM	Ferramentas
Cesar (2022)	✓	✓		✓				Controle de Acesso, GraphQL, JWT
Silvério & Guardia (2021)	✓	✓		✓				Filtragem de Kernel, Raspberry Pi, Energia
Kraus (2021)	✓			✓				Desempenho 5G, Latência
Negri (2025)	✓	✓					✓	Vulnerabs. Echo, Autenticação SMS/Hash
Schenfeld (2017)	✓	✓	✓		✓			Arquitetura Híbrida, TLS/DTLS
Trabalho Proposto	✓	✓	✓	✓	✓	✓	✓	Criptografia Híbrida, Defesa MitM

Legenda: ACB = Aplica Computação de Borda; FPS = Foco Principal em Segurança; FCI = Foco em Confidencialidade/Integridade; FDD = Foco em Disponibilidade/Desempenho; UCF = Usa Criptografia no Fluxo; ICH = Implementa Criptografia Híbrida; AAM = Aborda Ataque MitM.

com GraphQL, mitigando a sobrecarga em um ponto central. A segurança é abordada sob a perspectiva de controle de acesso baseado em políticas e contexto , utilizando autenticação via JWT. Embora utilize a borda e tenha um componente de segurança, difere deste trabalho por focar na disponibilidade e no controle de acesso, enquanto neste trabalho concentra-se na confidencialidade e integridade dos dados em trânsito contra ataques MitM por meio de criptografia híbrida.

Silverio e Guardia [38] focam diretamente no conceito de “Edge Security”, investigando a filtragem de pacotes na borda da rede. O trabalho compara diferentes mecanismos de filtragem disponíveis no kernel Linux (iptables, nftables, BPF, XDP) implementados em um dispositivo de borda (Raspberry Pi) , analisando o impacto no consumo de energia e na carga da CPU. A motivação é processar/descartar pacotes maliciosos o mais cedo possível , considerando as restrições de dispositivos de borda. Assim como neste trabalho, aplica a segurança diretamente na borda, mas a abordagem é diferente: filtragem base-

ada em regras para bloquear tráfego versus criptografia para proteger o conteúdo contra interceptação e adulteração (MitM).

Kraus [39] explora a aplicação da Computação de Borda habilitada por redes 5G para aplicações industriais que exigem latência ultrabaixa e alta confiabilidade. Utilizando simulação com Free5GC e UERANSIM , o trabalho quantifica a redução de latência ao acessar uma rede de dados local (Borda) em comparação com uma rede remota (*Data Center*) , utilizando mecanismos como UL CL para direcionamento de tráfego. A segurança não é o foco principal da análise. Este trabalho é relevante por demonstrar experimentalmente um dos principais motivadores da Computação de Borda (redução de latência), cenário no qual a segurança do fluxo de dados, que é o foco deste TCC, torna-se crucial.

Negri [40] investiga vulnerabilidades em dispositivos IoT de borda, especificamente os Amazon Echo (com Alexa). O estudo discute diversas ameaças, incluindo ataques MitM , e implementa um *proof-of-concept* de autenticação de usuário via SMS (Short Message Service), utilizando uma *skill* da Alexa e armazenamento seguro de credenciais (hashes SHA-256) no *backend*. Embora aborde a segurança em um dispositivo de borda popular e mencione MitM, o foco difere deste TCC, pois se concentra na autenticação do usuário e análise geral de vulnerabilidades, em vez de proteger especificamente o fluxo de dados em trânsito com criptografia contra interceptação/adulteração.

Schenfeld [41] apresenta uma arquitetura híbrida *Fog/Edge Computing* para IoT, visando reduzir latência e dependência da nuvem. A arquitetura define uma camada "Edge" nos próprios dispositivos e uma camada "Fog" em gateways (SoCs). Um aspecto relevante é a inclusão explícita de segurança na comunicação entre as camadas (*Edge-Fog e Fog-Cloud*) por meio dos protocolos TLS/DTLS. Esse trabalho se aproxima deste TCC por implementar segurança para dados em trânsito em uma arquitetura de borda/névoa. Contudo, utiliza os protocolos padrão TLS/DTLS. Embora estes também operem com criptografia híbrida, a proposta deste trabalho se distingue pela implementação dedicada e controlada dos algoritmos AES e RSA, visando demonstrar experimentalmente a eficácia mecânica dessa defesa contra ataques MitM em um cenário de borda, permitindo uma análise detalhada do fluxo cifrado.

Em suma, os trabalhos relacionados demonstram a relevância da Computação de Borda para IoT, seja por benefícios de desempenho (latência, disponibilidade) ou como ponto estratégico para aplicação de segurança (controle de acesso, filtragem, autenticação). Este trabalho contribui para esta área ao focar especificamente na vulnerabilidade do fluxo de dados Borda-Servidor a ataques Man-in-the-Middle e ao implementar e validar a eficácia da criptografia híbrida (AES+RSA) como solução prática para garantir a confidencialidade e a integridade de dados sensíveis neste contexto.

3 MÉTODO DE PESQUISA

Este capítulo detalha a metodologia empregada para o desenvolvimento prático da pesquisa deste trabalho. A pesquisa é classificada como aplicada, pois visa a concepção e implementação de uma solução tecnológica para um problema prático, a vulnerabilidade de dados sensíveis em trânsito em sistemas de borda, e experimental, pois envolveu a construção de um protótipo funcional em um ambiente de laboratório controlado, onde variáveis foram manipuladas para simular ameaças reais e validar a eficácia da contramedida de segurança proposta.

Os objetivos práticos deste método foram estruturados para responder à questão de pesquisa: construir um protótipo de borda capaz de realizar o reconhecimento de placas, analisar sua vulnerabilidade em um cenário padrão, implementar uma camada de criptografia híbrida robusta e, por fim, validar a solução por meio da simulação controlada de ataques MitM e DoS.

3.1 Arquitetura e Ambiente Experimental

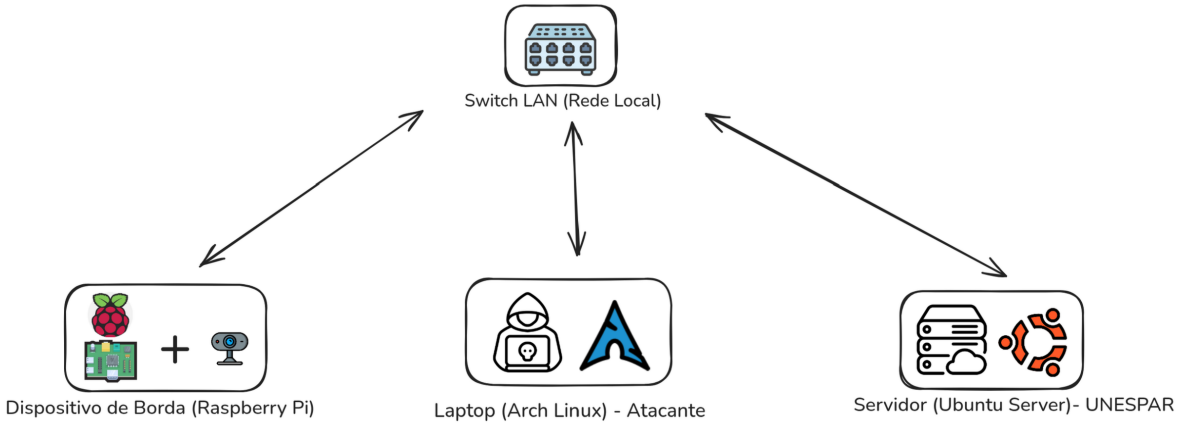
Para a realização dos experimentos, foi configurada uma infraestrutura de rede local (LAN) composta por três componentes de *hardware* principais, simulando um ecossistema de Computação de Borda e a presença de um agente malicioso.

- **Dispositivo de Borda:** Foi utilizado um Raspberry Pi 4 Model B (4 GB de RAM), operando com o sistema Raspberry Pi OS Lite (64 bits) em modo *headless* (sem interface gráfica), com o objetivo de otimizar o uso de recursos. Este dispositivo, equipado com uma *webcam* USB, tinha a função de capturar o vídeo, executar o processamento de LPR localmente e enviar os dados da placa ao servidor.
- **Servidor Central (*Cloud/Backend*):** Um Desktop PC (Intel i5, 8GB RAM), executando o sistema Ubuntu Server representou o ambiente de *backend* do sistema. Este servidor hospedou a API de validação e a base de dados de veículos autorizados, com a aplicação sendo gerenciada via Docker para garantir portabilidade e isolamento.
- **Estação do Atacante:** Um notebook com Arch Linux foi posicionado na mesma rede, destinado à execução das ferramentas de análise de tráfego e à realização dos ataques de interceptação e negação de serviço.

A topologia de rede consistiu em conectar os três dispositivos (borda, servidor e atacante) via cabo a um mesmo switch de rede. A Figura 6 demonstra a topologia

montada para o experimento.

Figura 6 – Topologia de rede montada para o experimento



Fonte: Imagem do Autor

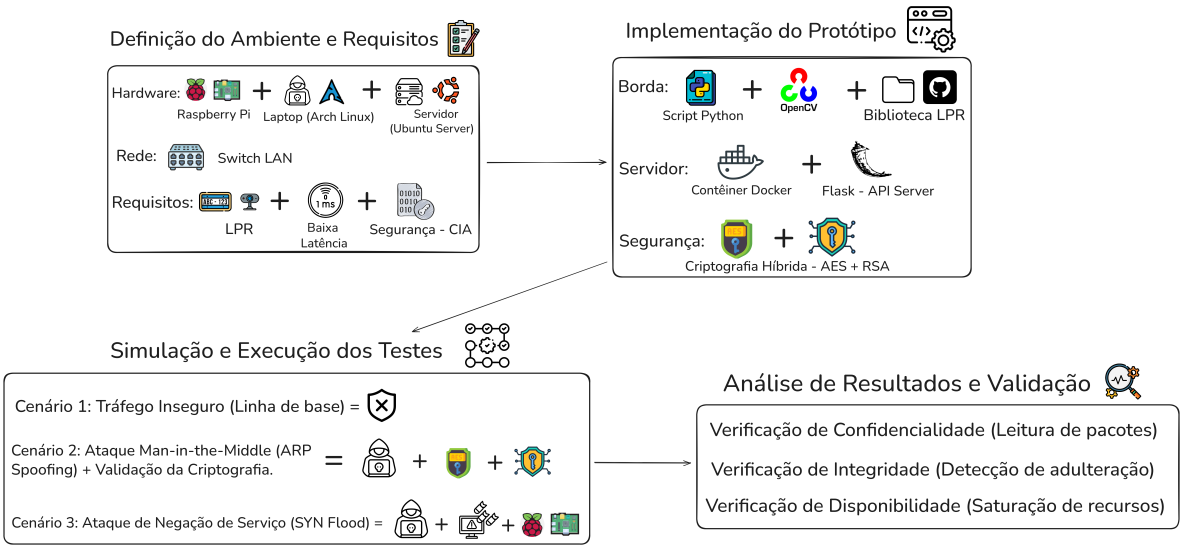
A opção pela utilização exclusiva de uma rede cabeada (Ethernet), em detrimento de uma rede sem fio (WLAN), deu-se por razões de estabilidade e controle do ambiente experimental. Anteriormente, foram realizados testes com interfaces de rede sem fio no servidor, porém foram observadas instabilidades críticas no subsistema de rede do sistema operacional Ubuntu Server. Tais falhas ocasionaram conflitos de roteamento e erros recorrentes durante tentativas de conexão SSH na mesma sub-rede, o que desencadeou um comportamento anormal no serviço de *logging* do sistema (processo *systemd-journald*).

Esse erro gerou uma "tempestade de logs" (*log storm*) que consumiu rapidamente o armazenamento em disco do servidor, comprometendo a execução dos contêineres e a validade dos testes. Portanto, para isolar variáveis de instabilidade de infraestrutura e focar exclusivamente na validação dos protocolos de segurança e criptografia propostos, adotou-se a topologia cabeada via *switch*, garantindo um canal de comunicação estável e previsível para a simulação dos ataques.

O desenvolvimento deste trabalho foi dividido em quatro etapas principais, concebidas para seguir um fluxo lógico desde a concepção até a validação final dos resultados. Para facilitar a compreensão do roteiro metodológico adotado, a Figura 7 apresenta visualmente a sequência das etapas percorridas.

Conforme ilustrado na Figura 7, a primeira etapa consistiu na definição do ambiente e dos requisitos, incluindo a montagem da infraestrutura de *hardware* (Raspberry Pi, servidor e máquina atacante) e a configuração da topologia de rede e a definição dos parâmetros de configuração. Os requisitos definidos foram: análise e processamento da placa do veículo na borda, comunicação constante e eficiente com o servidor, e garantia de confidencialidade e integridade dos dados.

Figura 7 – Fluxograma das etapas do desenvolvimento da pesquisa



Fonte: Imagem do Autor

A segunda etapa foi a implementação do protótipo, na qual os componentes de *software* foram desenvolvidos. Isso incluiu a refatoração e adaptação do código LPR de um projeto de código aberto para uma biblioteca, o desenvolvimento do código do cliente de borda em Python com sua lógica de captura (usando OpenCV), e a criação do servidor de API em Flask e sua conteinerização com Docker, além da integração da criptografia híbrida.

A terceira etapa compreendeu a simulação e execução dos testes, onde foram planejados três cenários distintos para avaliar o sistema: um teste de interceptação (MitM) em tráfego inseguro (texto puro), um teste de interceptação em tráfego seguro (com criptografia híbrida), e um teste de ataque DoS contra o dispositivo de borda.

Por fim, a quarta etapa envolveu a análise de resultados e validação. Nesta fase, os dados coletados foram examinados para verificar se os objetivos de segurança foram atingidos, validando especificamente as métricas de Confidencialidade, Integridade e Disponibilidade do sistema.

3.2 Implementação da Mitigação de Segurança

Para solucionar a vulnerabilidade de interceptação de dados, o método escolhido foi a implementação de um esquema de criptografia híbrida, combinando a velocidade do AES (simétrico) com a segurança do RSA (assimétrico) para a troca de chaves. O procedimento de comunicação segura foi definido da seguinte forma:

- **Preparação:** Um par de chaves RSA (2048 bits) é gerado. A chave privada é armazenada com segurança no servidor (dentro do contêiner Docker), e a chave pública é

transferida para o cliente Raspberry Pi utilizando o protocolo de rede SCP(*Secure Copy Protocol*) durante a fase de provisionamento do ambiente, garantindo que o dispositivo possua a credencial necessária antes do início da execução..

- **Requisição (Cliente):** Ao capturar uma placa, o cliente gera uma chave de sessão AES (128 bits) de uso único. Em seguida, criptografa o dado principal (JSON da placa) com essa chave AES e protege a própria chave AES utilizando a chave pública RSA do servidor. Ambos os dados cifrados são enviados ao servidor.

- **Processamento (Servidor):** O servidor utiliza sua chave privada RSA para descriptografar a chave de sessão AES. Com a chave AES em mãos, ele descriptografa o JSON da placa e realiza a validação.

- **Resposta (Servidor):** O servidor reutiliza a mesma chave de sessão AES para criptografar a resposta do acesso (autorizado/negado) e a envia de volta ao cliente.

- **Validação (Cliente):** O cliente usa sua chave AES original (ainda em memória) para descriptografar a resposta e confirmar a autorização.

Considerando um vetor de ataque restrito à interceptação e adulteração de tráfego de rede, a solução garante a proteção dos dados. Em conformidade com o princípio de Kerckhoffs [42], a segurança do sistema independe do segredo sobre o algoritmo utilizado. Dessa forma, um atacante só obteria êxito na decifragem do conteúdo caso fosse capaz de violar a segurança matemática dos protocolos RSA (2048 bits) e AES (128 bits) ou comprometer a chave privada armazenada no servidor.

3.3 Métricas e Critérios de Avaliação

Para avaliar a segurança da arquitetura, foram planejados três cenários de teste, utilizando ferramentas específicas para simular ameaças. As ferramentas de análise selecionadas foram o Wireshark [43] e o `tshark` [44], utilizadas para a captura e inspeção detalhada de pacotes de rede. Para a simulação dos ataques, empregaram-se as ferramentas `hping3` [45], para gerar o ataque DoS do tipo *SYN Flood*, e a ferramenta `arpspoof` [46] (do pacote `dsniff` [47]), para realizar o envenenamento de cache ARP, etapa fundamental para a execução do ataque *Man-in-the-Middle*.

O plano de testes, cujos resultados serão apresentados no Capítulo 4, foi dividido nos seguintes cenários:

- Cenário 1 – Teste de Vulnerabilidade: consistiu na execução do sistema sem criptografia, a fim de verificar a visibilidade dos dados (placa) em texto puro;

- Cenário 2 – Teste de Validação da Solução: repetiu o mesmo ataque MitM contra o sistema protegido com criptografia híbrida, buscando validar a confidencialidade e a integridade dos dados;
- Cenário 3 – Teste de Disponibilidade: concentrou-se na execução de um ataque DoS com o hping3, direcionado ao dispositivo de borda, avaliando o impacto desse ataque sobre a sua operação.

A avaliação dos resultados foi conduzida de forma qualitativa e comparativa, contrastando o comportamento do sistema e a natureza dos pacotes capturados em cada cenário. Os critérios de avaliação foram fundamentados nos pilares da segurança da informação:

- Confidencialidade: Avaliada pela verificação da legibilidade dos dados interceptados. O critério de sucesso para a solução é que os dados trafegados sejam ininteligíveis para o atacante;
- Integridade: Avaliada pela capacidade de detectar adulterações. O critério de sucesso é que qualquer modificação no pacote cifrado resulte em falha na descriptografia, impedindo a injeção de dados falsos;
- Disponibilidade: Avaliada pela observação do impacto do ataque de negação de serviço sobre a operação do dispositivo de borda, medindo a saturação de recursos (CPU) e a interrupção do serviço de comunicação.

Essas métricas permitiram comprovar experimentalmente se a criptografia híbrida adotada era suficiente para garantir a confidencialidade e integridade dos dados em trânsito, bem como identificar vulnerabilidades adicionais relacionadas à disponibilidade.

3.4 Uso de Ferramentas de Inteligência Artificial

Este trabalho contou com o apoio de ferramentas de Inteligência Artificial generativa (Gemini, modelo 2.5, do Google), utilizadas para revisão linguística sob supervisão e validação do autor.

4 EXPERIMENTOS

Este capítulo documenta a execução dos testes práticos descritos na metodologia de pesquisa. O foco é apresentar os três cenários experimentais que foram projetados para avaliar a arquitetura do sistema. Cada cenário foi elaborado para testar um aspecto específico da segurança: a operação normal e sua vulnerabilidade inerente, a capacidade de um atacante interceptar e adulterar os dados (Ataque *Man-in-the-Middle*), e a resiliência do dispositivo de borda a ataques de disponibilidade (Negação de Serviço). Os resultados detalhados, capturas de tela e evidências de cada experimento serão apresentados e analisados no Capítulo 5.

4.1 Configuração do Ambiente Experimental

A primeira etapa da execução consistiu na montagem e validação do ambiente de testes. A Figura 8 ilustra a estrutura de arquivos utilizada no servidor de API, conteinerizado por meio do Docker.

Figura 8 – Arquivos utilizados no ambiente do Servidor

Fonte: Imagem do Autor

Os componentes essenciais do ambiente de servidor são:

- `servidor.py`: *Script* principal da aplicação, escrito em Python utilizando o micro-*framework* Flask. Este arquivo é o responsável por criar a API de validação, escutar requisições HTTP, receber os dados da placa e verificar sua existência no arquivo `placas_cadastradas.json` para retornar a autorização.
- `Dockerfile`: Arquivo de receita para a conteinerização da aplicação. Automatiza a instalação do sistema operacional base e das dependências listadas em `requirements.txt`, garantindo um ambiente isolado e reprodutível.
- `placas_cadastradas.json`: Arquivo JSON que simula uma base de dados NoSQL, contendo a lista de placas autorizadas.

- `requirements.txt`: Lista as bibliotecas Python necessárias, incluindo Flask (para a API) e `pycryptodome` (para as operações criptográficas).
- `chave_privada.pem`: Componente crítico de segurança, a chave privada RSA de 2048 bits, mantida em segredo no servidor para descriptografar a chave de sessão AES.

A Figura 9 mostra a estrutura de arquivos montada para o ambiente do dispositivo de borda (Raspberry Pi).

Figura 9 – Arquivos utilizados no ambiente do Raspberry Pi

Fonte: Imagem do Autor

Os componentes do ambiente de borda são:

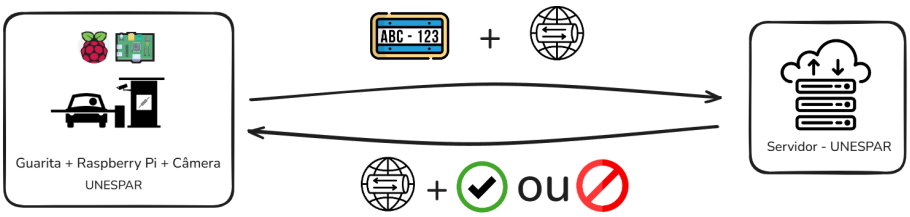
- `cliente_borda.py`: *Script* principal que orquestra a lógica de borda. Utiliza OpenCV para captura de vídeo, aciona a biblioteca `lpr_lib` para processamento e `requests` para comunicação com o servidor.
- `chave_publica.pem`: Chave pública RSA correspondente à chave privada do servidor, usada para criptografar a chave de sessão AES antes do envio.
- `requirements.txt`: Define as bibliotecas do cliente, incluindo `opencv-python`, `requests`, `pycryptodome` e `matplotlib`.
- `lpr_lib` (Pasta): Biblioteca modular resultante da refatoração do código LPR, permitindo a importação da função `"extrair_placa_do_frame()"` pelo *script* principal.

4.2 Cenário 1: Fluxo de Dados Básico (Inseguro)

O primeiro cenário estabeleceu a base funcional do sistema, representando a comunicação em seu estado mais simples e inseguro.

Como ilustrado na Figura 10, este cenário corresponde ao modelo de arquitetura fundamental do projeto. O dispositivo de borda (Guarita + Raspberry Pi + Câmera)

Figura 10 – Tráfego inseguro entre a Borda e o Servidor



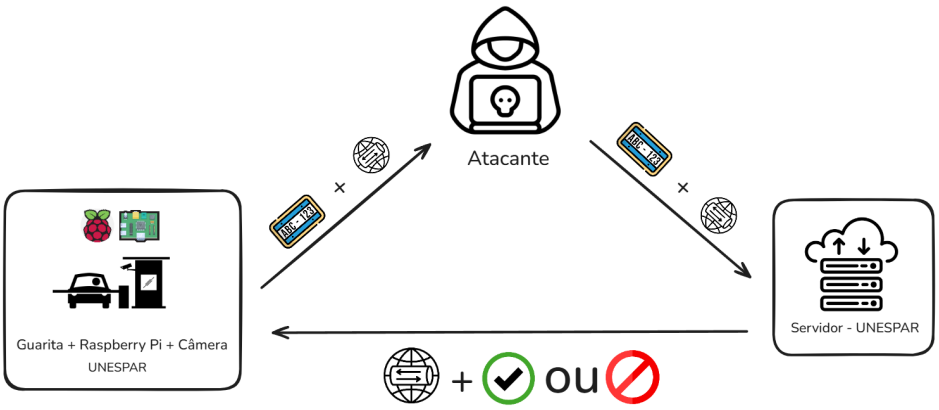
Fonte: Imagem do Autor

executa o processamento local de LPR para identificar uma placa de veículo (ex: “ABC-123”). Após a extração, o texto da placa é enviado pela rede LAN para o servidor. O servidor verifica na base de dados se a placa está cadastrada e retorna uma resposta de autorização (Aprovado ou Negado). Neste cenário, não há aplicação de criptografia, servindo como linha de base para a análise de vulnerabilidade.

4.3 Cenário 2: Demonstração da Vulnerabilidade (MitM em Tráfego Inseguro e Seguro)

O segundo cenário introduziu um “Atacante” na mesma rede local para simular o ataque MitM e testar a eficácia da criptografia híbrida implementada. O diagrama do ataque é apresentado na Figura 11.

Figura 11 – Diagrama demonstrando o ataque MitM



Fonte: Imagem do Autor

Utilizando a técnica de *ARP Spoofing*, a máquina do atacante foi posicionada logicamente entre o dispositivo de borda e o servidor, desviando todo o fluxo de tráfego. É fundamental destacar que, neste experimento, o *ARP Spoofing* foi utilizado estritamente como método de interceptação para viabilizar o ataque do tipo passivo, conhecido como *eavesdropping*. Dessa forma, o atacante limitou-se a capturar e analisar os pacotes em

trânsito para verificar a exposição dos dados, sem realizar injeção ou alteração ativa de pacotes durante a captura. Este experimento foi dividido em duas fases:

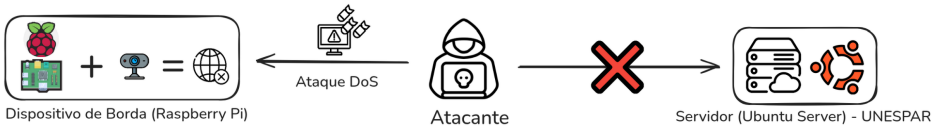
1. Teste de Vulnerabilidade: O ataque MitM foi executado contra o fluxo de dados inseguro do Cenário 1. A hipótese era que o atacante conseguiria violar a confidencialidade (lendo os dados em texto puro) e a integridade (tendo a capacidade de adulterar os dados).

2. Teste de Validação: O mesmo ataque MitM foi repetido contra o sistema com a criptografia híbrida (AES+RSA) ativada. A hipótese era que a solução se mostraria eficaz, tornando os dados interceptados cifrados e indecifráveis, impedindo a leitura e a adulteração.

4.4 Cenário 3: Teste de Disponibilidade (Ataque DoS)

O terceiro cenário avaliou um vetor de ataque distinto, focado na disponibilidade do *hardware* de borda. O ataque MitM posiciona o atacante na rede, permitindo não apenas a interceptação, mas também ataques de negação de serviço.

Figura 12 – Diagrama demonstrando o Ataque DoS do tipo *SYN Flood*



Fonte: Imagem do Autor

Como demonstrado na Figura 12, este experimento investigou se um atacante na mesma rede poderia paralisar a operação do dispositivo de borda. A hipótese é que o Raspberry Pi, por ser um *hardware* de recursos limitados, é suscetível a ataques de inundação de rede, como um *SYN Flood*. O atacante direcionou um alto volume de pacotes maliciosos ao IP do Raspberry Pi com o objetivo de saturar seus recursos de CPU e rede, impedindo a execução de suas tarefas essenciais. O “X” vermelho no diagrama simboliza a falha do dispositivo e a interrupção do serviço.

5 RESULTADOS

Este capítulo apresenta os resultados e as evidências práticas coletadas durante a execução dos cenários definidos no Capítulo 4, documentando as observações e analisando os dados à luz das métricas de segurança.

5.1 Resultados do Cenário 1: Tráfego Inseguro (Man-in-theMiddle)

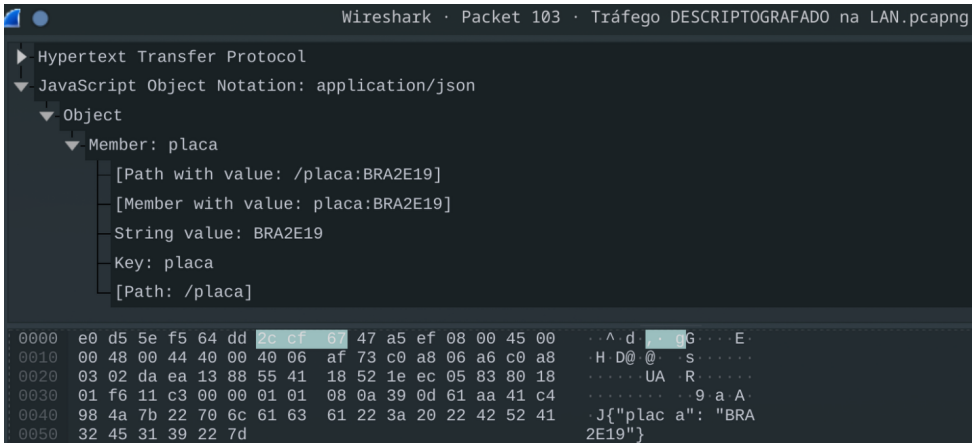
Nesta fase, o sistema foi executado sem criptografia. A Figura 13 mostra o processamento local bem-sucedido da placa "BRA2E19" pelo Raspberry Pi.

Figura 13 – Placa analisada e processada pelo Raspberry Pi

Fonte: Imagem do Autor

Em seguida, na Figura 14 apresenta a evidência crucial capturada pelo Wireshark. Como pode ser observado, o tráfego inseguro expõe o pacote HTTP `POST` com o *payload* JSON ("placa": "BRA2E19") em texto puro. Esta captura comprova uma falha crítica de confidencialidade, permitindo que qualquer pessoa na rede leia os dados sensíveis. Embora a ação realizada tenha sido apenas de escuta passiva (*eavesdropping*), a exposição em texto claro confirma que não existem barreiras técnicas impedindo a perda de integridade, pois um atacante que pode ler este formato pode facilmente replicá-lo ou adulterá-lo, como explorado na metodologia.

Figura 14 – Placa Interceptada no Wireshark

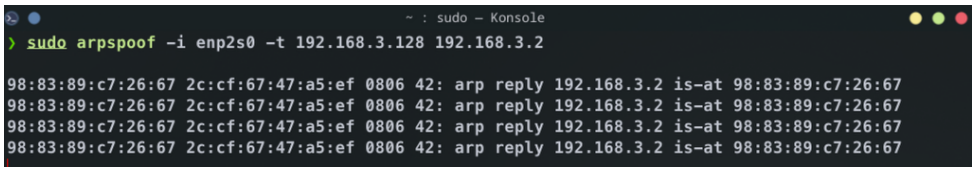


Fonte: Imagem do Autor

5.2 Resultados do Cenário 2: Tráfego Seguro (Validação da Criptografia Híbrida)

Este experimento validou a solução de criptografia híbrida sob as mesmas condições de ataque MitM descritas no Cenário 2. Primeiro, para evidenciar a execução do ataque MitM, foram utilizados dois comandos `arpspoof` em terminais separados na máquina do atacante. A Figura 15 documenta o envenenamento da tabela ARP do Raspberry Pi (-t 192.168.3.128), instruindo-o a enviar pacotes destinados ao servidor (192.168.3.2) para o endereço MAC do atacante.

Figura 15 – Ataque de *ARP spoofing* no Raspebrry Pi



Fonte: Imagem do Autor

```
sudo arpspoof -i enp2s0 -t 192.168.3.128 192.168.3.2
```

Comando 1: Envenenando o Raspberry Pi

`sudo:` Permite executar o comando com privilégios de administrador.

`arpspoof:` É o nome do comando que executa o ataque de envenenamento de cache ARP (*ARP Spoofing*).

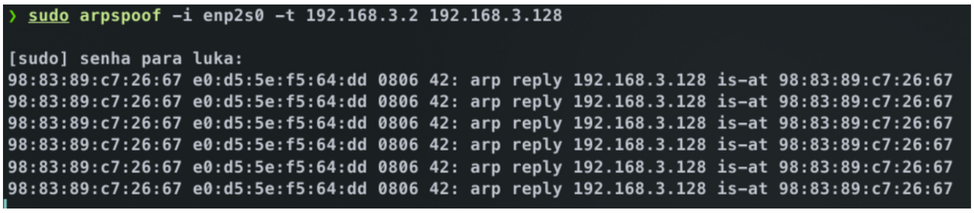
`-i enp2s0:` Define a **i**nterface de rede que será usada para enviar os pacotes ARP falsificados.

`-t 192.168.3.128:` Define o IP do **t**arget (alvo) que será envenenado (neste caso, o Raspberry Pi).

`192.168.3.2:` Define o endereço IP que o atacante irá personificar (neste caso, o Servidor).

Simultaneamente, a Figura 16 documenta o envenenamento do servidor (-t 192.168.3.2), instruindo-o a enviar as respostas destinadas ao Raspberry Pi (192.168.3.128) também para o atacante. Com estes dois comandos, o atacante se posicionou com sucesso como o intermediário de toda a comunicação.

Figura 16 – Ataque de *Arpspoofing* no Servidor



Fonte: Imagem do Autor

```
sudo arpspoof -i enp2s0 -t 192.168.3.2 192.168.3.128
```

Comando 2: Envenenando o Servidor

`sudo:` Permite executar o comando com privilégios de administrador.

`arpspoof:` É o nome do comando que executa o ataque de envenenamento de cache ARP (*ARP Spoofing*).

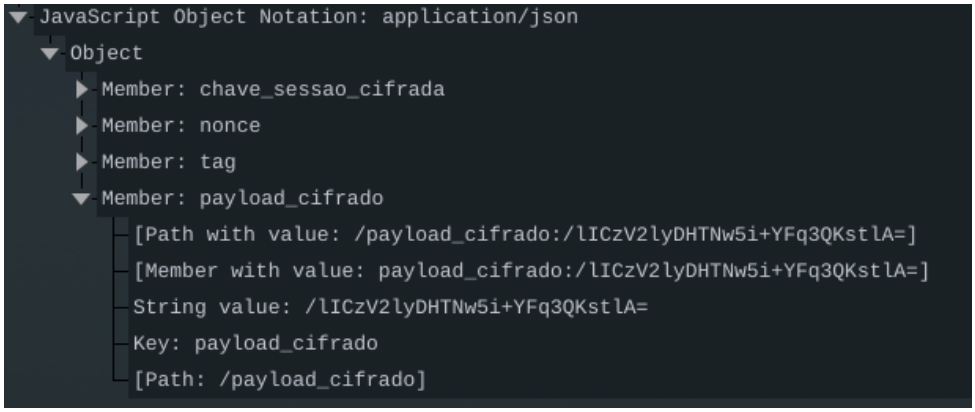
`-i enp2s0:` Define a **i**nterface de rede que será usada para enviar os pacotes ARP falsificados.

`-t 192.168.3.2:` Define o IP do **t**arget (alvo) que será envenenado (neste caso, o Raspberry Pi).

`192.168.3.128:` Define o endereço IP que o atacante irá personificar (neste caso, o Servidor).

Com o tráfego interceptado, os dados do sistema seguro foram analisados. A Figura 17 exibe o *payload* capturado do cliente. Diferentemente do Cenário 1, o conteúdo da placa ("BRA2E19") não é visível. Em seu lugar, observa-se um objeto JSON contendo a chave de sessão criptografada, o vetor de inicialização (*nonce*), a *tag* de autenticação e o `payload_cifrado`, que é uma sequência de caracteres codificados em Base64.

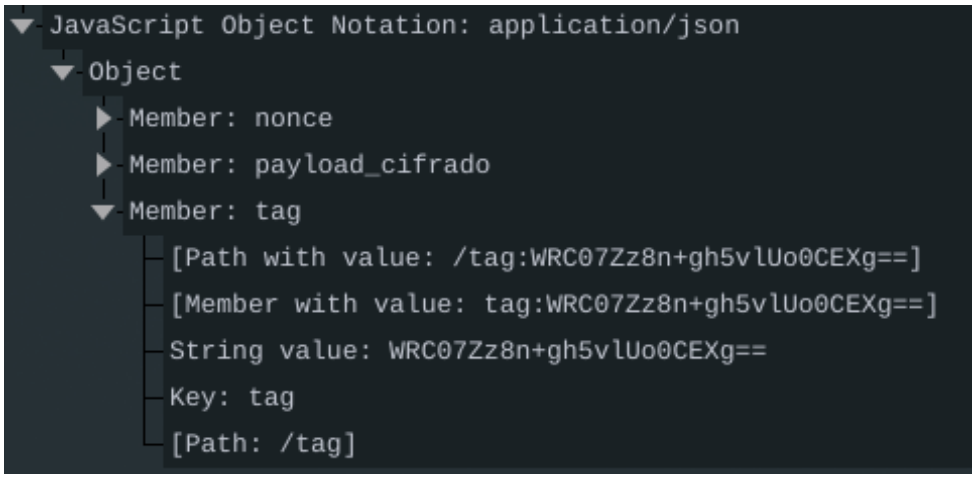
Figura 17 – Tráfego criptografado pela criptografia híbrida - *payload* do cliente



Fonte: Imagem do Autor

Da mesma forma, a Figura 18 mostra que a resposta do servidor também está cifrada, ocultando se a placa foi autorizada ou não.

Figura 18 – Tráfego criptografado pela criptografia híbrida - *payload* do servidor



Fonte: Imagem do Autor

Estas evidências comprovam que a solução foi eficaz em garantir a Confidencialidade e a Integridade. O atacante não consegue ler o conteúdo (confidencialidade) e qualquer alteração no texto cifrado invalidaria a *tag* de autenticação gerada pelo AES-GCM, fazendo com que o servidor rejeitasse o pacote (integridade).

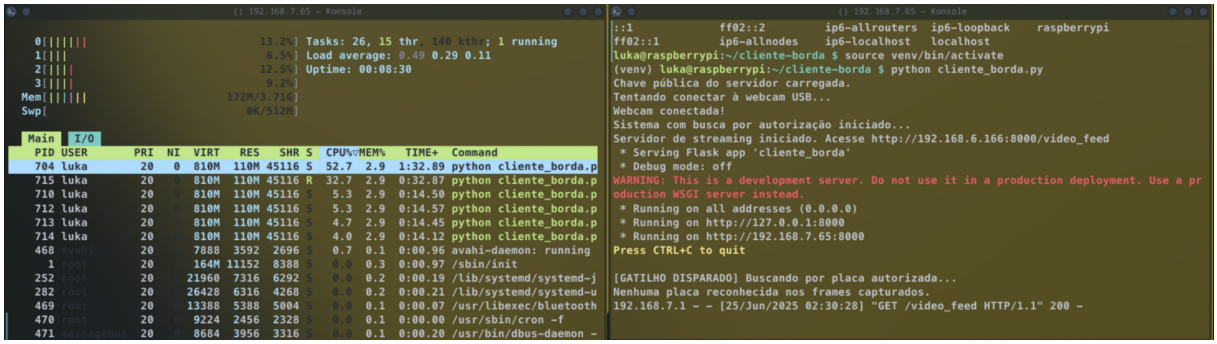
No entanto, é importante destacar o custo da segurança observado nesta análise. Ao comparar o tamanho do pacote HTTP no Cenário 1 (Figura 14), que continha apenas um JSON simples e leve, com o pacote do Cenário 2 (Figura 17), nota-se um aumento significativo no tamanho do *payload* (overhead). Esse aumento deve-se à necessidade de transmitir metadados adicionais de segurança, especialmente a chave de sessão AES criptografada com RSA de 2048 bits, que ocupa um espaço considerável, além do *nonce* e da

tag. Esse incremento no consumo de banda é o compromisso (*trade-off*) necessário para assegurar a proteção dos dados em um meio não confiável.

5.3 Resultados do Cenário 3: Ataque DoS

O último experimento avaliou a vulnerabilidade de Disponibilidade. A Figura 19 estabelece a linha de base, mostrando o Raspberry Pi em operação normal com carga de CPU estável.

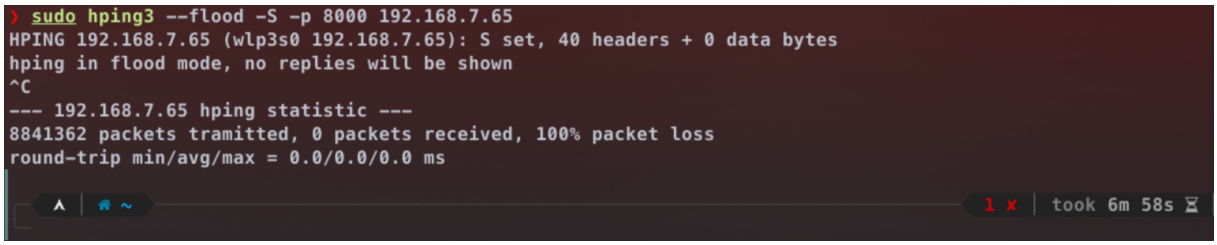
Figura 19 – Raspberry Pi e a aplicação operando normalmente



Fonte: Imagem do Autor

Em seguida, o ataque SYN Flood foi iniciado usando a ferramenta `hping3`, conforme demonstrado na Figura 20, registrando o envio massivo de milhões de pacotes.

Figura 20 – Terminal do atacante executando o ataque com o hping3



Fonte: Imagem do Autor

O comando utilizado é detalhado abaixo:

```
sudo hping3 --flood -S -p 8000 192.168.7.65
```

Comando 3: Ataque DoS (SYN Flood) ao Raspberry Pi

`sudo`: Permite executar o comando com privilégios de administrador.

`hping3`: É o nome da ferramenta, um gerador e analisador de pacotes de rede utilizado para testes de segurança.

`-flood:` Instrui a ferramenta a enviar pacotes o mais rápido possível (modo "inundação"), sem aguardar por respostas.

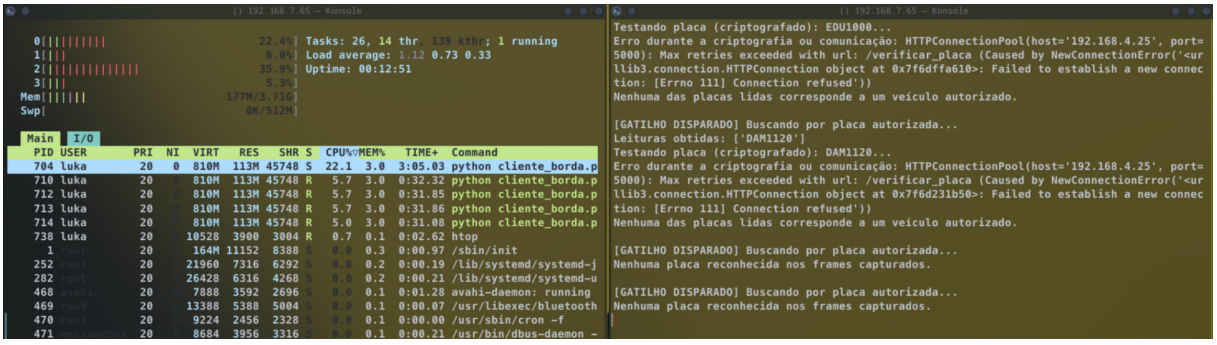
`-S:` Define o *flag* **S**YN (de sincronização) no cabeçalho do pacote TCP, caracterizando o ataque como um *SYN Flood*.

`-p 8000:` Define a **p**orta de destino do ataque (neste caso, a porta 8000, onde o serviço de streaming de vídeo do cliente estava a ser executado).

`192.168.7.65:` Define o endereço IP do alvo que receberá a inundação de pacotes (neste caso, o Raspberry Pi).

A Figura 21 documenta o impacto imediato no dispositivo. O painel `htop` mostra um aumento significativo na carga média (*Load Average* para 1.12), indicando sobrecarga no processamento de interrupções de rede. O resultado crítico é observado no log da aplicação, que passa a registrar erros sistemáticos de "[Errno 111] Connection refused".

Figura 21 – Raspberry Pi durante o ataque de *SYN Flood*



Fonte: Imagem do Autor

Este experimento evidenciou uma vulnerabilidade crítica na Disponibilidade. O subsistema de rede do Raspberry Pi foi saturado, exaurindo o *backlog* de conexões e impedindo o estabelecimento de novas conexões legítimas, paralisando efetivamente o serviço. Para mitigar este cenário em trabalhos futuros e fortalecer a disponibilidade, recomenda-se a implementação de regras de *firewall* (como `iptables`) diretamente no dispositivo de borda para limitar a taxa de novas conexões (*rate limiting*) ou o uso de mecanismos de filtragem de pacotes de alto desempenho no *kernel*, como o eXpress Data Path (XDP), para descartar tráfego malicioso antes que ele consuma recursos significativos do sistema.

6 CONCLUSÃO

Este trabalho alcançou seu objetivo geral de avaliar experimentalmente a segurança em Computação de Borda, validando a eficácia da criptografia híbrida na proteção do fluxo de dados. Em relação aos objetivos específicos, o protótipo utilizando Raspberry Pi foi implementado com sucesso, permitindo a simulação realista de um cenário de controle de acesso. A análise de vulnerabilidade confirmou, por meio de interceptação passiva, a exposição crítica de dados em redes inseguras. A implementação da criptografia híbrida mitigou esse risco, garantindo a confidencialidade e integridade das informações. Por fim, os testes de estresse demonstraram as limitações de disponibilidade do dispositivo de borda frente a ataques de negação de serviço

A execução do Cenário 1 (Inseguro) forneceu evidências claras de falhas de confidencialidade e integridade, onde os dados sensíveis (placas de veículos) foram interceptados e lidos em texto puro. Para solucionar esta vulnerabilidade, a implementação de uma criptografia híbrida (AES+RSA), detalhada na metodologia, foi validada no Cenário 2. Os resultados comprovaram que a solução foi eficaz, tornando os dados interceptados indecifráveis e garantindo a segurança do fluxo de dados nos dois sentidos (requisição e resposta) contra adulteração e espionagem.

Adicionalmente, a pesquisa expôs uma segunda vulnerabilidade crítica, investigada no Cenário 3. Foi demonstrado que a disponibilidade do dispositivo de borda (Raspberry Pi), por ser um *hardware* de recursos limitados, é um vetor de ataque distinto. Um ataque DoS do tipo *SYN Flood* foi capaz de saturar os recursos do dispositivo e paralisar a aplicação, provando que a segurança de borda deve ir além da criptografia. Como limitações, este estudo focou na validação funcional da criptografia sem uma análise quantitativa do *overhead* de desempenho (latência/CPU) e não implementou contramedidas para o ataque DoS.

Para trabalhos futuros, recomenda-se a realização dessa análise de desempenho da criptografia, a implementação e teste de mecanismos de mitigação de DoS na "borda" (como `iptables`) e a investigação de métodos escaláveis para o gerenciamento de chaves em um ambiente com múltiplos dispositivos. Conclui-se que a Computação de Borda exige uma estratégia de segurança deliberada, e que a criptografia híbrida é uma solução prática e eficaz para proteger a confidencialidade e a integridade dos dados gerados por ela.

REFERÊNCIAS

[1] FERASOLI, M. A. B. Detecção e reconhecimento de placas de automóveis para controle de acesso em condomínios residenciais por meio de visão computacional. Centro Universitário Sagrado Coração-UNISAGRADO, 2017.

[2] YILDIRIM, F. et al. Comprehensive review of edge computing for power systems: State of the art, architecture, and applications. *Applied Sciences (2076-3417)*, v. 15, n. 8, 2025.

[3] GILL, B.; RAO, S. Technology insight: Edge computing in support of the internet of things. *Gartner Research*, 2017.

[4] FONTES, E. L. G. Segurança da informação. *Saraiva Educação SA*, p. 2, 2017.

[5] COUTO, K. S. et al. Os três pilares da segurança da informação na internet chinesa. *Journal of Technology & Information (JTnI)*, v. 2, n. 2, 2022.

[6] KHAN, M. M. et al. License plate recognition methods employing neural networks. *IEEE access*, IEEE, v. 11, p. 73613–73646, 2023.

[7] DUSTDAR, S.; PUJOL, V. C.; DONTA, P. K. On distributed computing continuum systems. *IEEE Transactions on Knowledge and Data Engineering*, IEEE, v. 35, n. 4, p. 4092–4105, 2022.

[8] KUMAR, L.; SHARMA, R. K. Examining interdependencies among solution dimensions for sustainable development in smes based on industry 4.0 concept. *Kybernetes*, Emerald Publishing Limited, v. 54, n. 4, p. 2137–2174, 2025.

[9] RAHIMI, M. R. et al. Mobile cloud computing: A survey, state of art and future directions. *Mobile Networks and Applications*, Springer, v. 19, n. 2, p. 133–143, 2014.

[10] Tecnoomega. *Conheça os modelos IaaS, PaaS e SaaS!* 2024. <https://tecnomega.com.br/blog/conheca-os-modelos-iaas-paas-saas/>. Acesso em: 13 out. 2025.

[11] SATYANARAYANAN, M. The emergence of edge computing. *Computer*, IEEE, v. 50, n. 1, p. 30–39, 2017.

[12] MTAWA, Y. A.; HAQUE, A.; BITAR, B. The mammoth internet: Are we ready? *IEEE Access*, IEEE, v. 7, p. 132894–132908, 2019.

[13] LOPES, M. H. d. M. D. Análise do potencial de sistemas híbridos lorawan-edge computing para aprimoramento da eficiência e segurança em operações de mineração subterrânea. 2025.

[14] GONÇALVES, A. L. d. J.; FREITAS, L. A.; OLIVEIRA-JR, A. Arquitetura de dimensionamento adaptativo com suporte ao aprendizado. In: SBC. *Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC)*. [S.l.], 2025. p. 350–363.

[15] MAHMOUDI, C.; MOURLIN, F.; BATTOU, A. Formal definition of edge computing: An emphasis on mobile cloud and iot composition. In: IEEE. *2018 Third international conference on fog and mobile edge computing (FMEC)*. [S.l.], 2018. p. 34–42.

[16] CRISTINO, L.; CAMINHA, P. Caracterização da latência de borda sob efeito de mobilidade a partir de dados reais. In: *Anais do XXIX Workshop de Gerência e Operação de Redes e Serviços*. Porto Alegre, RS, Brasil: SBC, 2024. p. 112–125. ISSN 2595-2722. Disponível em: <https://sol.sbc.org.br/index.php/wgrs/article/view/30091>.

[17] IEEE Innovation at Work. *Real-Life Edge Computing Use Cases*. 2021. <https://innovationatwork.ieee.org/real-life-edge-computing-use-cases/>. Acesso em: 18 out. 2025.

[18] SAKASHITA, N.; KOSLOVSKI, G. Um estudo preliminar sobre o impacto de protocolos para controle de congestionamento em edge-cloud continuum. In: *Anais da XXIV Escola Regional de Alto Desempenho da Região Sul*. Porto Alegre, RS, Brasil: SBC, 2024. p. 53–56. ISSN 2595-4164. Disponível em: <https://sol.sbc.org.br/index.php/eradrs/article/view/28001>.

[19] NETO, V. J. d. S.; BONACELLI, M. B. M.; PACHECO, C. A. O sistema tecnológico digital: inteligência artificial, computação em nuvem e big data. *Revista Brasileira de Inovação*, v. 19, n. 0, p. e0200024, dec 2020.

[20] JUNIOR, F. M. R.; KAMIENSKI, C. A. Resiliência de dados da bruma computacional na internet das coisas. In: SBC. *Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC)*. [S.l.], 2021. p. 504–517.

[21] PIANTINO, J. M. *VISÃO COMPUTACIONAL E INSPEÇÃO AUTOMATIZADA: UMA ABORDAGEM PARA A MELHORIA CONTÍNUA NA PRODUÇÃO INDUSTRIAL*. Lavras, 2025.

[22] MATOS, I. de et al. Sistemas de visão computacional para monitoramento e automação de centros de distribuição: Computer vision systems for monitoring and automation of distribution centers. *RCMOS-Revista Científica Multidisciplinar O Saber*, v. 1, n. 1, 2023.

[23] SILVA, J. V. S. *AtalaIA: Uso de Deep Learning para reconhecimento automático de placas de licença automotiva em dispositivos com recursos limitados*. São Cristóvão, 2022.

[24] MATOS, I. d. Sistemas de Visão Computacional para Monitoramento e Automação de Centros de Distribuição. *RCMOS - Revista Científica Multidisciplinar O Saber*, III, n. 1, 2023. ISSN 2675-9128.

[25] QUEIROZ, A. A. L. et al. Autenticação com suporte à Computação de Borda 5G para a Internet de Veículos. *Brazilian Journal of Development*, Brazilian Journal of Development, v. 9, n. 5, p. 14613–14631, 2023.

[26] VARELLA, J. ComputaÇÃo em nÉvoa. *Revista EDUC-Faculdade de Duque de Caxias*, v. 6, n. 1, p. 31–46, 2019. Acesso em: 20 out. 2025. Disponível em: <https://uniesp.edu.br/sites/_biblioteca/revistas/20200910104431.pdf>.

[27] AMORIM, R. H. B. *MONITORANDO REDES EM INTERNET DAS COISAS (IoT): segurança de baixo custo com Suricata*. Dissertação (Dissertação de Mestrado) — Universidade Federal de Pernambuco, Recife, 2024.

[28] ALENCAR, G. H. G. COMO O DOCKER PODE AJUDAR A DIMINUIR CUSTOS DAS SUAS IMPLANTAÇÕES. *Revista Contemporânea*, v. 5, n. 5, p. e8141, 2025. Acesso em: 18 out. 2025. Disponível em: <https://ojs.revistacontemporanea.com/ojs/index.php/home/article/view/8141>.

[29] Podman project. *Podman documentation*. 2025. <https://docs.podman.io/en/latest/>. Acesso em: 18 out. 2025.

[30] The Kubernetes Authors. *Overview - Kubernetes*. 2025. <https://kubernetes.io/docs/concepts/overview/>. Acesso em: 18 out. 2025.

[31] CASELLA, M. M. *Estudo comparativo de performance entre máquina virtual e container Docker*. Brasília, 2023. Acesso em: 18 out. 2025. Disponível em: <https://bdm.unb.br/bitstream/10483/39100/1/2023_MaysaMeirellesCasella_tcc.pdf>.

[32] MIERS, C. C. et al. Uma análise de segurança no uso de contêineres docker em nuvens iaas openstack. *Anais do Computer on the Beach*, v. 10, p. 021–030, 2019.

[33] NASCIMENTO, A. J. do; COSTA, D. B. de A. A segurança da informação: uma observação sobre os aspectos técnicos, humanos, sociais e jurídicos conhecidos. *Revista Brasileira em Tecnologia da Informação*, v. 6, n. 1, p. 59–68, 2024.

[34] FERREIRA, J. P. d. S. *Segurança da informação nas pequenas empresas*. Brasília, 2021. 36 p.

[35] LIMA, D. A. d. Virtualização de sistema de detecção de ataque de negação de serviço. Universidade Federal do Rio Grande do Norte, 2021.

[36] FERREIRA, G. G.; CASTALDIN, A. G. Estudo de ataque man-in-the-middle com software cain&abel. In: *FatecSeg-Congresso de Segurança da Informação*. [S.l.: s.n.], 2022.

[37] CESAR, R. L. V. *UMA ARQUITETURA DE CONTROLE DE ACESSO PARA INTERNET DAS COISAS*. Dissertação (Dissertação (Mestrado Acadêmico em Computação)) — Universidade Federal do Ceará, Quixadá, 2022.

[38] SILVERIO, A. E.; GUARDIA, H. C. *FILTRAGEM DE PACOTES NA BORDA DA REDE: UMA ANÁLISE COMPARATIVA COM FOCO NO CONSUMO DE ENERGIA*. São Carlos, 2023.

[39] KRAUS, D. *Computação de borda para indústria utilizando a rede 5G*. Blumenau, 2021.

[40] NEGRI, G. B. P. *VULNERABILIDADES EM IOT: DISPOSITIVOS ECHO*. Ouro Preto, 2025.

[41] SCHENFELD, M. C. *FOG E EDGE COMPUTING: UMA ARQUITETURA HÍBRIDA EM UM AMBIENTE DE INTERNET DAS COISAS*. Dissertação (Dissertação (Mestre em Ciência da Computação)) — Pontifícia Universidade Católica do Rio Grande do Sul, Porto Alegre, 2017.

[42] MAKOKHA, F. The validity of kerckhoff's principle in the era of emerging technologies: A case study of cryptology in wireless telephony services. *DS Journal of Digital Science and Technology*, Dream Science, v. 4, 2025.

[43] Wireshark Foundation. *Wireshark: Go deep.* 2025. <https://www.wireshark.org/>. Acesso em: 3 nov. 2025.

[44] Wireshark Foundation. *TShark(1) Manual Page*. 2025. <https://www.wireshark.org/docs/man-pages/tshark.html>. Acesso em: 3 nov. 2025.

[45] Kali Linux. *hping3*. 2025. <https://www.kali.org/tools/hping3/>. Acesso em: 3 nov. 2025.

[46] SMIKIMS. *arpspoof: A simple arpspoofer in python*. 2025. <https://github.com/smikims/arpspoof>. Acesso em: 3 nov. 2025.

[47] Kali Linux. *dsniff*. 2025. <https://www.kali.org/tools/dsniff/>. Acesso em: 3 nov. 2025.